Agnes Camilla Tysland

Optimal Operation and Design of a Thermal Energy Storage Tank

Master's thesis in Chemical Engineering and Biotechnology
Supervisor: Johannes Jäschke
Co-supervisor: Caroline Satye Nakama
June 2021

Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Natural Sciences
Department of Chemical Engineering

NTNU
Norwegian University of
Science and Technology

Agnes Camilla Tysland

Optimal Operation and Design of a Thermal Energy Storage Tank

Master's thesis in Chemical Engineering and Biotechnology
Supervisor: Johannes Jäschke
Co-supervisor: Caroline Satye Nakama
June 2021

Norwegian University of Science and Technology
Faculty of Natural Sciences
Department of Chemical Engineering

NTNU
Norwegian University of
Science and Technology

Summary

Thermal energy storage (TES) provides an opportunity of reducing energy waste and a possibility of increasing network flexibility in district heating systems. Sensible TES systems in the form of pressurized hot water tanks may be used for short-term local energy storage. For efficient utilization of such systems, the sizing of the tank is crucial. In addition, the optimal size will dependent on waste heat availability and the corresponding energy demand. In this work, a simplified process is defined, and a nonlinear dynamic optimization problem is formulated. Formulations including complementarity constraints are contemplated.

This thesis focuses on the combined optimal operation and optimal design of a TES tank in a district heating network located in northern Norway, utilizing waste heat from a nearby ferrosilicon production plant. Electricity and CO gas are normally used as peak heating sources when demand is too high, and are considered as two different case studies for the economic evaluation in this work. Based on historical data, optimal operation and optimal sizing of the tank were obtained. The optimal volume of the TES tank was found to be 6322.95 m^3 for both cases, causing a decrease in peak heating of 421 MW for the 30 day-period investigated. The payback period for this investment is expected to be 13.74/12.18 years when using electricity/CO gas, respectively. However, if the announced tax increase from the Norwegian government is resolved in the parliament, this payback period is decreased to 5.7 years for the CO gas case. Sensitivity analyses have been conducted for both cases, considering a relative change in various factors affecting the payback time, such as interest rate, emission taxes and price of the energy source. The analyses suggest the active period of the TES tank throughout a year to be the most impactful parameter, closely followed by the price of the respective energy source and the interest rate. For CO gas, the payback period is prone to changes in the CO_2 tax as well.

Sammendrag

Lagring av termisk energi (TES) gir en mulighet for å redusere energisvinn og mulighet for å øke nettverksfleksibiliteten i fjernvarmesystemer. Fornuftige TES-systemer i form av varmtvannsbeholdere under trykk kan brukes til kortsiktig lokal energilagring. For effektiv utnyttelse av slike systemer er størrelsen på tanken avgjørende. I tillegg vil den optimale størrelsen avhenge av tilgjengeligheten av spillvarme og tilhørende energibehov. I denne oppgaven defineres en forenklet prosess, og det formuleres et ikke-lineært dynamisk optimeringsproblem. Formuleringer inkludert komplementaritet er overveid.

Denne oppgaven fokuserer på kombinert optimal drift og optimalt design av en TES-tank i et fjernvarmenett i Nord-Norge, som tar i bruk spillvarme fra et nærliggende ferrosilisium produksjonsanlegg. Elektrisitet og CO-gass brukes normalt som energikilder når etterspørselen er for høy, og betraktes som to forskjellige case-studier for den økonomiske evalueringen i dette arbeidet. Basert på historisk data ble optimal drift og optimal dimensjonering av tanken oppnådd. Optimalt volum ble funnet til å være 6322,95 m^3 for begge tilfeller, som med optimal drift forårsaket en reduksjon i energiforbruket på 421 MW for de 30 dagene undersøkt. Tilbakebetalingsperioden for denne investeringen er forventet å være henholdsvis 13,74/12,18 år ved bruk av strøm/CO-gass. Dersom skatteøkningen varslet fra den norske regjeringen blir vedtatt i Stortinget, vil imidlertid tilbakebetalingsperioden reduseres til 5,7 år for CO-gass tilfellet. Sensitivitetsanalyser er gjort for begge tilfeller. En relativ endring i ulike faktorer som påvirker tilbakebetalingsperioden, slik som rentesats, utslippsavgifter og pris på energikilden, er inkludert. Analysene tyder på at mengden aktiv tid TES-tanken har gjennom et år er den mest avgjørende parameteren, tett etterfulgt av prisen på den respektive energikilden og renten. CO-gass tilfellet påvirkes også av endringer i CO_2-avgiften.

Preface

This Master's thesis was written during the spring of 2021, concluding the 5-year Master's Degree Programme "Chemical engineering and biotechnology", leading to an M.Sc. in Chemical Engineering.

I would like to thank my supervisor Johannes Jäschke for the guidance and the opportunity to work on this topic. A big thank you to my co-supervisor Caroline Satye Nakama for all the appreciated help and support, and for introducing me to Julia and complementarity constraints. In addition, I would like to thank Brage Rugstad Knudsen for insightful information and interesting discussions.

Finally, thank you to all of my friends here in Trondheim, for five incredible years and for all the good memories.

Declaration of Compliance

I, Agnes Camilla Tysland, hereby declare that this is an independent work according to the exam regulations of the Norwegian University of Science and Technology (NTNU).
Signature:

Place and Date: Trondheim - Gløshaugen, June 2021

Table of Contents

List of Tables

List of Figures

Nomenclature

Abbreviations

DAE	differential algebraic equations
DH	district heating
DHC	district heating and cooling
DHN	district heating network
DOF	degree of freedom
DV	disturbance variable
MINLP	mixed integer non linear programming
MPCC	mathematical program with complementarity constraints
MPEC	mathematical program with equilibrium constraints
NLP	nonlinear programming
ODE	ordinary differential equation
PHB	peak heat boiler
TES	thermal energy storage
WHB	waste heat boiler

Symbol	***Definition***	***Unit***
ρ^{dh}	density of district heating stream	$\frac{kg}{m^3}$
B	savings	NOK
C_p	specific heat capacity	$\frac{kJ}{K \cdot kg}$
C^{peak}	cost of peak heating	$\frac{NOK}{kWh}$
C^{dump}	cost of peak heating	$\frac{NOK}{kWh}$
I	investment cost	NOK
$\dot{m}$	mass flow	$\frac{kg}{s}$
N_0	payback period	$years$
T	temperature	°C
T^A	Temperature from node A	°C
T^B	Temperature from node B	°C
T^C	Temperature from node C	°C
$T^{dh,Ret}$	return temperature from district	°C
$T^{dh,Sup}$	supply temperature to district	°C
T^{tes}	TES tank temperature	°C
V	volume	m^3
V^{tes}	volume of TES tank	m^3
Q^{demand}	heat demand	W
Q^{phb}	peak heating	W
Q^{whb}	available waste heat	W
$Q^{whb,used}$	available waste heat used	W
$Q^{whb,dump}$	available waste heat discarded	W
q	volumetric flow	$\frac{m^3}{s}$
q^A	discharging flow	$\frac{m^3}{s}$
q^B	charging flow	$\frac{m^3}{s}$
q^{bp}	district mass flow	$\frac{m^3}{s}$
q^{dh}	district volumetric flow	$\frac{m^3}{s}$
q^{whb}	flow through WHB	$\frac{m^3}{s}$
r	interest rate	$\%$

CHAPTER 1

Introduction

Increasing energy demand in the world in combination with the desire to reduce global emissions of greenhouse gases has brought on many debates and groundbreaking ways of generating energy [2]. As of today, there are several ways of generating renewable energy, and improvement of these methods are still in focus [3]. However, an inconvenience with many of these renewable energy sources is that they are intermittent, and cannot be dispatched on demand [4]. Therefore, in addition to generate renewable energy and increase the share of renewable energy in global consumption, ways of utilizing energy already generated but not yet exploited are equally important. This will reduce energy waste, whilst feeding more energy into the power grid. District heating (DH) plays an increasing role in the operation of low-carbon energy systems. Studies have shown that district heating in urban areas has great potential for reducing primary energy supply and carbon dioxide emissions, in addition to being more affordable [5]. The flexibility of DH systems enables possibilities of integrating various energy sources, in addition to more efficient use of these sources. Consequently, this may maximize the benefits of co-generation and economy of scale [6].

An important contribution to utilize energy more efficiently is to find and exploit smart ways of conserving energy. Thermal energy storage (TES) is an efficient solution for local storage, using the thermal properties of a medium by for example charging and discharging a pressurized tank with excess heat from process industry or similar. When TES is to be used, designing the storage tank may be a challenging task. It is desirable to make the tank small to lessen capital investment and avoid occupying unnecessary space. However, the volume of the tank should be large enough to store amounts of heat that can provide significant savings. Therefore, when designing a TES tank, the volume should be chosen based on an optimal point that is just large enough to facilitate operation but also avoids overinvestment. In addition, for the utilization of DH systems to be of interest, a positive investment incentive is an important factor. If both smaller companies and larger-scale

industries are expected to green-light such financial investments, they must be profitable. The potential savings and benefits of introducing a TES tank should surpass the expected fixed investment and operational cost, within a reasonable amount of time.

1.1 Previous Work

This work is based on the recent studies by Knudsen et al. [7]. A different approach is explored in this work, but the waste heat supplier and district heating network considered is the same. Knudsen et al. proposed a combined dynamic simulation and model predictive control approach, accounting for dynamic and optimal control of the heating plant including the TES. An iterative, parametric evaluation of various TES volumes was carried out, while evaluating the effective peak heating reduction and energy-to-heat-flow-ratio.

Another study based on the same data was conducted by Thombre et al. [8]. They formed a scenario-tree formulation based on principal component analysis of the historical data. A methodology for a multistage nonlinear model predictive control algorithm was presented, optimizing energy storage and discharging decisions while handling the uncertainties of the daily operation.

Two more studies are based on the same data [9, 10].

Various TES technologies have been thoroughly reviewed and compared by Sarbu and Sebarchievici [1]. They state that water is the most popular and commercial heat storage medium when considering Sensible Heating Systems (SHS), and conclude that SHS is applicable to district heating and industrial needs. However, for TES overall, they emphasise that investment incentives and support for research and development are essential for deployment to be fostered.

More specific studies also underline the potential of TES, and the importance of further investigating such tachnologies. The effect of applying a hot water storage tank into a network supplied with geothermal energy to minimize the use of peak heating boilers was studied by Kyriakis and Younger [11]. They develop a model for sizing the system, in addition to a model studying the daily and annual operation of the system, concluding that based on the results, introduction of the tank is financially beneficial. Simeoni et al. [12] developed a multi-objective optimization model to perform a sustainable evaluation of a district heating network. The network utilized waste heat from a nearby industrial facility, considering the conflicting objectives related to various stakeholders, e.g. waste heat source, residential consumers, DHN provider and public authorities. Their model allows for analysis of the trade-off between various stakeholders' perspectives. Usage of TES in district heating and cooling systems (DHC) have been thoroughly reviewed by Guelpa and Verda [13]. They conclude that a common unique best solution for thermal storage to DH is not possible to identify, as DHC systems may depend on several case-specific configurations, such as network topology, possible connection to plants, control strategies etc. However, they underline fields to be more investigated in future research, such as possibilities in long-term TES storage and installation of alternating hot and cold storage between seasons. Wang et al. [14] estimated TES tank volumes by running simulation of different boiler capacities, in combination with various heat demand profiles. Considering a combination of discharge efficiency, boiler on- and off-time and maximum output temperature, linear correlations between boiler capacities and optimum TES tank volume

were obtained. The smallest sizing factor for the TES tank was obtained when average building heat demand was 45 % of boiler nominal capacity. Benefits of using TES tank were discovered to significantly decrease if the average building heat demand exceeded 60% of boiler nominal capacity.

1.2 Motivation & Objective

This thesis focuses on a district heating network in northern Norway, including the city of Mo i Rana and Mo Industrial Park. Waste heat from a ferrosilicon production plant in Mo Industrial Park is currently the main heating source in the district heating network (DHN), supplying heat to the city of Mo i Rana. The remaining heat is supplied by boilers driven by either fossil fuel, biofuel, CO-gas or electricity [9, 15]. In total, the amount of waste heat available surmounts the DH demand, but still, peak heating boilers (PHB) are often used. This is due to fluctuations in waste heat availability and variational DH demand. For example, an unexpected halt in the production of ferrosilicon or a sudden wave of cold weather affecting the city's demand will increase the discrepancy causing a need for peak heating. Contrarily, during periods of lower demand, excess waste heat will arise and be discarded.

The aim is to reduce the need for peak heating, by introducing thermal energy storage to the DHN. When the nearby process plant provides heat in excess, the heat may be stored in a TES tank and utilized at later occasions when the waste heat is not sufficient. Expected natural benefits of installing a TES tank are therefore energy savings, as optimal operation of the TES tank will lead to better utilization of the surplus heat currently being discarded, and reduced operational costs since less peak heating will be used. Furthermore, reducing the amount of peak heating now covered by fossil fuel, biofuel, CO gas and electricity will reduce the amount of greenhouse gas emissions.

Nonetheless, installing a TES tank needs to be a desirable investment, hence, the possible savings must be greater than the capital investment cost of the tank in the long run. Therefore, the objective of this study is to determine an optimal TES tank size for the district heating network supplying heat to Mo i Rana, taking into account both operational and economic aspects in a single model, using historical data. To determine the optimal tank size, a model yielding optimal operation should first be obtained and applied using historical data. Thereafter, the model must be adapted to consider economic aspects whilst obtaining optimal control. Various economic parameters, such as CO_2 emission taxes, interest rates and price of peak heating sources, may have a great impact on the designing decisions of the tank. The significance of these factors should therefore be analyzed.

1.3 Outline

In Chapter 2 relevant theory and explanation of concepts is presented. Thermal energy storage is briefly explained, followed by dynamic optimization including differential algebraic equations and the implicit Euler approach. Since the construction of the optimal operation model led to exploring the use of complementarity constraints, mathematical

programs with complementarity constraints (MPCCs) are discussed. A methodology of how to reformulate MPCCs as continuous nonlinear optimization problems is included.

The following chapter includes the process description and introduces the mathematical model used to represent the system. In addition, historical heat data provided by Mo Fjernvarme is presented and discussed. Chapter 4 addresses the optimal operation problem. The optimization problem and relevant parameters are stated, and the problem is optimized for different heat input profiles. The problem is then optimized using historical data.

After the optimal operation model is obtained, the next chapter deals with optimal sizing of the TES tank. Investment costs and operational costs are considered, and the problem is optimized with respect to payback time, while simultaneously maintaining optimal operation. A sensitivity analysis is carried out, considering various economic factors such as the price of the peak heating source, greenhouse gas emission taxes and interest rate. Finally, Chapter 6 concludes the thesis, summarizing the main findings and discussing possibilities for future work.

CHAPTER 2

Theoretical Background

This chapter provides theory and explanation of basic concepts relevant to this thesis. First, the concepts of thermal energy storage and sensible heat are briefly explained. Thereafter, dynamic optimization and an introduction to basic optimization problems are given. This includes dynamic system representation and discretization. This is important as the DHN is expressed as differential algebraic equations. A continuous nonlinear program express the model behavior and objective, which must be discretized in order to be handled by the optimizer. Lastly, mathematical programs with complementarity constraints and how they can be handled are presented. Logic constraint are present in the system, and are primarily approached as complementarity constraints. Such constraints exhibits non-smoothness and needs reformulation to be handled by a nonlinear solver.

2.1 Thermal Energy Storage

Thermal energy storage (TES) is a technology that utilizes the thermal properties of a medium in order to store energy. TES technologies may vary with respect to the phase and characterization of the medium used, and may be further divided into sensible heat storage (SHS), latent heat storage (LHS) and thermo-chemical heat storage[1]. In this thesis, SHS in a pressurized hot-water tank is considered.

2.1.1 Sensible Heat Storage

SHS is amongst the simplest and safest TES technologies. In addition, water is commonly used as heat storage medium, which is both cheap and, not to mention, thoroughly researched. Water requires minimum treatment and is usually readily available, making it a great substance to work with. Energy is stored by raising the temperature of a chosen

medium (charging), and may be used at a later occasion (discharging). The temperature increase in the storage medium is due to increasing internal energy, i.e. the energy is stored as sensible heat and no change in phase or composition occurs, as illustrated in Figure 2.1.1.

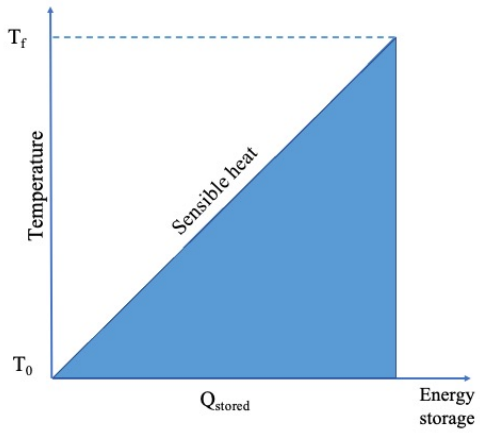

Figure 2.1.1: Illustration of sensible heat[1]

Specific heat, amount of storage material and temperature difference are thus important parameters when deciding the amount of stored heat[16]. The correlation is given by the heat equation,

$$Q_{stored} = \int_{T_0}^{T_f} mC_p dT, \tag{2.1.1}$$

where Q_{stored} represents the amount of heat stored, m the amount of storage medium, C_p is the specific heat capacity, and T_0 and T_f the initial and final temperature, respectively. Assuming specific heat capacity is independent of temperature, Equation 2.1.1 yields

$$Q_{stored} = mC_p(T_f - T_0) \tag{2.1.2}$$

2.2 Dynamic Optimization

Dynamic optimization is the optimization of dynamic systems, i.e. systems changing with time. For such systems to be optimized, they must first be formulated as an optimization problem, or a nonlinear program (NLP). Optimization problems consist of three main parts, namely an objective function, decision variables and constraints. In the objective function a scalar variable or property, for example constituting the system cost, is minimized or maximized. Decision variables can be integers, binary variables, real variables or belong to other spaces[17]. Constraints may consist of equality and inequality constraints, and represent the system. An optimization problem may thus be formulated:

$$\min_{z \in \mathbb{R}^n} \quad f(\mathbf{y}) \tag{2.2.1a}$$

$$s.t. \quad c^j(\mathbf{y}) = 0, \qquad j \in \mathcal{E} \tag{2.2.1b}$$

$$c^j(\mathbf{y}) \geq 0, \qquad j \in \mathcal{I} \tag{2.2.1c}$$

where $\mathcal{E}$ and $\mathcal{I}$ are equality and inequality constraint index sets, respectively, and $\mathbf{y}$ is the vector of decision variables. Equation 2.2.1a is the objective function, subject to constraints in Equations 2.2.1b and 2.2.1c. If either the objective function or some of the constraints are nonlinear, Equation 2.2.1 yields an NLP.

Dynamic systems may be modeled in different ways. Mathematically, they can be represented using differential equations. A dynamic system consisting of a fully implicit set of differential algebraic equations (DAEs), can be formulated as an initial value problem with initial conditions at zero, as follows

$$F\left(\mathbf{y}(t), \frac{d\mathbf{y}(t)}{dt}, \mathbf{p}, t\right) = 0, \tag{2.2.2a}$$

$$h(\mathbf{y}(0)) = 0, \tag{2.2.2b}$$

where $t \geq 0$, $\mathbf{y}(t) \in \mathbb{R}^{n_\mathbf{y}}$ denotes the variables as functions of time, $\frac{d\mathbf{y}(t)}{dt}$ is the time derivative of these variables and $\mathbf{p} \in \mathbb{R}^{n_\mathbf{p}}$ are remaining, time-independent parameters.

A simpler way of formulating the implicit DAE in Equation 2.2.2, is obtained by splitting the variables into two groups; differential variables, $\mathbf{x}(t)$, and algebraic variables, $\mathbf{z}(t)$. The DAE system may then be given on the semi-explicit form,

$$\frac{d\mathbf{x}}{dt} = f(\mathbf{x}(t), \mathbf{z}(t), \mathbf{p}), \tag{2.2.3a}$$

$$\mathbf{x}(t_0) = \mathbf{x}_0, \tag{2.2.3b}$$

$$g(\mathbf{x}(t), \mathbf{z}(t), \mathbf{p}) = 0. \tag{2.2.3c}$$

By taking a dynamic model, e.g. a DAE system, as constraints, the dynamic optimiza-

tion problem may thus be formulated as

$$\min \quad \int_{t_0}^{t_f} \phi(\mathbf{x}(t), \mathbf{z}(t), \mathbf{p}) \tag{2.2.4a}$$

$$\text{s.t.} \quad \frac{d\mathbf{x}}{dt} = f(\mathbf{x}(t), \mathbf{z}(t), \mathbf{p}), \qquad \mathbf{x}(t_0) = \mathbf{x}_0, \tag{2.2.4b}$$

$$g(\mathbf{x}(t), \mathbf{z}(t), \mathbf{p}) = 0, \tag{2.2.4c}$$

$$\mathbf{x}_L \leq \mathbf{x}(t) \leq \mathbf{x}_U, \tag{2.2.4d}$$

$$\mathbf{z}_L \leq \mathbf{z}(t) \leq \mathbf{z}_U \tag{2.2.4e}$$

$$\mathbf{p}_L \leq \mathbf{p} \leq \mathbf{p}_U \tag{2.2.4f}$$

$$\tag{2.2.4g}$$

Here, the objective function is minimizing a certain cost, subject to the dynamic model equations and boundaries. The differential equations with initial conditions at time equal to zero is given in Equation 2.2.4b and the algebraic equations in Equation 2.2.4c. Equations 2.2.4d-2.2.4f gives the system bounds, i.e. process specifications or constraints for feasible operation.

2.2.1 Index of DAEs

Dynamic models may be divided into two groups based on their index. DAEs of *low index* has an index of 0 or 1, while *high index* is considered to be 2 or higher. Many commercial solvers can only handle low index DAEs. The higher the index of the DAE, the more difficulties are expected for the solver converge. Hence, it is desired to develop models consisting of lower index DAEs when possible. The index of the problem is determined by the number of differentiation steps it is required to obtain an implicit ODE from the DAE system [18]. Thus, an ODE have index 0. A semi-explicit DAE as the one considered above has index 1 if g is solvable for $\mathbf{z}$. That is, if by differentiation of the algebraic equations

$$0 = \partial_{\mathbf{x}} g(\mathbf{x}(t), \mathbf{z}(t), \mathbf{p})\mathbf{x}' + \partial_{\mathbf{z}} g(\mathbf{x}(t), \mathbf{z}(t), \mathbf{p})\mathbf{z}', \tag{2.2.5}$$

$\partial_{\mathbf{z}} g$ is nonsingular,

$$\det(\partial_{\mathbf{z}} g(\mathbf{x}(t), \mathbf{z}(t), \mathbf{p})) \neq 0, \tag{2.2.6}$$

the system is an implicit ODE and has index 1. This is also known as the index-1 property.

2.2.2 Solving Dynamic Optimization Problems

A dynamic optimization problem may be translated into an NLP using a direct method. The given time period is discretized into multiple smaller finite elements, and, by using parameterization, the state trajectory is obtained. The resulting discretized system may be solved using a sequential or simultaneous approach. In general, simultaneous approaches are better suited when the NLP becomes large[19]. Using a simultaneous approach, the dynamic state variables may be discretized in time using Radau collocation with the Runge-Kutta polynomial representation. This method is equal to an implicit Euler approach with

first order accuracy when only one collocation point is chosen[20], also known as one stage Runge-Kutta method.

Consider the semi-explicit DAE

$$\frac{d\mathbf{x}}{dt} = f(\mathbf{x}(t), \mathbf{z}(t), \mathbf{p}), \qquad \mathbf{x}(t_0) = \mathbf{x}_0 \tag{2.2.7}$$

for a known function f, with initial value data t_0 and $\mathbf{x}_0$. As previously defined, $\mathbf{x}$ contains the time-dependent state variables, $\mathbf{z}$ the algebraic variables and $\mathbf{p}$ the time-independent parameters. The implicit Euler approximation is calculated using the expression[21]

$$\mathbf{x}(t_i) = \mathbf{x}(t_{i-1}) + hf(\mathbf{x}(t_i), \mathbf{z}(t_i), \mathbf{p}), \tag{2.2.8}$$

where h is the size of the finite element, i, and $i = 1, ..., N_T$, for N_T number of finite elements.

Applying the implicit Euler approach with a fixed time step h, the discretized dynamic optimization problem from Equation 2.2.4 becomes

$$\begin{aligned}
\min \quad & \sum_{i=1}^{N_T} \phi(\mathbf{x}_i, \mathbf{z}_i, \mathbf{p}) && && \text{(2.2.9a)}\\
\text{s.t.} \quad & \mathbf{x}_i = \mathbf{x}_{i-1} + hf(\mathbf{x}_i, \mathbf{z}_i, \mathbf{p}), && \mathbf{x}(t_0) = \mathbf{x}_0, && \text{(2.2.9b)}\\
& g(\mathbf{x}_i, \mathbf{z}_i, \mathbf{p}) = 0, && && \text{(2.2.9c)}\\
& \mathbf{x}_{L,i} \leq \mathbf{x}_i \leq \mathbf{x}_{U,i}, && && \text{(2.2.9d)}\\
& \mathbf{z}_{L,i} \leq \mathbf{z}_i \leq \mathbf{z}_{U,i}, && && \text{(2.2.9e)}\\
& \mathbf{p}_L \leq \mathbf{p} \leq \mathbf{p}_U, && && \text{(2.2.9f)}\\
& && && \text{(2.2.9g)}
\end{aligned}$$

for N_T number of finite elements and $i = 1, ..., N_T$.

2.3 Mathematical Programs with Complementarity Constraints (MPCC)

The relationship between variables where one or both of the variables must be at a boundary is called a complementarity. A complementarity problem is an optimization problem in which at least one constraint is a complementarity, e.g. if x complements y, then at least one of them must be equal to zero in the solution of the problem [22]. A mathematical program with complementarity constraints (MPCC) may be formulated as [23]

$$\begin{aligned}
\min \quad & f(\mathbf{x}, \mathbf{y}, \mathbf{z}) && \text{(2.3.1a)}\\
\text{s.t.} \quad & h(\mathbf{x}, \mathbf{y}, \mathbf{z}) = 0, && \text{(2.3.1b)}\\
& g(\mathbf{x}, \mathbf{y}, \mathbf{z}) \geq 0, && \text{(2.3.1c)}\\
& 0 \leq \mathbf{x} \perp \mathbf{y} \geq 0, && \text{(2.3.1d)}
\end{aligned}$$

where Equation 2.3.1d is the complementarity constraint with the complementarity operator $\perp$, implying

$$x_i = 0 \text{ or } y_i = 0 \qquad \text{for i = 1,...,N}_T$$
$$\mathbf{x}, \mathbf{y} \geq 0$$

This complementarity is a logical condition, which makes the problem nonsmooth. If a nonlinear solver is to be used, the MPCC needs to be reformulated. A common formulation is [24]

$$\begin{aligned} \min \quad & f(\mathbf{x}, \mathbf{y}, \mathbf{z}) && \text{(2.3.2a)} \\ \text{s.t.} \quad & h(\mathbf{x}, \mathbf{y}, \mathbf{z}) = 0, && \text{(2.3.2b)} \\ & g(\mathbf{x}, \mathbf{y}, \mathbf{z}) \geq 0, && \text{(2.3.2c)} \\ & \mathbf{x}, \mathbf{y} \geq 0, && \text{(2.3.2d)} \\ & \mathbf{x}^T \mathbf{y} \leq 0. && \text{(2.3.2e)} \end{aligned}$$

This way, the solution set of the MPCC remains the same, but the problem is ill-conditioned as it may lead to inconsistency due to violation of constraint qualifications. Interior methods, e.g. IPOPT, may prove to be inefficient when solving such problems, as they are fed conflicting goals of enforcing the complementarity in Equation 2.3.2e, bringing the product of the variables to zero, while striving to keep the variables away from their bounds.

In order to deal with these inefficiencies, relaxation approaches, such as

$$x_i \cdot y_i \leq \theta, \qquad i = 1, ..., N_T \tag{2.3.3}$$

may be applied [24]. Here, θ is a relaxation parameter which is driven to zero.

Another approach that can enhance the efficiency of the nonlinear solver, is to reformulate the MPCC such that the inequality constraint in Equation 2.3.2e is given a penalty parameter and is moved to the objective function:

$$f(\mathbf{x}, \mathbf{y}, \mathbf{z}) + \pi \mathbf{x}^T \mathbf{y}. \tag{2.3.4}$$

This is also known as an l_1-penalty term, where the value of the penalty parameter π must be above zero. The value of this parameter may not be easy to tune. However, algorithms have been proposed that estimate its value during the course of the optimization calculation [24].

CHAPTER 3

Model Development

This chapter includes a description of the thermal energy storage system and a presentation of the mathematical model and the nonlinear programming (NLP) problem. The main challenge is to obtain a desired behavior for the charging and discharging of the TES tank, considering the main objective of minimizing the peak heating and wasting as little energy as possible. The model developed is a semi-explicit DAE consisting of a differential equation describing the temperature in the TES tank, and algebraic equations for the remaining heat equations and mass balances. Notations presented in this chapter will be used consistently throughout the thesis.

3.1 Process Description

For the purpose of investigating the potential benefits of introducing a TES tank into the district heating network in northern Norway, a simplified process is defined and modelled, presented in Figure 3.1.1. The district itself is not included in the simplified flow diagram, as the district demand is given by historical data and included in the model through the flow rate to the district, return temperature and supply temperature.

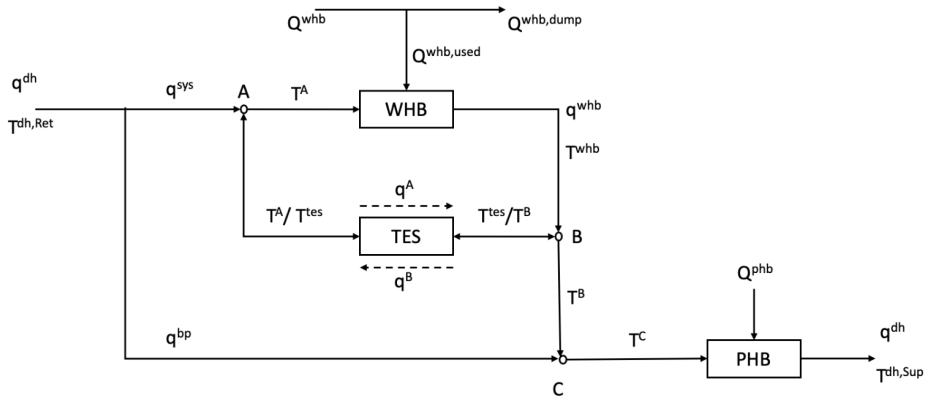

Figure 3.1.1: Flow diagram of thermal energy storage problem.

The volumetric flow rate to and from the district is denoted q_{dh}. It is assumed that there is no mass accumulation in the system, hence, flows in and out of the system are instantaneously the same. The district return and supply temperatures are represented by $T_{dh,Ret}$ and $T_{dh,Sup}$, respectively, the latter being equal to the temperature exiting the peak heating boiler (PHB), T_{phb}. Part of the inlet flow will pass on to node A, as q_{sys}. The flow q_{sys} has temperature equal to the return temperature, and will either go through the waste heat boiler (WHB) or into the TES tank if the tank is to be discharged, $q_A > 0$. On the other hand, if the tank is charged, $q_B > 0$, q_B at temperature T_{tes} and q_{sys} at temperature $T_{dh,Ret}$ will go from node A to the WHB, with inlet temperature T_A. As the tank is considered to be bidirectional, q_A and q_B may not be non-zero simultaneously. The remaining inlet flow may bypass to node C, as q_{bp}, if the temperature from node B is higher than the desired supply temperature. Q_{whb} represents the excess heat available from the ferrosilicon plant, and $Q_{whb,used}$ denotes the amount of the available heat exploited by the TES system. The remaining excess heat is dumped, illustrated by Q_{dump}. The volumetric flow rate through the WHB is denoted by q_{whb}, and the boiler outlet temperature is T_{whb}. The TES tank is assumed to be perfectly-mixed, i.e. with homogeneous temperature, T_{tes}, throughout the tank. In addition, no mass accumulation is assumed for the tank, and the system over-all. In node B, the flow from the WHB is respectively mixed or split with the discharge or charging flow from the TES. The resulting flow is at temperature T_B and has volumetric flow rate q_{sys}. This flow merges with the bypass flow at node C, resulting in a flow at temperature T_C going to the PHB. If the temperature T_C is not at the desired supply temperature, additional heat must be added, Q_{phb}.

3.2 Mathematical Model

In order to solve the optimal operation problem, a dynamic model of the process must be established. For the system in general, all fluid streams are assumed to have constant properties, i.e. density and specific heat capacity do not change with temperature. Heat losses to the surroundings are disregarded and no mass accumulation is assumed, hence

$\frac{dV}{dt} = 0$.

The total mass balance, for the system presented in Figure 3.1.1 is given by

$$q^{dh} = q^{sys} + q^{bp}. \tag{3.2.1}$$

The interior mass balance, showing the charging and discharging, is

$$q^{sys} = q^{whb} + q^{A} - q^{B}. \tag{3.2.2}$$

As no mass accumulation is assumed, charging and discharging cannot occur simultaneously. This logic constraint may be enforced through the complementarity

$$0 \leq q^{A} \perp q^{B} \geq 0. \tag{3.2.3}$$

The energy balance over point A may be expressed as

$$\rho^{dh} q^{sys} C_p^{dh} T^{dh,Ret} + \rho^{dh} q^{B} C_p^{dh} T^{tes} = \rho^{dh} q^{whb} C_p^{dh} T^{A} + \rho^{dh} q^{A} C_p^{dh} T^{A}, \tag{3.2.4}$$

where ρ^{dh} is the density of the flow through the DHN, and C_p^{dh} is the specific heat capacity. Equivalently, temperature from point A is given by

$$T^{A} = \frac{q^{sys} T^{dh,Ret} + q^{B} T^{tes}}{q^{whb} + q^{A}} \tag{3.2.5}$$

Likewise, the energy balance over point B is

$$\rho^{dh} q^{whb} C_p^{dh} T^{whb} + \rho^{dh} q^{A} C_p^{dh} T^{tes} = \rho^{dh} q^{sys} C_p^{dh} T^{B} + \rho^{dh} q^{B} C_p^{dh} T^{B}, \tag{3.2.6}$$

yielding temperature at point B,

$$T^{B} = \frac{q^{whb} T^{whb} + q^{A} T^{tes}}{q^{sys} + q^{B}}. \tag{3.2.7}$$

Finally, the energy balance over point C is then

$$\rho^{dh} q^{bp} C_p^{dh} T^{dh,Ret} + \rho^{dh} q^{sys} C_p^{dh} T^{B} = \rho^{dh} q^{dh} C_p^{dh} T^{C}, \tag{3.2.8}$$

giving temperature T^{C} at the entrance of the PHB

$$T^{C} = \frac{q^{bp} T^{dh,Ret} + q^{sys} T^{B}}{q^{dh}}. \tag{3.2.9}$$

Heat transferred through the heat exchangers may be obtained using the heat equation,

$$Q = \dot{m} C_p \Delta T. \tag{3.2.10}$$

Considering that $\dot{m} = \rho q$, the equation for the waste heat boiler becomes

$$Q^{whb} = \rho^{dh} q^{whb} C_p^{dh} (T^{whb} - T^{A}) \tag{3.2.11}$$

and, equivalently for the peak heating exchanger

$$Q^{phb} = \rho^{dh} q^{dh} C_p^{dh} (T^{phb} - T^C). \tag{3.2.12}$$

Additionally, the available waste heat from the nearby process industry minus the heat utilized in the TES system, yields the amount of heat dumped. This may be mathematically expressed as the heat balance

$$Q^{\textit{available waste heat}} = Q^{whb} = Q^{whb,used} + Q^{whb,dump}. \tag{3.2.13}$$

Equation 3.2.10 may also be used to calculate the total heat demand for the TES system, Q^{demand}, as follows

$$Q^{demand} = \rho^{dh} q^{dh} C_p^{dh} (T^{phb} - T^{dh,Ret}). \tag{3.2.14}$$

Assuming the TES tank is well-mixed, i.e. the spatial temperature gradient throughout the tank is zero, the dynamic mass balance may be expressed as

$$\frac{d}{dt}(\rho^{dh} V^{tes}) = \rho^{dh}(q^A - q^B). \tag{3.2.15}$$

No mass accumulation and constant density are assumed, yielding the energy balance

$$\frac{d}{dt}(\rho^{dh} V^{tes} C_p^{dh} T^{tes}) = \rho^{dh} C_p^{dh} (q^A (T^{tes} - T^A) - q^B (T^{tes} - T^B)) \tag{3.2.16}$$

Rearranging this expression results in the ODE

$$\frac{dT^{tes}}{dt} = \frac{q^A}{V^{tes}}(T^A - T^{tes}) + \frac{q^B}{V^{tes}}(T^B - T^{tes}). \tag{3.2.17}$$

3.3 Historical Data

One year of historical data from Mo Fjernvarme is used. Heat data from various boilers, in addition to return and supply temperatures were used for obtaining optimal operation and designing the TES tank. Figure 3.3.1 shows three subplots; available waste heat and heat demand at the top, mass flow through the district heating network in the middle and temperatures of the flow returned and supplied to the district at the bottom. The data is collected hourly, throughout the year.

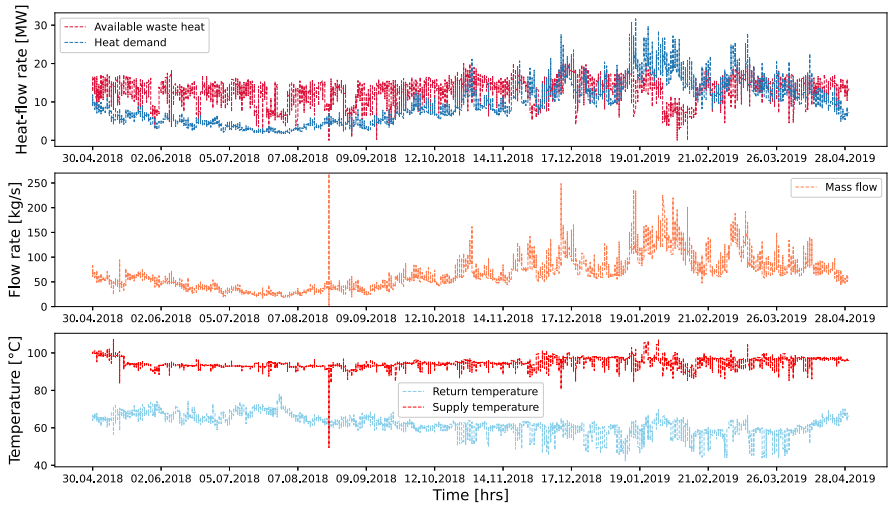

Figure 3.3.1: Data from Mo Fjernvarme for the period April 30^{th} 2018 to April 29^{th} 2019. Available waste heat and heat demand is shown in the top plot, mass flow through the DH system in the middle plot and return and supply temperature from and to the district in the bottom plot.

An outlier is clearly present on August 22nd 2018, probably due to measurement error. The mass flow rate is calculated using the heat demand, return temperature and supply temperature, using the equation

$$q^{dh} = \frac{Q^{demand}}{\rho^{dh} C_p^{dh} (T^{dh,Sup} - T^{dh,Ret})}. \tag{3.3.1}$$

The bottom plot in Figure 3.3.1 shows that the supply temperature is below the return temperature at this specific hour, causing an irregularly high mass flow rate. The measurements for this hour at this date is therefore neglected.

For the purpose of sizing a TES tank to minimize the need for peak heating, time periods presenting alternating highs of demanded heat and waste heat are of interest. A representative period with this behaviour is March 2019, which will be used for further investigation of optimal operation and optimal sizing. Figure 3.3.2 displays the relevant information for March 2019.

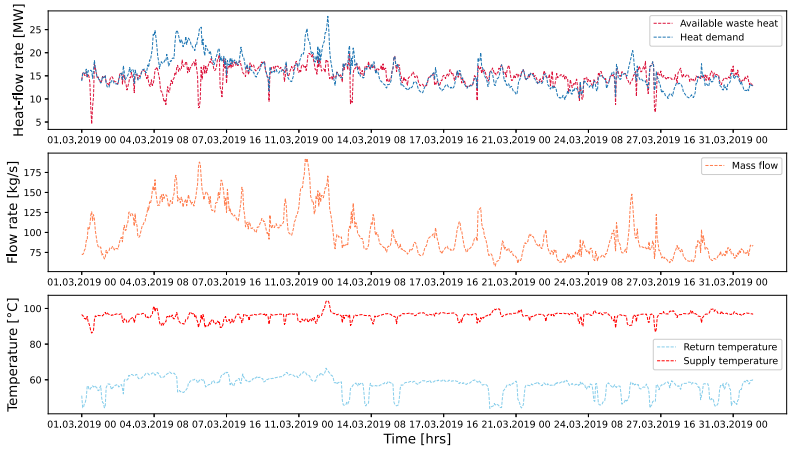

Figure 3.3.2: Data from Mo Fjernvarme for March 2019. Available waste heat and heat demand is shown in the top plot, mass flow through the DH system in the middle plot and return and supply temperature from and to the district in the bottom plot.

CHAPTER 4

Optimal Operation

Considering the district heating system with a TES tank in Mo i Rana, a dynamic optimization model must first be defined, in order to obtain optimal operation. As mentioned in Section 2.2, an optimization problem consists of an objective function, decision variables and constraints. The process and model equations representing the system, were defined in the previous chapter. These will constitute the constraints of the optimization problem, together with the decision variable bounds. Moreover, the logic constraint presented previously will be included as complementarity constraints in the model, converting the problem to a mathematical program with complementarity constraints (MPCC). The optimization model objective is to minimize the amount of peak heating needed in the system and maximize utilization of waste heat from the ferrosilicon production process.

The dynamic model will first be simulated using test data, to ensure that the model behaves as desired. After this, historical data will be applied over an increasing time horizon. First, periods of 24 hours will be simulated, followed by seven days and, finally, a total period of one month. The obtained optimal operation for this month, will form the basis for the optimal design problem and economic evaluations in the next chapter.

As mentioned in the previous chapter, the notation used in this chapter is as stated before.

4.1 Mathematical Formulation

The mathematical model of the system presented in Section 3.2 may be formulated as a set of DAEs:

$$\frac{d\mathbf{x}(t)}{dt} = f(\mathbf{x}(t), \mathbf{z}(t), \mathbf{p}) \tag{4.1.1a}$$

$$0 = g(\mathbf{x}(t), \mathbf{z}(t), \mathbf{p}) \tag{4.1.1b}$$

where **x** is a vector containing the differential variables, here the temperature of the TES tank

$$\mathbf{x} = [T^{tes}], \tag{4.1.2}$$

vector **z** contains the algebraic variables

$$\begin{aligned}\mathbf{z} = [&T^{A}, T^{B}, T^{C}, T^{whb}, Q^{whb,used}, Q^{whb,dump}, Q^{phb}, q^{whb},\\ &q^{A}, q^{B}, q^{bp}, q^{sys}, q^{dh}, T^{dh,ret}, T^{dh,sup}, Q^{whb}, Q^{demand}],\end{aligned} \tag{4.1.3}$$

and **p** is a vector of the known, time-independent variables

$$\mathbf{p} = [\rho^{dh}, C_p^{dh}, V^{tes}]. \tag{4.1.4}$$

The optimal operations problem may thus be expressed as the following dynamic optimization problem

$$\min_{x,u,z} \quad \int (Q^{phb}(t))^2 + (Q^{dump}(t))^2 dt \tag{4.1.5a}$$

$$\text{s.t.} \quad \frac{dT^{tes}(t)}{dt} = \frac{q^A(t)}{V^{tes}}(T^A(t) - T^{tes}(t)) + \frac{q^B(t)}{V^{tes}}(T^B(t) - T^{tes}(t)) \tag{4.1.5b}$$

$$q^{dh}(t) = q^{bp}(t) + q^{sys}(t) \tag{4.1.5c}$$

$$q^{sys}(t) = q^{whb}(t) + q^{A}(t) - q^{B}(t) \tag{4.1.5d}$$

$$T^{A}(t) = \frac{T^{dh,Ret}(t) \cdot q^{sys}(t) + T^{tes}(t) \cdot q^{B}(t)}{q^{whb}(t) + q^{A}(t)} \tag{4.1.5e}$$

$$T^{B}(t) = \frac{T^{whb}(t) \cdot q^{whb}(t) + T^{tes}(t) \cdot q^{A}(t)}{q^{sys}(t) + q^{B}(t)} \tag{4.1.5f}$$

$$T^{C}(t) = \frac{T^{B}(t) \cdot q^{sys}(t) + T^{dh,Ret}(t) \cdot q^{bp}(t)}{q^{dh}(t)} \tag{4.1.5g}$$

$$Q^{whb}(t) = Q^{whb,used}(t) + Q^{whb,dump}(t) \tag{4.1.5h}$$

$$Q^{whb,used}(t) = \rho^{dh} C_p^{dh} q^{whb}(t)(T^{whb}(t) - T^{A}(t)) \tag{4.1.5i}$$

$$Q^{phb}(t) = \rho^{dh} C_p^{dh} q^{dh}(t)(T^{phb}(t) - T^{C}(t)) \tag{4.1.5j}$$

$$Q^{demand}(t) = \rho^{dh} C_p^{dh} q^{dh}(t)(T^{dh,Sup}(t) - T^{dh,Ret}(t)) \tag{4.1.5k}$$

$$0 \leq q^{A}(t) \perp q^{B}(t) \geq 0 \tag{4.1.5l}$$

$$\mathbf{x}_L \leq \mathbf{x}(t) \leq \mathbf{x}_U \tag{4.1.5m}$$

$$\mathbf{z}_L \leq \mathbf{z}(t) \leq \mathbf{z}_U \tag{4.1.5n}$$

As the intention is to minimize the amount of peak heating needed in the DHN and utilize as much of the waste heat available as possible, the objective function in Equation 4.1.5a is set to minimize the variable for peak heating and waste heat dumped. The temperature change in the TES tank is accounted for in the model through the differential equation in Equation 4.1.5b. Mass and energy balances are present through equations 4.1.5c-4.1.5k, and the logic constraint to avoid simultaneous charging and discharging of the TES tank

is enforced through the complementarity constraint in Equation 4.1.5l. Equations 4.1.5m-4.1.5n enforces physical bounds to the differential and algebraic variables. The numerical values of these bounds are given in Table 4.1. Table 4.2 presents the values of all the time-independent parameters used.

Table 4.1: Variable bounds for optimal operations problem.

Description	**Lower bound**	**Variable**	**Upper bound**	Unit
Temperature in TES tank	40.0	T^{tes}	120.0	[°C]
Temperature after node A	40.0	T^{A}	120.0	[°C]
Temperature after node B	40.0	T^{B}	120.0	[°C]
Temperature after node C	40.0	T^{C}	120.0	[°C]
Temperature after WHB	40.0	T^{whb}	120.0	[°C]
Peak heating applied	0.0	Q^{phb}	-	[MW]
Waste heat utilized	0.0	$Q^{whb,used}$	22	[MW]
Waste heat dumped	0.0	$Q^{whb,dumped}$	Q^{whb}	[MW]
Flow through WHB	0.0	q^{whb}	333.33	$[\frac{kg}{s}]$
TES charging flow	0.0	q^{A}	1388.89	$[\frac{kg}{s}]$
TES discharging flow	0.0	q^{B}	1388.89	$[\frac{kg}{s}]$
Bypass flow	0.0	q^{bp}	q^{dh}	$[\frac{kg}{s}]$
System flow (not bypassing)	0.0	q^{sys}	q^{dh}	$[\frac{kg}{s}]$

Table 4.2: Description and value of known time-independent parameters.

Variable	**Description**	**Value**	**Unit**
ρ^{dh}	Density	1000.00	$[\frac{kg}{m^3}]$
C_p	Specific heat	4.18	$[\frac{kJ}{kg \cdot K}]$
V^{tes}	Volume TES tank	5000.00	m^3

In order to be solved as an NLP problem, the continuous formulation in Equation 4.1.5 is discretized using Eulers method, as described in Section 2.2.2. The ODE in Equation 4.1.5b is approximated using

$$\mathbf{x}_i = \mathbf{x}_{i-1} + hf(\mathbf{x}_i, \mathbf{z}_i, \mathbf{p}) \tag{4.1.6}$$

where h is the step length of one finite element, in this study corresponding to one hour, and $i = 1, ..., N_T$ for N_T number of finite elements.

The resulting discretized operations problem is formulated as follows

$$\min_{x,z} \quad \sum_{i=1}^{N_T} 10^{-6}(Q_i^{phb})^2 + 10^{-6}(Q_i^{whb,dump})^2 \tag{4.1.7a}$$

$$\text{s.t.} \quad \mathbf{x}_i = \mathbf{x}_{i-1} + hf(\mathbf{x}_i, \mathbf{z}_i, \mathbf{p}) \qquad i = 1, ..., N_T \tag{4.1.7b}$$

$$\mathbf{x}_0 = \mathbf{x}(0) \tag{4.1.7c}$$

$$\mathbf{g}_i(\mathbf{x}_i, \mathbf{z}_i, \mathbf{p}) = \mathbf{0} \qquad i = 1, ..., N_T \tag{4.1.7d}$$

$$\mathbf{x}_{L,i} \leq \mathbf{x}_i \leq \mathbf{x}_{U,i} \qquad i = 1, ..., N_T \tag{4.1.7e}$$

$$\mathbf{z}_{L,i} \leq \mathbf{z}_i \leq \mathbf{z}_{U,i} \qquad i = 1, ..., N_T \tag{4.1.7f}$$

$$0 \leq q_i^A \perp q_i^B \geq 0 \qquad i = 1, ..., N_T \tag{4.1.7g}$$

The MPCC in Equation 4.1.7 is reformulated as an NLP using the algorithm proposed in [24] and presented in Section 2.3. The l_1-penalty term formulation is used, adding a penalty term to the objective function and gradually increasing the associated weight parameter. The NLP is implemented in **julia**[25], using JuMP[26] as mathematical programming language and optimization solver IPOPT[27] with linear solver MA97. The script $tes\_logic.jl$ contains the implementation, presented in Appendix D.7.

4.2 Implementation Using Test Data

To ensure that the model behaves as desired, the system is first simulated using the test data presented in Figure 4.2.1 as input.

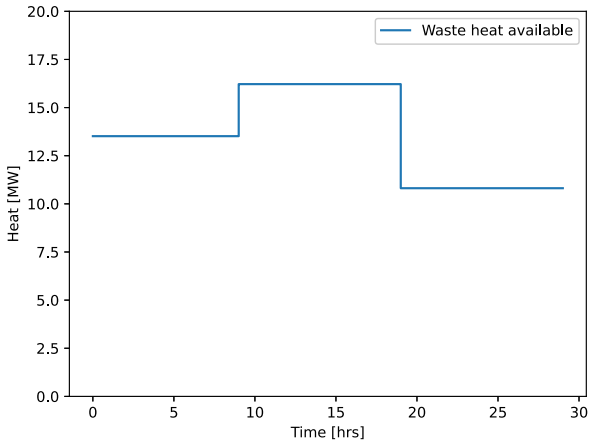

Figure 4.2.1: Waste heat profile given as input.

The given waste heat profile together with fixed return and supply temperatures at $T^{dh,ret}$=55°C and $T^{dh,sup}$=95°C, respectively, and district flow rate of $q^{dh} = 80.83$ kg/s are given to the optimizer. The result is presented in Figure 4.2.2. The top left plot shows the temperature from the TES tank in blue, temperature from the WHB in green and temperature from the PHB in orange. The mass flow from the PHB is the mass flow supplied to the district, hence $T^{phb} = T^{dh,Sup}$. The plot to the top right gives the waste heat available in blue and demand in orange. Next, the plot in the middle left position presents the flow through the WHB in blue, discharging flow in orange and charging flow in green. The middle right plot shows peak heating in blue, dumped waste heat in orange and waste heat utilized in green. At the bottom left, the plot gives the bypass stream in blue and the total district heating network stream in orange. Lastly, the flow temperatures from each of the nodes, A, B and C are given in blue, orange and green, respectively.

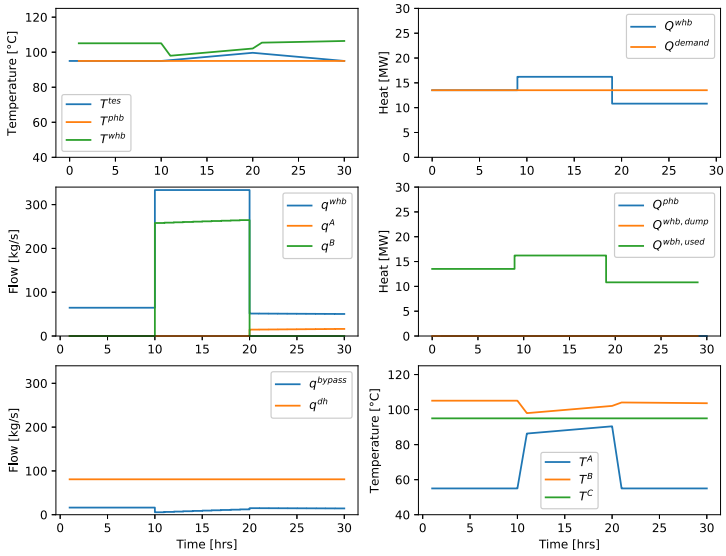

Figure 4.2.2: Optimal operation model simulated using test data with initial tank temperature at 95 °C.

The waste heat available in the first 10 hours of the test data is the exact amount of heat the system needs to heat the district flow from 55 °C to 95 °C. Part of the district flow is sent through the WHB, exploiting all of the heat available, mixing the flow with the bypass, yielding the desired supply temperature of 95 °C. No peak heating is needed and no waste heat is dumped the first 10 hours, as seen in the middle right plot. Intuitively, it was expected that the entire district flow would pass through the waste heat boiler, resulting in a temperature out of the WHB at 95 °C and no charging, discharging or bypass would be needed. However, these results are mathematically equivalent, and the situation of

Q^{demand} exactly matching Q^{whb} is highly unlikely when considering actual values. After 10 hours the waste heat exceeds the required amount, as seen in the top, right plot in Figure 4.2.2. The surplus heat is therefore stored in the tank. This is seen from the middle plot to the left as q_B is positive, and from the top left plot, where there is an increase in the temperature of the tank, T_{tes}. After 20 hours, the waste heat alone is not sufficient, and the tank is discharged. The amount of heat stored from 10-20 hours is the exact amount of heat needed for 20-30 hours. The heat is drained from the tank, and no further peak heating is needed. The system behaves as desired for the given input.

Next, the influence of the initial tank temperature is to be examined. The system will be simulated using the same waste heat profile as previously considered, but with initial tank temperatures of 60 and 115 °C. The results are presented in Figure 4.2.3 and Figure 4.2.4.

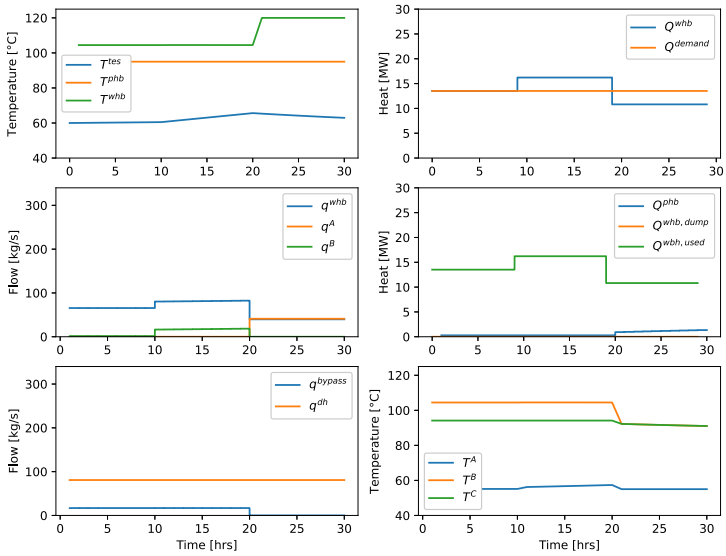

Figure 4.2.3: Optimal operation model simulated using test data with initial tank temperature at 60 °C.

For this scenario when the tank temperature is initially lower than the supply temperature, it is seen from the middle right plot in Figure 4.2.3 that some peak heating is needed. This is as expected as the test data provided equals the exact amount of the heat demanded for the 30 hours considered. As the previous scenario, some heat is stored from 10-20 hours, but as the volume of the tank is large, and the temperature is relatively cold, the system is not able to retrieve all the heat stored, and some peak heating is needed for the last 10 hours.

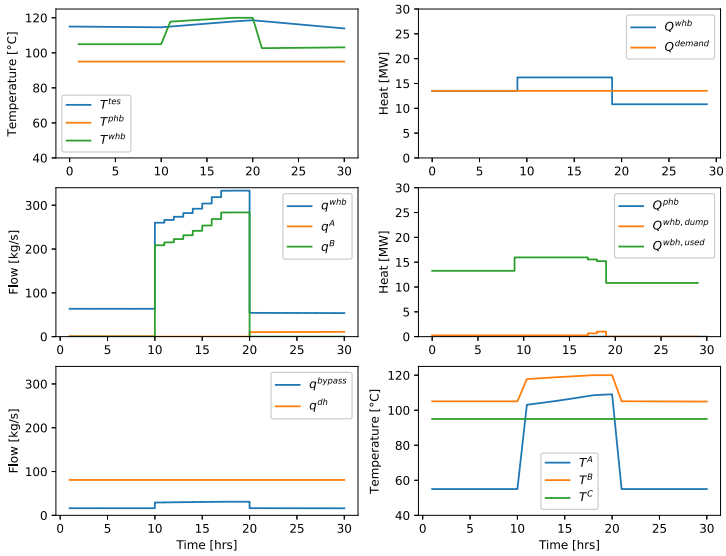

Figure 4.2.4: Optimal operation model simulated using test data with initial tank temperature at 115 °C.

Contrarily, for the scenario shown in Figure 4.2.4 where the initial tank temperature is high, it is seen from the top left plot that the tank is charged and reaches max capacity when the temperature of the tank is at 120 °C. Consequently, some heat is dumped, visible in the middle right plot. However, even though the amount of waste heat available is exactly equal to the total amount demanded and some waste heat is dumped, no peak heating is needed for the last 10 hours. This is because the tank initially was stored with surplus heat available to the district.

It is thus concluded that the starting temperature of the tank may affect the result of the testing quite significantly, as the time period for now is 30 hours. The energy initially stored in the tank may constitute a significant share of the overall heat transferred in the DHN. However, the impact of this uncertainty is expected to lessen as the time horizon expands, especially after a long period of excess or shortage of available energy, as this will bring the TES to its upper or lower bounds, respectively. It is therefore important to obtain optimal operation for a significant amount of hours, in order for this system to yield a representative time horizon to be used for the optimal sizing in the next chapter.

4.3 Applying Historical Data

In this section, the impact of using a TES tank in an existing district heating network is investigated. The optimal operations problem in Equation 4.1.7 is to be solved, using historical data provided by Mo Fjernvarme presented in Section 3.3.

4.3.1 24 Hour-Scenarios

To ensure that the model is sturdy and able to handle various initial tank temperatures and availability of waste heat, three different 24 hours scenarios are investigated. One period of alternating waste heat surplus and shortage, one period with majority of waste heat surplus and one with majority of waste heat shortage. The starting temperature of the TES tank is varied between $T = 60$°C, $T = 95$°C and $T = 115$°C.

Optimal operation result after applying historical data from February 1^{st} 2019, a day of waste heat shortage, is presented in Figure 4.3.1. The initial tank temperature is here 95 °C, as seen from the upper left plot. The waste heat trajectory in the upper right plot shows a dip after approximately 11 hours, and peak heating is needed, as seen in the middle right plot. At this time, the bypass is zero, and the entire district flow is either going through the WHB or to the tank which is being discharged. As the tank initially starts at a temperature above the return temperature of 55 °C, it is being discharged throughout the majority of the period, and little peak heating is needed, despite the shortage.

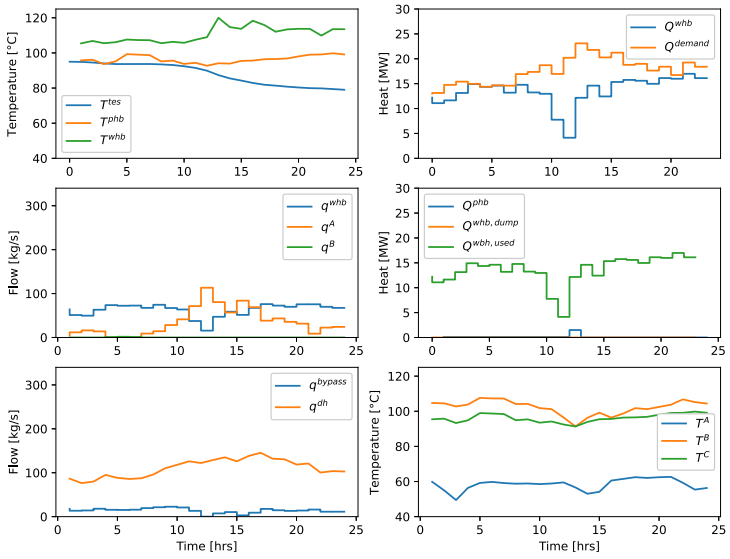

Figure 4.3.1: Optimal operations problem solved using historical heat data from Ferbruary 1^{st} 2019, with initial tank temperature at 95 °C.

March 30^{th} 2019 contains heat data consisting of mostly waste heat surplus. The optimized result is shown in Figure 4.3.2. The tank is charged for the majority of the time period, indicated by the middle left plot when $q^B > 0$. No peak heating is needed and no heat is dumped as the tank does not reach max capacity.

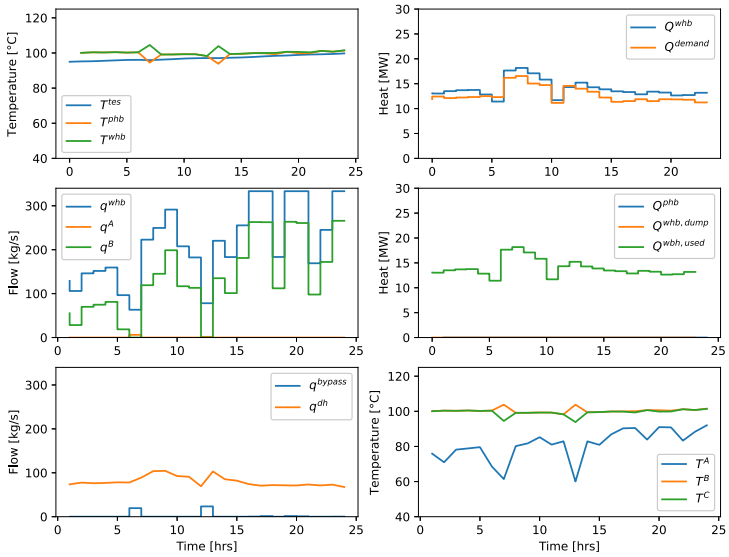

Figure 4.3.2: Optimal operations problem solved using historical heat data from March 30^{th} 2019, with initial tank temperature at 95 °C.

A day exhibiting alternating energy surplus and shortage is March 15^{th} 2019, presented in Figure 4.3.3. The tank is charged at hours when there is energy surplus and discharged when there is shortage. No peak heating is needed and no heat is dumped. Some bypass occurs during the shortage as the tank is discharged, and the remaining mass flow goes to the WHB yielding an outlet temperature from the boiler above the desired district supply temperature.

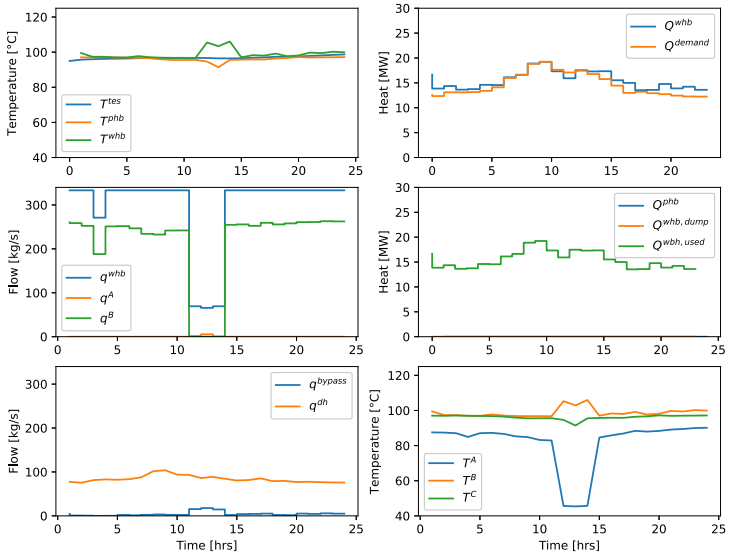

Figure 4.3.3: Optimal operations problem solved using historical heat data from March 15^{th} 2019, with initial tank temperature at 95 °C.

The results for initial tank temperatures of 60 °C and 115 °C for all three dates are shown in Appendix A. The solver was able to converge to local solutions for all three initial temperatures and for the variety of waste heat availability, and the optimal operations model presented in Equation 4.1.7 is used for further expansion of the time horizon.

4.3.2 7 Days

The optimal operations model presented in the previous section is used for further application of historical data. While the solver converges for up to six days, it is experienced that whether the NLP converges to a local solution or not greatly depends on the initial guesses. Therefore, it is decided to create separate functions to ensure that the initial guesses for the variables lies within the solutions feasible region. Instead of only creating one model for optimization, an additional model is created to be used only for simulation of one finite element. This simulation model will be given input data and guesses are provided for the free variables. The simulation model is then simulated at each finite element, for only one hour. The results of all of the short simulations are used as initial guesses in the optimization model, to be solved over the entire time horizon. This way, mostly feasible initial guesses are ensured, facilitating local convergence for the solver. The scripts $main\_operation.jl$, $tes\_create\_model.jl$, $tes\_sim.jl$, $tes\_bounds.jl$ and $read\_file.jl$ contains the implementation and are given in Appendix D.

Using the implementation as described above, the optimal operations problem is solved

for 7 days. To obtain local convergence, l_1-regularization terms for q_{bp} and q_{whb} were added.

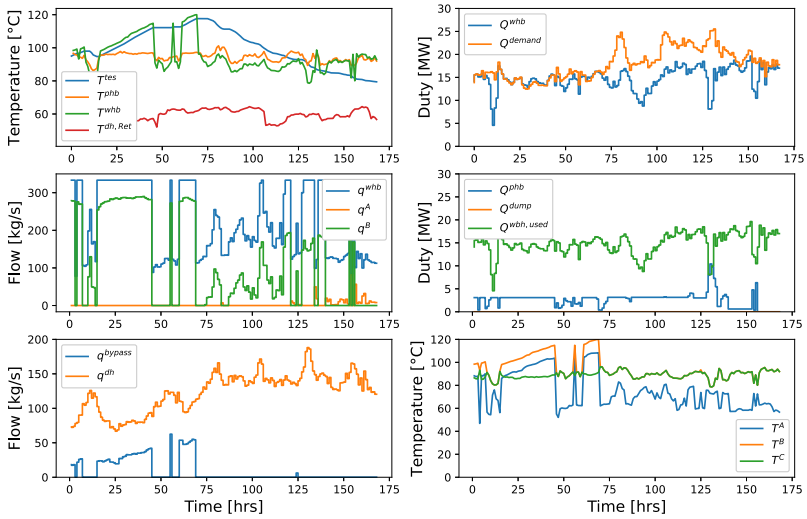

Figure 4.3.4: Optimal operations problem simulated for the first week of March 2019, with initial tank temperature at 95 °C.

The model successfully converges to a local optimum for the 7 day-time horizon. However, it is seen that the optimizer chooses to charge the tank even though the tank is warmer than the flow out of the WHB. This is not ideal as the temperature in the tank falls, and unnecessary amounts of water have to be pumped through the system instead of simply discharging the tank. In contrast, for different initial temperatures of the tank, this problem was not as prominent for the solutions obtained, as seen in Figure 4.3.5 and Figure 4.3.6.

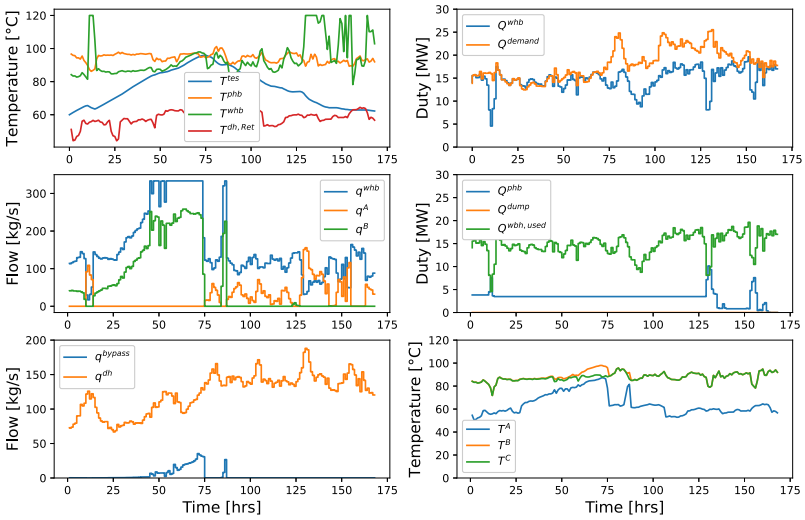

Figure 4.3.5: Optimal operations problem simulated for the first week of March 2019, with initial tank temperature at 60 °C.

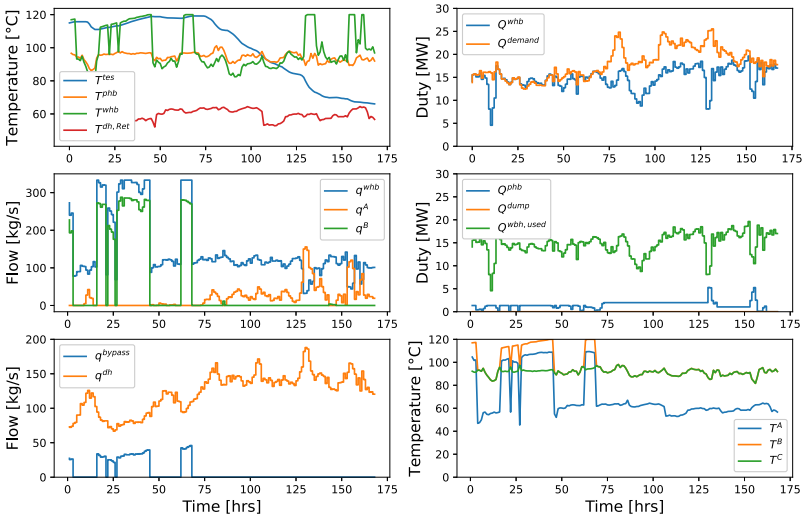

Figure 4.3.6: Optimal operations problem simulated for the first week of March 2019, with initial tank temperature at 115 °C.

However, as this behaviour is not desired, various complementarity constraints and regularization terms were tested to account for this, though none were able to converge for more than a few of days. The MPCC becomes too large, and the complementarities are not necessarily strict, which can present challenges for the NLP formulation [24, 28, 20]. Considering this fact in combination with the possibility of recycling in the system, the solver is not able to find a solution, despite good initial guesses. Therefore, to continue expanding the time horizon using historical data and obtain optimal operation for a longer representative time period which may be applied in the optimal design of the tank, it is decided to modify the formulation of the problem. The complementarity constraints are replaced with regular equality constraints that is defined for each finite element separately. As the availability of waste heat and the heat demand at each finite element is given, this information will be used to decide the possibility of charging and discharging the tank. In other words, the complementarity constraints previously present to avoid simultaneous charging and discharging will be replaced by specified constraints at each finite element, fixing either q_i^A or q_i^B to zero. In addition, two regularization terms with small penalties in the objective function were necessary to ensure convergence. The updated model used for obtaining optimal operation is thus formulated as

$$\min_{x,u,z} \quad \sum_{i=1}^{N_T} (Q_i^{phb})^2 + (Q_i^{whb,dump})^2 + \alpha q_i^{bp} + \beta q_i^{whb} \tag{4.3.1a}$$

$$\text{s.t.} \quad \mathbf{x}_i = \mathbf{x}_{i-1} + hf(\mathbf{x}_i, \mathbf{z}_i, \mathbf{p}) \qquad i = 1, ..., N_T \tag{4.3.1b}$$

$$\mathbf{x}_0 = \mathbf{x}(0) \tag{4.3.1c}$$

$$\mathbf{g}_i(\mathbf{x}_i, \mathbf{z}_i, \mathbf{u}_i, \mathbf{p}_i) = \mathbf{0} \qquad i = 1, ..., N_T \tag{4.3.1d}$$

$$\mathbf{x}_{L,i} \leq \mathbf{x}_i \leq \mathbf{x}_{U,i} \qquad i = 1, ..., N_T \tag{4.3.1e}$$

$$\mathbf{z}_{L,i} \leq \mathbf{z}_i \leq \mathbf{z}_{U,i} \qquad i = 1, ..., N_T \tag{4.3.1f}$$

$$\mathbf{u}_{L,i} \leq \mathbf{u}_i \leq \mathbf{u}_{U,i} \qquad i = 1, ..., N_T \tag{4.3.1g}$$

$$q_i^A = 0 \text{ if } Q_i^{whb} < Q_i^{demand} \qquad i = 1, ..., N_T \tag{4.3.1h}$$

$$q_i^B = 0 \text{ if } Q_i^{whb} \geq Q_i^{demand} \qquad i = 1, ..., N_T \tag{4.3.1i}$$

This formulation yields the result shown in Figure 4.3.7, when an initial tank temperature of 95 °C is used.

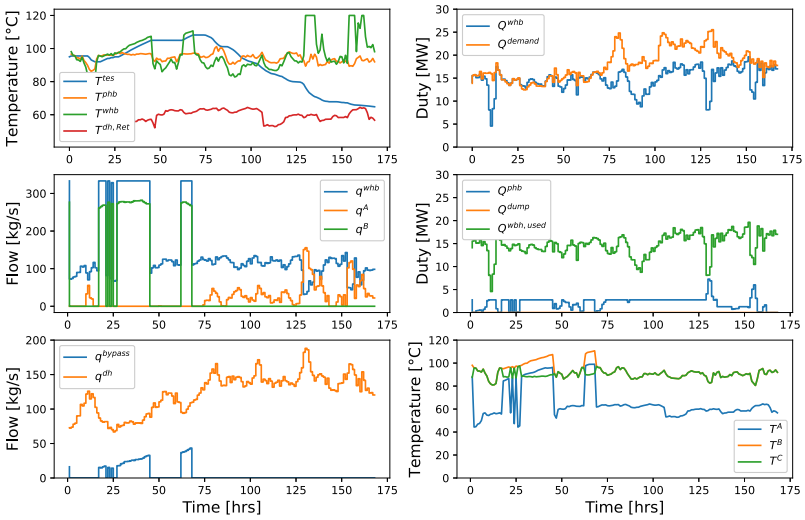

Figure 4.3.7: Optimal operations problem simulated for the first week of March 2019, with initial tank temperature at 95 °C, using the updated optimal operations model.

When comparing Figure 4.3.4 and Figure 4.3.7 it can be seen that the obtained results using the two models are quite similar, except less recycling occurs in the updated formulation. Still, with less recycling the updated formulation is able to utilize all the waste heat available, and no heat is dumped using either of the models. However, it is also seen from the middle right plot that more peak heating is used with the original formulation, especially after 125 hours have passed. This is because the updated formulation discharges the tank and the final tank temperature after simulating for the whole week is lower than for the original formulation, but as this causes less peak heating, it is the desired scenario. The updated model utilizes all of the heat available and demands less peak heating. The updated optimal operations model thus yields desired results, and will be used for further expansion of the time horizon.

4.4 Expanding the Horizon

Using the optimal operations model as formulated in Equation 4.3.1, a larger horizon is considered. As the order of magnitude between the mass flows and the heat is large, the weights of the regularization terms are increased so that the regularization terms influence the objective function.

As stated in Section 3.3 March 2019 is considered to be a representative month for utilization of the tank. Applying the data for this month, the NLP in Equation 4.3.1 is optimized for different initial values of the tank, using 720 finite elements. The results are shown in Figure 4.4.1, Figure 4.4.2 and Figure 4.4.3, for initial values $T_{tes,0} = 60$°C, $T_{tes,0} = 95$°C and $T_{tes,0} = 115$°C, respectively.

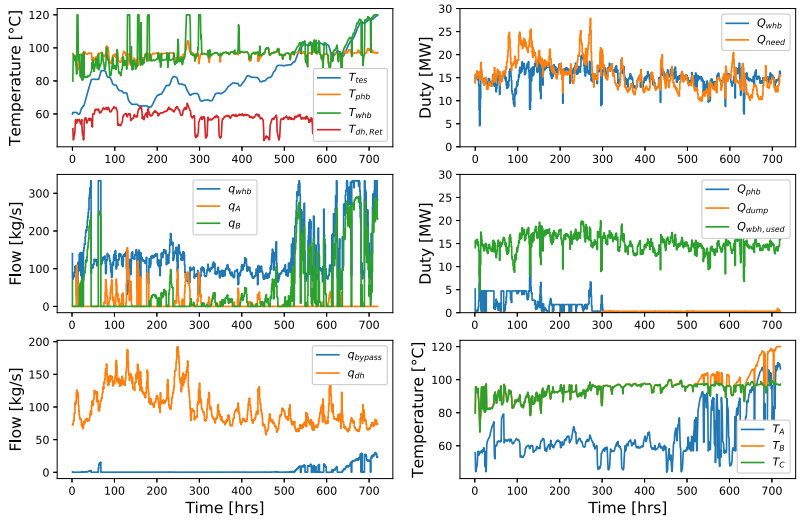

Figure 4.4.1: Optimal operations problem simulated using historical data from March 2019, with initial tank temperature at 60 °C.

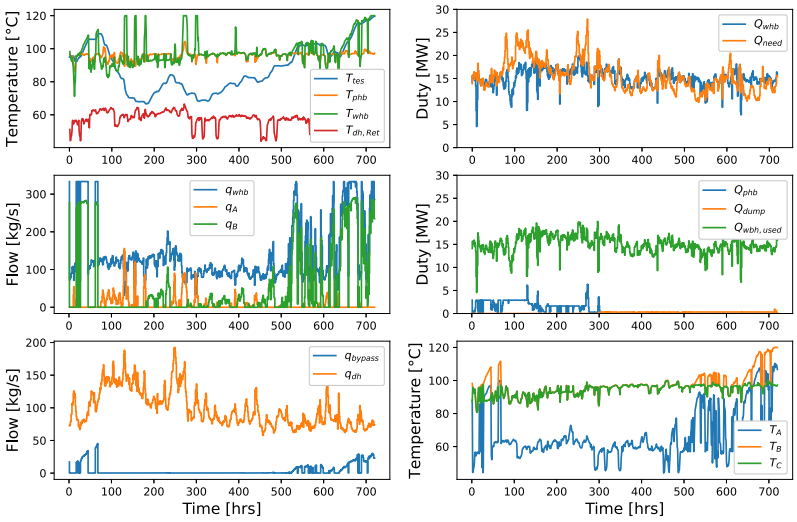

Figure 4.4.2: Optimal operations problem simulated using historical data from March 2019, with initial tank temperature at 95 °C.

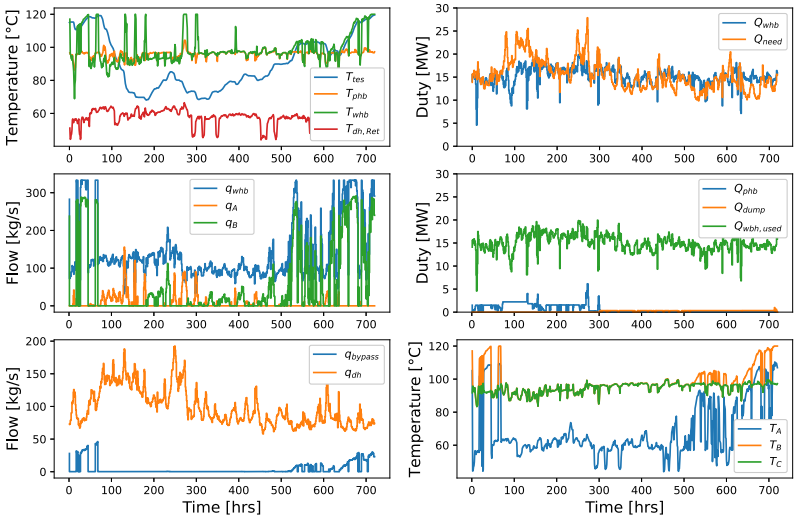

Figure 4.4.3: Optimal operations problem simulated using historical data from March 2019, with initial tank temperature at 115 °C.

When comparing the results after optimizing with the three different initial guesses, clear trends are visible. The three scenarios all have active bypass streams and peak heating at the same points in time, however of different magnitude. The scenario starting with the highest initial temperature yields the results needing the least amount of peak heating, as expected. This tank initially holds more energy to be drained than the other scenarios. Corroborative, the scenario with the lowest initial tank temperature requires the highest amount of peak heating. Moreover, less bypass is used for this scenario, than for the other two. This too, is as expected, as the mass flow primarily passes through the waste heat boiler and less recycling occurs. The tank is colder, so recycling with the lower initial temperature demands more heat as the inlet temperature to the WHB decreases when recycling occurs in this scenario. Contrarily, when recycling occurs at high initial temperatures of the tank, the inlet temperature to the WHB increases, hence the outlet temperature may be heated up to a higher final temperature. When the outlet temperature of the WHB is above the desired supply temperature to the district, bypass is needed. This effect is therefore more prominent for the two latter cases. Furthermore, from approximately 300 hours until the end of the horizon, the solution of all three cases are quite similar. It is seen that all three cases dump approximately the same amount of heat, and the final tank temperature is approximately the same.

Thus, it is concluded that for longer horizons, the initial temperature indeed loses influence as the results are similar. Initial temperature does however affect the total peak heating and potential for heat dumping. As the average district supply temperature is approximately 95 °C, this may be viewed as an equilibrium, and will therefore be used as the initial temperature when considering the optimal design study in the next chapter.

CHAPTER 5

Optimal Design

Once a model providing optimal operation using historical data is obtained, the system is also optimized with respect to the tank volume. As mentioned in the introduction, a smaller tank is beneficial both economically and physically. Moreover, for the investment to be of interest to the plant owners, in this case Mo Fjernvarme, it needs to be profitable in the long run. Therefore, the payback period is an interesting metric to consider. The size of the tank may thus be found through minimizing the payback period as a function of the investment cost and potential savings, while simultaneously obtaining optimal operation of the tank. This chapter first explains the cost calculations and present the optimal sizing problem. As mentioned in the introduction, Mo Fjernvarme's main source of energy is the off-gases from the ferrosilicon production process, while the remaining energy is obtained through electricity, fossil fuel, biofuel and CO gas. However, as the amount of fossil fuel and bio fuel are rarely used in practice [15], these energy sources are not considered. Accordingly, two cases are studied, one where electricity is used as peak heating source and, in the other, CO gas as peak heating source. The results are analyzed with respect to payback time and carbon emissions. Lastly, sensitivity analyses of the payback period for the two cases are conducted, considering various impact factors, such as emission taxes, tank size, interest rate and tank operating time throughout a year.

5.1 Cost Calculations and Problem Formulation

For a TES tank the payback period can be calculated by[9]

$$N_0 = \frac{ln(B/(B - I(V)r))}{ln(1 + r)}, \tag{5.1.1}$$

where B denotes the savings, $I(V)$ is the investment cost for a tank of a certain volume, V, and r indicates the yearly interest rate. The savings for this particular case is considered

to be the total reduction in operating cost, namely the difference in peak heating expenses for the base case, i.e. no TES tank at all, and the optimized case,

$$B = C^{peak} \sum_{i=1}^{nfe} (Q_i^{phb,noTES} - Q_i^{phb}). \tag{5.1.2}$$

C^{peak} is the total peak heating cost given in NOK/kWh, which includes cost of the energy source and emission taxes. Q_i^{phb} and $Q_i^{phb,noTES}$ is the peak heating needed at each finite element, with and without a TES tank installed, respectively. Q_i^{phb} is obtained from the model output, while $Q_i^{phb,noTES}$ is obtained from the given data from Mo Fjernvarme, equal to the red bars in Figure 5.1.1. The figure shows historical heat data (district demand in green and waste heat available in yellow), in addition to peak heating in red and heat dumped in blue, for the base case.

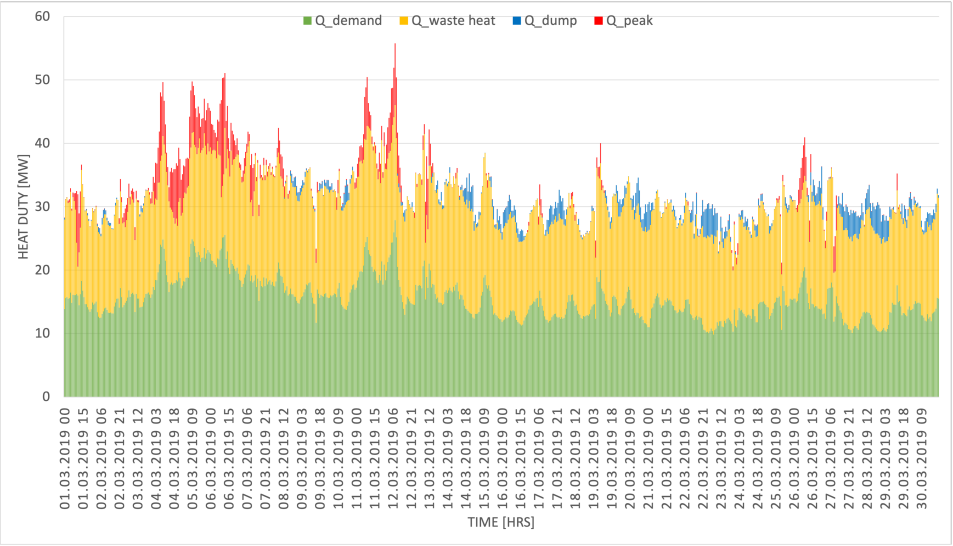

Figure 5.1.1: Historical heat data for March 2019. Heat demand in green, available waste heat in yellow. For the base case the waste heat dumped is shown in blue and peak heating needed in red.

For the investment cost, Haoran Li et al. [29] established a relationship between initial investment and water tank size based on previous projects. The investment cost of the water tank is thus given by the following power function approximation

$$I(V) = 0.0047(V)^{0.6218} \tag{5.1.3}$$

for $I(V)$ given in million euros and V given in m^3.

Total price per kWh for using either electricity or CO gas as peak heating sources is simply price of the fuel itself in addition to the emission fees on the fuel used,

$$C^{peak} = C^{fuel(electric/CO-gas)} + C^{tax}. \tag{5.1.4}$$

The price of electricity depends on the electricity price from the distributor at that time of year, hence it is volatile. For the case studied, the electricity price for March 2019 is used, which is 0.8721 NOK/kWh [30, 31, 32]. It is here considered that Mo i Rana is in a county granted VAT exemption[31]. As the tank is expected to be operational mainly during the transitional months, electricity prices for March are assumed to be representative [32]. However, electricity prices in 2019 was higher than the average of past years [31]. The impact of these uncertainties is further investigated in the sensitivity analysis.

CO gas is a byproduct from Mo Industrial Park and used in varying extent for peak heating, depending on the availability, demand from other plants in the park and other aspects [9]. The price is confidential.

As stated in Equation 5.1.4, the price of the government taxes on greenhouse gas emissions is needed in addition to the price of the energy source itself, to obtain the total cost per kWh. The Norwegian carbon tax rate is currently set to NOK 590 per tonne CO_2 equivalents, and is announced to increase significantly, to a value of NOK 2,000 per tonne CO_2 equivalents in the year 2030[33]. For NO_x-emission, the tax is currently set to 23,480 NOK per tonne NO_x[34]. According to The Norwegian Water Resources and Energy Directorate (NVE), the calculated CO_2-factor for power consumption in Norway is 17 gram CO_2 equivalents per kilowatt hour[35]. For CO gas 574 gCO_2 equivalents per kilowatt hour and 0.55 g NO_x per kWh is used[9].

The CO_2 tax per kilowatt hour is thereby given,

$$C^{tax(elec/COgas)} = \text{Tax on emission}\left[\frac{\text{NOK}}{\text{tonnes CO}_2}\right] \cdot \text{Emission}\left[\frac{\text{tonnes CO}_2}{\text{kWh}}\right], \quad (5.1.5)$$

and likewise for the NO_x tax,

$$C^{tax(elec/COgas)} = \text{Tax on emission}\left[\frac{\text{NOK}}{\text{kg NO}_x}\right] \cdot \text{Emission}\left[\frac{\text{kg NO}_x}{\text{kWh}}\right]. \quad (5.1.6)$$

The cost of electricity and CO gas is calculated using Equations 5.1.4-5.1.6 and values in Table 5.1. The resulting cost values used for optimization are summarized in Table 5.2.

Table 5.1: Values used for calculating parameters.

Description	Value	Unit
Interest rate[36]	5.00	[%]
CO_2 tax 2021[33]	590.00	[NOK/tonne CO_2e]
NO_x tax 2021[34]	23,480.00	[NOK/tonne NO_x]
Conversion rate EUR to NOK	10.03	[NOK/EUR]
Estimated electricity price March 2019	0.8721	[NOK/kWh]
Estimated price for CO-gas	*[1]	[NOK/kWh]
CO_2-factor for power consumption[35]	17	[g CO_2e/kWh]
NO_x-factor for power consumption	0	[g NO_xe/kWh]
CO_2-factor for CO gas[9]	574	[g CO_2e/kWh]
NO_x-factor for CO gas[9]	0.55	[g NO_xe/kWh]

[1] The price of CO gas is confidential.

Table 5.2: Parameters used for optimization.

Variable	Description	Value	Unit
C^{elec}	Cost of electricity	0.8721	[NOK/kWh]
$C^{elec,tax}$	Emission tax on electricity	0.0100	[NOK/kWh]
$C^{elec,total}$	Total cost of using electricity	0.0.8821	[NOK/kWh]
C^{COgas}	Cost of CO gas	$*$[2]	[NOK/kWh]
$C^{COgas,tax}$	Emission tax on CO gas	0.3516	[NOK/kWh]
$C^{COgas,total}$	Total cost of using CO gas	$*$[3]	[NOK/kWh]

Considering the presented cost equations and relevant parameters, a new optimization problem that incorporates operational and economic aspects is formulated for obtaining the optimal size of the TES tank. The discretized optimal sizing problem is presented in the following equation.

$$\min_{x,u,z} \quad N_0 + \sum_{i=1}^{nfe} C^{dump} Q_i^{whb,dump} + 10^{-5} q_i^{bp} + 10^{-7} q_i^{whb} \tag{5.1.7a}$$

$$\text{s.t.} \quad N_0 = \frac{ln(B/(B-(I(V^{tes})r)))}{ln(1+r)} \tag{5.1.7b}$$

$$B = C^{peak}(Q^{phb,noTES} - Q^{phb}) \tag{5.1.7c}$$

$$I(V^{tes}) = 0.0047(V^{tes})^{0.6218} \tag{5.1.7d}$$

$$\mathbf{x}_i = \mathbf{x}_{i-1} + hf(\mathbf{x}_i, \mathbf{z}_i, \mathbf{p}_i) \qquad i = 1, ..., nfe \tag{5.1.7e}$$

$$\mathbf{x}_0 = \mathbf{x}(0) \tag{5.1.7f}$$

$$\mathbf{g}_i(\mathbf{x}_i, \mathbf{z}_i, \mathbf{p}_i) = \mathbf{0} \qquad i = 1, ..., nfe \tag{5.1.7g}$$

$$\mathbf{x}_{L,i} \leq \mathbf{x}_i \leq \mathbf{x}_{U,i} \qquad i = 1, ..., nfe \tag{5.1.7h}$$

$$\mathbf{z}_{L,i} \leq \mathbf{z}_i \leq \mathbf{z}_{U,i} \qquad i = 1, ..., nfe \tag{5.1.7i}$$

$$q_i^A = 0 \text{ if } Q_i^{whb} < Q_i^{demand} \qquad i = 1, ..., nfe \tag{5.1.7j}$$

$$q_i^B = 0 \text{ if } Q_i^{whb} \geq Q_i^{demand} \qquad i = 1, ..., nfe \tag{5.1.7k}$$

The objective function, Equation 5.1.7a, is minimizing the payback period with respect to savings, investment cost and tank volume, while regularization terms and a weighted penalty for total amount of waste heat dumped are added to maintain optimal operation. Equations 5.1.7e-5.1.7k are the differential and algebraic equations and constraints as presented in the previous chapter. Bounds used for tank volume, payback time and savings are given in Table 5.3. A lower bound for savings of above zero was enforced to avoid numerical issues, as the payback period is a logarithmic expression, Equation 5.1.1.

[2]The price of CO gas is confidential.
[3]The price of CO gas is confidential.

Table 5.3: Dependant variables

Description	Lower bound	Variable	Upper bound	Unit
TES volume	0.0	V^{tes}	50,000.0	$[m^3]$
Payback time	0.0	N_0	-	[years]
Savings	100.0	B	-	$[\frac{NOK}{30days}]$

The TES optimization problem is solved for 30 days using historical data from Mo Fjernvarme, as presented in Section 3.3. The sampling time is one hour and an initial tank temperature of 95 °C is used. As for the optimal operations problem, the implementation is done in **julia**[25], using JuMP[26] as mathematical programming language and optimization solver IPOPT[27] with linear solver MA97. The scripts $main\_cost.jl$, $tes\_create\_model.jl$, $tes\_sim.jl$, $tes\_bounds.jl$ and $read\_file.jl$ contains the implementations, presented in Appendix D.

5.2 Results

The optimal design problem in Equation 5.1.7 has been solved using historical data from Mo Fjernvarme for the time period March 1^{st} 2019 to March 30^{th} 2019. The output for both cases is presented through six different plots in Figure 5.2.1 and Figure 5.2.2 for electricity and CO gas, respectively. The positioning and contents of each plot are similar to those in the previous chapter.

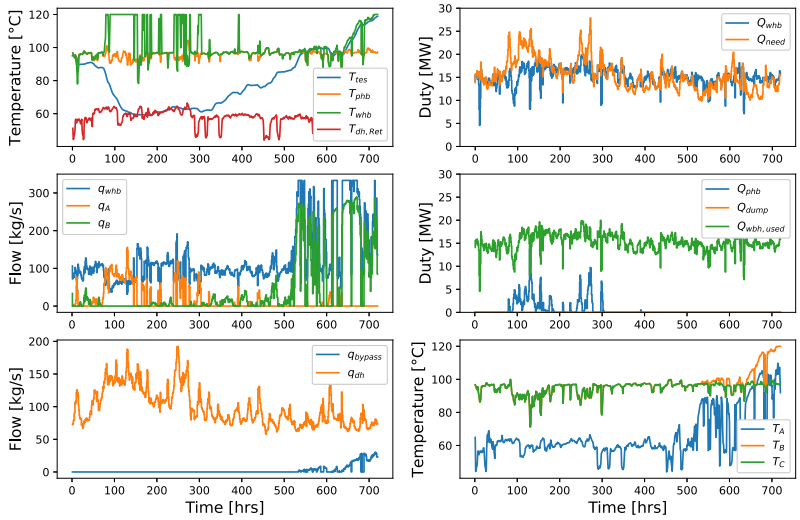

Figure 5.2.1: Optimal sizing result using electricity as peak heating source.

Optimizing the NLP in Equation 5.1.7 using electricity as peak heating source yielded a payback time of N_0 = 13.74 years, and an optimal volume for the TES tank of V^{tes} = 6322.95m^3. Total peak heating, i.e. the sum of all Q^{phb} in the middle right plot of Figure 5.2.1, is found to be 455.46 MW. Similarly, the total heat dumped, $Q^{whb,dump}$, also in the middle right plot, is found to be 0 MW.

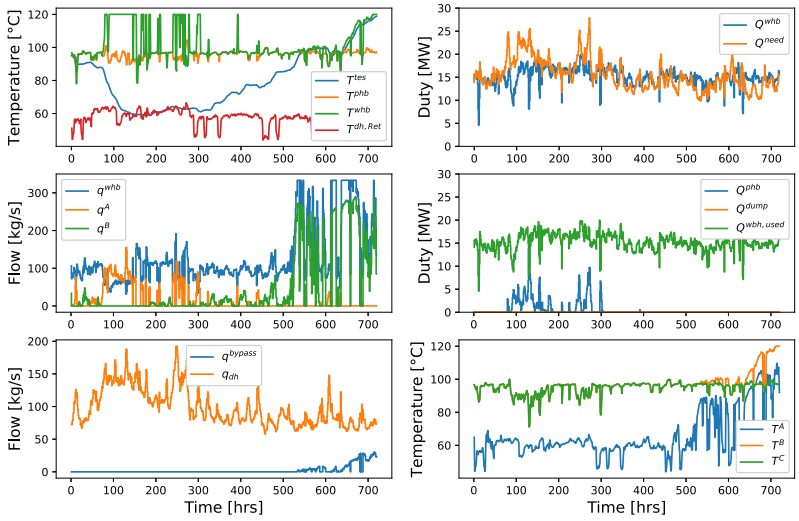

Figure 5.2.2: Optimal sizing result using CO gas as peak heating source.

Equivalently, when using CO gas as peak heating source the payback time was found to be 12.18 years, while the optimal size of the tank is equal for the two cases, i.e. 6322.95m^3. Total peak heating and waste heat dumped is also identical, as the operation and size of the tank was found to be the same.

The difference in available waste heat and heat demanded for the first 30 days of March 2019 is -280.40 MW. It is therefore expected at least 280 MW of peak heating for this month. However, it is seen from the top right plot in Figure 5.2.1 and Figure 5.2.2 that heat demanded is greater than waste heat available more or less throughout the first 300 hours of the month, approximately. Thus, it is expected that demanded peak heating must exceed 280 MW, and the optimized total peak heating result of 455.46 MW is as presumed.

The peak heating is reduced from 876.40 MW to 455.46 MW after introducing the TES tank. Considering the first case using electric peak heating, this yields a reduction in peak heating costs of NOK 371,323 for the investigated 30 days. For the CO gas case the reduction in tax expenses alone constitutes NOK 148,003 for the 30 days. These savings further increases when the price of CO is also considered. Amount of peak heating used and operating costs have thus decreased by 48%, regardless of the case, as they operate identically. Furthermore, $Q^{whb,dump}$ for the base case is 596.00 MW. For the optimized cases 420.94 MW of the peak heating is covered by the waste heat and there is no waste

heat dumped. Hence, at least 175.06 MW of the waste heat is stored in the tank at the end of the optimization to be used at later convenience. This is confirmed by the top left plots in both figures as the final value for T^{tes} is approximately 25 °C higher than the initial value.

As discussed in Section 3.3 it is the periods throughout the year of alternating waste heat surplus and deficit that are interesting operational periods for the tank. This is expected to be the case at least during transitional seasons, when the climate is neither very hot nor very cold. However, this is grounds for uncertainties in the payback time, and is investigated in the sensitivity analysis in the next section.

A summary of the numerical results are presented in Table 5.4, for both energy sources. Reduction are calculated with respect to the base case.

Table 5.4: Selected numerical results from optimization.

Description	**Electricity**	**CO**
Payback period, N_0	13.74 years	12.18 years
Tank size, V^{tes}	6322.95 m^3	6322.95 m^3
Peak heating, Q^{phb}	455.46 MW	455.46 MW
Peak heating reduction	420.94 MW	420.94 MW
Peak heating reduction	48 %	48 %
Waste heat dumped, $Q^{whb,dump}$	0 MW	0 MW
Waste heat savings	100 %	100 %
Peak heating CO_2 emission	23.23 tonnes CO_2e/year	784.30 tonnes CO_2e/year
Reduction in CO_2 emission	21.47 tonnes CO_2e/year	724.86 tonnes CO_2e/year
Reduction in CO_2 emission	48 %	48 %
Peak heating NO_x emission	0.00 kg NO_x/year	751.51 kgNO_x/year
Reduction in NO_x emission	0.00 kg NO_x/year	694.55 kgNO_x/year
Reduction in NO_x emission	0.00 %	48 %

Figure 5.2.3 illustrates the expected yearly amount of tonnes CO_2e, with and without the TES tank, assuming that there are three such periods corresponding to the 30 days considered. The blue bars represents the amount of CO_2 expected to be emitted while utilizing the TES tank, and the orange bars represents the amount of CO_2 expected to be emitted for the base case. The series to the left is for the scenario of using CO gas as peak heating source, the middle is the amount using 50 % CO gas and 50 % electricity and the series to the right is the emissions if the peak heating is covered by only electricity boilers. Similarly, Figure 5.2.4 illustrates the yearly amount of kg NO_x expected emitted, with and without the TES tank. The colors and placements of the series is as described above.

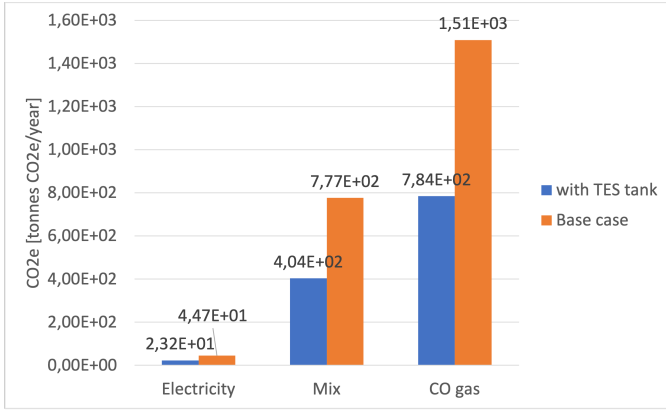

Figure 5.2.3: Comparison of the yearly calculated emission of CO_2 with and without the TES tank. Case 1 using electricity to the left, case 2 using CO gas to the right and a mix of the two in the middle.

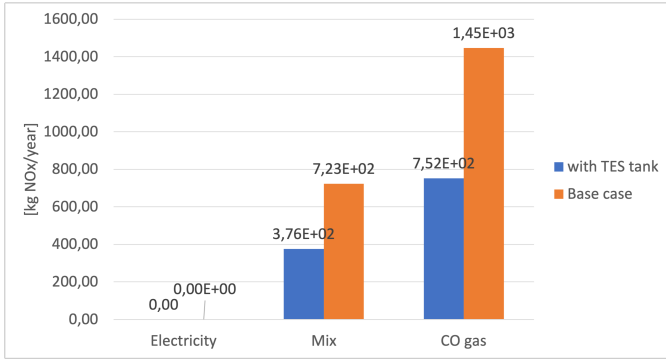

Figure 5.2.4: Comparison of the yearly calculated emission of NO_x with and without the TES tank. Case 1 using electricity to the left, case 2 using CO gas to the right and a mix of the two in the middle.

The yearly emissions are clearly lower for the optimized cases as they demand less supplementary energy. Furthermore, emission rates are evidently higher for the CO gas case. It is thus expected that this case is more prone to changes in tax rates than electricity. With increasing greenhouse gas emission taxes, it is expected that this case will yield the lowest payback period.

5.3 Sensitivity Analysis

To investigate the impact various factors have on the payback time, sensitivity analyses have been conducted. One analysis considering the first case using electricity as peak heating source and one considering the second case, where CO gas is used as peak heating

source. The two sensitivity plots are presented in Figure 5.3.1 and Figure 5.3.2, for case 1 and case 2, respectively. The payback period is plotted against relative change in various parameters. One parameter is changed at a time, while the rest is held constant. The parameters selected were based on inherent uncertainties they might present. The numerical results are given in Appendix C.

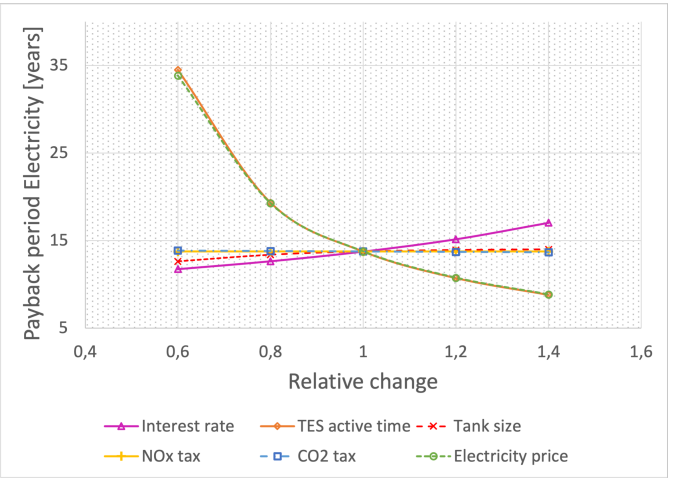

Figure 5.3.1: Sensitivity analysis when using electricity as peak heating source. Payback period plotted against the relative change in various parameters.

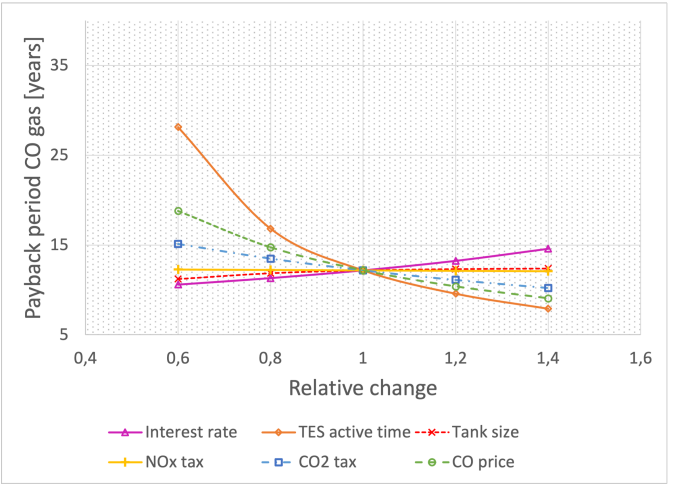

Figure 5.3.2: Sensitivity analysis when using CO gas as peak heating source. Payback period plotted against the relative change in various parameters.

For both cases it is clear that the presumed active period of the tank is the most promi-

nent factor when it comes to payback period sensitivity. In this work it is assumed that the tank is active for three months during a year. However, had this value been increased by 40 % till an active period of just over four months, the payback period for the two cases would have decreased from 13.74/12.18 till 8.80/7.90 years for cases 1/2.

The price of the peak heating source itself also greatly affects the value of the payback time, especially for the case considering electricity. As mentioned previously, electricity prices are volatile, giving this parameter high uncertainty.

Naturally, interest rate also affects the payback period for both cases. As mentioned, in this work the interest rate is assumed to be 5 %. This rate will depend on the policy rate, the lender and on the liquidity of the borrower. An interest rate of 5 % may be a conservative value. Thus, the result of 11.74/10.59 years from the sensitivity analysis when using the 40 % lower estimate may also be interesting.

Regarding changes in the tank volume, this factor is shown to be of less impact. A larger tank saves more energy and uses less peak heating, thus increasing the total savings for the period. However, the investment cost increases as well. Contrarily, the expenses saved on reducing the tank size are settled by the decreased savings of the waste heat not stored. Further, the smaller tank sizes discards large amount of heat which is heat lost for future shortages. The plotted results for various tank sizes are shown in Appendix B.

As the emissions from electricity is low, changes in the CO_2 tax does not greatly affect the payback period for the electricity case. In contrast, for the case considering CO gas, it is seen that a small change in the CO_2 tax impacts the payback period significantly. As stated in Section 5.1 this tax is expected to increase heavily over the upcoming years. Table 5.5 shows the expected payback time for various CO_2 taxes in the interval proposed by the Norwegian Government[33].

Table 5.5: Payback time for CO gas case with respect to various possible CO_2 taxes.

CO_2 tax [NOK/tonnes CO_2]	**Calculated Payback time CO gas [years]**
590	12.18
872	9.91
1154	8.37
1436	7.24
1718	6.39
2000	5.71

Changing the CO_2 tax from the current level of 590 NOK/tonnes CO_2e to the announced price of 2000 NOK/tonnes CO_2e, will cause a reduction in the payback time of approximately 6 years, as shown in Table 5.5. That is equal to half the payback time using the current tax level, for the case investigated.

CHAPTER 6

Final Remarks

Optimal operation and optimal design of a thermal energy storage tank have been obtained, based on historical data from a district heating network located in northern Norway. The operational objective was to minimize the need for peak heating supplied by boilers using either electricity or CO gas as energy sources, in addition to maximize the utilization of waste heat from a nearby ferrosilicon production plant. A simplified process of the DHN with a TES tank was defined, and after a model yielding optimal operation was obtained, historical data over an increasing time horizon was applied. Considering bidirectional flows using an MPCC formulation, optimal operation was obtained for a week. When trying to expand the time horizon past the seven days, convergence issues were met. For this reason it was decided to replace the complementarity constraints with equality constraints specified at each finite element, in order to obtain optimal operation for 30 days of historical data.

The obtained results after considering both the operational and economic aspects yielded a tank volume of 6322.95 m^3. This tank volume in combination with the optimal operation of the TES lead to a 48 % reduction in the amount of peak heating needed for the 30 days investigated, constituting a cut-back of NOK 371,323 using electricity as peak heating source, and a cut-back of NOK 148,003 in emission costs alone, when using energy from CO gas. Furthermore, all of the available waste heat for the period was utilized, i.e. no energy was discarded. The payback period using electricity as peak heating source was found to be 13.74 years, however very prone to changes in the electricity price. Similarly, for the case using CO gas as energy source, the optimized payback period is 12.18 years. It must be emphasized that this value is expected to significantly decrease should the announced CO_2 tax increase accede. The sensitivity analysis for both cases reveals the yearly assumed operational time of the TES system to be the variable introducing the highest uncertainty to the calculated payback time, followed by the cost of the energy source and interest rate.

It is concluded that introducing a thermal energy storage into the district heating network in northern Norway indeed decreases the amount of peak heating needed significantly. An expected payback time of 12-14 years is reasonable, and is additionally expected to decrease with increasing CO_2 tax.

6.1 Suggestion for Future Work

The model aims to optimize the payback period while optimally operating the tank. Future work could include modeling the thermal storage tank in a different way, such that the physical dynamics in the tank are taken into consideration. The tank may for example be modelled as several layers instead of perfectly mixed. Considering various types of insulation and heat loss from the tank and/or the pipes in the district heating network can also be a useful addition.

As discussed in Section 5.3 the price of the peak heating source greatly affects the value of the payback time, especially for the case considering electricity. Considering this fact, in addition to electricity prices being volatile, implementations considering live-electricity prices and taking advantage of this source when prices are low could be very interesting to consider.

Another approach could be to consider a variety of months. Had another month other than March 2019 been chosen for this study, the results would have been altered. For this scenario there was energy shortage in the beginning of the month, and later on a surplus. Hence, had it been the opposite, the amount of peak heating would have decreased, and the optimal volume would have been dependent on the inlet temperature, since this is crucial as to when the tank reaches max capacity. Considering several different months and evaluating how probable they each are, may yield a result based on a broader data foundation.

Chance-constrained optimization is also a suitable formulation for an optimization problem under uncertainty, such as this. A chance-constrained problem may be formulated for various scenarios each with different probabilities, ensuring profitable investment within a certain amount of years.

Bibliography

[1] Ioan Sarbu and Calin Sebarchievici. A comprehensive review of thermal energy storage, 1 2018. ISSN 20711050.

[2] European Commission. 2030 climate & energy framework. URL https://ec.europa.eu/clima/policies/strategies/2030_en#tab-0-0.

[3] Alberto Mirandola Mirandola and Enrico Lorenzini. Energy, environment and climate: From the past to the future. *International Journal of Heat and Technology*, 34 (2):159–164, 2016. ISSN 03928764. doi: 10.18280/ijht.340201.

[4] Henrik Zsiborács, Nóra Hegedűsné Baranyai, András Vincze, László Zentkó, Zoltán Birkner, Kinga Máté, and Gábor Pintér. Intermittent renewable energy sources: The role of energy storage in the european power system of 2040. *Electronics*, 8(7), 2019. ISSN 2079-9292. doi: 10.3390/electronics8070729. URL https://www.mdpi.com/2079-9292/8/7/729.

[5] D. Connolly, H. Lund, B.V. Mathiesen, S. Werner, B. Möller, U. Persson, T. Boermans, D. Trier, P.A. Østergaard, and S. Nielsen. Heat roadmap europe: Combining district heating with heat savings to decarbonise the eu energy system. *Energy Policy*, 65:475–489, 2014. ISSN 0301-4215. doi: https://doi.org/10.1016/j.enpol.2013.10.035. URL https://www.sciencedirect.com/science/article/pii/S0301421513010574.

[6] Aira Hast, Sanna Syri, Vidas Lekavičius, and Arvydas Galinis. District heating in cities as a part of low-carbon energy system. *Energy*, 152:627–639, 2018. ISSN 0360-5442. doi: https://doi.org/10.1016/j.energy.2018.03.156. URL https://www.sciencedirect.com/science/article/pii/S0360544218305656.

[7] Brage Rugstad Knudsen, Daniel Rohde, and Hanne Kauko. Thermal energy storage sizing for industrial waste-heat utilization in district heating: A model predictive control approach. *Energies*, (2021), To appear.

[8] Mandar Thombre, Zawadi Mdoe, and Johannes Jäschke. Data-driven robust optimal operation of thermal energy storage in industrial clusters. *Processes*, 8(2), 2020. ISSN 2227-9717. doi: 10.3390/pr8020194. URL https://www.mdpi.com/2227-9717/8/2/194.

[9] Hanne Kauko, Daniel Rohde, Brage Rugstad Knudsen, and Terje Sund-Olsen. Potential of thermal energy storage for a district heating system utilizing industrial waste heat. *Energies*, 13(15), 2020. ISSN 1996-1073. doi: 10.3390/en13153923. URL https://www.mdpi.com/1996-1073/13/15/3923.

[10] Mandar Thombre, Sandeep Prakash, Brage Rugstad Knudsen, and Johannes Jäschke. Optimizing the capacity of thermal energy storage in industrial clusters. In Sauro Pierucci, Flavio Manenti, Giulia Luisa Bozzano, and Davide Manca, editors, *30th European Symposium on Computer Aided Process Engineering*, volume 48 of *Computer Aided Chemical Engineering*, pages 1459–1464. Elsevier, 2020. doi: https://doi.org/10.1016/B978-0-12-823377-1.50244-5. URL https://www.sciencedirect.com/science/article/pii/B9780128233771502445.

[11] Sotirios A. Kyriakis and Paul L. Younger. Towards the increased utilisation of geothermal energy in a district heating network through the use of a heat storage. *Applied Thermal Engineering*, 94:99–110, 2016. ISSN 1359-4311. doi: https://doi.org/10.1016/j.applthermaleng.2015.10.094. URL https://www.sciencedirect.com/science/article/pii/S1359431115011412.

[12] Patrizia Simeoni, Gellio Ciotti, Mattia Cottes, and Antonella Meneghetti. Integrating industrial waste heat recovery into sustainable smart energy systems. *Energy*, 175:941–951, 2019. ISSN 0360-5442. doi: https://doi.org/10.1016/j.energy.2019.03.104. URL https://www.sciencedirect.com/science/article/pii/S0360544219305080.

[13] Elisa Guelpa and Vittorio Verda. Thermal energy storage in district heating and cooling systems: A review. *Applied Energy*, 252:113474, 2019. ISSN 0306-2619. doi: https://doi.org/10.1016/j.apenergy.2019.113474. URL https://www.sciencedirect.com/science/article/pii/S0306261919311481.

[14] Kui Wang, Marco A. Satyro, Ross Taylor, and Philip K. Hopke. Thermal energy storage tank sizing for biomass boiler heating systems using process dynamic simulation. *Energy and Buildings*, 175:199–207, 2018. ISSN 0378-7788. doi: https://doi.org/10.1016/j.enbuild.2018.07.023. URL https://www.sciencedirect.com/science/article/pii/S0378778818306844.

[15] Norsk Fjernvarme. Mo fjernvarme. URL https://www.fjernkontrollen.no/mo-fjernvarme/.

[16] Arun Kumar and S.K. Shukla. A review on thermal energy storage unit for solar thermal power plant application. *Energy Procedia*, 74:462–469, 2015. ISSN 1876-6102. doi: https://doi.org/10.1016/j.egypro.2015.07.

728. URL https://www.sciencedirect.com/science/article/pii/S1876610215014964. The International Conference on Technologies and Materials for Renewable Energy, Environment and Sustainability –TMREES15.

[17] Bjarne Foss and Tor Aksel N. Heirung. *Merging Optimization and Control*. 03 2016. ISBN 978-82-7842-201-4.

[18] *2. Theory of DAE's*, pages 15–39. doi: 10.1137/1.9781611971224.ch2. URL https://epubs.siam.org/doi/abs/10.1137/1.9781611971224.ch2.

[19] Lorenz T. Biegler. An overview of simultaneous strategies for dynamic optimization. *Chemical Engineering and Processing: Process Intensification*, 46 (11):1043–1053, 2007. ISSN 0255-2701. doi: https://doi.org/10.1016/j.cep.2006.06.021. URL https://www.sciencedirect.com/science/article/pii/S0255270107001122. Special Issue on Process Optimization and Control in Chemical Engineering and Processing.

[20] Lorenz T. Biegler. *Nonlinear Programming: Concepts, Algorithms, and Applications to Chemical Processes*. SIAM, 2010.

[21] J.C Butcher. Numerical methods for ordinary differential equations, 2008.

[22] Stephen C Billups and Katta G Murty. Complementarity problems. *Journal of computational and applied mathematics*, 124(1):303–318, 2000. ISSN 0377-0427.

[23] B.T. Baumrucker, J.G. Renfro, and L.T. Biegler. Mpec problem formulations and solution strategies with chemical engineering applications. *Computers & Chemical Engineering*, 32(12):2903–2913, 2008. ISSN 0098-1354. doi: https://doi.org/10.1016/j.compchemeng.2008.02.010. URL https://www.sciencedirect.com/science/article/pii/S0098135408000367.

[24] Sven Leyffer, Gabriel López-Calva, and Jorge Nocedal. Interior methods for mathematical programs with complementarity constraints. *SIAM Journal on Optimization*, 17(1):52–77, 2006.

[25] Jeff Bezanson, Alan Edelman, Stefan Karpinski, and Viral B Shah. Julia: A fresh approach to numerical computing. *SIAM review*, 59(1):65–98, 2017. URL https://doi.org/10.1137/141000671.

[26] Iain Dunning, Joey Huchette, and Miles Lubin. Jump: A modeling language for mathematical optimization. *SIAM Review*, 59(2):295–320, 2017. doi: 10.1137/15M1020575.

[27] Andreas Wächter and Lorenz Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical programming*, 106:25–57, 03 2006. doi: 10.1007/s10107-004-0559-y.

[28] Sven Leyffer and Todd Munson. A globally convergent filter method for mpecs. 11 2007.

[29] Haoran Li, Juan Hou, Tianzhen Hong, Yuemin Ding, and Natasa Nord. Energy, economic, and environmental analysis of integration of thermal energy storage into district heating systems using waste heat from data centres. *Energy*, 219:119582, 2021. ISSN 0360-5442. doi: https://doi.org/10.1016/j.energy.2020.119582. URL https://www.sciencedirect.com/science/article/pii/S036054422032689X.

[30] Robert Skotvold & Magne Holstad. Markant fall i strømprisen. URL https://www.ssb.no/energi-og-industri/artikler-og-publikasjoner/markant-fall-i-stromprisen.

[31] Robert Skotvold & Thomas Aanensen. Veldig lav strømpris i 2020. URL https://www.ssb.no/energi-og-industri/artikler-og-publikasjoner/veldig-lav-strompris-i-2020.

[32] LOS. Historiske strømpriser. URL https://www.los.no/dagens-strompris/historiske-strompriser/.

[33] Jon Berg. Norway's comprehensive climate action plan. URL https://www.regjeringen.no/en/aktuelt/heilskapeleg-plan-for-a-na-klimamalet/id2827600/.

[34] Skatteetaten. Nox-avgift. URL https://www.skatteetaten.no/satser/saravgift---nox/?year=2021#rateShowYear.

[35] Kristian Løksa. Strømforbruk i norge har lavt klimagassutslipp. URL https://www.nve.no/nytt-fra-nve/nyheter-energi/stromforbruk-i-norge-har-lavt-klimagassutslipp/.

[36] Norges Bank. The policy rate. URL https://www.norges-bank.no/en/topics/Monetary-policy/Policy-rate/.

APPENDIX A

Supplementing Results of Optimal Operation

Here, the results of the optimal operations problem in Equation 4.1.7 using initial tank temperatures of 60 °C and 115 °C for February 1^{st} 2019, March 30^{th} and March 15^{th} are presented.

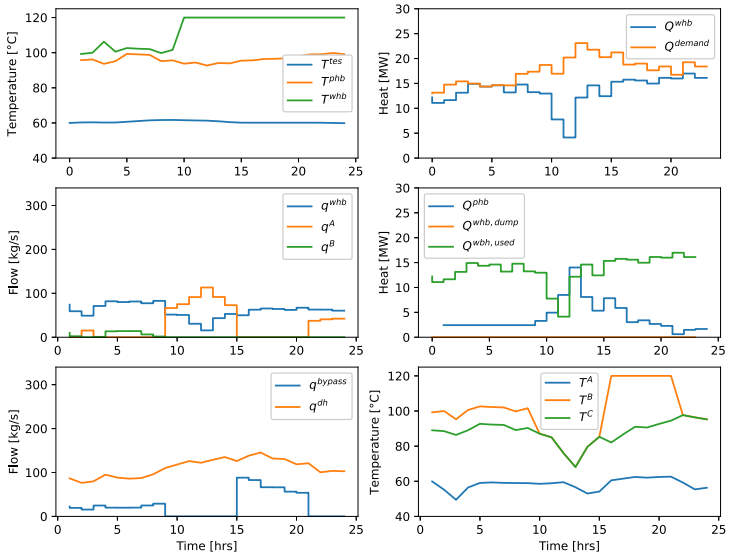

Figure A.0.1: Optimal operations problem from Equation 4.1.7 solved using historical heat data from February 1^{st} 2019, with initial tank temperature at 60 °C.

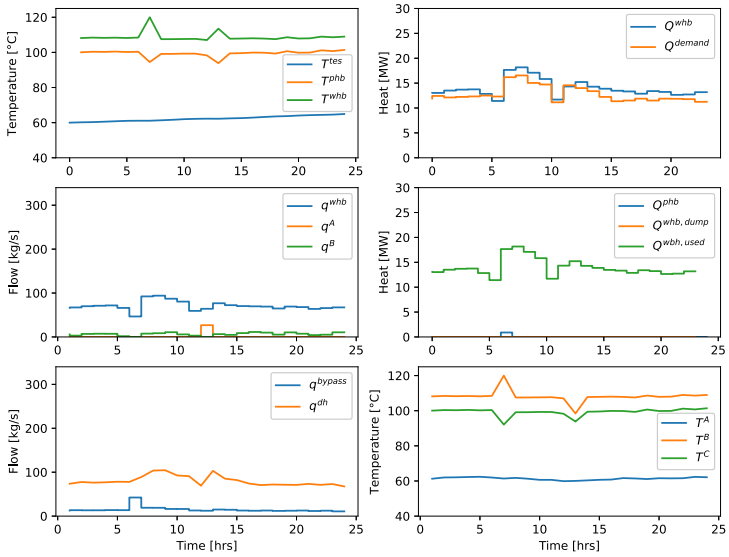

Figure A.0.2: Optimal operations problem from Equation 4.1.7 solved using historical heat data from March 30^{th} 2019, with initial tank temperature at 60 °C.

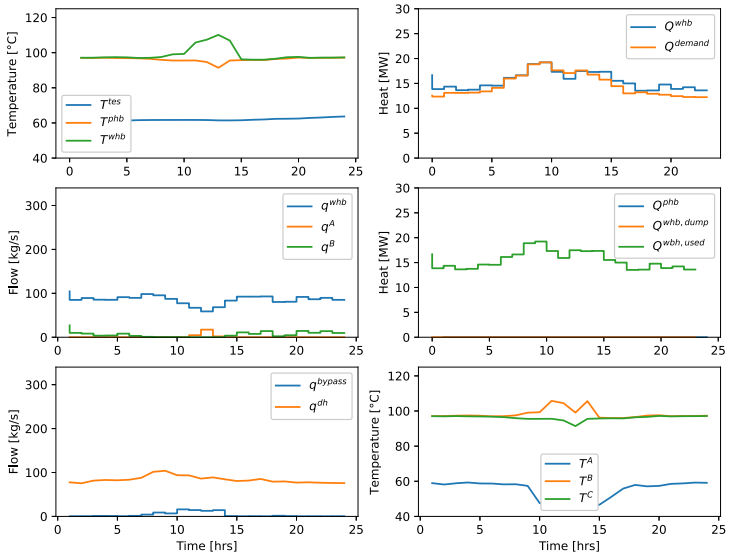

Figure A.0.3: Optimal operations problem from Equation 4.1.7 solved using historical heat data from March 15^{th} 2019, with initial tank temperature at 60 °C.

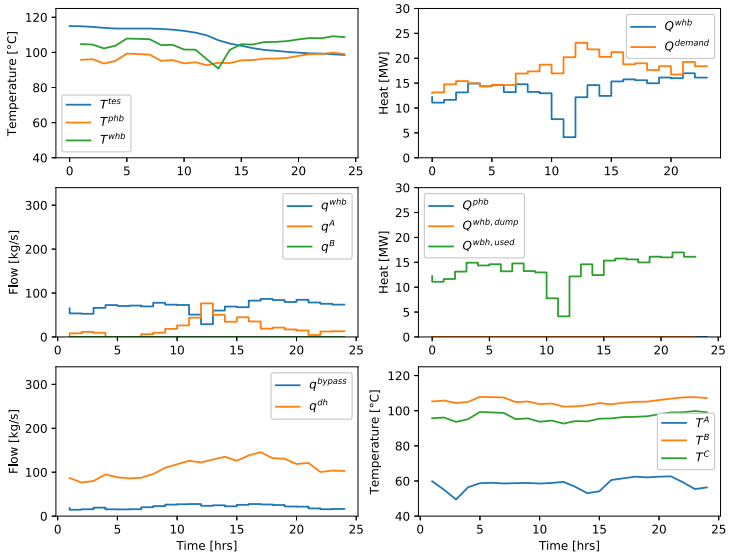

Figure A.0.4: Optimal operations problem from Equation 4.1.7 solved using historical heat data from February 1^{st} 2019, with initial tank temperature at 115 °C.

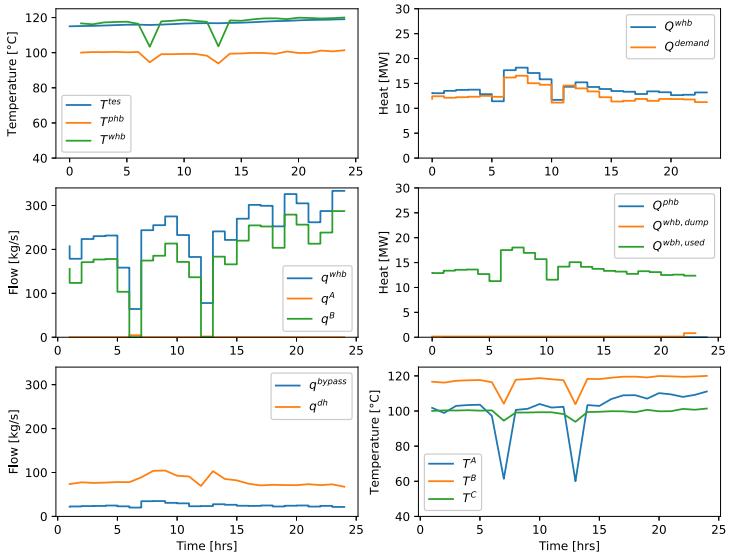

Figure A.0.5: Optimal operations problem from Equation 4.1.7 solved using historical heat data from March 30^{th} 2019, with initial tank temperature at 115 °C.

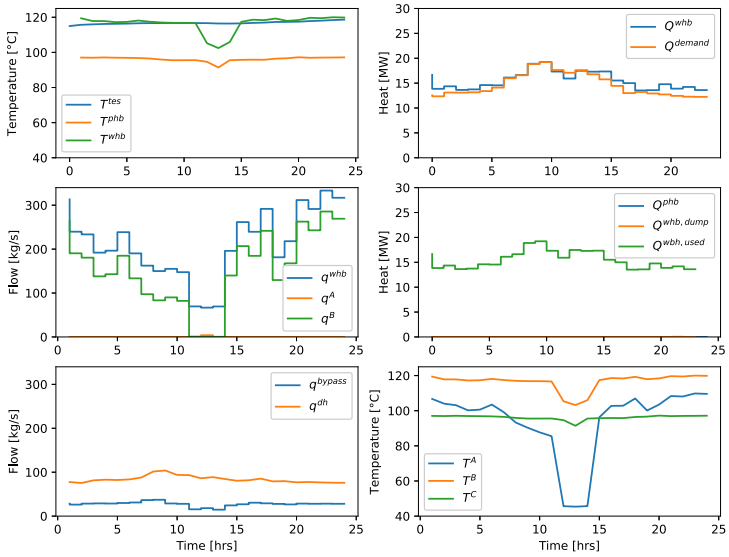

Figure A.0.6: Optimal operations problem from Equation 4.1.7 solved using historical heat data from March 15^{th} 2019, with initial tank temperature at 115 °C.

APPENDIX B

Results of Various Tank Volumes

Optimal operation results when fixing the tank volume to 3794 m^3, 5058 m^3, 6323 m^3, 7588 m^3 and 8852 m^3, respectively.

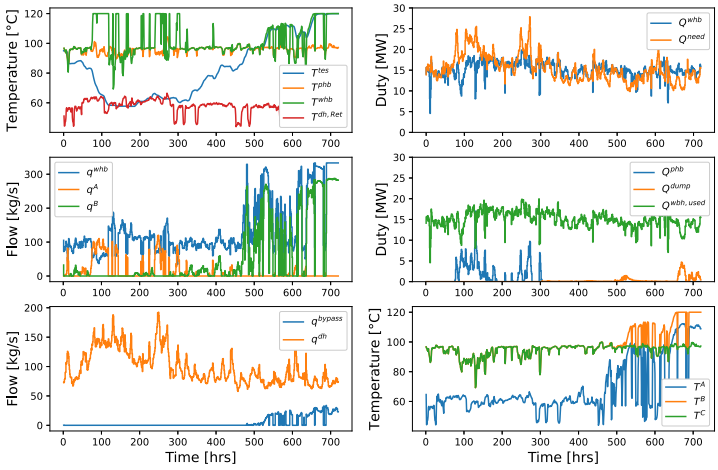

Figure B.0.1: Optimal operation when V^{tes}=3794 m^3.

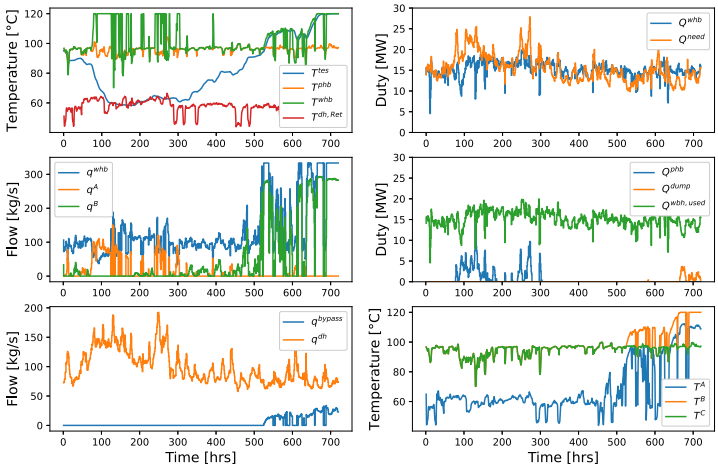

Figure B.0.2: Optimal operation when V^{tes}=5058 m^3.

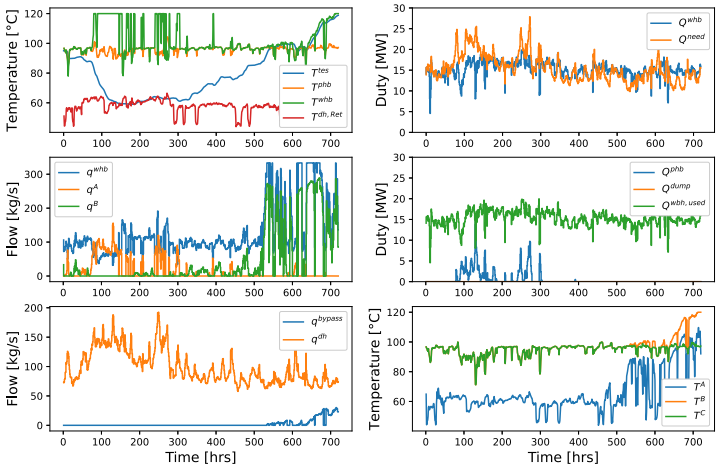

Figure B.0.3: Optimal operation when V^{tes}=6323 m^3.

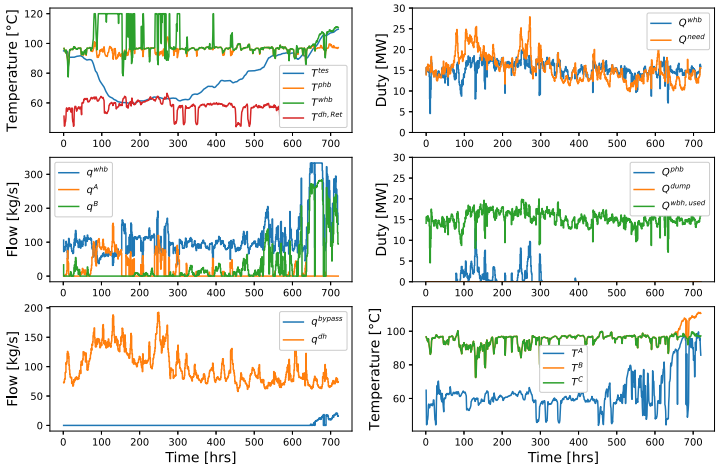

Figure B.0.4: Optimal operation when V^{tes}=7588 m^3.

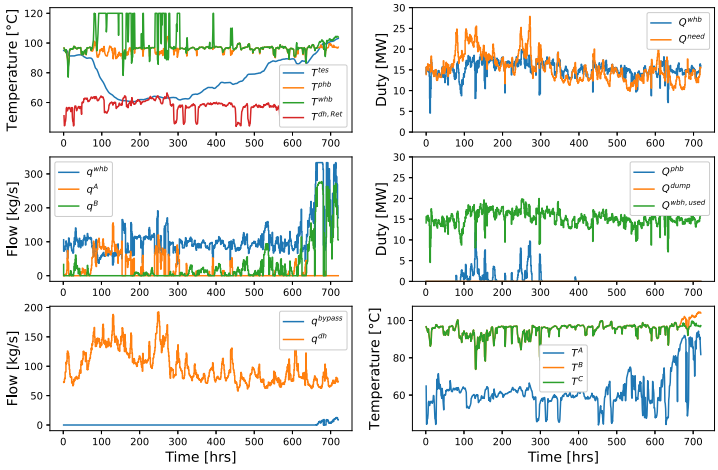

Figure B.0.5: Optimal operation when V^{tes}=8852 m^3.

APPENDIX C

Numerical Results of the Sensitivity Analyses

This chapter presents the numerical results from the sensitivity analyses conducted, presented in Section 5.3.

Table C.1: Numerical result of sensitivity analysis for electricity case.

Relative change	CO_2 tax	NO_x tax	Tank size	TES active time	Interest rate	Electricity price
0.6	13.83	13.74	12.61	34.50	11.74	33.84
0.8	13.79	13.74	13.37	19.34	12.64	19.24
1.0	13.74	13.74	13.74	13.74	13.74	13.74
1.2	13.70	13.74	13.91	10.71	15.15	10.74
1.4	13.65	13.74	13.98	8.80	17.03	8.83

Table C.2: Numerical result of sensitivity analysis for CO gas case.

Relative change	CO_2 tax	NO_x tax	Tank size	TES active time	Interest rate	Electricity price
0.6	15.10	12.27	11.21	28.14	10.59	18.80
0.8	13.48	12.22	11.86	16.82	11.31	14.75
1.0	12.18	12.18	12.18	12.18	12.18	12.18
1.2	11.11	12.13	12.32	9.58	13.24	10.39
1.4	10.22	12.09	12.38	7.90	14.58	9.06

APPENDIX D

Julia Codes

This section presents files for the implementation of the optimal operations and optimal design problem. The code is written in julia using JuMP[26] as mathematical language and solver IPOPTs MA97[27].

D.1 "main_operation.jl"

This script contains the implementation for running the optimal operations problem. The functions $read\_file()$, $tes\_create\_model()$, $simulation()$ and $set_bounds()$ are called, and the model is optimized.

```
using PyPlot, LaTeXStrings

include("tes_create_model.jl");
include("tes_sim.jl");
include("tes_bounds.jl");
include("read_file.jl");

default = Dict(
    :ρ_dh => 1000,          #kg/m3
    :Cp_dh => 4.18,         #kJ/(kg-K)
    :T_dh_minSup => 92.5,     #minimum for supply
    :T_dh_maxSup => 95,       #maximum for supply
    :T_whb_max => 120,
    :V_tes => 5000,          #m3
    :Tlb => 40,
    :Tub => 120,
    :Q_used_ub => 22e3*3600,          # kJ/hr
    :q_whb_init => 291,              # [m3/hr]
    :T_tes_init => 95,               # Starting value T_tes
    :nfe => 720,
    :ncp => 1,
```

```
    :h => 1.0,
    :S => ["T_tes"],
    :C => ["q_whb", "q_A", "q_B", "q_bp", "q_sys"],
    :A => ["T_A", "T_B", "T_C", "T_phb", "T_whb"],
    :A2 => ["Q_phb", "Q_whb_used", "Q_whb_dump", "Q_need"]
);
nfe = default[:nfe];
ncp = default[:ncp];
S = default[:S];
C = default[:C];
A = default[:A];
A2 = default[:A2];

Q_whb = [];
q_dh = [];
T_dh_ret = [];

file1 = "15032019.csv";
file2 = "data_jan_18.csv";
file3 = "march19.csv";
Q_whb, q_dh, T_dh_ret, T_dh_sup, dates = read_file(file3, nfe);

initial_guesses = ["Ttes_guess","Tphb_guess","Twhb_guess","qwhb_guess","qA_guess",
                   "qB_guess","Qphb_guess","Qdump_guess","Qused_guess",
                   "qbp_guess","TA_guess","TB_guess","TC_guess"];

sim_mod = tes_create_mod(Q_whb[1],q_dh[1],T_dh_ret[1],T_dh_sup[1],printlevel=0);
opt_mod = tes_create_mod(Q_whb,q_dh,T_dh_ret,T_dh_sup, printlevel=5);
opt_mod, guesses = simulation(sim_mod,opt_mod,Q_whb,q_dh,T_dh_ret,T_dh_sup);
opt_mod = set_bounds(opt_mod,Q_whb,q_dh,T_dh_ret,T_dh_sup);
init_guess = Dict(initial_guesses .=> guesses);

@NLobjective(opt_mod, Min, sum(opt_mod[:y]["Q_phb",i]^2 for i in 1:nfe)
            + 1e10*sum(opt_mod[:u]["q_A",i]*opt_mod[:u]["q_B",i] for i in 1:nfe)
            + sum(opt_mod[:y]["Q_whb_dump",i]^2 for i in 1:nfe)
            + sum(1e4*opt_mod[:u]["q_bp",i]^2 for i in 1:nfe)
            + sum(1e4*opt_mod[:u]["q_whb",i] for i in 1:nfe)
            );

optimize!(opt_mod);
println(termination_status(opt_mod));

######################
### PLOTTING
######################
Q_units = 1/3600/1000; # [kJ/hr] -> [MW]
q_units = 1/3600*1000; # [m3/hr] -> [kg/s]

tspan = (0.0, length(Q_whb));
tot_Q_phb = sum(value.(opt_mod[:y]["Q_phb",:])[i] for i in 1:nfe)*Q_units;
tot_Q_dump = sum(value.(opt_mod[:y]["Q_whb_dump",:])[i] for i in 1:nfe)*Q_units;

fig = figure(figsize = (12,12));

PyPlot.subplot(321);
ax1 = PyPlot.plot(tspan[1]:tspan[2], vcat(value.(opt_mod[:x0]["T_tes"]),
                  value.(opt_mod[:x]["T_tes",:,end])), label = L"T^{tes}");
ax1 = PyPlot.plot(tspan[1]+1:tspan[2], value.(opt_mod[:z]["T_phb",:,end]),
                  label = L"T^{phb}");
ax1 = PyPlot.plot(tspan[1]+1:tspan[2], value.(opt_mod[:z]["T_whb",:,end]),
                  label = L"T^{whb}");
ax1 = PyPlot.plot(tspan[1]+1:tspan[2], T_dh_ret, label = L"T^{dh,Ret}");
PyPlot.ylabel(L"Temperature [$  $C]",fontsize=14)
PyPlot.legend();

PyPlot.subplot(322);
ax6 = PyPlot.step(tspan[1]:tspan[2]-1, Q_whb*Q_units, label = L"Q^{whb}");
ax6 = PyPlot.step(tspan[1]:tspan[2]-1, value.(opt_mod[:y]["Q_need",:])*Q_units,
```

```
                    label = L"Q^{demand}");
PyPlot.ylim([0.0, 30]);
PyPlot.ylabel("Duty [MW]",fontsize=14)
PyPlot.legend();

PyPlot.subplot(323);
ax2 = PyPlot.step(tspan[1]+1:tspan[2], value.(opt_mod[:u]["q_whb",:])*q_units,
                    label = L"q^{whb}");
ax2 = PyPlot.step(tspan[1]+1:tspan[2], value.(opt_mod[:u]["q_A",:])*q_units,
                    label = L"q^A");
ax2 = PyPlot.step(tspan[1]+1:tspan[2], value.(opt_mod[:u]["q_B",:])*q_units,
                    label = L"q^B");
PyPlot.ylabel("Flow [kg/s]",fontsize=14)
PyPlot.legend();

PyPlot.subplot(324);
ax3 = PyPlot.step(tspan[1]+1:tspan[2], value.(opt_mod[:y]["Q_phb",:])*Q_units,
                    label = L"Q^{phb}");
ax3 = PyPlot.step(tspan[1]+1:tspan[2], value.(opt_mod[:y]["Q_whb_dump",:])
                    *Q_units, label = L"Q^{dump}");
ax3 = PyPlot.step(tspan[1]+1:tspan[2], value.(opt_mod[:y]["Q_whb_used",:])
                    *Q_units, label = L"Q^{wbh,used}");
PyPlot.ylim([0.0, 30]);
PyPlot.ylabel("Duty [MW]",fontsize=14)
PyPlot.legend();

PyPlot.subplot(325);
ax4 = PyPlot.step(tspan[1]+1:tspan[2], value.(opt_mod[:u]["q_bp",:])*q_units,
                    label = L"q^{bypass}");
ax4 = PyPlot.step(tspan[1]+1:tspan[2], q_dh*q_units, label = L"q^{dh}");
PyPlot.xlabel("Time [hrs]",fontsize=14);
PyPlot.ylabel("Flow [kg/s]",fontsize=14)
PyPlot.ylim([0.0, 200.0]);
PyPlot.legend();

PyPlot.subplot(326);
ax5 = PyPlot.plot(tspan[1]+1:tspan[2], value.(opt_mod[:z]["T_A",:,end]),
                    label = L"T^A");
ax5 = PyPlot.plot(tspan[1]+1:tspan[2], value.(opt_mod[:z]["T_B",:,end]),
                    label = L"T^B");
ax5 = PyPlot.plot(tspan[1]+1:tspan[2], value.(opt_mod[:z]["T_C",:,end]),
                    label = L"T^C");
PyPlot.xlabel("Time [hrs]",fontsize=14);
PyPlot.ylabel(L"Temperature [$  $C]",fontsize=14);
PyPlot.ylim([0.0, 120.0]);
PyPlot.legend();

fig.show();

########################################
# Plotting initial guesses
########################################

fig2 = figure(figsize = (12,12));

PyPlot.subplot(321);
ax1 = PyPlot.plot(tspan[1]+1:tspan[2], init_guess["Ttes_guess"][:,end],
                    label = L"T_{tes,guess}");
ax1 = PyPlot.plot(tspan[1]+1:tspan[2], init_guess["Tphb_guess"][:,end],
                    label = L"T_{phb,guess}");
ax1 = PyPlot.plot(tspan[1]+1:tspan[2], init_guess["Twhb_guess"][:,end],
                    label = L"T_{whb,guess}");
ax1 = PyPlot.plot(tspan[1]+1:tspan[2], T_dh_ret, label = L"T_{dh,Ret,given}");
PyPlot.ylabel(L"Temperature [$  $C]",fontsize=14)
PyPlot.legend();

PyPlot.subplot(322);
ax6 = PyPlot.step(tspan[1]:tspan[2]-1, Q_whb*Q_units, label = L"Q_{whb,given}");
```

```
ax6 = PyPlot.step(tspan[1]:tspan[2]-1, value.(opt_mod[:y]["Q_need",:])*Q_units,
                  label = L"Q_{need,given}");
PyPlot.ylim([0.0, 30]);
PyPlot.ylabel("Duty [MW]",fontsize=14)
PyPlot.legend();

PyPlot.subplot(323);
ax2 = PyPlot.step(tspan[1]+1:tspan[2], init_guess["qwhb_guess"]*q_units,
                  label = L"q_{whb,guess}");
ax2 = PyPlot.step(tspan[1]+1:tspan[2], init_guess["qA_guess"]*q_units,
                  label = L"q_{A,guess}");
ax2 = PyPlot.step(tspan[1]+1:tspan[2], init_guess["qB_guess"]*q_units,
                  label = L"q_{B,guess}");
PyPlot.ylabel("Flow [kg/s]",fontsize=14)
PyPlot.legend();

PyPlot.subplot(324);
ax3 = PyPlot.step(tspan[1]+1:tspan[2], init_guess["Qphb_guess"]*Q_units,
                  label = L"Q_{phb,guess}");
ax3 = PyPlot.step(tspan[1]+1:tspan[2], init_guess["Qdump_guess"]*Q_units,
                  label = L"Q_{dump,guess}");
ax3 = PyPlot.step(tspan[1]+1:tspan[2], init_guess["Qused_guess"]*Q_units,
                  label = L"Q_{wbh,used,guess}");
PyPlot.ylim([0.0, 30]);
PyPlot.ylabel("Duty [MW]",fontsize=14);
PyPlot.legend();

PyPlot.subplot(325);
ax4 = PyPlot.step(tspan[1]+1:tspan[2], init_guess["qbp_guess"]*q_units,
                  label = L"q_{bypass,guess}");
ax4 = PyPlot.step(tspan[1]+1:tspan[2], q_dh*q_units, label = L"q_{dh,given}");
PyPlot.xlabel("Time [hrs]",fontsize=14);
PyPlot.ylabel("Flow [kg/s]",fontsize=14);
PyPlot.legend();

PyPlot.subplot(326);
ax5 = PyPlot.plot(tspan[1]+1:tspan[2], init_guess["TA_guess"][:,end],
                  label = L"T_{A,guess}");
ax5 = PyPlot.plot(tspan[1]+1:tspan[2], init_guess["TB_guess"][:,end],
                  label = L"T_{B,guess}");
ax5 = PyPlot.plot(tspan[1]+1:tspan[2], init_guess["TC_guess"][:,end],
                  label = L"T_{C,guess}");
PyPlot.xlabel("Time [hrs]",fontsize=14);
PyPlot.ylabel(L"Temperature [$  $C]",fontsize=14);
PyPlot.legend();

fig2.show();
```

D.2 "main_cost.jl"

This script contains the implementation for running the optimal operations problem. The functions $read\_file()$, $tes\_create\_mod\_cost()$, $simulation\_cost()$ and $set\_bounds\_cost()$ are called, and the model is optimized.

```
# using Revise;
using MPCCLibrary;
using PyPlot, LaTeXStrings

include("tes_create_model.jl");
include("tes_sim.jl");
```

```
include("tes_bounds.jl");
include("read_file.jl");


default = Dict(
    :ρ_dh => 1000,                          #kg/m3
    :Cp_dh => 4.18,                         #kJ/(kg-K)
    :T_dh_minSup => 92.5,                   #minimum for supply
    :T_dh_maxSup => 95,                     #maximum for supply
    :T_whb_max => 120,
    :V_tes => 5000,                         #m3
    :Tlb => 40,
    :Tub => 120,
    :Q_used_ub => 22e3*3600,                # kJ/hr
    :q_whb_init => 291,                     # [m3/hr] Q_used_ub/rho*Cp*(120-55)
    :T_tes_init => 95,                      # Starting value T_tes
    #:nfe => length(Q_whb),
    :nfe => 720,
    :ncp => 1,
    :h => 1.0,
    :t_active => 3,                         # Months tank is expected to be active
    :S => ["T_tes"],
    :C => ["q_whb", "q_A", "q_B", "q_bp", "q_sys"],
    :A => ["T_A", "T_B", "T_C", "T_phb", "T_whb"],
    :A2 => ["Q_phb", "Q_whb_used", "Q_whb_dump", "Q_need"],
    :r => 0.05,                             # interest rate pr year
    :Q_phb_noTES => 876398422/1000*3600,    # kJ/hr ACTUAL DATA 30 days march 2019
    :USDtoNOK => 8.22,                          # NOK/USD
    :EURtoNOK => 10.03,                          # NOK/USD
    :C_peak => 0.9620440/3600,
    :C_dump => 0.6104700/3600
);

nfe = default[:nfe];
ncp = default[:ncp];
S = default[:S];
C = default[:C];
A = default[:A];
A2 = default[:A2];

Q_whb = [];
q_dh = [];
T_dh_ret = [];
T_dh_sup = [];

file1 = "15032019.csv";
file2 = "data_jan_18.csv";
file3 = "march19.csv";
Q_whb, q_dh, T_dh_ret, T_dh_sup, dates = read_file(file3, nfe);

initial_guesses = ["Ttes_guess","Tphb_guess","Twhb_guess","qwhb_guess","qA_guess",
                   "qB_guess","Qphb_guess","Qdump_guess","Qused_guess",
                   "qbp_guess","TA_guess","TB_guess","TC_guess"];

sim_mod = tes_create_mod_cost(Q_whb[1],q_dh[1],T_dh_ret[1],T_dh_sup[1],
                                printlevel=0);
opt_mod = tes_create_mod_cost(Q_whb,q_dh,T_dh_ret,T_dh_sup, printlevel=5);
opt_mod, guesses = simulation_cost(sim_mod,opt_mod,Q_whb,q_dh,T_dh_ret,T_dh_sup);
opt_mod = set_bounds_cost(opt_mod,Q_whb,q_dh,T_dh_ret,T_dh_sup);
init_guess = Dict(initial_guesses .=> guesses);

C_peak = default[:C_peak];
C_dump = default[:C_dump];

@NLobjective(opt_mod, Min, opt_mod[:N_0]
            + sum(C_dump*opt_mod[:y]["Q_whb_dump",i] for i in 1:nfe)
            + 1e-5*sum(opt_mod[:u]["q_bp",i] for i in 1:nfe)
            + 1e-7*sum(opt_mod[:u]["q_whb",i] for i in 1:nfe)
```

```
            );

optimize!(opt_mod);
println(termination_status(opt_mod));


######################
### PLOTTING
######################
Q_units = 1/3600/1000; # [kJ/hr] -> [MW]
q_units = 1/3600*1000; # [m3/hr] -> [kg/s]

tspan = (0.0, length(Q_whb));
tot_Q_phb = sum(value.(opt_mod[:y]["Q_phb",:])[i] for i in 1:nfe)*Q_units;
tot_Q_dump = sum(value.(opt_mod[:y]["Q_whb_dump",:])[i] for i in 1:nfe)*Q_units;

fig = figure(figsize = (12,12));

PyPlot.subplot(321);
ax1 = PyPlot.plot(tspan[1]:tspan[2], vcat(value.(opt_mod[:x0]["T_tes"]),
                    value.(opt_mod[:x]["T_tes",:,end])), label = L"T^{tes}");
ax1 = PyPlot.plot(tspan[1]+1:tspan[2], value.(opt_mod[:z]["T_phb",:,end]),
                                label = L"T^{phb}");
ax1 = PyPlot.plot(tspan[1]+1:tspan[2], value.(opt_mod[:z]["T_whb",:,end]),
                                label = L"T^{whb}");
ax1 = PyPlot.plot(tspan[1]+1:tspan[2], T_dh_ret, label = L"T^{dh,Ret}");
PyPlot.ylabel(L"Temperature [$  $C]",fontsize=14)
PyPlot.legend();

PyPlot.subplot(322);
ax6 = PyPlot.step(tspan[1]:tspan[2]-1, Q_whb*Q_units, label = L"Q^{whb}");
ax6 = PyPlot.step(tspan[1]:tspan[2]-1, value.(opt_mod[:y]["Q_need",:])*Q_units,
                                label = L"Q^{need}");
PyPlot.ylim([0.0, 30]);
PyPlot.ylabel("Duty [MW]",fontsize=14)
PyPlot.legend();

PyPlot.subplot(323);
ax2 = PyPlot.step(tspan[1]+1:tspan[2], value.(opt_mod[:u]["q_whb",:])*q_units,
                                label = L"q^{whb}");
ax2 = PyPlot.step(tspan[1]+1:tspan[2], value.(opt_mod[:u]["q_A",:])*q_units,
                                label = L"q^A");
ax2 = PyPlot.step(tspan[1]+1:tspan[2], value.(opt_mod[:u]["q_B",:])*q_units,
                                label = L"q^B");
PyPlot.ylabel("Flow [kg/s]",fontsize=14)
PyPlot.legend();

PyPlot.subplot(324);
ax3 = PyPlot.step(tspan[1]+1:tspan[2], value.(opt_mod[:y]["Q_phb",:])*Q_units,
                                label = L"Q^{phb}");
ax3 = PyPlot.step(tspan[1]+1:tspan[2], value.(opt_mod[:y]["Q_whb_dump",:])*Q_units,
                                label = L"Q^{dump}");
ax3 = PyPlot.step(tspan[1]+1:tspan[2], value.(opt_mod[:y]["Q_whb_used",:])*Q_units,
                                label = L"Q^{wbh,used}");
PyPlot.ylim([0.0, 30]);
PyPlot.ylabel("Duty [MW]",fontsize=14)
PyPlot.legend();

PyPlot.subplot(325);
ax4 = PyPlot.step(tspan[1]+1:tspan[2], value.(opt_mod[:u]["q_bp",:])*q_units,
                                label = L"q^{bypass}");
ax4 = PyPlot.step(tspan[1]+1:tspan[2], q_dh*q_units, label = L"q^{dh}");
PyPlot.xlabel("Time [hrs]",fontsize=14);
PyPlot.ylabel("Flow [kg/s]",fontsize=14)
PyPlot.legend();

PyPlot.subplot(326);
ax5 = PyPlot.plot(tspan[1]+1:tspan[2], value.(opt_mod[:z]["T_A",:,end]),
```

```
                                label = L"T^A");
ax5 = PyPlot.plot(tspan[1]+1:tspan[2], value.(opt_mod[:z]["T_B",:,end]),
                                label = L"T^B");
ax5 = PyPlot.plot(tspan[1]+1:tspan[2], value.(opt_mod[:z]["T_C",:,end]),
                                label = L"T^C");
PyPlot.xlabel("Time [hrs]",fontsize=14);
PyPlot.ylabel(L"Temperature [$  $C]",fontsize=14);
PyPlot.legend();
fig.show();

########################################
# Plotting initial guesses
########################################
fig2 = figure(figsize = (12,12));
PyPlot.subplot(321);
ax1 = PyPlot.plot(tspan[1]+1:tspan[2], init_guess["Ttes_guess"][:,end],
                                        label = L"T_{tes,guess}");
ax1 = PyPlot.plot(tspan[1]+1:tspan[2], init_guess["Tphb_guess"][:,end],
                                        label = L"T_{phb,guess}");
ax1 = PyPlot.plot(tspan[1]+1:tspan[2], init_guess["Twhb_guess"][:,end],
                                        label = L"T_{whb,guess}");
ax1 = PyPlot.plot(tspan[1]+1:tspan[2], T_dh_ret, label = L"T_{dh,Ret,given}");

PyPlot.ylabel(L"Temperature [$  $C]",fontsize=14)
PyPlot.legend();

PyPlot.subplot(322);
ax6 = PyPlot.step(tspan[1]:tspan[2]-1, Q_whb*Q_units, label = L"Q_{whb,given}");
ax6 = PyPlot.step(tspan[1]:tspan[2]-1, value.(opt_mod[:y]["Q_need",:])*Q_units,
                                        label = L"Q_{need,given}");

PyPlot.ylim([0.0, 30]);
PyPlot.ylabel("Duty [MW]",fontsize=14)
PyPlot.legend();

PyPlot.subplot(323);
ax2 = PyPlot.step(tspan[1]+1:tspan[2], init_guess["qwhb_guess"]*q_units,
                                label = L"q_{whb,guess}");
ax2 = PyPlot.step(tspan[1]+1:tspan[2], init_guess["qA_guess"]*q_units,
                                label = L"q_{A,guess}");
ax2 = PyPlot.step(tspan[1]+1:tspan[2], init_guess["qB_guess"]*q_units,
                                label = L"q_{B,guess}");
PyPlot.ylabel("Flow [kg/s]",fontsize=14)
PyPlot.legend();

PyPlot.subplot(324);
ax3 = PyPlot.step(tspan[1]+1:tspan[2], init_guess["Qphb_guess"]*Q_units,
                                label = L"Q_{phb,guess}");
ax3 = PyPlot.step(tspan[1]+1:tspan[2], init_guess["Qdump_guess"]*Q_units,
                                label = L"Q_{dump,guess}");
ax3 = PyPlot.step(tspan[1]+1:tspan[2], init_guess["Qused_guess"]*Q_units,
                                label = L"Q_{wbh,used,guess}");
PyPlot.ylim([0.0, 30]);
PyPlot.ylabel("Duty [MW]",fontsize=14);
PyPlot.legend();

PyPlot.subplot(325);
ax4 = PyPlot.step(tspan[1]+1:tspan[2], init_guess["qbp_guess"]*q_units,
                                label = L"q_{bypass,guess}");
ax4 = PyPlot.step(tspan[1]+1:tspan[2], q_dh*q_units, label = L"q_{dh,given}");
PyPlot.xlabel("Time [hrs]",fontsize=14);
PyPlot.ylabel("Flow [kg/s]",fontsize=14);
PyPlot.legend();

PyPlot.subplot(326);
ax5 = PyPlot.plot(tspan[1]+1:tspan[2], init_guess["TA_guess"][:,end],
                                label = L"T_{A,guess}");
ax5 = PyPlot.plot(tspan[1]+1:tspan[2], init_guess["TB_guess"][:,end],
```

```
                                label = L"T_{B,guess}");
ax5 = PyPlot.plot(tspan[1]+1:tspan[2], init_guess["TC_guess"][:,end],
                                label = L"T_{C,guess}");
PyPlot.xlabel("Time [hrs]",fontsize=14);
PyPlot.ylabel(L"Temperature [$  $C]",fontsize=14);
PyPlot.legend();

fig2.show();
```

D.3 "tes_create_model.jl"

The script $tes\_create\_model.jl$ contains two functions. The functions $tes\_create\_model()$ and $tes\_create\_mod\_cost()$ are similar. They take as input waste heat available, district flow, return and supply temperature and return a model for optimal operation and optimal sizing, respectively.

```
function tes_create_mod(Q_whb_in, q_dh_in, T_dh_ret_in, T_dh_sup_in; kwargs...)
    V_tes = get(kwargs, :V_tes, default[:V_tes]);
    ρ_dh = get(kwargs, :ρ_dh, default[:ρ_dh]);          #kg/m3
    Cp_dh = get(kwargs, :Cp_dh, default[:Cp_dh]);         #kJ/(kg-K)
    T_dh_minSup = get(kwargs, :T_dh_minSup, default[:T_dh_minSup]);
    T_dh_maxSup = get(kwargs, :T_dh_maxSup, default[:T_dh_maxSup]);
    T_whb_max = get(kwargs, :T_whb_max, default[:T_whb_max]);
    Tlb = get(kwargs, :Tlb, default[:Tlb]);
    Tub = get(kwargs, :Tub, default[:Tub]);
    T_tes_init = get(kwargs, :T_tes_init, default[:T_tes_init]);
    Q_used_ub = get(kwargs, :Q_used_ub, default[:Q_used_ub]);
    ncp = get(kwargs, :ncp, default[:ncp]);
    h = get(kwargs, :h, default[:h]);
    printlevel = get(kwargs, :printlevel, 0)

    nfe = length(Q_whb_in);

    ### Defining model
    model = create_model(linear_solver = "ma97", max_iter = 5000, print_level
                         = printlevel);

    ### Defining sets for variables
    S = get(kwargs, :S, default[:S]);
    C = get(kwargs, :C, default[:C]);
    A = get(kwargs, :A, default[:A]);
    A2 = get(kwargs, :A2, default[:A2]);

    adot = collocation_matrix(ncp,"Radau");

    ### Variables
    @variable(model, 0 <= x[S, 1:nfe, 1:ncp+1] <= Tub, start = 95);
    @variable(model, rk[S, 1:nfe, 2:ncp+1], start = 0);
    @variable(model, 0 <= u[C, 1:nfe] <= 5000, start = 0);
    @variable(model, 0 <= z[A, 1:nfe, 2:ncp+1] <= Tub, start = 95);
    @variable(model, 0 <= y[A2, 1:nfe]);
    @variable(model, x0[S]);

    @variable(model, 0 <= Q_whb[1:nfe]);
    @variable(model, 0 <= q_dh[1:nfe]);
    @variable(model, 0 <= T_dh_ret[1:nfe]);

    set_start_value.(x["T_tes",:,:], T_tes_init);
    set_start_value.(z["T_A",:,:], 55.0);
```

```
    @expression(model, ODE[k in S, i in 1:nfe, j in 2:ncp+1], u["q_A",i]/V_tes
                               *(z["T_A",i,j] - x["T_tes",i,j]) + u["q_B",i]
                               /V_tes*(z["T_B",i,j] - x["T_tes",i,j]));

    ### Model constraints
    # discrtized ODE model
    @constraint(model, rkConstr[k in S, i in 1:nfe, j in 2:ncp+1], 0 == rk[k,i,j]
                               - sum(x[k,i,l] * adot[l,j] for l in 1:ncp+1))

    @constraint(model, dODE[k in S, i in 1:nfe, j in 2:ncp+1], rk[k,i,j]
                               - h*ODE[k,i,j] == 0)
    # initial condition
    @constraint(model, initConstrs[k in S], x[k,1,1] - x0[k] == 0)
    # Closing shooting gap
    @constraint(model, gapConstrs[k in S, j in 1:nfe-1], x[k,j,end]
                               - x[k,j+1,1] == 0)

    # algebraic constraints (balances)
    @constraint(model, split[i in 1:nfe], 0 == u["q_bp",i]+u["q_sys",i]-q_dh[i]);
    @constraint(model, mass[i in 1:nfe],  0 == u["q_whb",i] + u["q_A",i]
                               - (u["q_sys",i] + u["q_B",i]));
    @constraint(model, energ1[i in 1:nfe, j in 2:ncp+1], 0 == u["q_sys",i]
                               *T_dh_ret[i] + u["q_B",i]*x["T_tes",i,j]
                               - z["T_A",i,j]*(u["q_whb",i] + u["q_A",i]));
    @constraint(model, energ2[i in 1:nfe, j in 2:ncp+1], 0 == (u["q_whb",i]
                               *z["T_whb",i,j] + u["q_A",i]*x["T_tes",i,j])
                               - z["T_B",i,j]*(u["q_sys",i] + u["q_B",i]));
    @constraint(model, energ3[i in 1:nfe, j in 2:ncp+1], 0 == u["q_sys",i]
                               *z["T_B",i,j] + u["q_bp",i]*T_dh_ret[i]
                               - z["T_C",i,j]*q_dh[i]);
    @constraint(model, excess[i in 1:nfe], 0 == Q_whb[i] - (y["Q_whb_used",i]
                               + y["Q_whb_dump",i]));

    @constraint(model, WHB[i in 1:nfe, j in 2:ncp+1], 0 == y["Q_whb_used",i]
                        - ρ_dh*Cp_dh*u["q_whb",i]*(z["T_whb",i,j] - z["T_A",i,j]));
    @constraint(model, PHB[i in 1:nfe, j in 2:ncp+1], 0 == y["Q_phb",i]
                        - ρ_dh*Cp_dh*q_dh[i]*(z["T_phb",i,j] - z["T_C",i,j]));
    @constraint(model, demand[i in 1:nfe], 0 == y["Q_need",i] - q_dh[i]
                        *Cp_dh*ρ_dh*(T_dh_sup_in[i]-T_dh_ret[i]))

    return model
end

##############################
# COST MODEL
##############################

function tes_create_mod_cost(Q_whb_in, q_dh_in, T_dh_ret_in,T_dh_sup_in; kwargs...)
    V_tes_init = get(kwargs, :V_tes, default[:V_tes]);
    ρ_dh = get(kwargs, :ρ_dh, default[:ρ_dh]);          #kg/m3
    Cp_dh = get(kwargs, :Cp_dh, default[:Cp_dh]);         #kJ/(kg-K)
    T_dh_minSup = get(kwargs, :T_dh_minSup, default[:T_dh_minSup]);
    T_dh_maxSup = get(kwargs, :T_dh_maxSup, default[:T_dh_maxSup]);
    T_whb_max = get(kwargs, :T_whb_max, default[:T_whb_max]);
    Tlb = get(kwargs, :Tlb, default[:Tlb]);
    Tub = get(kwargs, :Tub, default[:Tub]);
    T_tes_init = get(kwargs, :T_tes_init, default[:T_tes_init]);
    Q_used_ub = get(kwargs, :Q_used_ub, default[:Q_used_ub]);
    ncp = get(kwargs, :ncp, default[:ncp]);
    h = get(kwargs, :h, default[:h]);
    printlevel = get(kwargs, :printlevel, 0);
    tolerance = get(kwargs, :tolerance, 1e-8);
    r = get(kwargs, :r, default[:r]);
    Q_phb_noTES = get(kwargs, :Q_phb_noTES, default[:Q_phb_noTES]);
    C_peak = get(kwargs, :C_peak, default[:C_peak]);
    EURtoNOK = get(kwargs, :EURtoNOK, default[:EURtoNOK]);
    USDtoNOK = get(kwargs, :USDtoNOK, default[:USDtoNOK]);
```

```
    t_active = get(kwargs, :t_active, default[:t_active]);

    nfe = length(Q_whb_in);

    ### Defining model
    model = create_model(linear_solver = "ma97", max_iter = 5000, print_level
                            = printlevel, tol = tolerance);

    ### Defining sets for variables
    S = get(kwargs, :S, default[:S]);           # state variables
    C = get(kwargs, :C, default[:C]);           # control variables
    A = get(kwargs, :A, default[:A]);           # algebraic variables
    A2 = get(kwargs, :A2, default[:A2]);        # algebraic variables

    adot = collocation_matrix(ncp,"Radau");

    ### Variables
    @variable(model, 0 <= x[S, 1:nfe, 1:ncp+1] <= Tub, start = 95);
    @variable(model, rk[S, 1:nfe, 2:ncp+1], start = 0);
    @variable(model, 0 <= u[C, 1:nfe] <= 5000, start = 0);
    @variable(model, 0 <= z[A, 1:nfe, 2:ncp+1] <= Tub, start = 95);
    @variable(model, 0 <= y[A2, 1:nfe]);
    @variable(model, x0[S]);

    @variable(model, 0 <= Q_whb[1:nfe]);
    @variable(model, 0 <= q_dh[1:nfe]);
    @variable(model, 0 <= T_dh_ret[1:nfe]);

    @variable(model, 0 <= V_tes, start = V_tes_init);

    @variable(model, 0 <= N_0, start = 10);
    @variable(model, 0 <= B, start = 350000);

    set_start_value.(x["T_tes",:,:], T_tes_init);
    set_start_value.(z["T_A",:,:], 55.0);

    @expression(model, ODE[k in S, i in 1:nfe, j in 2:ncp+1], u["q_A",i]
                                *(z["T_A",i,j] - x["T_tes",i,j]) + u["q_B",i]
                                *(z["T_B",i,j] - x["T_tes",i,j]));


    ### Model constraints
    # discrtized ODE model
    @constraint(model, rkConstr[k in S, i in 1:nfe, j in 2:ncp+1], 0
                                == rk[k,i,j] - sum(x[k,i,l] * adot[l,j]
                                for l in 1:ncp+1))

    @constraint(model, dODE[k in S, i in 1:nfe, j in 2:ncp+1], rk[k,i,j]
                                *V_tes - h*ODE[k,i,j] == 0)
    # initial condition
    @constraint(model, initConstrs[k in S], x[k,1,1] - x0[k] == 0)
    # Closing shooting gap
    @constraint(model, gapConstrs[k in S, j in 1:nfe-1], x[k,j,end] - x[k,j+1,1] ==
0)

    # algebraic constraints (balances)
    @constraint(model, split[i in 1:nfe], 0 == u["q_bp",i]+u["q_sys",i]-q_dh[i]);
    @constraint(model, mass[i in 1:nfe],  0 == u["q_whb",i] + u["q_A",i]
                                - (u["q_sys",i] + u["q_B",i]));
    @constraint(model, energ1[i in 1:nfe, j in 2:ncp+1], 0 == u["q_sys",i]
                                *T_dh_ret[i] + u["q_B",i]*x["T_tes",i,j]
                                - z["T_A",i,j]*(u["q_whb",i] + u["q_A",i]));
    @constraint(model, energ2[i in 1:nfe, j in 2:ncp+1], 0 == (u["q_whb",i]
                                *z["T_whb",i,j] + u["q_A",i]*x["T_tes",i,j])
                                - z["T_B",i,j]*(u["q_sys",i] + u["q_B",i]));
    @constraint(model, energ3[i in 1:nfe, j in 2:ncp+1], 0 == u["q_sys",i]
                                *z["T_B",i,j] + u["q_bp",i]*T_dh_ret[i]
                                - z["T_C",i,j]*q_dh[i]);
```

```
    @constraint(model, excess[i in 1:nfe], 0 == Q_whb[i] - (y["Q_whb_used",i]
                                + y["Q_whb_dump",i]));

    @constraint(model, WHB[i in 1:nfe, j in 2:ncp+1], 0 == y["Q_whb_used",i]
                            - ρ_dh*Cp_dh*u["q_whb",i]*(z["T_whb",i,j] - z["T_A",i,j]));
    @constraint(model, PHB[i in 1:nfe, j in 2:ncp+1], 0 == y["Q_phb",i]
                            - ρ_dh*Cp_dh*q_dh[i]*(z["T_phb",i,j] - z["T_C",i,j]));
    @constraint(model, demand[i in 1:nfe], 0 == y["Q_need",i] - q_dh[i]
                            *Cp_dh*ρ_dh*(T_dh_sup_in[i]-T_dh_ret[i]));

    @NLexpression(model, tank_cost, (0.0047*V_tes^0.6218)*1e6*EURtoNOK);

    @NLconstraint(model, payback, 0 == log(1+r)*N_0 - (log(t_active*B)
                            - log(t_active*B-tank_cost*r)));
    @constraint(model, savings, 0 == B - C_peak*(Q_phb_noTES - sum(y["Q_phb",i]
                            for i in 1:nfe )));

    return model
end
```

D.4 "tes_bounds.jl"

The script $tes\_bounds.jl$ contains two functions. The functions $set\_bounds()$ and $set\_bounds\_cost()$ are similar. They take as input a model, the optimal operations model or the optimal design model, respectively, waste heat available, district flow, return and supply temperature and return the model updated with applicable bounds for optimal operation or optimal design.

```
function set_bounds(model, Q_whb_in, q_dh_in, T_dh_ret_in, T_dh_sup_in; kwargs...)
    Q_used_ub = get(kwargs, :Q_used_ub, default[:Q_used_ub]);
    T_dh_maxSup = get(kwargs, :T_dh_maxSup, default[:T_dh_maxSup]);
    ρ_dh = get(kwargs, :ρ_dh, default[:ρ_dh]);
    Cp_dh = get(kwargs, :Cp_dh, default[:Cp_dh]);
    T_dh_minSup = get(kwargs, :T_dh_minSup, default[:T_dh_minSup]);
    T_dh_maxSup = get(kwargs, :T_dh_maxSup, default[:T_dh_maxSup]);
    T_whb_max = get(kwargs, :T_whb_max, default[:T_whb_max]);
    Tlb = get(kwargs, :Tlb, default[:Tlb]);
    Tub = get(kwargs, :Tub, default[:Tub]);
    T_tes_init = get(kwargs, :T_tes_init, default[:T_tes_init]);

    fix.(model[:Q_whb][:],Q_whb_in[:], force=true);
    fix.(model[:q_dh][:],q_dh_in[:], force=true);
    fix.(model[:T_dh_ret][:],T_dh_ret_in[:], force=true);

    fix.(model[:x0], T_tes_init, force=true);

    set_upper_bound.(model[:x]["T_tes",:,:], Tub);
    set_lower_bound.(model[:x]["T_tes",:,:], Tlb);
    set_upper_bound.(model[:z][:,:,:], Tub);
    set_lower_bound.(model[:z][:,:,:], Tlb);

    set_lower_bound.(model[:z]["T_phb",:,:], T_dh_minSup);
    set_upper_bound.(model[:z]["T_phb",:,:], T_dh_maxSup);
    set_upper_bound.(model[:z]["T_whb",:,:], T_whb_max);

    set_upper_bound.(model[:u]["q_whb",:], 1200);
    set_upper_bound.(model[:y]["Q_whb_used",:], Q_used_ub);


    for i in 1:length(Q_whb)
```

```
        excess = Q_whb[i] - ρ_dh*Cp_dh*q_dh[i]*(T_dh_sup_in[i]-T_dh_ret[i]);
        if excess > 0
            fix.(model[:u]["q_A",i], 0., force=true);
        else
            fix.(model[:u]["q_B",i], 0., force=true);
        end
    end

    for i in 1:length(Q_whb)
        set_upper_bound.(model[:u]["q_sys",i], q_dh[i]);
        set_upper_bound.(model[:u]["q_bp",i], q_dh[i]);
        set_upper_bound.(model[:y]["Q_whb_dump",i], Q_whb[i]);

        fix.(model[:z]["T_phb",i,:],T_dh_sup_in[i], force=true)
    end
    return model
end



function set_bounds_cost(model, Q_whb_in, q_dh_in, T_dh_ret_in, T_dh_sup_in; kwargs...)
    Q_used_ub = get(kwargs, :Q_used_ub, default[:Q_used_ub]);
    T_dh_maxSup = get(kwargs, :T_dh_maxSup, default[:T_dh_maxSup]);
    ρ_dh = get(kwargs, :ρ_dh, default[:ρ_dh]);
    Cp_dh = get(kwargs, :Cp_dh, default[:Cp_dh]);
    T_dh_maxSup = get(kwargs, :T_dh_maxSup, default[:T_dh_maxSup]);
    T_whb_max = get(kwargs, :T_whb_max, default[:T_whb_max]);
    Tlb = get(kwargs, :Tlb, default[:Tlb]);
    Tub = get(kwargs, :Tub, default[:Tub]);
    T_tes_init = get(kwargs, :T_tes_init, default[:T_tes_init]);
    fix_tes= get(kwargs, :fix_tes, true);

    fix.(model[:Q_whb][:],Q_whb_in[:], force=true);
    fix.(model[:q_dh][:],q_dh_in[:], force=true);
    fix.(model[:T_dh_ret][:],T_dh_ret_in[:], force=true);

    fix.(model[:x0], T_tes_init, force=true);

    set_upper_bound.(model[:V_tes], 50000);
    #fix.(model[:V_tes], 6322.9517512*1.4, force = true)
    #fix.(model[:V_tes], 5000, force = true)

    set_lower_bound.(model[:B], 100);
    set_upper_bound.(model[:x]["T_tes",:,:], Tub);
    set_lower_bound.(model[:x]["T_tes",:,:], Tlb);
    set_upper_bound.(model[:z][:,:,:], Tub);
    set_lower_bound.(model[:z][:,:,:], Tlb);

    set_upper_bound.(model[:z]["T_whb",:,:], T_whb_max);

    set_upper_bound.(model[:u]["q_whb",:], 1200);
    set_upper_bound.(model[:y]["Q_whb_used",:], Q_used_ub);

    if fix_tes
        for i in 1:length(Q_whb)
            excess = Q_whb[i] - ρ_dh*Cp_dh*q_dh[i]*(T_dh_sup_in[i]-T_dh_ret[i]);
            if excess > 0
                fix.(model[:u]["q_A",i], 0., force=true);
            else
                fix.(model[:u]["q_B",i], 0., force=true);
            end
        end
    end

    for i in 1:length(Q_whb)
        set_upper_bound.(model[:u]["q_sys",i], q_dh[i]);
        set_upper_bound.(model[:u]["q_bp",i], q_dh[i]);
        set_upper_bound.(model[:y]["Q_whb_dump",i], Q_whb[i]);
```

```
        fix.(model[:z]["T_phb",i,:],T_dh_sup_in[i], force=true)
    end
    return model
end
```

D.5 "tes_sim.jl"

The script $tes\_sim.jl$ contains two functions. The functions $simulation()$ and $simulation\_cost()$ are similar. They take as input the model to be simulated and the model to be optimized, and also waste heat available, district flow, return and supply temperature. The function simulates the simulation model at each finite element, and add the results as initial guesses to the optimization model. The optimization model and the initial guesses are returned.

```
function simulation(sim_mod, opt_mod, Q_whb, q_dh, T_dh_ret, T_dh_sup; kwargs...)
    T_dh_maxSup = get(kwargs, :T_dh_maxSup, default[:T_dh_maxSup]);
    ρ_dh = get(kwargs, :ρ_dh, default[:ρ_dh]);
    Cp_dh = get(kwargs, :Cp_dh, default[:Cp_dh]);
    nfe = get(kwargs, :nfe, default[:nfe]);
    ncp = get(kwargs, :ncp, default[:ncp]);
    S = get(kwargs, :S, default[:S]);
    C = get(kwargs, :C, default[:C]);
    A = get(kwargs, :A, default[:A]);
    A2 = get(kwargs, :A2, default[:A2]);
    x0 = get(kwargs, :x0, default[:T_tes_init]);

    Ttes_guess = Array{Float64,1}(undef,nfe);
    Tphb_guess = Array{Float64,1}(undef,nfe);
    Twhb_guess = Array{Float64,1}(undef,nfe);
    qwhb_guess = Array{Float64,1}(undef,nfe);
    qA_guess = Array{Float64,1}(undef,nfe);
    qB_guess = Array{Float64,1}(undef,nfe);
    Qphb_guess = Array{Float64,1}(undef,nfe);
    Qdump_guess = Array{Float64,1}(undef,nfe);
    Qused_guess = Array{Float64,1}(undef,nfe);
    qbp_guess = Array{Float64,1}(undef,nfe);
    TA_guess = Array{Float64,1}(undef,nfe);
    TB_guess = Array{Float64,1}(undef,nfe);
    TC_guess = Array{Float64,1}(undef,nfe);

    # Simulation loop
    for i in 1:nfe
        excess = Q_whb[i] - ρ_dh*Cp_dh*q_dh[i]*(T_dh_sup[i]-T_dh_ret[i]);

        set_start_value(sim_mod[:y]["Q_need",1], ρ_dh*Cp_dh*q_dh[i]*(T_dh_sup[i]
                                    -T_dh_ret[i]))

        fix.(sim_mod[:z]["T_phb",1,end], T_dh_sup[i], force=true);
        fix.(sim_mod[:x0], x0, force=true);

        # Updating input
        fix.(sim_mod[:Q_whb],Q_whb[i], force=true)
        fix.(sim_mod[:q_dh],q_dh[i], force=true)
        fix.(sim_mod[:T_dh_ret],T_dh_ret[i], force=true)

        if excess < 0
            if is_fixed(sim_mod[:u]["q_A",1])
                unfix(sim_mod[:u]["q_A",1])
                set_lower_bound(sim_mod[:u]["q_A",1], 0.0)
            end
```

```
            if is_fixed(sim_mod[:y]["Q_phb",1])
                unfix(sim_mod[:y]["Q_phb",1])
                set_lower_bound(sim_mod[:y]["Q_phb",1], 0.0)
            end
            fix.(sim_mod[:u]["q_B",1], 0, force = true)
            fix.(sim_mod[:y]["Q_whb_dump",1], 0, force = true)
            set_start_value(sim_mod[:u]["q_sys",1], q_dh[i])
            if x0 > T_dh_maxSup
                @objective(sim_mod, Min, sim_mod[:y]["Q_phb",1]^2
                                + sim_mod[:u]["q_bp",1])
            else
                set_start_value(sim_mod[:u]["q_whb",1], q_dh[i])
                @objective(sim_mod, Min, sim_mod[:u]["q_A",1]^2
                                + sim_mod[:u]["q_bp",1])
            end
        else
            if is_fixed(sim_mod[:u]["q_B",1])
                unfix(sim_mod[:u]["q_B",1])
                set_lower_bound(sim_mod[:u]["q_B",1], 0.0)
            end
            if is_fixed(sim_mod[:y]["Q_whb_dump",1])
                unfix(sim_mod[:y]["Q_whb_dump",1])
                set_lower_bound(sim_mod[:y]["Q_whb_dump",1], 0.0)
            end
            if is_fixed(sim_mod[:u]["q_bp",1])
                unfix(sim_mod[:u]["q_bp",1])
                set_lower_bound(sim_mod[:u]["q_bp",1], 0.0)
            end
            fix(sim_mod[:u]["q_A",1], 0, force = true)
            fix.(sim_mod[:y]["Q_phb",1], 0, force = true)
            @objective(sim_mod, Min, sim_mod[:y]["Q_whb_dump",1]^2)
        end

        fix.(sim_mod[:z]["T_phb",1,end], T_dh_sup[i], force = true)

        optimize!(sim_mod);

        x0 = value.(sim_mod[:x]["T_tes",1,end]);
        println(termination_status(sim_mod));
        if termination_status(sim_mod) != MOI.LOCALLY_SOLVED
            println(excess)
            println(value.(all_variables(sim_mod)))
        end

        Ttes_guess[i] = value.(sim_mod[:x]["T_tes",1,end]);
        Tphb_guess[i] = value.(sim_mod[:z]["T_phb",1,end]);
        Twhb_guess[i] = value.(sim_mod[:z]["T_whb",1,end]);
        qwhb_guess[i] = value.(sim_mod[:u]["q_whb",1]);
        qA_guess[i] = value.(sim_mod[:u]["q_A",1]);
        qB_guess[i] = value.(sim_mod[:u]["q_B",1]);
        Qphb_guess[i] = value.(sim_mod[:y]["Q_phb",1]);
        Qdump_guess[i] = value.(sim_mod[:y]["Q_whb_dump",1]);
        Qused_guess[i] = value.(sim_mod[:y]["Q_whb_used",1]);
        qbp_guess[i] = value.(sim_mod[:u]["q_bp",1]);
        TA_guess[i] = value.(sim_mod[:z]["T_A",1,end]);
        TB_guess[i] = value.(sim_mod[:z]["T_B",1,end]);
        TC_guess[i] = value.(sim_mod[:z]["T_C",1,end]);

        for j in S
            set_start_value.(opt_mod[:x][j,i,1], value.(sim_mod[:x][j,1,1]));
            set_start_value.(opt_mod[:x][j,i,end], value.(sim_mod[:x][j,1,end]));
            set_start_value.(opt_mod[:rk][j,i,:], value.(sim_mod[:rk][j,1,end]));
        end
        for j in C
            set_start_value.(opt_mod[:u][j,i], value.(sim_mod[:u][j,end]));
        end
        for j in A
            set_start_value.(opt_mod[:z][j,i,:], value.(sim_mod[:z][j,1,end]));
```

```
        end
        for j in A2
            set_start_value.(opt_mod[:y][j,i], value.(sim_mod[:y][j,end]));
        end
        fix.(sim_mod[:x0], value.(sim_mod[:x]["T_tes",1,end]), force=true);
    end
    guesses = [Ttes_guess,Tphb_guess,Twhb_guess,qwhb_guess,qA_guess,qB_guess,
                    Qphb_guess,Qdump_guess,Qused_guess,
                    qbp_guess,TA_guess,TB_guess,TC_guess];
    return opt_mod, guesses
end


#################################
# Cost simulation
#################################

function simulation_cost(sim_mod, opt_mod, Q_whb, q_dh, T_dh_ret, T_dh_sup; kwargs...)
    T_dh_maxSup = get(kwargs, :T_dh_maxSup, default[:T_dh_maxSup]);
    ρ_dh = get(kwargs, :ρ_dh, default[:ρ_dh]);
    Cp_dh = get(kwargs, :Cp_dh, default[:Cp_dh]);
    nfe = get(kwargs, :nfe, default[:nfe]);
    ncp = get(kwargs, :ncp, default[:ncp]);
    S = get(kwargs, :S, default[:S]);
    C = get(kwargs, :C, default[:C]);
    A = get(kwargs, :A, default[:A]);
    A2 = get(kwargs, :A2, default[:A2]);
    x0 = get(kwargs, :x0, default[:T_tes_init]);
    V_tes_init = get(kwargs, :V_tes, default[:V_tes]);

    Ttes_guess = Array{Float64,1}(undef,nfe);
    Tphb_guess = Array{Float64,1}(undef,nfe);
    Twhb_guess = Array{Float64,1}(undef,nfe);
    qwhb_guess = Array{Float64,1}(undef,nfe);
    qA_guess = Array{Float64,1}(undef,nfe);
    qB_guess = Array{Float64,1}(undef,nfe);
    Qphb_guess = Array{Float64,1}(undef,nfe);
    Qdump_guess = Array{Float64,1}(undef,nfe);
    Qused_guess = Array{Float64,1}(undef,nfe);
    qbp_guess = Array{Float64,1}(undef,nfe);
    TA_guess = Array{Float64,1}(undef,nfe);
    TB_guess = Array{Float64,1}(undef,nfe);
    TC_guess = Array{Float64,1}(undef,nfe);

    # Simulation loop
    for i in 1:nfe
        excess = Q_whb[i] - ρ_dh*Cp_dh*q_dh[i]*(T_dh_sup[i]-T_dh_ret[i]);

        set_start_value(sim_mod[:y]["Q_need",1], ρ_dh*Cp_dh*q_dh[i]*(T_dh_sup[i]
                                    -T_dh_ret[i]))

        fix.(sim_mod[:z]["T_phb",1,end], T_dh_sup[i], force=true);
        fix.(sim_mod[:x0], x0, force=true);
        fix.(sim_mod[:V_tes],V_tes_init, force=true);

        # Updating input
        fix.(sim_mod[:Q_whb],Q_whb[i], force=true)
        fix.(sim_mod[:q_dh],q_dh[i], force=true)
        fix.(sim_mod[:T_dh_ret],T_dh_ret[i], force=true)

        if excess < 0
            if is_fixed(sim_mod[:u]["q_A",1])
                unfix(sim_mod[:u]["q_A",1])
                set_lower_bound(sim_mod[:u]["q_A",1], 0.0)
            end
            if is_fixed(sim_mod[:y]["Q_phb",1])
                unfix(sim_mod[:y]["Q_phb",1])
                set_lower_bound(sim_mod[:y]["Q_phb",1], 0.0)
```

```
            end
            fix.(sim_mod[:u]["q_B",1], 0, force = true)
            fix.(sim_mod[:y]["Q_whb_dump",1], 0, force = true)
            set_start_value(sim_mod[:u]["q_sys",1], q_dh[i])
            if x0 > T_dh_maxSup
                @objective(sim_mod, Min, sim_mod[:y]["Q_phb",1]^2
                                + sim_mod[:u]["q_bp",1])
            else
                set_start_value(sim_mod[:u]["q_whb",1], q_dh[i])
                @objective(sim_mod, Min, sim_mod[:u]["q_A",1]^2
                                + sim_mod[:u]["q_bp",1])
            end
        else
            if is_fixed(sim_mod[:u]["q_B",1])
                unfix(sim_mod[:u]["q_B",1])
                set_lower_bound(sim_mod[:u]["q_B",1], 0.0)
            end
            if is_fixed(sim_mod[:y]["Q_whb_dump",1])
                unfix(sim_mod[:y]["Q_whb_dump",1])
                set_lower_bound(sim_mod[:y]["Q_whb_dump",1], 0.0)
            end
            if is_fixed(sim_mod[:u]["q_bp",1])
                unfix(sim_mod[:u]["q_bp",1])
                set_lower_bound(sim_mod[:u]["q_bp",1], 0.0)
            end
            fix(sim_mod[:u]["q_A",1], 0, force = true)
            fix.(sim_mod[:y]["Q_phb",1], 0, force = true)

            @objective(sim_mod, Min, sim_mod[:y]["Q_whb_dump",1]^2)
        end

        fix.(sim_mod[:z]["T_phb",1,end], T_dh_sup[i], force = true)

        optimize!(sim_mod);

        x0 = value.(sim_mod[:x]["T_tes",1,end]);
        println(termination_status(sim_mod));
        if termination_status(sim_mod) != MOI.LOCALLY_SOLVED
            println(excess)
            println(value.(all_variables(sim_mod)))
        end

        Ttes_guess[i] = value.(sim_mod[:x]["T_tes",1,end]);
        Tphb_guess[i] = value.(sim_mod[:z]["T_phb",1,end]);
        Twhb_guess[i] = value.(sim_mod[:z]["T_whb",1,end]);
        qwhb_guess[i] = value.(sim_mod[:u]["q_whb",1]);
        qA_guess[i] = value.(sim_mod[:u]["q_A",1]);
        qB_guess[i] = value.(sim_mod[:u]["q_B",1]);
        Qphb_guess[i] = value.(sim_mod[:y]["Q_phb",1]);
        Qdump_guess[i] = value.(sim_mod[:y]["Q_whb_dump",1]);
        Qused_guess[i] = value.(sim_mod[:y]["Q_whb_used",1]);
        qbp_guess[i] = value.(sim_mod[:u]["q_bp",1]);
        TA_guess[i] = value.(sim_mod[:z]["T_A",1,end]);
        TB_guess[i] = value.(sim_mod[:z]["T_B",1,end]);
        TC_guess[i] = value.(sim_mod[:z]["T_C",1,end]);

        for j in S
            set_start_value.(opt_mod[:x][j,i,1], value.(sim_mod[:x][j,1,1]));
            set_start_value.(opt_mod[:x][j,i,end], value.(sim_mod[:x][j,1,end]));
            set_start_value.(opt_mod[:rk][j,i,:], value.(sim_mod[:rk][j,1,end]));
        end
        for j in C
            set_start_value.(opt_mod[:u][j,i], value.(sim_mod[:u][j,end]));
        end
        for j in A
            set_start_value.(opt_mod[:z][j,i,:], value.(sim_mod[:z][j,1,end]));
        end
        for j in A2
```

```
            set_start_value.(opt_mod[:y][j,i], value.(sim_mod[:y][j,end]));
        end
        set_start_value.(all_variables(sim_mod), value.(all_variables(sim_mod)));
        fix.(sim_mod[:x0], value.(sim_mod[:x]["T_tes",1,end]), force=true);
    end
    guesses = [Ttes_guess,Tphb_guess,Twhb_guess,qwhb_guess,qA_guess,qB_guess,
                    Qphb_guess,Qdump_guess,Qused_guess,
                    qbp_guess,TA_guess,TB_guess,TC_guess];
    return opt_mod, guesses
end
```

D.6 "read_file.jl"

The script $read\_file.jl$ contains one function, $read\_file()$, taking as input the filename as a string and the number of finite elements. The function returns arrays of available waste heat, district flow, return temperature, supply temperature and the dates when the heat data was obtained.

```
using CSV;
using MPCCLibrary;

function read_file(filename, nfe; kwargs...)
    T_dh_maxSup = get(kwargs, :T_dh_maxSup, default[:T_dh_maxSup]);
    ρ_dh = get(kwargs, :ρ_dh, default[:ρ_dh]);          # kg/m3
    Cp_dh = get(kwargs, :Cp_dh, default[:Cp_dh]);       # kJ/(kg-K)
    file = CSV.read(filename);

    Q_whb_file = [];
    Q_need_file = [];
    q_dh_file = [];
    T_ret_file = [];
    T_sup_file = [];

    for row in eachrow(file)[3:nfe+2]
        Q_whb_iter = parse(Float64,row.Column9)+parse(Float64,row.Column10)
                +parse(Float64,row.Column11)+parse(Float64,row.Column12);
        Q_need_iter = parse(Float64,row.Heat_flow_rates)+parse(Float64,row.Column7)
                +parse(Float64,row.Column8)+parse(Float64,row.Column9)
                +parse(Float64,row.Column11);
        q_iter = (Q_need_iter/1000*3600)/(ρ_dh*Cp_dh*(parse(Float64,row.Column17)
                -parse(Float64,row.Column15)));

        append!(Q_whb_file,Q_whb_iter/1000*3600);
        append!(Q_need_file,Q_need_iter/1000*3600)
        append!(q_dh_file,q_iter);
        append!(T_ret_file,parse(Float64,row.Column15));
        append!(T_sup_file,parse(Float64,row.Column17));
    end
    return Q_whb_file, q_dh_file, T_ret_file, T_sup_file, file[3:nfe+2,1]
end
```

D.7 "tes_logic.jl"

The file *tes_logic.jl* was the initial file for the optimal operations problem, used for optimizing the shorter time horizon using NLP in Equation 4.1.7. The model with bounds and same initial guesses for all finite elements is made and optimized.

```
include("read_file.jl");
using PyPlot, LaTeXStrings;

### Parameters

# Heat Parameters
ρ_dh = 1000          #kg/m3
Cp_dh = 4.18         #kJ/(kg-K)

# Temperature
T_dh_ret = 55          #return, btw 40-70 depending on season
T_dh_minSup = 92.5     #minimum for supply
T_dh_maxSup = 95       #maximum for supply
T_whb_max = 120

T_tes_init = 95;

# Capacity
V_tes = 5000;          #m3

# Bounds
Tlb = 40;
Tub = 120;
Q_used_ub = 22e3*3600;           # kJ/hr
q_whb_ub = 291.0;         # m3/hr          q = Q/(ρ_dh*Cp_dh*(95-55))

### Input
#nfe=24;
file1 = "15032019.csv";
file3 = "march19.csv";
#Q_whb, q_dh, T_dh_ret, T_dh_sup = read_file(file3,nfe);

Q_whb = vcat(1.0*ones(10,1), 1.2*ones(10,1), 0.8*ones(10,1)) *1.672e5*q_whb_ub
q_dh = vcat(1.0*ones(10,1), 1.0*ones(10,1), 1.0*ones(10,1)) *291;
T_dh_ret = vcat(1.0*ones(10,1), 1.0*ones(10,1), 1.0*ones(10,1)) *55;
T_dh_sup = vcat(1.0*ones(10,1), 1.0*ones(10,1), 1.0*ones(10,1)) *95;


### Defining sets for variables
S = ["T_tes"];
C = ["q_whb", "q_A", "q_B", "Q_phb", "q_bp", "q_sys"];
A = ["T_A", "T_B", "T_C"];
A2 = ["T_phb", "T_whb"];

### Discretization
ncp = 1;
tspan = (0.0, 30);
h = 1.0
nfe = Int32((tspan[2] - tspan[1])/h);
adot = collocation_matrix(ncp,"Radau");

### Defining model
model = create_model(linear_solver = "ma97", max_iter = 5000);

### Variables
@variable(model, Tlb <= x[S, 1:nfe, 1:ncp+1] <= Tub, start = 95);
@variable(model, Tlb <= y[A2, 1:nfe, 2:ncp+1] <= Tub, start = 95);
@variable(model, rk[S, 1:nfe, 2:ncp+1], start = 0);
```

```
@variable(model, u[C, 1:nfe] >= 0, start = 0);
@variable(model, Tlb <= z[A, 1:nfe, 2:ncp+1] <= Tub, start = 95);
@variable(model, x0[S]);

@variable(model, Q_used_ub >= Q_whb_used[i in 1:nfe] >= 0, start = Q_whb[i]);
@variable(model, Q_whb[i] >= Q_dump[i in 1:nfe] >= 0, start = 0);



set_lower_bound.(y["T_phb",:,:], T_dh_minSup);
set_upper_bound.(y["T_phb",:,:], T_dh_maxSup);
set_upper_bound.(y["T_whb",:,:], T_whb_max);
fix.(y["T_phb",:,:], 95, force=true);

set_upper_bound.(u["q_bp",:], 291.0);
set_upper_bound.(u["q_sys",:], 291.0);

set_upper_bound.(u["q_whb",:], 1200);
fix.(x0, T_tes_init, force=true);
set_start_value.(x["T_tes",:,:], 55.0);
set_start_value.(z["T_A",:,:], 55.0);
set_start_value.(u["q_bp",:], 0.0);
set_start_value.(u["q_whb",:], q_whb_ub);

set_start_value.(u["Q_phb",:], 0);


### Mathematical model
tmp = Dict();
tmp["T_tes"] = @NLexpression(model, [i in 1:nfe, j in 2:ncp+1], u["q_A",i]/V_tes
                                    *(z["T_A",i,j] - x["T_tes",i,j])
                                    + u["q_B",i]/V_tes*(z["T_B",i,j]
                                    - x["T_tes",i,j]));


@NLexpression(model, ODE[k in ["T_tes"], i in 1:nfe, j in 2:ncp+1], tmp[k][i,j]);

### Model constraints
# discretized ODE model
@NLconstraint(model, rkConstr[k in ["T_tes"], i in 1:nfe, j in 2:ncp+1], rk[k,i,j]
                                    == sum(x[k,i,l] * adot[l,j] for l in 1:ncp+1))

@NLconstraint(model, dODE[k in ["T_tes"], i in 1:nfe, j in 2:ncp+1], rk[k,i,j]
                                    - h*ODE[k,i,j] == 0)
# initial condition
@constraint(model, initConstrs[k in ["T_tes"]], x[k,1,1] - x0[k] == 0)
# Closing shooting gap
@constraint(model, gapConstrs[k in ["T_tes"], j in 1:nfe-1], x[k,j,end]
                                    - x[k,j+1,1] == 0)


# algebraic constraints (balances)
@constraint(model, split[i in 1:nfe], 0 == u["q_bp",i] + u["q_sys",i] - q_dh[i]);
@constraint(model, mass[i in 1:nfe],  0 == u["q_whb",i] + u["q_A",i]
                                    - (u["q_sys",i] + u["q_B",i]));
@constraint(model, energ1[i in 1:nfe, j in 2:ncp+1], 0 == u["q_sys",i]*T_dh_ret[i]
                                    + u["q_B",i]*x["T_tes",i,j] - z["T_A",i,j]
                                    *(u["q_whb",i] + u["q_A",i]));
@constraint(model, energ2[i in 1:nfe, j in 2:ncp+1], 0 == (u["q_whb",i]
                                    *y["T_whb",i,j] + u["q_A",i]*x["T_tes",i,j])
                                    - z["T_B",i,j]*(u["q_sys",i] + u["q_B",i]));
@constraint(model, energ3[i in 1:nfe, j in 2:ncp+1], 0 == u["q_sys",i]
                                    *z["T_B",i,j] + u["q_bp",i]*T_dh_ret[i]
                                    - z["T_C",i,j]*q_dh[i]);

@constraint(model, [i in 1:nfe], Q_whb[i] == Q_whb_used[i] + Q_dump[i]);

@constraint(model, WHB[i in 1:nfe, j in 2:ncp+1], Q_whb_used[i] == ρ_dh*Cp_dh
```

```
                                    *u["q_whb",i] * (y["T_whb",i,j]
                                    - z["T_A",i,j]))
@constraint(model, PHB[i in 1:nfe, j in 2:ncp+1], u["Q_phb",i] == ρ_dh*Cp_dh
                                    *q_dh[i]*(y["T_phb",i,j] - z["T_C",i,j]))



@variable(model, Q_need[i in 1:nfe]);
@NLconstraint(model, [i in 1:nfe], Q_need[i] == q_dh[i]*Cp_dh*ρ_dh
                                    *(T_dh_sup[i]-T_dh_ret[i]))


@NLobjective(model, Min,   1e-6*sum(u["Q_phb",i]^2 for i in 1:nfe)
                         + sum(u["q_A",i]*u["q_B",i] for i in 1:nfe)
                         + 1e-6*sum(Q_dump[i]^2 for i in 1:nfe)
                         #+ 1e-3*sum(u["q_bp",i] for i in 1:nfe)
                         #+ 1e-3*sum(u["q_whb",i]^2 for i in 1:nfe)
                         );


optimize!(model);
cost = sum(value.(u["Q_phb",:])[i] for i in 1:nfe);

Q_units = 1/3600/1000; # [kJ/hr] -> [MW]
q_units = 1/3600*1000; # [m3/hr] -> [kg/s]

### Plotting results
fig = figure(figsize = (10,10));

subplot(321);
ax1 = plot(tspan[1]:tspan[2], vcat(value.(x0["T_tes"]), value.(x["T_tes",:,end])),
                                    label = L"T^{tes}");
ax1 = plot(tspan[1]+1:tspan[2], value.(y["T_phb",:,end]), label = L"T^{phb}");
ax1 = plot(tspan[1]+1:tspan[2], value.(y["T_whb",:,end]), label = L"T^{whb}");
ylim([40.0, 125.0]);
ylabel(L"Temperature [$  $C]");
legend();

subplot(322);
ax6 = step(tspan[1]:tspan[2]-1, Q_whb*Q_units, label = L"Q^{whb}");
ax6 = step(tspan[1]:tspan[2]-1, value.(Q_need)*Q_units, label = L"Q^{demand}")
ylim([0.0, 30.0]);
ylabel("Heat [MW]")
legend();

subplot(323);
ax2 = step(tspan[1]+1:tspan[2], value.(u["q_whb",:])*q_units, label = L"q^{whb}");
ax2 = step(tspan[1]+1:tspan[2], value.(u["q_A",:])*q_units, label = L"q^A");
ax2 = step(tspan[1]+1:tspan[2], value.(u["q_B",:])*q_units, label = L"q^B");
ylabel("Flow [kg/s]");
ylim([0.0, 340.0]);
legend();

subplot(324);
ax3 = step(tspan[1]+1:tspan[2], value.(u["Q_phb",:])*Q_units, label = L"Q^{phb}");
ax3 = step(tspan[1]:tspan[2]-1, value.(Q_dump[:])*Q_units, label = L"Q^{whb,dump}");
ax3 = step(tspan[1]:tspan[2]-1, value.(Q_whb_used[:])*Q_units, label
                                    = L"Q^{wbh,used}");
ylim([0.0, 30.0]);
ylabel("Heat [MW]");
legend();

subplot(325);
ax4 = step(tspan[1]+1:tspan[2], value.(u["q_bp",:])*q_units, label
                                    = L"q^{bypass}");
ax4 = plot(tspan[1]+1:tspan[2], value.(q_dh[:])*q_units, label = L"q^{dh}");
ylabel("Flow [kg/s]");
xlabel("Time [hrs]");
```

```
ylim([0.0, 340.0]);
legend();

subplot(326);
ax5 = plot(tspan[1]+1:tspan[2], value.(z["T_A",:,end]), label = L"T^A");
ax5 = plot(tspan[1]+1:tspan[2], value.(z["T_B",:,end]), label = L"T^B");
ax5 = plot(tspan[1]+1:tspan[2], value.(z["T_C",:,end]), label = L"T^C");
ylim([40.0, 125.0]);
ylabel(L"Temperature [$  $C]");
xlabel("Time [hrs]");
legend();

fig.show();
```

NTNU

Norwegian University of
Science and Technology