Master's thesis

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

Alireza Mirzaei

Optimal Control of Energy Storage Systems

Master's thesis in Chemical Engineering
Supervisor: Sigurd Skogestad | David Pérez Piñeiro

September 2021

NTNU
Norwegian University of
Science and Technology

NTNU
Kunnskap for en bedre verden

Department of Chemical Engineering

TKP4900 - Master Thesis

Optimal Control of Energy Storage Systems

Author:
Alireza Mirzaei

Supervisors:
Sigurd Skogestad — David Pérez Piñeiro

Septmber, 2021

I dedicate this work to my parents who have always been supportive of my decisions and challenges

Abstract

Fossil fuel consumption and global warming are among the most pressing problems in present time. This, in turn, has necessitated the application of renewable sources of energy like as thermal energy of earth. Development of advanced technologies to improve the thermal energy performance of the buildings by using earth is an example of such technology and called thermal energy storage (TES).

Energy storage problems are highly stochastic in nature. Not only the generation of the renewable energy is uncertain, but also the energy prices are not constant. Therefore implementation of efficient control and optimization policies will decrease energy consumption and cost. In this thesis, we studied the optimal control of borehole energy storage system. In order to minimize the annual energy costs, we developed two methods of parameterization in Cost Function Aprroximation (CFA) policy which itself integrated with a direct lookahead policy. These algorithms decide about the daily energy flows and calculate the optimal values of tuning parameters. We applied the forecast error technique to predict randomness in future and integrated it with MPC Control. The results prove that these parameterized CFA-DLA policies are suitable to be used in reality.

Acknowledgements

This thesis did not reached end with kind supports and help of many individual. I would like to extend my thanks to all of them.

I highly appreciate chemical process system engineering group especially my supervisor, Professor Sigurd Skogestad, to provided me with enough knowledge and relevant prerequisites needed to complete this thesis.

After that, I would like to express my special gratitude to my co-supervisor David Pérez Piñeiro who supported me till end of this thesis and spent a lot of time and endeavor for different part of the project. Without his coordination, it was not possible to finish it on time.

I am highly indebted to Jose Otavio Assumpcao Matias for his advices and troubleshooting of the Codes.

My thanks and appreciations also go to classmates who have willingly helped me out with their knowledge.

Contents

Abbreviations

ADP : Approximate Dynamic Programming

ADDP :Approximate Dual Dynamic Programming

ASHP : Air Source Heat Pump

BDP : Backward Dynamic Programing

CFA : Cost Function Approximation

COP : Coefficient of Performance

DLA : Direct Lookahead Approximation

GSHP : Ground Source Heat Pump

MDP : Markovian Decision Process

MPC : Model Predictive Control

SDDP : Stochastic Dual Dynamic Programming

TES : Thermal Energy Storage

PFA : Policy Function Approximation

VFA : Value Function Approximation

Nomenclature

Gereral Symbols

C	Stage cost	-
$\mathbb{E}$	Average or expectation	-
$\bar{F}^{\pi}$	Approximation of total cost obtained by policy π	-
K_t	Input coefficient in Ricatti equation	-
N	Number of sample paths	-
p_t	Price at time t	-
$\bar{p}_t$	Estimate of price at time t	-
S_0	Initial state	-
S_t	State at time t	-
$\tilde{S}_{t,t'}$	State approximation at time t' being made at time t	-
S^M	Transition function of states	-
t	discrete time	-
t'	discrete time in prediction horizon	-
T	Control horizon	-
V_t	Value function at time t	-
$\bar{V}_t$	Approximation of value function at time t	-
W_t	Exogenous information at time t	-
$\tilde{W}_{t,t'}$	Approximation of the exogenous information at time t' being made at time t	-
X	Set of possible actions	-
x_t	Action at time t	-
$\tilde{x}_{t,t'}$	Action based on approximation of states and exogenous information at time t' being made at time t	-
α	A coefficient between 0 and 1	-
θ	Parameter	-
π	Policy	-
Π	Set of all possible Policies	-
ω_t	Sample realization of random variable	-

Symbols Used in our Model

A_d^s	Maximum value of heating demand	MWh
A_d^w	Maximum value of cooling demand	MWh
$A^{\inf}$	Domain of change of underground temperature	$^{\circ}K$
C_p	Water heat capacity	$MWh/Kg^{\circ}K$
D	Diameter of pipes in borehole	$Meter$
D_t	Demand at time t	MWh

$\hat{D}_t$	change of demand between time t and t+1	MWh
$D_{t,t'}$	Approximation of demand at time t' being made at time t	MWh
D^w_{std}	Standard deviation of randomness of demand in summer	MWh
D^s_{std}	Standard deviation of randomness of demand in winter	MWh
F^{Norm}	Cost of unparameterized model	NOK
F^{Opt}	Real optimal cost	NOK
$f^D_{t,t'}$	Forecast of demand at time t' being made at time t	MWh
$f^{P^c}_{t,t'}$	Forecast of cooling price at time t' being made at time t	NOK
$f^{P^h}_{t,t'}$	Forecast of heating price at time t' being made at time t	NOK
$f^{T^{\inf}}_{t,t'}$	Forecast of underground temperature at time t' being made at time t	$^\circ K$
L	Length of the pipes in borehole	$Meter$
m	Water mass in borehole	Kg
NOP	Number of pipes	$-$
P^c_t	Cooling price at time t	NOK
P^h_t	Heating price at time t	NOK
$\hat{P}^c_t$	change of cooling price between time t and t+1	NOK
$\hat{P}^h_t$	change of heating price between time t and t+1	NOK
$\hat{P}^c_{avg}$	Average cooling price	NOK
$\hat{P}^h_{avg}$	Average heating price	NOK
$\hat{P}^c_{min}$	Minimum cooling price	NOK
$\hat{P}^h_{min}$	Minimum heating price	NOK
$\hat{P}^c_{std}$	Standard deviation of randomness in cooling price	NOK
$\hat{P}^h_{std}$	Standard deviation of randomness in heating price	NOK
Q^b_t	Discrete energy flow between borehole and demand	MWh
Q^c_t	Discrete energy flow of cooler	MWh
Q^h_t	Discrete energy flow of heater	MWh
$\tilde{Q}^b_{t,t'}$	Estimation of discrete energy flow between borehole and demand based on forecast of exogenous information and states at time t' being made at time t	MWh
$\tilde{Q}^c_{t,t'}$	Estimation of discrete energy flow of cooler based on forecast of exogenous information and states at time t' being made at time t	MWh
$\tilde{Q}^h_{t,t'}$	Estimation of discrete energy flow of heater based on forecast of exogenous information and states at time t' being made at time t	MWh
R_{max}	Maximum allowable energy flow between demand and borehole	MWh
T^b	Borehole temperature at time t	$^\circ K$
$\tilde{T}^b_{t,t'}$	Estimation of borehole temperature based on forecast of exogenous information at time t' being made at time t	$^\circ K$

T_t^{inf}	Underground temperature at time t	$^\circ K$
$\hat{T}_t^{inf}$	Change of underground temperature between time t and t+1	$^\circ K$
T_{avg}^{inf}	Average underground temperature	$^\circ K$
$\hat{T}_{std}^{inf}$	Standard deviation of randomness in underground temperature	$^\circ K$
$\varepsilon_{t,t'}^{D}$	Vector of forecast noise of demand at time t' being made at time t	MWh
$\varepsilon_{t,t'}^{P^c}$	Vector of forecast noise of cooler price at time t' being made at time t	NOK
$\varepsilon_{t,t'}^{P^h}$	Vector of forecast noise of heater price at time t' being made at time t	NOK
$\varepsilon_{t,t'}^{T^{\text{inf}}}$	Vector of forecast noise of underground temperature at time t' being made at time t	$^\circ K$
λ	Heat transfer coefficient between borehole and underground	$MWh/^\circ K$
θ	Single scalar parameter	-
θ_{month}	i^{th} coordinate of parameter	-
$\Delta F^\pi(\theta)$	Policy cost index with rexpect to parameters	-
$CR^\pi(\theta)$	Cost Reduction with respect to parameters	-

Mathematical Symbols

∇_θ	Derivative with respect to	–
∂	Partial Derivative	–

List of Figures

List of Tables

1 Introduction

1.1 Importance of Renewable Energy and the Control of Energy Storage Systems

Today the world needs more energy and the statistics show that energy demand is continuously increasing in order to support economic and social growth of countries and build a better life standards, but providing this huge amount of energy around the world should be taken with responsibility and commitment to developing and using our resources more efficiently. These days all of us have to be committed to protect both people and the environment and make positive economic contributions.

The problem here is not only that the conventional energy resources such as fossil fuels are depleting as fast as possible, but also there are increasing concerns about the adverse environmental effects of hydrocarbons since they are almost used as the primary source of energy in all countries. It is said that electricity production is the first source of greenhouse gases, more than sum of all driving and flying travels. Electricity generation has been reported as the second leading cause of industrial air pollution in the USA. Most of our electricity comes from coal, nuclear and gas cycle power plants. Generating energy from these resources takes a severe toll on our environment and pollutes our air, land, and water. Renewable and clean sources of energy is one of the most important measures we can take to reduce these adverse impacts on our planet. Renewable energy is the energy derived from natural resources that replenish themselves over a period of time without depleting. Renewable sources of energy also have the benefit of being abundant, available in different capacities nearly everywhere, and they cause little, if any, environmental damage. Energy from the sun, wind, and thermal energy stored in the Earth's crust are most well-known examples. These clean sources of energy reduce harmful smog and toxic buildups in our air and water while they have not negative impacts caused by coal mining and hydrocarbon exploration and development. Due to increasing trend of population around the world and in consequence energy consumption in residential buildings, many countries offer subsidy for applying renewable sources of energy. A familiar example is installing solar cells in roof of the buildings. Also governments are showing more interest in wind turbines for electricity generation and boreholes for district heating. These resources have fewer environmental impacts e.g. producing less carbon dioxide (CO_2), which is the leading cause of global climate change.

Alongside with gradual technical improvement in manufacturing and installation of renewable energy equipment, the public intention to replacing these sources with fossil fuels increased within recent years. The strongest motivation for renewable sources is that renewable energy is nearly free of charge compared to conventional one. In many cases such as solar cells it is even possible to sell extra amount of generated electricity to the grid.

However, replacing fossil-fuel infrastructures and conventional electricity with renewable sources will take time because renewable sources have always been associated with technical problems. One of the most important issues with these kinds of energy is unreliability. This comes from the fact that the most renewable resources are out of our control. In case of wind turbines or solar cells, we are uncertain that how we have to manage to use the produced energy, because we do not know how much energy we are able to produce over the next days. An effective approach for solving this issue is using an energy storage device or battery. Without a battery if we can not generate energy from renewables we have to supply energy from grid in the demand peak time even if it is expensive. In addition if we succeed to produce more energy than what is really needed in a specific time, extra amount is wasted. The battery enables us to store the energy and use it in more appropriate time later. In other words, the role of battery here is to create a degree of freedom for our decisions that let us choose that when we can charge the battery or discharge it to supply the demand. In energy storage domain, the essence of some optimal control policies that help us to reduce energy consumption cost is highly needed from economical point of view because most of the time the prices are fluctuating. Moreover due to uncertain nature of renewable energy availability in different times, such policies should ensure us that we can always provide required energy from both conventional energy (grid) and these renewable resources. Efficiencies of these policies depends on many factors such as uncertainty patterns, quality of prediction and model of

case study.

1.2 Contributions of the Master Thesis

In this thesis, we develop a stochastic optimization approach for a specific energy storage problem and compare its efficiency with optimal point. The case study in this thesis is to provide the thermal energy for a number of buildings. We demonstrate that how it is possible to feed the buildings economically with required thermal energy by a borehole device in parallel with conventional fire gas heaters and electricity based coolers while the demand is uncertain, energy prices experience random variation and the storage device is subject to stochastic heat exchange with surrounding underground.

We evaluate the performance of a parameterized version of hybrid Cost Function Approximation and Direct Lookahead Approximation method which is tuned to give better results under uncertainty. This method can be an effective way for handling uncertainty. In this approach the computation is simpler than methods like as scenario tree Model Predictive Control and it has been subject of interest recently by some articles and researchers. We do following tasks in this thesis:

1. We create a mathematical model of borehole (storage device), heater and cooler and heat exchange system between these components for control.

2. we create uncertainty models of the heating and cooling demand, heat exchange between borehole and underground and the energy prices to estimate the expected cost of the designed control policy.

3. We parametrize our model and implement a direct lookahead policy and tune the parameters offline in above policy.

4. We check the performance of our control policy and compare it with the optimal cost when we know everything about the future.

1.3 Organization of the Report

We start by defining the base model and then describe that how we can formulate the problem by manipulating the underlying data. We then discuss the policy which is used to minimize annual energy cost of our energy storage problem and compare it with the real optimum cost in the situation that we have perfect forecast. This thesis includes 8 chapters. The content of rest chapters are as following:

- **Chapter 2. Literature Review of Stochastic Optimization and Energy Storage**

We review other researches about energy storage systems, uncertainty modeling and different policies to handle stochastic optimization.

- **Chapter 3. Stochastic Optimization**

We describe the main components in a stochastic optimization problem and go further in detail to four groups of policies, that are generally applied to solve this problems, by presenting main formulations.

- **Chapter 4. Case study: A Borehole Energy Storage**

In this chapter we present a base model of the energy storage problem and demonstrate different variables and parameters of the system. Then we provide the formulation of the cost and show how the stochastic variables could be simulated.

- **Chapter 5. Designing Policies**

The algorithms used in this thesis and the benchmark policy are explained in this chapter.

- **Chapter 6. Evaluating Policies Performances**

We implement the hybrid parameterized CFA-DLA policy in the context of our problem and provide performance of this policy in terms of numerical results and profiles of energy flow in this chapter.

- **Chapter 7. Discussion**

Our results are analyzed in this chapter. We also discuss about the development of our simplification and derive possible future research on this domain.

- **Chapter 8. Conclusion**

Final conclusion in connection with our findings are presented in this chapter.

2 Literature Review of Stochastic Optimization and Energy Storage

Optimal control in energy storage domain under stochastic condition has been subject of interest over recent years. Researchers have investigated on different energy storage models and developed appropriate control policies in each case. However, according to our knowledge, few number of researches have been carried out on optimal control of borehole systems. In this chapter we provide with a short review of the classification of methods used to handle stochastic optimization and afterwards we present previous works of academic communities on optimal control of energy storage systems by above methods.

Control of energy storage systems under stochastic conditions is a kind of sequential decision making problems. Every discrete time, the controller decides to make some decisions about the further actions according to pre-defined policy while taking model characteristics and predictions into account and then system is updated. These actions are normally energy flows between components of the system. W. B. Powell 2019 classified the strategies of the policy functions to four different groups which are normally applied in sequential decision making problems:

a. Policy function approximations (PFA)

This approach is to work with analytical functions that map a state to an action and includes all the information we need to know. These functions might be linear or nonlinear.

b. Cost function approximations (CFA)

This group of policies minimize or maximize some analytical and parameterized cost functions that normally include analytical modified set of constraints. We recognize the CFA as the cost over a single time period, but it is common to use hybrid methods and combine the concept of CFA policy with a look-ahead policy.

c. Value function approximations (VFA)

This group includes the policies developing Bellman's equation and are very popular in the literature on dynamic programming and reinforcement learning. These methods are known as the cost-to-go function in control theory. In this approach, the quality of the policy is quantified by the so-called value function. Computing an approximation of this value function is an important subtopic of reinforcement learning.

d. Direct lookahead approximation (DLA)

In many problems it is not possible to compute sufficiently accurate VFAs. In such cases we can replace the lookahead expectation with an approximate lookahead model as an alternative option. The most well known approximation is to use a deterministic lookahead, often called Model Predictive Control (MPC).

The two first approaches are policies based on policy search and the two last methods are policies based on look ahead approximations. In policy search strategies an objective function is used to search within a family of functions to find a function that works best. Lookahead approximations are based on approximating the impact of a control action on the future. A wide range of energy storage problems have been previously studied by researchers and policies from each of the four classes or some hybrid policies have been investigated before.

PFA policies have been investigated in several articles in the energy storage domain. Warrington et al. 2012 effectively controlled storage systems in one day horizon based on predefined functions which maps state to action by wind forecast errors. Kim and W. Powell 2011 derived an optimal policy for making advance commitments of energy from a renewable wind source in presence of a storage device and a mean-reverting process for electricity prices. They configured the policy in such a way that stored energy always used in next discrete time and assumed stationarity in the wind and price processes.Han and E 2016 applied the neural network technique to conduct energy flows from wind turbines, battery and the power grid to meet a time-varying demand.

Neural network approach makes control decisions based on the state of the system and therefore is classified as a subset of PFA.

Parametrized CFA, however, have been overlooked by academic communities. Perkins and W. Powell 2017 implemented a hybrid CFA/DLA policy by rolling forecasts to tune parameters of an energy storage problem with stochastic energy from wind and demand. They analyzed the performance of the different forms of parameterization on one of the constraints by changing forecast error. They proved that quality of forecast can change the efficiency of one formulation compared to other forms. Moreover, they demonstrated that in high level of uncertainty the policies further underestimate the forecast to limit the risk of paying penalties for constraint violation. Ghadimi et al. 2019 studied almost the same problem using rolling forecasts of varying quality, but with different assumption in the model. He parameterized the model constraints and examined the performance of lookup table parameterization policy under perfect forecast and noisy one. He showed that modifying deterministic approximation by parameters can handle uncertainty and at the same time capture the dynamic of full base model. By this way there is no need to make approximation which is used in stochastic lookahead model. In same manner, Simão et al. 2017 implemented CFA in the form of deterministic look ahead associated with some parameters to control an energy storage where energy of the battery was managed to handle the changes of renewable energy.

VFA policies, usually referred to as dynamic programming, have been very popular in machine learning domain. For example, Zhou et al. 2018 performed Backward Dynamic Programming (BDP) to find an optimal policy for managing wind energy and storage system in presence of price variation and wind unreliability. One of the main drawbacks of dynamic programming is the cost of the computation especially in large state space systems. The complexity of dynamic programming techniques increases linearly with the number of stages, but exponentially with the number of state variables. This matter is called "curse of dimensionality". The practical meaning for energy storage domain is that the complexity linearly with the number of time samples, but exponentially with the number of storage devices, and with the number of state variables describing each device as well. Energy storage problems in reality have large state space and stochastic future processes of such complex problems are infeasible to develop. Many authors studied on a so-called Approximate Dynamic Programming (ADP) technique that applies an approximation of the base model. Lohndorf and Minner 2010 implemented an ADP least-squares policy evaluation technique, which worked based on temporal differences, to find the optimal infinite-horizon storage for an energy storage system combined with renewables. They studied on a system with a renewable source and a storage device and proposed the model bidding processes as continuous-state Markov decision processes. The problem solved via stochastic dynamic programming. To avoid the curse of dimensionality, the value function was approximated by a set of linearly independent functions and the results were compared to the ones obtained by linear programming. They later presented an approach named approximate stochastic dual dynamic programming (ADDP) which integrated ideas from ADP with stochastic dual dynamic programming (SDDP), which approximated the value of energy storage using multidimensional Bender's cuts (Lohndorf, D.Wozabal et al. 2013). Backward ADP was also tested and showed near-optimal solution in Durante et al. 2017 and Cheng et al. 2018. Salas and W. Powell 2018 benchmarked a scalable ADP algorithm with piece-wise, separable VFAs for stochastic control of Grid-Level Energy Storage and showed that this algorithm is able to design storage policies that are within 0.08% of optimal on deterministic models, and within 0.86% on stochastic models. They tested their model with 5 energy storage device at the same time and proved that this method learn similar complex behavior for such system.

The most popular approximation strategy is to use a deterministic lookahead, often called model predictive control or rolling/receding horizon procedure. This group of policies are usually the first policy that comes into mind when we are going to solve a problem which needs a lookahead policy. Deterministic prediction of random exogenous processes has been applied in the energy domain. Wallace and Fleten 2003 implemented deterministic forecasts of wind, solar and electricity loads to manage energy storage. Sioshansi et al. 2014 studied the benefits of integrating wind turbines and storage devices by utilizing a deterministic lookahead which makes decision based on a mixed integer program with prediction for two-weeks horizon. Deterministic lookahead policies are subject to argument by the research communities that they do not take uncertainty into account and forecast error cannot be controlled in such models. Some researchers studied on stochastic DLA

policies in different domains. Jacobs et al. 1995 used stochastic lookahead model for hydroelectric power planning with streamflow forecasting models and a database containing hydrological information. In another article Takriti et al. 1996 developed a model and a solution technique for generating of electricity power when demands are uncertain. His results indicate large saving in terms of cost of power generating systems with stochastic model instead of the deterministic model. In energy storage domain Arnold and Andersson 2011 implemented MPC to control a storage hub with both battery and hot water storage devices to minimize the cost of satisfying loads of a bunch of households in presence of uncertain renewable energy load as well as electricity and natural gas prices. Rahmani 2017 showed the remarkable potential for minimizing the energy loss and operation cost by optimal control of energy storage systems and application of stochastic MPC. In this work, MPC increased the robustness of optimization procedure with respect to the prediction errors. Kou et al. 2018 developed a new stochastic energy scheduling scheme for microgrids. In his scheme the energy scheduling was formulated as a stochastic model predictive control problem which incorporates the uncertainties in both sides of supply and demand. By machine learning techniques, he converted his problem to a standard convex quadratic programming which solved efficiently. This method worked well with both Gaussian and non-Gaussian uncertainties.

In lookahead approximation policies, it is very important to perform a good uncertainty modelling to handle stochastic optimization. There are several articles that addressed uncertainty in their energy storage case studies. Backward DP was tested to find near-optimal solution in Durante et al. 2017. Zhou et al. 2018 also performed Backward Dynamic Programming (BDP) to find an optimal policy for managing wind energy storage system in presence of price variation and wind unreliability. However, exact Backward DP have not been widely used in the energy storage domain due to its potential problems with large state space. Another challenge in direct lookahead policies is model simplification which should be taken carefully. This is essential and have been performed in most of the literature to allow the controller to solve the problem, but this have to keep the simplified model near the reality. In most of the literature simplifications performed in the modelling of the generation of renewable energy and price processes. For example Ridder et al. 2011 assumes simple model to describe the dynamics of the underground storage system to be able to implement dynamic programming. He obtains a low root-mean-square value of the prediction error showing that his model is sufficiently good in predicting the temperature in the complex model and used by the controller. Price changes and production of renewables have also been subject to some simplifications in other previous works. For example Jiang et al. (2014), Cheng et al. 2018 and Mokrian et al. (2006) modeled it as a first order Markov chain in their works.

Almost all previous researches were about two specific energy source, one stochastic wind or solar energy and one electricity grid, which are used to charge both storage device and supply the demand. The models are simple and not characterize the real system in detail which is partly inevitable due to complexity of computations. Borehole as an efficient system for recovery of geothermal energy storage purposes have also been increasingly trended over recent years. But according to our search, articles about the borehole system limited to modeling, structure, sizing and pipe arrangement of these underground systems. As an research in control domain, Rink has provided with some papers about heat pumps control extended with a battery subjecting to variable energy prices, but his works does not cover long term energy storage (Rink et al. 1988 and Rink 1994). we did not find any article about optimal control of borehole system under stochastic condition.

In this thesis, we design different policies for optimal control of thermal energy storage in a typical borehole system and evaluate their performance. We also benchmark these polices against real optimal cost (perfect forecast). We describe our policy in chapter 4.

3 Stochastic Optimization

There is a wide spectrum of optimal control problems that need making decisions while they are associated with uncertainty. The sequence of these problems is decisions, information, decisions, information and so on. This kind of sequential decision making often referred to as decisions under uncertainty or stochastic optimization. Such problems falls into limitless domains such as: control engineering, economic and finance, stock market, games, energy storage, freight transportation and logistics scheduling. Application of optimal control under uncertainty in energy domain include unit commitment, energy storage, bidding energy resource, pricing electricity contracts and investment planning. In such problems every time that we make a decision, new information enters to our system in a variety of forms such as customer demands failures, stock price changes, unforeseen delay in scheduling, control noise and so on. These information usually affect and update our understanding of the system. This overall sequence forces us to make control decisions before this new information has been known or, in other words, means that we have to optimize the system under uncertainty. These problems have been developed and addressed in different domains with different notation and mathematical styles in modeling conventions by various research communities under the names such as dynamic programming, optimal control, stochastic programming, and robust optimization. The limits of these names are not clear and solutions of these communities might overlap or sometimes applicable to each other. According to literature there is a wide variation in the types of stochastic optimization problems in terms of the nature of the decisions, the uncertainties and the dynamics of the system. W. B. Powell 2019 refers to this as jungle of stochastic optimization and proposed a canonical framework for modeling and solving these problems. The main advantage of this approach is to completely separate the design of the model from the design of policies used to solve these models. Here we provide with a discussion of the system in Powell's framework. Afterwards, we continue with the canonical model for sequential stochastic optimization (control) problems and finally we sum up with design of policies in this canonical framework.

3.1 Canonical Model in Stochastic Optimization

In this framework there are five key elements in any sequential, stochastic decision problem which are needed for modeling of the system. These are states, actions, exogenous information, transition function and objective function. In this section we review these five elements in more detail. Afterwards, we explain the policies which are used to solve a sequential decision making problem under stochastic condition.

3.1.1 State Variables

The state variables ($S_t{}^n$) capture all the information from the system available to us at time t or at iteration n. States are necessary and sufficient information enable us to realize and model the system from time t or iteration n onward. States are used to calculate the cost function, make the decisions and compute next states by the transition function. According to Powell framework, state variables are often placed into three different categories (for simplicity we overlook iteration):

1. Physical resources (R_t) which could be for example inventory, temperature, or number of a specific items. We have physical realization of this subgroup variables.

2. Other information (I_t) such as prices, weather, the state of the economy or history of previous decisions. These are information which simply are not a kind of the first group.

3. Probabilistic information about the quantities that we cannot realize directly (K_t), for example probability of failure of an equipment.

In another classification, some states are controllable while other such as weather condition are not. There are third group of states which are partly controllable. The state space S_t usually

consists of a number of dimensions which are written as:

$$S_t = (S1_t, S2_t, S3_t, ..., Send_t) \tag{1}$$

The set of initial state, denoted with S_0, includes initial values of dynamic states and is deterministic. we normally put all other fixed parameters and initial beliefs about unknown parameters in this set.

3.1.2 Decision Variables

Depending on the research community that study on sequential optimal control problem, decisions might be represented as actions (a), low dimensional controls (u), or general vectors (x) that can be anything like as binary, scalar, continuous or discrete vectors. In control domain, they are continuous, but in sequential stochastic decision problems they are a vector of discrete variables being made over specific time steps. Here, we denote them with x. We call X a set of possible values of all decisions (x_t) and denote policy with $X^\pi(S_t)$. The relation between actions and policy is as following:

$$x_t = X^\pi(S_t) \tag{2}$$

3.1.3 Exogenous Information Variables

Exogenous information is any kind of information that becomes known to us at each time t, i.e. between t-1 and t and usually denoted by W_t. These could be any type of information such as prices, energy loads, number of failures or amount of precipitation and the sequence is W_1, W_2, W_3,... Taking this information to the account, we can show that sequence of the states, decisions and information evolve with below sequence:

$$(S_0, x_0, W_1, S_1, x_1, W_2, S_2, x_2, W_3, ..., S_t, x_t, W_{t+1}) \tag{3}$$

A typical way of showing exogenous information is the notation with $\hat{}$ sign. For example when a energy demand changes between two discrete times randomly, we can demonstrate this exogenous change with below equation:

$$D_t = D_{t-1} + \hat{D}_t \tag{4}$$

The exogenous information might be scalar or a vector with innumerous dimensions. It might depend on the states and sometimes even the actions, e.g large supply could lower the price, or it is purely independent like as amount of wind generation. The notation in the first case is $W_{t+1}(S_t, x_t)$. The exogenous information can enter to our model in the form of observational uncertainty, forecasting, model uncertainty, and uncertainty in the implementation of decisions. In some cases there should be a stochastic model of uncertainty such as probability distributions, but it is rare in case.

3.1.4 Transition Function

Transition function is referred to as the system dynamics and show the evolution of the states from one discrete time to next one. This function is normally shown with below equation:

$$S_{t+1} = S^M(S_t, x_t, W_{t+1}) \tag{5}$$

Above function demonstrates that $S^M(.)$ is depended to previous states, decisions and exogenous information that become known at every discrete time. There are two types of transition function which classified to model based and model free. The first type implies that transition is a system of analytical equations or in other words there is a model for transition of states, while in case of the latter type there is not. In this type we observe a state S_t and then we take an action x_t, but after that all we can do is observe the next state S_{t+1} without any estimation or calculation tools.

3.1.5 Objective Function

Performance of a system could be investigated through many ways. It might be single or multiple goals interested in terms of cost, performance, number of system failures and time. In stochastic optimization we will set $C_t(S_t, x_t, W_{t+1})$ to the cost contribution of being in state S_t, taking action x_t and transition to next state while taking exogenous information W_{t+1} into account. The objective is to find the policy that minimizes expected cost, which is written as:

$$\min_{\pi \in \Pi} \mathbb{E}^{\pi} \sum_{t=0}^{T} C(S_t, X_t^{\pi}(S_t)) \tag{6}$$

considering transition function and the fact that the expectation is over all possible paths of $W_1, W_2, ..., W_T$. The π sign in superscript of expectation implies that the exogenous information might depend on prior actions. As it is not practically possible to calculate the expectation in above formula, we can take the average of cumulative cost over a series of simulations as following:

$$\bar{F}^{\pi} = \frac{1}{N} \sum_{n=1}^{N} \sum_{t=0}^{T} C(S_t(\omega^n), X_t^{\pi}(S_t(w^n))) \tag{7}$$

Here, ω is a sample realization of the random variables (W) an n is the number of the simulation paths of exogenous information.

3.2 Control Policies

3.2.1 Policy Function Approximation (PFA)

In this class of policies we normally have a good knowledge that how we have to make decisions. The examples covers a wide range including stock trading, frequent charge and discharge of batteries according to price and loading and unloading of an inventory. PFA's are analytical functions and map states to actions while do not use an embedded optimization. We can classify them as following:

1. Lookup tables: The simplest and most sensible class is lookup tables meaning that if a condition is true then controller will do something specific. As an example we can define that if temperature is above 50 °C, then the controller turn on the cooler.

2. Parametric functions: The second class is parametric functions. As an example we might hold a stock ($x_t = 0$) or sell ($x_t = 1$) it if the price at every discrete time (p_t) goes down below a smoothed estimate $\bar{p}_t$ with below formula:

$$\bar{p}_t = (1 - \alpha)\bar{p}_{t-1} + \alpha p_t \tag{8}$$

The policy then could be written as:

$$X^{\Pi}(S_t|\theta) = \begin{cases} 1, & \text{if } p_t \leq \bar{p}_t - \theta \\ 0, & \text{otherwise} \end{cases} \tag{9}$$

The parameter θ in above is then manually tuned via grid search. We call this class of policies monotone policies because actions change monotonically.

3. Statistical models: These are also parametric functions in the form of linear or nonlinear equations. A linear form can be written as below equation:

$$X^{\pi}(S_t|\theta) = \theta_0 + \theta_1\Phi_1(S_t) + \theta_2\Phi_2(S_t) \tag{10}$$

This is also know as affine policy or linear decision rule. The task is to is find the best vector of θ. If the objective function is a type of quadratic form, we can show that the best policy in absence of any constraint derived as:

$$X^{*}(S_t) = K_t S_t \tag{11}$$

which is the famous Ricatti equation.

We may also use other types of PFA such as Boltzmann policies for discrete action, nonlinear function such as an order-up-to inventory policy, a nonparametric/locally linear policies or a neural network. The first one chooses a discrete action according to the probability distribution. Here the actions with the largest so-called estimated value are given the highest probability of being accepted. The third one is again a kind of affine policy with user defined region of the state space in which the responses are locally linear. In neural networks the care should be taken while they have very high dimensional architectures which means that they require large training data sets and in addition they are very sensitive to noise; therefore they are suitable for systems with less noise. Typically there is no guarantee that a PFA is in the optimal class of policies. Instead, we search for the best performance in a class by searching for a specific parameters that maximize or minimize cost function as following:

$$F^{\pi}(\theta) = \mathbb{E}\left\{\sum_{t=0}^{T} C(S_t, {X_t}^{\pi}(S_t|\theta))|S_0\right\} \tag{12}$$

Which is subject to below transition function:

$$S_{t+1} = S^{M}(S_t, x_t, W_{t+1}) \tag{13}$$

As it mentioned before, we are not able to calculate expectation. We can take two strategies of derivative-free and derivative -based method for minimizing $F^{\pi}(\theta)$. In the first strategy the policy simulator is seen as a black box. The second strategy can be implemented numerically by Batch Learning or Adaptive Learning techniques. Both of these techniques use numerical methods for computation of derivatives. The first technique uses the average of cumulative cost over a series of simulations in the way described in section 3.1.5. The formula becomes as following:

$$\bar{F}^{\pi}(\theta) = \frac{1}{N}\sum_{n=1}^{N}\sum_{t=0}^{T} C(S_t(\omega^n), {X_t}^{\pi}(S_t(\omega^n)|\theta)) \tag{14}$$

and the sequence of states generated by sample path ω^n is:

$$S_{t+1}(\omega^n) = S^{M}(S_t(\omega^n), X^{\pi}(S_t(\omega^n)), W_{t+1}(\omega^n)) \tag{15}$$

In the second numerical technique we use stochastic gradient of objective function for updating parameters in every forward pass of simulation as following:

$$\theta^{n+1} = \theta^n + \alpha_n \nabla_{\theta} F^{\pi}(\theta^n, W^{n+1}) \tag{16}$$

Under the conditions that cost function, policy function and transition function are differentiable, we obtain below equation by chain rule and considering the fact that contribution cost ($C(S_t, x_t)$) is a function of both state (S_t) and action x_t, the policy $X^\pi(S_t|\theta)$ is a function of both state (S_t) and parameters (θ), and the state (S_t) is a function of the previous state (S_{t-1}), the previous control decision (x_{t-1}) and the most recent exogenous information (W_t):

$$\nabla_\theta F^\pi(\theta,\omega) = \left(\frac{\partial C_0(S_0,x_0)}{\partial x_0}\right)\left(\frac{\partial {X_0}^\pi(S_0|\theta)}{\partial \theta}\right) + \sum_{t'=1}^{T}\left[\left(\frac{\partial C_{t'}(S_{t'},{X_{t'}}^\pi(S_{t'}))}{\partial S_{t'}}\frac{\partial S_{t'}}{\partial \theta}\right) + \frac{\partial C_{t'}(S_{t'},x_{t'})}{\partial x_{t'}}\left(\frac{\partial {X_{t'}}^\pi(S_{t'}|\theta)}{\partial S_{t'}}\frac{\partial S_{t'}}{\partial \theta} + \frac{\partial {X_{t'}}^\pi(S_{t'}|\theta)}{\partial \theta}\right)\right] \quad (17)$$

where:

$$\frac{\partial S_{t'}}{\partial \theta} = \frac{\partial S_{t'}}{\partial S_{t'-1}}\frac{\partial S_{t'-1}}{\partial \theta} + \frac{\partial S_{t'}}{\partial x_{t'-1}}\left[\frac{\partial X^\pi_{t'-1}(S_{t'-1}|\theta)}{\partial S_{t'-1}}\frac{\partial S_{t'-1}}{\partial \theta} + \frac{\partial X^\pi_{t'-1}(S_{t'-1})}{\partial \theta}\right] \quad (18)$$

The derivatives ($\frac{\partial S_{t'}}{\partial \theta}$) are calculated by above equation starting at $t' = 0$ and stepping forward in time.

PFAs do not scale to larger and more complex problems such as scheduling an airline or managing an international supply chain and is not able to handle an inventory problem with rolling forecasts.

3.2.2 Cost Function Approximation (CFA)

These policies minimize or maximize some analytical parametrized function subject to analytically modified constraints. In CFA's, we require to solve an imbedded optimization problem and usually there is one or more tunable parameters θ. CFA is suited for high dimensional stochastic optimization problems that require the use of solvers for linear, integer or nonlinear programs such as scheduling an airline. There are three steps in the implementation of parametric CFAs:

1. Designing the parameterization

Parameterization should be chosen to improve what can be achieved with the original deterministic approximation. There is no universal rule that how the parametrization should be implemented. Any parametrization form might be applicable to any problem due to circumstances.

2. Evaluating a parametric CFA

The form of CFA optimization equation is as following:

$$X^{CFA}(St|\theta) = \underset{x\in X_t(\theta)}{\arg\max}\, \bar{C}_t(S_t, x|\theta) \quad (19)$$

where $\bar{C}_t(S_t, x|\theta)$ is the cost function having one or more parameters and could be subject to parametrized constraints.

3. Tuning the parameters

The most common search procedures is either derivative-based stochastic search using numerical derivatives or derivative-free stochastic optimization.

Like as PFA's, CFA methods do not provide optimal policies in a wide range of problems.

3.2.3 Value Function Approximation (VFA)

Here, the system to be controlled is usually modelled as a Markovian Decision Process (MDP). An MDP includes a set of states which present configuration of the system, a set of actions which are reason of change of the system's states and a set of transition probabilities. These probabilities change from one state to another under a specific action and just depend on the current state-action pair. Usually a reward function associating a scalar to each transition and a discounting factor is defined. The role of discounting factor is to decreases the influence of long-term rewards. The quality of such policy is quantified by a so-called value function which associates to each state, the expected cumulative discounted reward from starting in that state and then following the given policy (expectation being done over all possible trajectories). An optimal policy is one of those which maximize the associated value function for each state.

If a system is in state S_t and controller takes an action x_t, the system goes to a new state S_{t+1} considering W_t as exogenous information. If we have a value function $V_{t+1}(S_{t+1})$ that estimates the value of being in state S_{t+1} it captures the impact of decision x_t.

The ideal VFA policy involves solving Bellman's equation as following:

$$V_t(S_t) = \max_{x_t} \mathbb{E}(C(S_t, x_t) + \{V_{t+1}(S_{t+1})|S_t, x_t\}) \tag{20}$$

where

$$V_{t+1}(S_{t+1}) = \max_{\pi} \mathbb{E}\left\{ \sum_{t'=t+1}^{T} C(S_{t'}, X_{t'}^{\pi}(S_{t'}))|S_{t+1} \right\} \tag{21}$$

The optimal policy of above formulation for a stochastic optimization problem is as following:

$$X_t^*(S_t) = \arg\max_{x_t \in X_t} (C(S_t, x_t) + \mathbb{E}\{V_{t+1}(S_{t+1})|S_t, x_t\}) \tag{22}$$

Since it is not possible to compute $V_{t+1}(S_{t+1})$, we use machine learning methods to replace it with approximation of value function ($\bar{V}_{t+1}(S_{t+1})$). This is called Value Function Approximation and the solution would be:

$$X_t^{VFA}(S_t) = \arg\max_{x_t \in X_t} (C(S_t, x_t) + \mathbb{E}\left\{\bar{V}_{t+1}(S_{t+1}|\theta)|S_t, x_t\right\}) \tag{23}$$

In order to remove the expectation, we can use post-decision state (S_t^x). This is the state immediately after a decision is made

$$X_t^{VFA}(S_t) = \arg\max_{x_t \in X_t} (C(S_t, x_t) + \bar{V}_t(S_t^x|\theta)) \tag{24}$$

This strategy works well especially for problems where x_t is a vector and $V_t^x\,(S_t^x)$ is a convex function of S^x.

3.2.4 Direct Lookahead Approximation (DLA)

These policies model the downstream trajectory of each decision making them now. We represent the sequence of states, actions and exogenous information same as Eq. 3, but with below differences

in notation:

$$\begin{aligned} S_t &\rightarrow \tilde{S}_{t,t'} \\ x_t &\rightarrow \tilde{x}_{t,t'} \\ W_t &\rightarrow \tilde{W}_{t,t'} \end{aligned} \tag{25}$$

Tilde sign in right hand side of above replacements shows the approximation of S, x and W in future consecutive discrete times t' when the system is in time t. The actions $\tilde{x}_{t,t'}$ determined by policy $\tilde{X}^{\tilde{\pi}}(\tilde{S}_{t,t'})$. Therefore the Eq. 3 is written for a lookahead model as following:

$$(S_t, x_t, \tilde{W}_{t,t+1}, \tilde{S}_{t,t+1}, \tilde{x}_{t,t+1}, \tilde{W}_{t,t+2}, ..., \tilde{S}_{t,t'}, \tilde{x}_{t,t'}, \tilde{W}_{t,t'+1}, ...) \tag{26}$$

We may introduce any approximation we think that it is suitable to our lookahead model and it is the real challenge. An example is change the belief model or to simplify the different types of uncertainty. We can then write the approximate lookahead policy as below:

$$X_t^{DLA}(S_t) = \arg\max_{x_t} \left(C(S_t, x_t) + \tilde{E} \left\{ \max_{\tilde{\pi}} \tilde{E} \left\{ \sum_{t'=t+1}^{T} C(\tilde{S}_{t,t'}, \tilde{X}^{\tilde{\pi}}(\tilde{S}_{t,t'})) | \tilde{S}_{t,t+1} \right\} | S_t, x_t \right\} \right) \tag{27}$$

Typically the approximate expectations ($\tilde{E}$) are computed using Monte Carlo sampling, although we can use a deterministic forecast. This policy is also known as a rollout policy. A simple solution could again be the parametrization of this policy and finding the best parameters. This policy requires that stochastic optimization problem to be solved at each time period. Fig. 1 shows the process of development of a direct deterministic lookahead policy, but the same process is used with any direct lookahead policy.

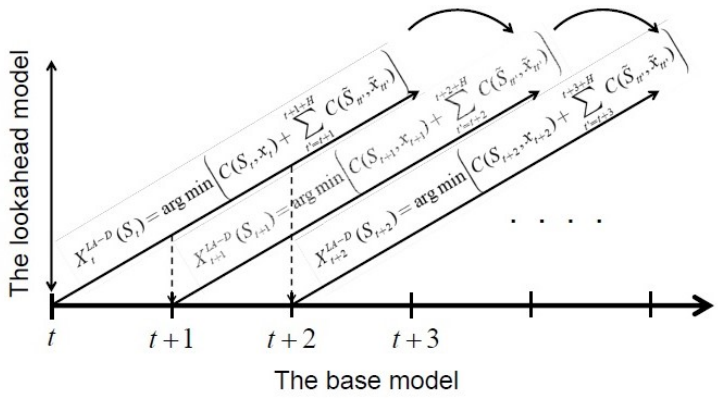

Figure 1: Development of a deterministic lookahead policy

4 Case study: A Borehole Energy Storage System

In this chapter we provide the basic model of the energy storage problem by defining state variables, decision variables, exogenous information, transition function and objective function. In our case study, we analyze optimal control of a heating and cooling system integrated with a borehole, as our energy storage device, to meet daily energy demand demand. Furthermore, we describe the methods that we use to simulate the uncertainties which enter to the system.

4.1 Basic Model

In the most buildings around 40 percent of the energy is consumed to heat and cool the building. An effective way to reduce the electricity energy consumption, operating cost, and consequently carbon dioxide emission is to heat and cool buildings by an underground thermal energy storage system. Briefly underground storage field consists a large underground volume. This borehole is loaded continuously with cold or heat quantities from demand over a year and can hold thermal energy for longer periods. Boreholes are the most well-known underground thermal energy storage systems. They typically include several horizontal and vertical pipes, drilled on the nodes of a square grid and are filled with water. This water is circulated between borehole and the building. A district heating network is a system that produces heat from a central location and might use a combination of gas, underground thermal energy or waste heat. Underground pipes then carry warm water to the buildings in a closed loop. The water is returned to the plant and heated and this cycle is repeated again and again. This borehole system looks like a big underground container. There are two types of boreholes in terms of heat exchange with underground surrounding. In the first type, the pipes are allowed to exchange the heat freely with the underground surrounding. Underground temperature variation is much less than the air and this stability is the main advantage of ground source heat pump (GSHP) when it is compared to a typical air source heat pump (ASHP). This type usually needs a heat pump to works both in summer and winter. In summers the heat is absorbed by cold circulating water through a heat pump and warm water returns to the borehole. The mass of circulated water is remarkably less than volume of underground water storage and temperature in borehole does not change largely. Therefore heat quantities are dumped out to the underground environment continuously. In winters the underground and consequently circulating water is slightly warmer than the air and the heat of water is conducted to the building through a heat pump. Cold water returns to the borehole and absorb the energy from warmer underground temperature and its temperature is somewhat recovered. In another type of borehole, the heat exchange between borehole and underground is minimized as much as possible by insulation. The idea here is to transfer the heat to the building by quite colder circulating water through a heat pump. Water in the storage borehole gradually become colder while summer passes and this cold water is stored and used directly to cool the building in the summer.

In our model we assumed a primary borehole network as our storage battery device in parallel with a conventional heater and cooler which are used to provide thermal energy for a number of buildings. Function of auxiliary heater and cooler is to supply the shortage of the energy when it is not possible to take all the heat or cold from borehole. Heater and cooler are working with fuel gas and electricity and their operating cost is subject to stochastic variations.

The variation in the underground temperature is lower than the air and this make underground more reliable energy source than outside air, however these variations depends on many factors such as geographical location, weather condition, depth of borehole and soil type and it is not possible to predict it directly. Therefore we consider the underground temperature variation as a stochastic energy loss in the storage system. We will use it further in the formulation of transition function in 4.1.6. We assume that there is no other heat loss in our model.

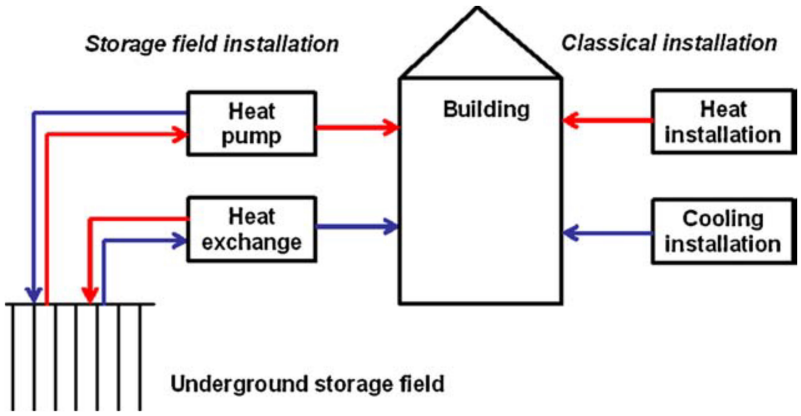

Figure 2: Heating and cooling supply in a building

In order to reduce the energy consumption cost, we use direct cooling and shutdown the heat pump in summers, therefore we decide to control the borehole temperature between 0 °C and 12 °C. Within this range a heat pump to be coupled with borehole storage system to provide the heat over winter, nevertheless, no building can be heated with such low temperatures. If the temperature goes below 0 °C, not only there is the risk of water freezing but also heat pump does not work efficiently due to high temperature difference. According to literature (Ridder et al. 2011) setting a maximum of 12 °C as higher bound helps us to use the cold water directly for cooling purpose. Therefore there is no need to heat pump and we shut it down over summer. A typical sketch of our model is presented in Fig. 2.

The objective is to control the daily energy flows between borehole, heater and cooler in such a way that we obtain the minimized energy cost. There are four nodes, three decision variables and five exogenous information in this model. In Fig. 3 We provide with a more conceptual sketch of the model.

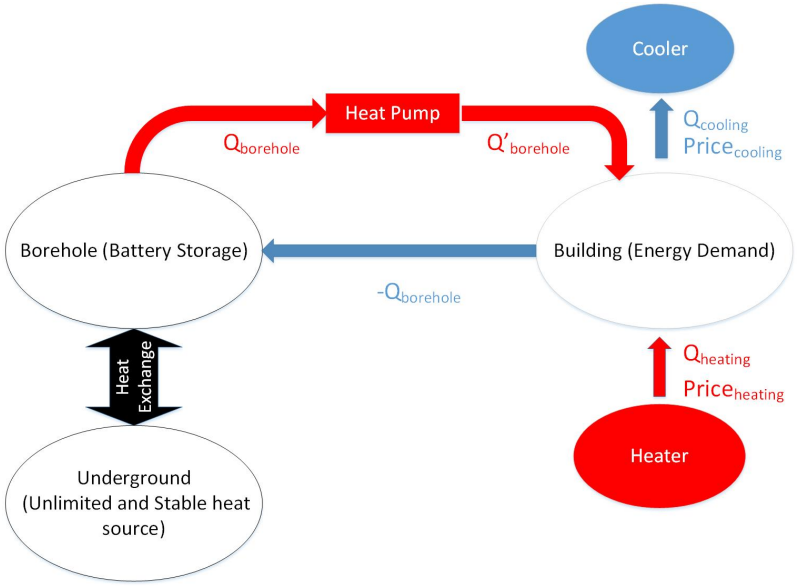

Figure 3: Basic model

In winters heat flows from borehole and heater to the demand. In summer, however, heat transferred from demand to borehole and the shortage of cooling will be supplied 'with removing the heat by the cooler. The borehole is continuously in heat exchange with underground and the rate of heat exchange depends on temperature difference and heat transfer coefficient. Borehole continuously receive heat from underground surrounding.

4.1.1 Static Parameters

In our model some parameters assumed to be constant for simplification, therefore we assigned fixed quantities to these parameters. First, the geometry of the borehole is fixed, therefore the number, diameter and length of pipes are constant. Second, we supposed that overall heat transfer coefficient between the borehole and the surrounding underground is not affected by dynamic of the system and is estimated to be constant. This makes sense due to nearly stagnant situation of water in borehole and negligible variation of underground temperature especially in depth. Third, due to physical limitation of the system, the maximum flow of the water from borehole to the system is limited to a certain value meaning that rate of the heat flow to or from the borehole in limited up to a fixed value. In addition, we suppose that our heat pump has a fixed COP. We sort the list of the static parameters as following:

R_{max}: Maximum allowable energy flow between the borehole and the demand in MWh.

n: The number of the pipes in the borehole system

D: Diameter of the pipes in meters

L: Length of the pipes in meters

m: Mass of the water in the borehole in Kg

COP: Coefficient of performance of the heat pump

λ: Heat transfer coefficient between borehole and surrounding in MWh/°C

C_p: Water heat capacity in MWh/Kg°C

For our case study, we consider the following numerical values for the above parameters in Table 1.

Parameter	Value	Unit
R_{max}	5	MWh
NOP	180	-
D	3	meters
L	10	meters
m	13000000	Kg
COP	3	
λ	0.2	MWh/°C
C_p	0.000001167	MWh/Kg°C

Table 1: Values of the static parameters

4.1.2 State Variables

The set of state variables includes all information that we need to characterize the model of our case study over time. This set is also sufficient for optimizer to make decisions and calculate final cost. We denote the initial values of states by S_0. Our thermal energy storage case study has five dynamic states (S_t), including the demand, prices, underground temperature and borehole temperature. We define two prices as heater price and cooler price which enable us to analyze the contribution of each of these energy sources. We also assume that that there is no need to pay for the cooling and heating by borehole and it is free of charge. Therefore the states of the system is given by:

$$S_t = (D_t, {T_t}^b, {T_t}^{\inf}, {P_t}^h, {P_t}^c) \tag{28}$$

D_t: Heat demand on day t in MWh.

$T_t{}^b$: Borehole temperature on day t in °C.

$T_t{}^{inf}$: Underground temperature on day t in °C.

$P_t{}^h$: Heating price by heater on day t in NOK/MWh

$P_t{}^c$: Cooling price by cooler on day t in NOK/MWh

We present the values of these initial states in Table 2:

Variable	Value	Unit
D_0	7.5	MWh
$T_0{}^b$	12	°C
$T_0{}^{inf}$	18	°C
$P_0{}^h$	70	NOK/MWh
$P_0{}^c$	1200	NOK/MWh

Table 2: Values of the initial states, S_0

we assume that control starts form November. We also suppose that each house needs a maximum of 100 KWh/day and 33 KWh/day for heating and cooling purposes respectively and our demand network includes 150 houses. Therefore the peak demand is 15 MWh/day for heating and 5 MWh/day for cooling. November is quite cold in Norway but we can assume that underground temperature is in its peak in range of 14 to 18 °C. This range is typical for underground depth of 12 ft and more. we set initial value of the demand around half of the peak in the coldest day which is 7.5 MWh. Household electricity price in first quarter of 2021 is around 1200 NOK/MW in Norway which could be a base for cooling price. For heating purpose, we use natural gas fire boiler. We set natural gas price is around 20.9 NOK/MMBTU or around 70 NOK/MWh.

4.1.3 Decision Variables

The set of decision variables ($Q_t{}^{b,h,c}$) includes all energy flow allocations which made everyday by controller. Heating of the buildings can take place by the heater and borehole in our model. Similarly cold could be supplied by borehole and cooler. Cold is normally the inverse of the heat. Therefore cooling of the building by borehole is considered to be as flowing of the heat from buildings to borehole. Taking above points to the account, set of decision variables is as following:

$$X_t = (Q_t{}^b, Q_t{}^h, Q_t{}^c) \tag{29}$$

$Q_t{}^b$: The amount of heat and cold which is provided by borehole on day t in MWh/Day

$Q_t{}^h$: The amount of heat which is provided by heater on day t in MWh/Day.

$Q_t{}^c$: The amount of cold which is provided by cooler on day t in MWh/Day.

4.1.4 Constraints

In our model state variables and actions, which were specified in previous parts, are subject to some constraints as following:

Due to maximum amount of water that can be circulated by pump and assuming 4 °C as approach temperature of inlet and outlet flow of the water to the evaporator of the heat pump, maximum flow of energy the borehole to buildings and inversely from buildings to the borehole is limited up

to a certain value. Due to this limitation there is a maximum value for the energy that can be taken from or given to the borehole. Here we assumed that this maximum is 5 MWh. Therefore:

$$0 \leq {Q_t}^b \leq 5MWh \tag{30}$$

As it was mentioned in basic model, we are interested in controlling the borehole temperature between 0 and 12 °C. Therefore the second constraint is:

$$0°C \leq {T_t}^b \leq 12°C \tag{31}$$

In addition we cannot have the negative values for our energy flows, therefore next constraint is that all energy flows have positive values:

$${Q_t}^b, {Q_t}^h, {Q_t}^c \geq 0 \tag{32}$$

Energy flows should be controlled in such a way that can meet the demand everyday. Here we assumed that we do not heat and cool the building at same time. Therefore we can write this constraint as following formula:

$$D_t = (\frac{COP}{COP-1}{Q_t}^b + {Q_t}^h) * 1\_\{D_t \geq 0\} + ({Q_t}^b + {Q_t}^c) * 1\_\{D_t \prec 0\} \tag{33}$$

Which means that our daily heating demand shall be equal to sum of heating flows from both borehole and heater and our daily cooling demand shall be equal to sum of the removing heat by both borehole and cooler.

4.1.5 Exogenous Information

We let the ω_t be the vector of our exogenous information. These are random variables or disturbances that we do not know anything about them at each discrete time that we make a decision. These information enter to our model later. We consider them as the exogenous change of states and we use "hat" sign to indicate these variables. In our model they are as following:

$\hat{D}_t$: The change of demand between t and t+1 i.e. two different days in MWh

$\hat{T}_t^{inf}$: The change of underground temperature between t and t+1 i.e. two different days in °C.

$\hat{P}_t^h$: The change of heating price by heater between t and t+1, i.e. two different days in NOK/MWh.

$\hat{P}_t^c$: The amount of cooling price change by cooler between t and t+1, i.e. two different days in NOK/MWh.

It is very critical to estimate a good model for probability distribution of the exogenous variables. This helps us to stay near optimal point in Lookahead Approximations or Value Function Approximation. We will discuss it in chapter 7.

4.1.6 State Transition Function

The transition function $S^M(S_t, X_t, \omega_{t+1})$ specifies the transition of the states from t to t+1 and represents the dynamic of the model. The set of transition functions of the states, which includes exogenous information, is obtained by following equations in our model:

$$D_{t+1} = D_t + \hat{D}_t \tag{34}$$

$$P_{t+1}^{h} = P_{t}^{h} + \hat{P}_{t}^{h} \tag{35}$$

$$P_{t+1}^{c} = P_{t}^{c} + \hat{P}_{t}^{c} \tag{36}$$

$$T_{t+1}^{\inf} = T_{t}^{\inf} + \hat{T}_{t}^{\inf} \tag{37}$$

In order to obtain transition function of the borehole temperature, we have to write an energy balance over the borehole:

$$mC_p \frac{dT^b}{dt} = \lambda(\mathrm{T}^{\inf}\text{-}\mathrm{T}^b) \pm Q^b \tag{38}$$

Here m is the mass of water in the borehole and C_p is the heat capacity of water. The sign $\pm$ before Q^b shows the direction of the heat flow. It can be from borehole to demand in case of heating or inverse in case of cooling. λ represents the overall heat transfer coefficient of borehole with underground. Based on our model assumption, we need a borehole network with low λ to minimize the loss, because we are interested that borehole can store the thermal energy inside. After discretizing and rearranging above equation we have the following equation for transition of the borehole temperature:

$${T_{t+1}}^b = (1 - \frac{\lambda}{mC_p}){T_t}^b + \frac{\lambda}{mC_p}\mathrm{T}_t^{\inf} \pm \frac{1}{mC_p}Q^b \tag{39}$$

4.1.7 Objective Function

Our cost function characterizes our system and helps us to know that how much we are far from the optimal operation. We assume that the cost of using borehole for heating/cooling purpose can be neglected while compared to heater and cooler. Therefore we have to pay every time that we use heater or cooler due to their energy consumption.

Therefore our stage cost can be written as below equation:

$$C(S_t, X_t) = {Q_t}^b {P_t}^h + {Q_t}^c {P_t}^c \tag{40}$$

And the total cost over a year is equal to:

$$Total\,\mathrm{Cos}\,t = \sum_{t=1}^{365} C(S_t, X_t) \tag{41}$$

We are interested to minimize the annual energy cost. Therefore our objective function is as following formula:

$$\min_{\pi}\ \mathbb{E}[\sum_{t=1}^{365} C(S_t, X_t)|S_0] \tag{42}$$

Our objective is to apply a policy π that minimizes this cost and we have to choose it from $\pi \in \Pi$. Π is the set of all possible policies that meet the constraints within entire control period.

4.2 Exogenous Information

In this section we present the models of the exogenous information processes. There are four sources of randomness in our case study by design. The first is underground temperature (T_t^{inf}) which can impact on our borehole temperature by heat exchange. The other exogenous information are demand, heating price and cooling price. In this section we provide with the formulas used to model the variations of underground temperature, demand and prices.

4.2.1 Price Model

Electricity prices are stochastic and might be updated daily, hourly and in some cases even every 5 minutes. In this work, we assume that heater and cooler prices follow first order Markov chain model. we use following formula and represent the randomness by Guassian noise as:

$$P_t^{h,c} = \max\left\{P_{\min}^{h,c}, P_{avg}^{h,c} + normrnd(0, P_{std}^{h,c})\right\} \tag{43}$$

In the above model prices fluctuates around an average value ($P_{avg}^{h,c}$) every day while they cannot be lower than a minimum value ($P_{min}^{h,c}$) . The standard deviation of both prices is denoted by $P_{std}^{h,c}$. These values are presented in Table 3.

Variable	Value	Unit
P_{avg}^{h}	70	NOK/MWh
P_{avg}^{c}	1200	NOK/MWh
P_{std}^{h}	5	NOK/MWh
P_{std}^{c}	300	NOK/MWh
P_{min}^{h}	20	NOK/MWh
P_{min}^{c}	500	NOK/MWh

Table 3: Values of average operating prices, minimum prices and standard deviation of randomness in prices for both heater and cooler

Fig. 4 provides a typical profile of prices. It can be seen that how the profile of stochastic prices over a year looks like.

4.2.2 Demand Model

We assume that control starts from first day of the November. we set half of the maximum as initial demand at this time. Our demand continues to increase and peak at a maximum and then it declines over the winter. Alternatively when summer starts the demand passes zero and becomes negative which means that cooling is required by the demand. As the summer days go ahead the negative demand increases, touches a minimum and then returns to its initial value. We characterize this model with a sinusoidal function over the year. We also assumed that duration of heating is around 9 months and we provide the cooling just for rest of the year, i.e. 3 months. Moreover, we consider a higher absolute value for heating compared to cooling which quite complies to Norway weather condition. In order to take randomness into account in daily demand, we add a Gaussian noise to both winter and summer demand functions as following:

$$D_t^w = -A_d^w \sin(\frac{0.85\pi t}{270} + \frac{7\pi}{6}) + normrnd(0, D_{std}^w) \tag{44}$$

$$D_t^s = -A_d^s \sin(\frac{\pi t}{95}) + normrnd(0, D_{std}^s) \tag{45}$$

Here D_t^w and D_t^s represent daily demand in winter and summer, respectively. We set 15 MWh as the maximum daily heating demand and 7 MWh for that of cooling. Therefore we set following values to the parameters in the demand equations:

A_d^s: 15 MWh which is Maximum value of heating demand

A_d^w: 7 MWh which is Maximum value of cooling demand

D_{std}^w: 3 MWh which is standard deviation of heating demand, i.e. 20 percent of the maximum heating

D_{std}^s: 1.4 MWh which is standard deviation of cooling demand, i.e. 20 percent of the maximum cooling

Fig. 4 provides that how the stochastic price demand over a year looks like.

4.2.3 Underground Temperature Model

Underground temperature, and in consequence, heat exchange between borehole and underground depends to different factors such as geographical location, weather variation, depth of installation and type of the soil. A number of quantitative models from geothermal low activity areas (i.e. on stable platforms outside tectonic and volcanically active areas) show that at shallow depths down to a few hundred meters, mean annual surface temperature is the main factor that controls subsurface temperature. But most of the resource agree that it fluctuates between 10 °C and 25 °C. In this model we assume that this temperature varies in a range between 14-18 °C and in first days of the winter (first day of November in our case study) it is equal or very close to its maximum value. When days go ahead through a year, it reaches a minimum and then returns again to the maximum after 365 days. These variation is simulated by a sinusoidal function as following:

$$T_t^{\inf} = T_{avg}{}^{\inf} - A^{\inf} \sin(\frac{2\pi t}{365} - \frac{\pi}{2}) + normrnd(0, T_{std}{}^{\inf}) \tag{46}$$

We set following values to the parameters of above equation:

$T_{avg}{}^{inf}$: 289 °K (16 °C) which is the average underground temperature

A^{inf}: 2 °K (°C) which is the half of domain of underground temperature change

$T_{std}{}^{inf}$: 0.3 °K (°C) which is the standard deviation of randomness in underground temperature

Fig. 4 shows that how the variation of underground temperature looks like over a year.

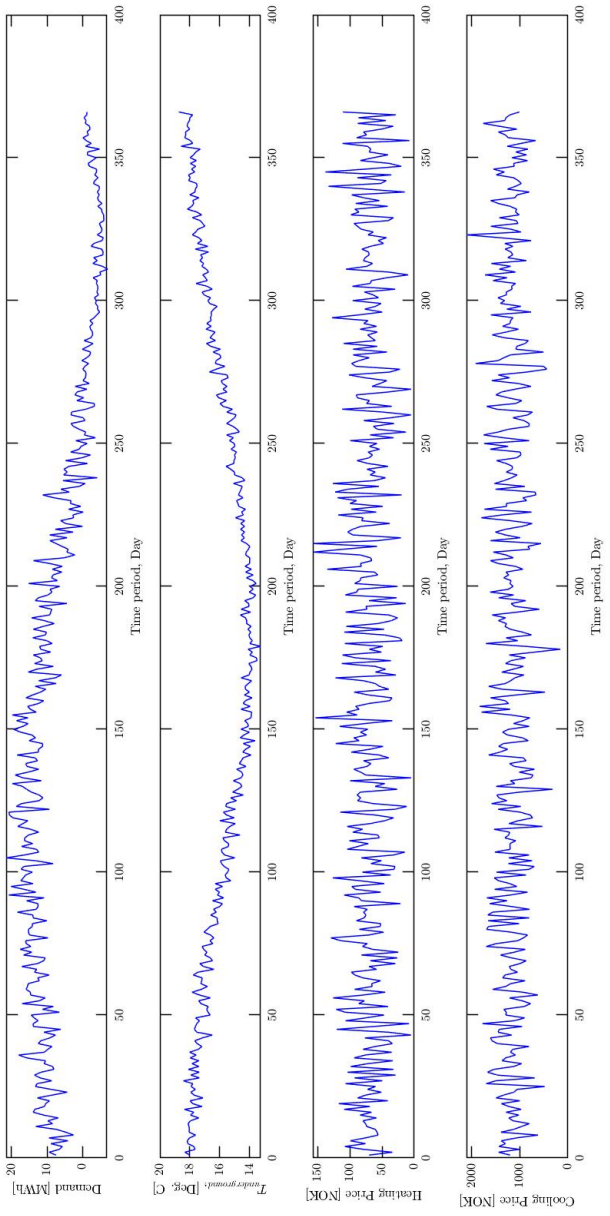

Figure 4: A sample path of exogenous information

We modeled our exogenous information as sinusoidal load, sinusoidal underground temperature function and first order Markov chain prices in the previous sections. We will set the parameterized CFA-DLA policy and optimize these parameters in our energy storage problem. In order to verify the performance of our policy we need to benchmark it against perfect forecast. The difference in total costs shows us how much we are from optimal cost under perfect forecast.

5 Designing Policies, Hybrid CFA-DLA

In this chapter we present the policies that are designed to allocate energy flows as decision variables in our case study. As mentioned before, W. B. Powell 2019 classified four different approaches for solving the stochastic optimization problem with sequential decision making as: policy function approximations (PFA), cost function approximations (CFA), value function approximations (VFA) and direct lookahead Approximation (DLA). Here we will take the parametrized version of Cost Function Approximation policy which itself is a hybrid method of both CFA and DLA. We describe that how we design this policy for our case study.

In order to test this policy and see how much the obtained cost via this policy is far from our real optimal value, we have to benchmark it against the perfect forecast. In the perfect forecast model, we assume that we know all stochastic changes of exogenous information which happen in the future. Therefore the problem can be addressed as a deterministic model and obtained cost is the real optimal cost of the optimization problem.

In order to have a solid understanding of our method, we start by presenting DLA metohd, then continue with CFA and finally describe that how these methods are integrated in our method. Based on the formulation provided in section 4, our goal is to optimize below analytical cost function:

$$\min_{\pi} \mathbb{E}[\sum_{t=1}^{365} {Q_t}^h {P_t}^h + {Q_t}^c {P_t}^c | S_0] \tag{47}$$

In order to remove the expectation sign, we take the technique mentioned in Eq. 14 and compute the average of cumulative cost over different simulation paths of exogenous price, demand and underground temperature. These exogenous information generated by the formulas presented in sections 4.2.1, 4.2.2 and 4.2.3. Now, we have to define a lookahead policy to minimize the above objective function.

5.1 Model Predictive Control

We use a deterministic lookahead policy for optimization of above cost function over a year, The deterministic lookeahad policy called called Model Predictive Control (MPC) in some resources. A deterministic lookahead model is a deterministic approximation of the base model. It is an advanced tool in process control which is used to control a process while satisfying the set of constraints each time the optimization repeated. It has been popular in the process industries since the 1980s and in recent years it has also been applied in power system balancing models and power electronics. MPC relies on dynamic models of the real process. It is based on iterative, finite-horizon open loop optimization of a system model in consecutive time steps. The current states of the system are measured every discrete time (t), and an optimal open loop control strategy is computed by a numerical solver for specified prediction time horizon, i.e. from time t to H in the future [t,t+H]. At the same time an online calculation is performed to explore state trajectories that emanate from the current state. After the set of actions is calculated by solver, only the first action is fed into the system and the set of states is updated. Then the states of the system are measured again and the calculations are repeated starting from the new current state with a new control and new predicted state path. Here the prediction horizon keeps being shifted forward. This approach is not always optimal but in practice it has given very good results. Fig. 5 shows a diagram of MPC control and demonstrates that how it works.

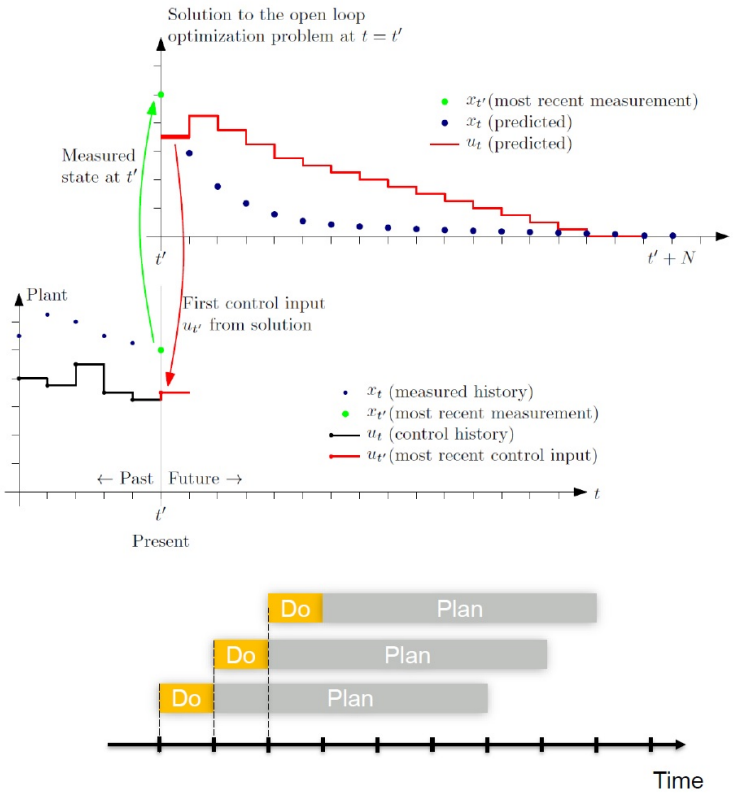

Figure 5: Model Predictive Control (MPC) Diagram

Our goal is to use above method to solve the optimization Eq. 47. At every discrete time we use lingprog solver to compute all energy flows and trajectories in whole prediction horizon and the first set of these energy flows update the set of states by transition functions. It is obvious that the formulation in Eq. 47 cannot be solved with lookeahed method due to presence of uncertainty in the form of future prices, demands and underground temperature which affect our system. Therefore we need to find a way to make forecast for these variables and solve the problem. We denote all of the forecast variables in the lookahead model with tildes and index them by both the time t at which we are making our decision and time $t^{'}$ representing a time in future. Now the variables become in the following forms:

- The set of decisions vector to be executed at time $t^{'}$ in the forecast horizon being made at time t $\left(\tilde{x}_{t,1}, \tilde{x}_{t,2}, \tilde{x}_{t,3}, ..., \tilde{x}_{t,t'}, .., \tilde{x}_{t,t+H}\right)$
- The set of states vector to be calculated at time $t^{'}$ in the forecast horizon being made at time t $\left(\tilde{S}_{t,1}, \tilde{S}_{t,2}, \tilde{S}_{t,3}, ..., \tilde{S}_{t,t'}, .., \tilde{S}_{t,t+H}\right)$
- The stage cost vector based on the above actions and states to be calculated at time $t^{'}$ in the forecast horizon being made at time t $(\tilde{C}_{t,t'})$

Our deterministic lookahead policy $X_t^{DLA}(S_t)$ is defined by following linear formula:

$$X_t^{DLA}(S_t) = \arg \min_{x_t, \tilde{x}_{t,t'} where\ t' \in [t+1, t+H]} \left[C(S_t, x_t) + \sum_{t'=t+1}^{t+H} C\left(\tilde{S}_{t,t'}, \tilde{x}_{t,t'}\right) \right] \tag{48}$$

Where we have:

$$C(S_t, x_t) = Q_t^h P_t^h + Q_t^c P_t^c \tag{49}$$

$$C(\tilde{S}_{t,t'}, \tilde{x}_{t,t'}) = \tilde{Q}^h_{t,t'} f^{P^h_{t,t'}} + \tilde{Q}^c_{t,t'} f^{P^c_{t,t'}} \tag{50}$$

Now, we have to solve optimization problem under the constraint mentioned in Eq. 30 to 33 and following constraint for $t' \in [t+1, t+H]$ forecast horizon:

$$0 \leq \tilde{Q}^b_{t,t'} \leq 5\ MWh \tag{51}$$

$$0\ ^\circ C \leq \tilde{T}^b_{t,t'} \leq 12\ ^\circ C \tag{52}$$

$$\tilde{Q}^b_{t,t'}, \tilde{Q}^h_{t,t'}, \tilde{Q}^c_{t,t'} \geq 0 \tag{53}$$

$$\tilde{D}_{t,t'} = (\frac{COP}{COP-1}\tilde{Q}^b_{t,t'} + \tilde{Q}^h_{t,t'}) * 1\_\{\tilde{D}_{t,t'} \geq 0\} + (\tilde{Q}^b_{t,t'} + \tilde{Q}^c_{t,t'}) * 1\_\{\tilde{D}_{t,t'} \prec 0\} \tag{54}$$

This linear program allows us to supply for our forecasts of demand. However, this strategy still cause that the cheap energy is not being used if demands are higher than forecast or, from the other side. On the other hand energy is wasted if our point forecasts are higher than demand. Ghadimi et al. 2019 used a series of recursive equations to realistically model the evolution of the amount of wind energy generated using rolling forecasts. The main attribute of this model is that it helps us to manipulate forecast error and control the quality of the our forecast in the optimization problem. In this thesis, we adopt this technique to simulate the rolling forecasts of the exogenous information in our model and calculate that how altering forecast errors affects our policy's ability to make sequential decisions.

5.2 Developing Forecast Error

Here we use some recursive equations to build a realistic model of the stochastic process describing the changes of both prices, demand and underground temperature. We define $f^{P^h}_{t,t'}$, $f^{P^c}_{t,t'}$, $f^{D}_{t,t'}$ and $f^{T^{\inf}}_{t,t'}$ as the forecast of the heating price, cooling price, demand and underground temperature at every discrete time respectively. We have:

$$f^{P^h}_{t+1,t'} = f^{P^h}_{t,t'} + \varepsilon^{P^h}_{t+1,t'} \quad t = 0, 1, ..., T-1 \qquad t' = t+1, ..., \min(t+H, T) \tag{55}$$

$$f^{P^c}_{t+1,t'} = f^{P^c}_{t,t'} + \varepsilon^{P^c}_{t+1,t'} \quad t = 0, 1, ..., T-1 \qquad t' = t+1, ..., \min(t+H, T) \tag{56}$$

$$f^{D}_{t+1,t'} = f^{D}_{t,t'} + \varepsilon^{D}_{t+1,t'} \quad t = 0, 1, ..., T-1 \qquad t' = t+1, ..., \min(t+H, T) \tag{57}$$

$$f^{T^{\inf}}_{t+1,t'} = f^{T^{\inf}}_{t,t'} + \varepsilon^{T^{\inf}}_{t+1,t'} \quad t = 0, 1, ..., T-1 \qquad t' = t+1, ..., \min(t+H, T) \tag{58}$$

where any of the above $\varepsilon_{t+1,t'}$ represents the level of forecast noise and their distribution depends on forecast. The vector of ε_t for each of exogenous information is defined by following:

$$\varepsilon_{t,t'} = \begin{bmatrix} \varepsilon_{t,t+1} \\ \varepsilon_{t,t+2} \\ . \\ . \\ \varepsilon_{t,\min(t+H,T)} \end{bmatrix} \tag{59}$$

We build this noise by constructing a symmetric matrix $\sum \in \mathbb{R}^{H\times H}$ in a way that $\sum(i,j) = \sigma_E^2 e^{-\alpha|i-j|} \forall i,j$ and $\sigma_E, \alpha > 0$ are constant real numbers. By this formulation we can adjust the quality of the forecast by altering σ_E. The art here is that $\sum$ acts as a covariance matrix which represents less correlation between the i-th and j-th elements when they are far from each other. Therefore our prediction is less reliable for future discrete times which are farther from the current time.

We compute normal noise vector as:

$$\varepsilon_{t+1,t'} = L_{H\times H}.Z \tag{60}$$

where $L_{H\times H}$ is the top-left H x H block of the lower triangular Cholesky decomposition of $\sum$ and $Z_t \sim \mathcal{N}(0, I_{H\times H})$, i.e. as following:

$$Z_t \sim \mathcal{N}\left(0, \sigma_E \begin{bmatrix} \sqrt{f_{t,t+1}} & 0 & . & . & 0 \\ 0 & \sqrt{f_{t,t+2}} & . & . & 0 \\ . & . & & & . \\ . & . & & & . \\ . & . & & & . \\ 0 & . & . & . & \sqrt{f_{t,\min(t+H,T)}} \end{bmatrix}\right) \tag{61}$$

Each element of above noise vector has a normal distribution with zero mean and variance of σ_E^2 with respect to the fact that $\sum = L.L^{\mathrm{T}}$ and $\sum(i,i) = 1$. The speciality of this model is that it allows us to manipulate σ_E, and thus control the quality of the forecast. In Fig. 6 and 7, we show how the noise in our price forecast increases by raising σ_E. Red lines represent real values of price and thin grey lines indicate the forecast.

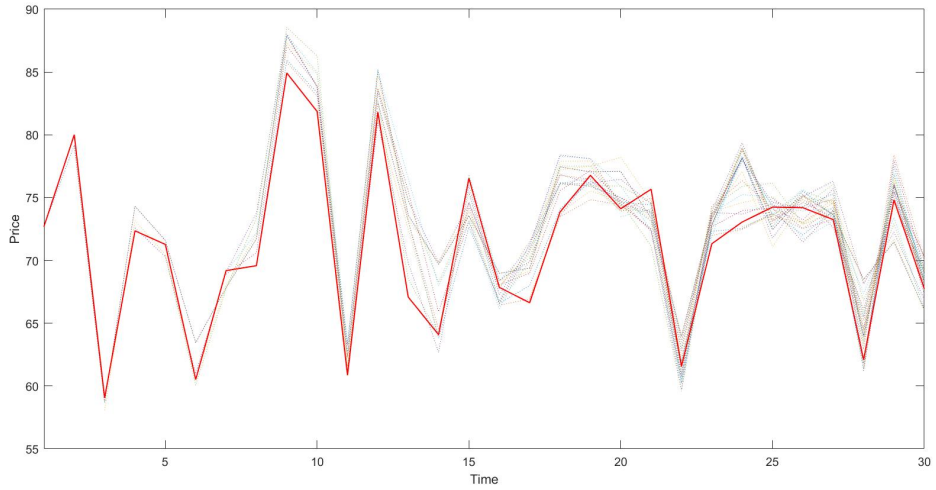

Figure 6: Evolution of price forecasts over 30 days period with $\sigma_E = 0.1$. The red line is the actual price

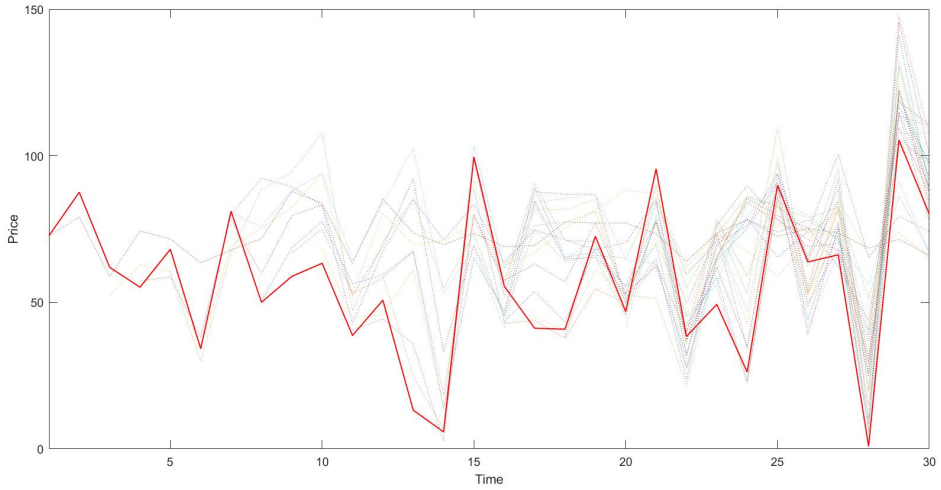

Figure 7: Evolution of price forecasts over 30 days period with $\sigma_E = 10$. The red line is the actual price

5.3 Policy Parameterization

For benchmark policy we assume that all exogenous information are known to us over entire year. We construct and run a deterministic MPC code including cost function in Eq. 47 and take the average cumulative cost over 30 simulation paths of exogenous information. Larger number of simulation paths is better, but our code becomes much more heavier due to long MPC horizon. we use this benchmark policy to evaluate the degree to which the parameterized policy is able to improve the cost in presence of uncertainty. We implement parameterization of the model to compensate for uncertainties. As mentioned before, there are different ways of parameterizing in lookahead models. Parameterization may be implemented in either constraints or transition function. The most common form is constant forecast parameterization. Here a single scalar parameter is used to modify the forecast amount of for example renewable energy for the entire horizon. We formulate two different parameterization on both transition function and demand constraint to see that which one works best.

5.3.1 Parameterization on Transition Function

In this section, we parameterize the heat transfer coefficient between borehole and underground surrounding. The reason is that it is difficult to calculate the exact values of this parameter. In addition λ represents the degree to which the borehole can charge with thermal energy by underground. First we define single scalar parameter (θ) and multiply it by heat transfer coefficient. The transition function in the prediction horizon will looks like as:

$$\tilde{T}^b_{t+1,t'} = (1 - \frac{\lambda}{mC_p})\tilde{T}^b_{t,t'} + \frac{\lambda.\theta}{mC_p}\tilde{\mathrm{T}}^{\mathrm{inf}}_{t,t'} \pm \frac{\tilde{Q}^b_{t,t'}}{mC_p} \tag{62}$$

After testing the performance of single scalar parameterization, we define 12 coordinates of θ based on 12 months of the year with the parameters $\{\theta_{Nov}, \theta_{Dec}, \theta_{Jan}, ..., \theta_{Oct}\}$. This parameterization is a lookup table representation because there is a different θ for each month in the lookahead policy. We call this monthly-based parameterization hereinafter. Here, the transition function changes to:

$$\tilde{T}^b_{t+1,t'} = (1 - \frac{\lambda}{mC_p})\tilde{T}^b_{t,t'} + \frac{\lambda.\theta_{month}}{mC_p}\tilde{\mathrm{T}}^{\mathrm{inf}}_{t,t'} \pm \frac{\tilde{Q}^b_{t,t'}}{mC_p} \tag{63}$$

Low values of coordinates of θ shows that the policy is more conservative about taking heat energy from borehole and decrease the risk of reaching to lower limits of borehole temperature. Inversely, if the coordinates of θ are large, the policy then acts more risky about maintaining heat energy of borehole.

5.3.2 Parameterization on Demand Forecast

In this set of experiments we parameterize the demand forecast and compare the cost with the one obtained in paramterization policy in previous section. Here we implement parametrization in Eq. 54. Therefore we have below formulas for single scalar and monthly-based parameterization:

$$\theta \times \tilde{D}_{t,t'} = (\frac{COP}{COP-1}\tilde{Q}^b_{t,t'} + \tilde{Q}^h_{t,t'}) * 1\_\{\theta \times \tilde{D}_{t,t'} \geq 0\} + (\tilde{Q}^b_{t,t'} + \tilde{Q}^c_{t,t'}) * 1\_\{\theta \times \tilde{D}_{t,t'} \prec 0\} \quad (64)$$

$$\theta_{month}\tilde{D}_{t,t'} = (\frac{COP}{COP-1}\tilde{Q}^b_{t,t'} + \tilde{Q}^h_{t,t'}) * 1\_\{\theta_{month}\tilde{D}_{t,t'} \geq 0\} + (\tilde{Q}^b_{t,t'} + \tilde{Q}^c_{t,t'}) * 1\_\{\theta_{month}\tilde{D}_{t,t'} \prec 0\} \quad (65)$$

Again high values of different coordinates of parameters shows that the policy is more conservative about taking heat energy from borehole and it decrease the risk of reaching to lower and upper limits of borehole temperature. Inversely, if the parameters are small the policy then acts less conservative about using borehole for heating and cooling purposes.

5.4 Policy Cost Index and Cost Reduction

In order to test the performance of this policy, we formulate the percentage of Policy Cost Index (PCI) by below formula:

$$\Delta F^{\pi}(\theta) = \frac{\mathrm{F}^{\pi}(\theta) - F^{Opt}}{F^{Opt}} \times 100 \quad (66)$$

Here $\mathrm{F}^{\pi}(\theta)$ is the minimum cost obtained by parameterization policy and F^{Opt} is the optimal cost when we assume that we have perfect knowledge of the future. The latter is sometimes called the "posterior optimal bound". The Policy Cost Index indicates that how much we are above the optimal cost. Lower values of Policy Cost Index shows that the performance of our parameterization policy is higher.

We also define the percentage of Cost Reduction by below formula:

$$CR^{\pi}(\theta) = \frac{\mathrm{F}^{DLA} - \mathrm{F}^{\pi}(\theta)}{\mathrm{F}^{DLA}} \times 100 \quad (67)$$

where F^{DLA} is the average cost generated by unparameterized deterministic lookahead policy. This is the cost when we set a certain forecast error in our system and we set all parameters to one. It shows us the maximum percentage we are able to reduce the cost by parameterization in comparison with the case that we do not implement parameterization while we have some error in prediction of the exogenous information. Both costs are obtained by taking average over an examining data set of same sample paths.

6 Evaluating Policies Performances

For each parameterization policy described in section 5, average final cost is calculated over 30 sample paths of simulated demand, heater price, cooler price and underground temperature. Daily energy flows are decided over a 365 days period starting from the first day of November and updated every day. First, we obtain the expected average cost in case of benchmark policy. Second, we present the impact of forecast error of different exogenous information on cost to see that which one is more critical. Last, we consider a noisy forecast and see the impact of parameterization on both transition function and demand forecast on cost. We present profiles of energy flows for each policy and calculate the cost showing that how much we expect to pay as annual cost of the energy consumption averaged over 30 tested trial. In the next section we will analyse and discuss these results in detail.

6.1 Benchmark Policy with Perfect Forecast

Fig. 8 shows the energy flows of our energy system in a typical sample path of exogenous information. Here σ_E is set to zero meaning that we have perfect realization about the exogenous information in upcoming discrete time. This case obviously shows perfect model and optimal control. The average annual cost is around 705,695 NOK.

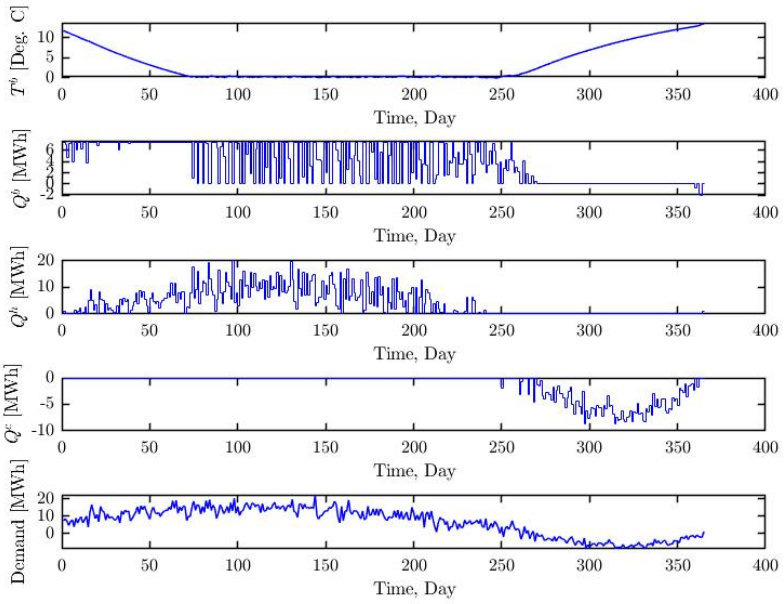

Figure 8: Profile of energy flows with error forecast $\sigma_E = 0$, monthly-based parameterization on transition function

We test the performance of the parameterization policy with different coordinates of θ under perfect forecasts. The results are presented in Fig. 9. Each coloured line represents the changes of Policy Cost Index obtained by changing one coordinate of parameter while other coordinates kept constant.

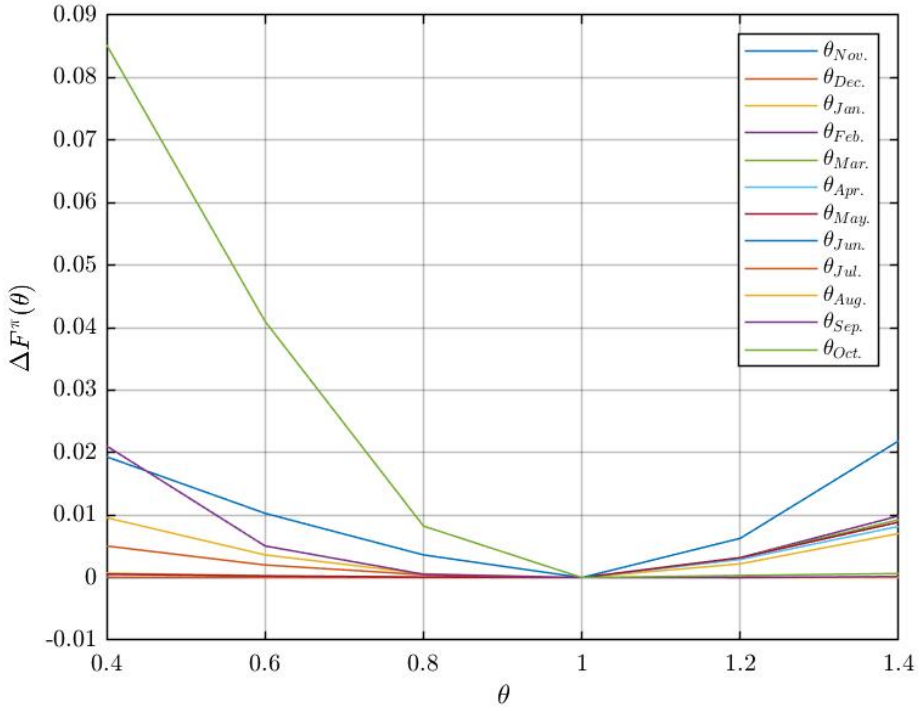

Figure 9: Average performance of lookup parameterization policy under perfect forecasts, $\sigma_E = 0$ for all types of exogenous information, monthly-based parameterization on transition

6.2 Effect of Different Forecast Error on Cost

In second set of experiments, we shift to noisy forecast. We set $\sigma_E = 15$ for each of D, P^h and P^c while it is zero for the others and do the same procedure as perfect forecast to see that which kind of exogenous information is more critical and has more impact on cost. The energy flows in a typical sample path and policy performance curves are presented in Fig. 10 to 15 respectively.

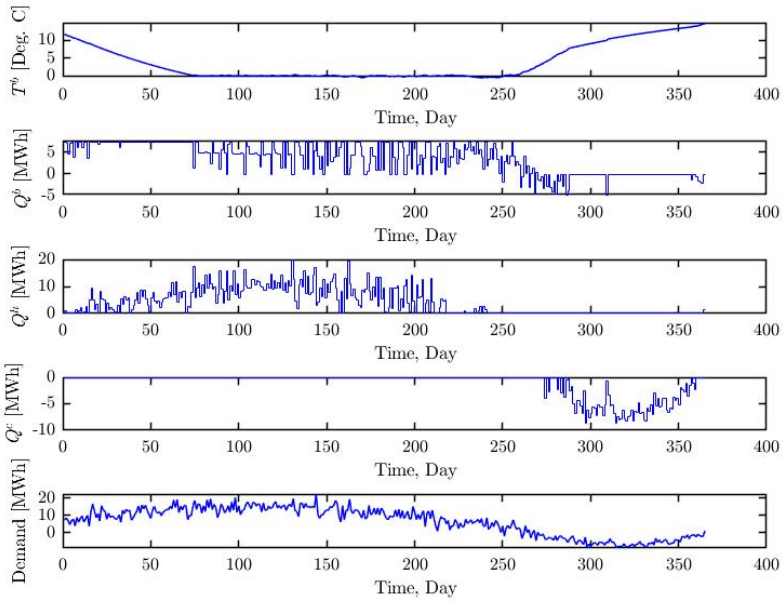

Figure 10: Profile of energy flows with error forecast $\sigma_E = 15$ for demand, monthly-based parameterization on transition function

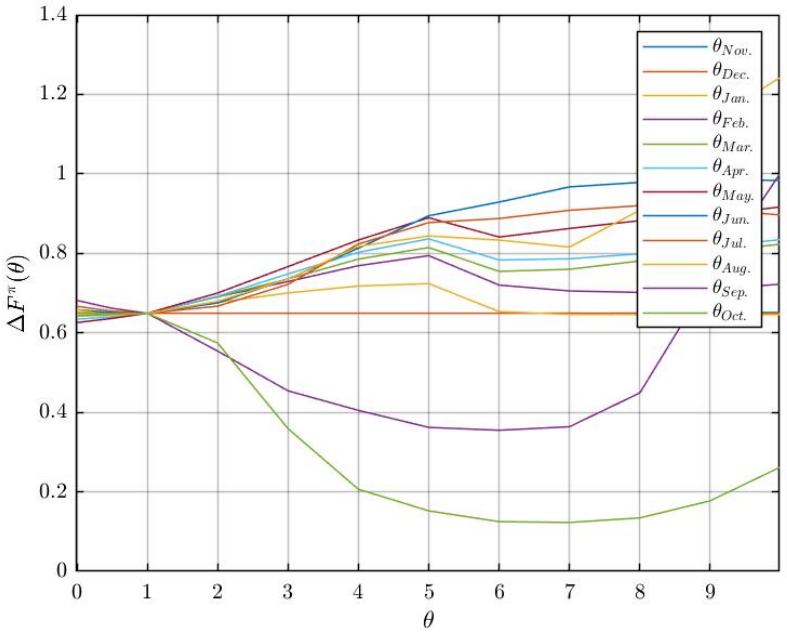

Figure 11: Average performance of lookup parameterization policy, $\sigma_E = 15$ for demand, monthly-based parameterization on transition function

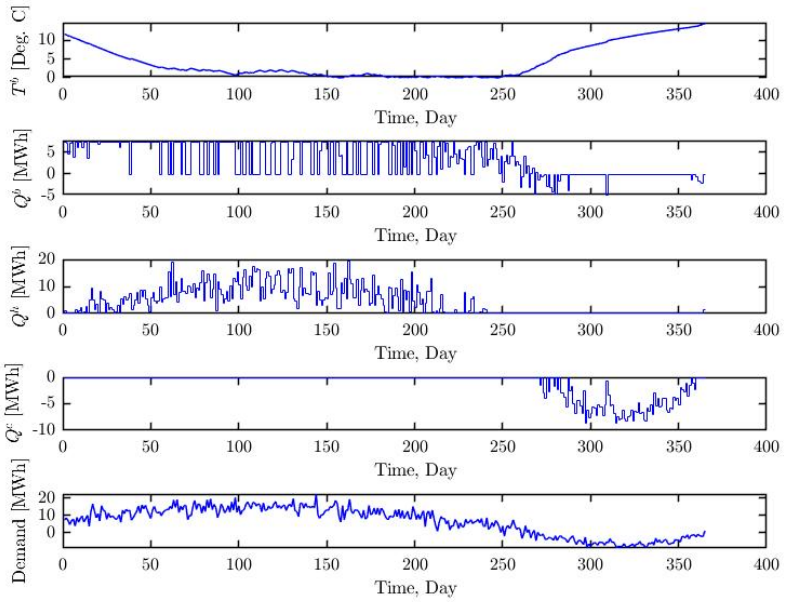

Figure 12: Profile of energy flows with error forecast $\sigma_E = 15$ for Heating Price, monthly-based parameterization on transition function

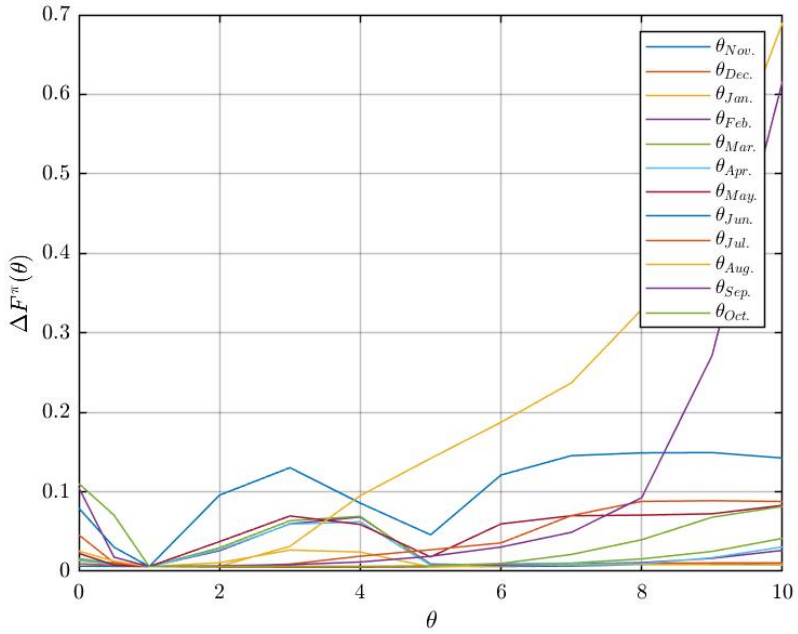

Figure 13: Average performance of lookup parameterization policy, $\sigma_E = 15$ for heating price, monthly-based parameterization on transition function

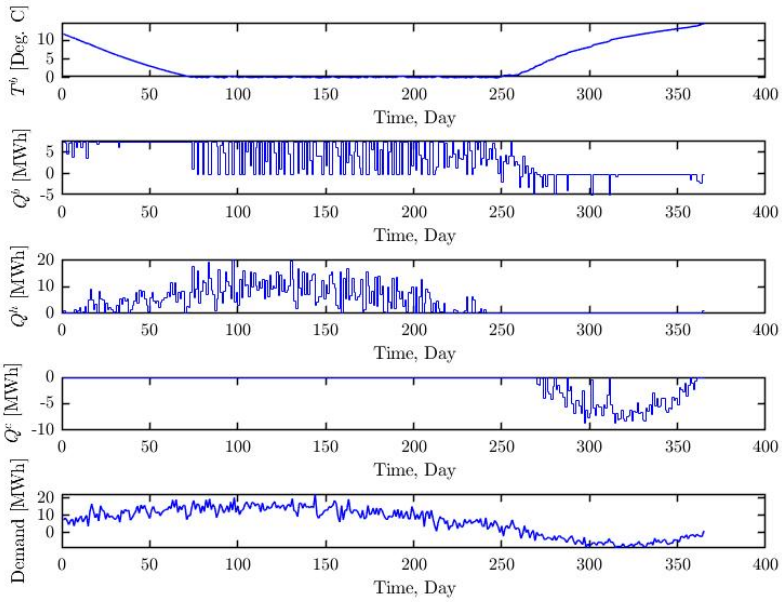

Figure 14: Profile of energy flows with error forecast $\sigma_E = 15$ for cooling price, monthly-based parameterization on transition function

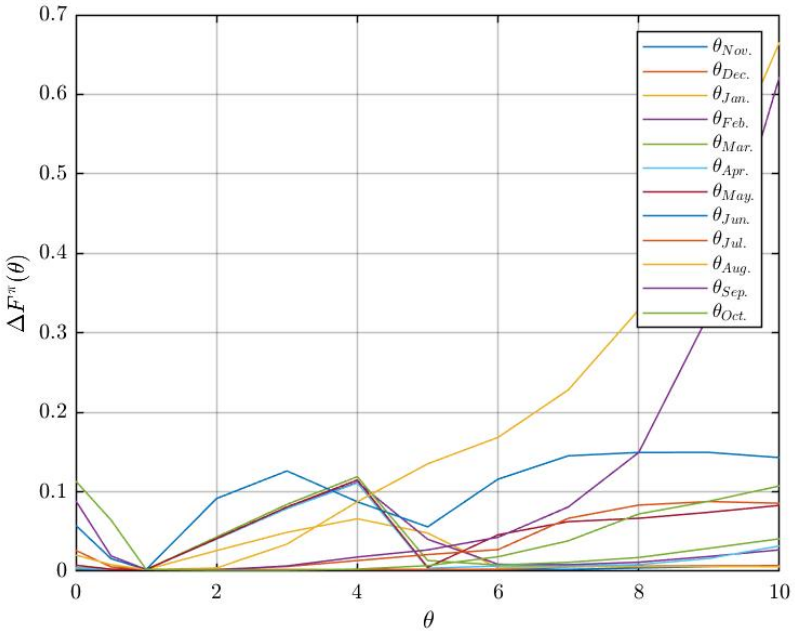

Figure 15: Average performance of lookup parameterization policy, $\sigma_E = 15$ for cooling price, monthly-based parameterization on transition function

6.3 Policy Performance of Parameterization on Transition Function

At this stage, we set $\sigma_E = 15$ for all D, P^h and P^c and we consider maximum possible forecast error on absolute value of underground temperature, i.e. $\sigma_E = 1$, and repeat the experiments. The energy flows over a typical sample path and effects of scalar parameterization presented in Fig. 16

and 17. In this case the minimum obtained cost is 1,205,647 NOK and Policy Cost Index is 70.8 %. The Cost Reduction is calculated to be as 8.8% .

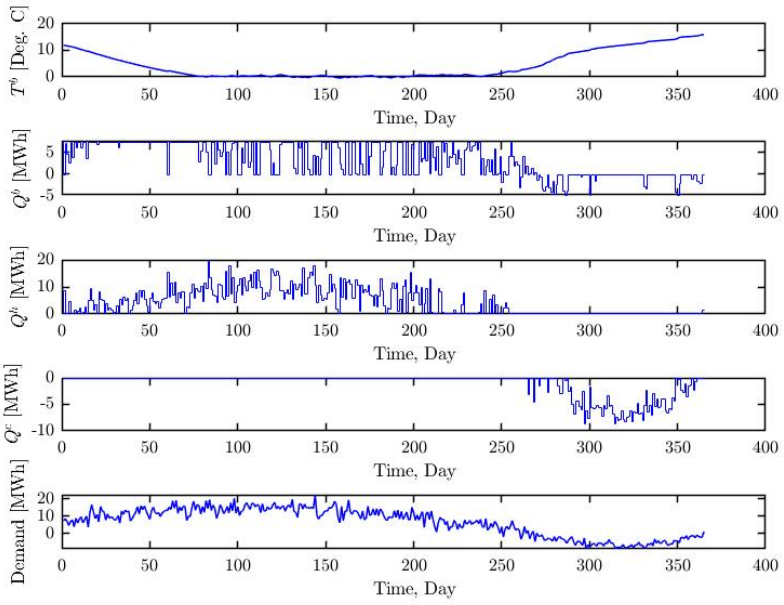

Figure 16: Profile of energy flows with error forecast $\sigma_E = 15$ for D, P^h,P^c and $\sigma_E = 1$ for T^{inf}, single scalar parameterization on transition function, $\theta = 0.4$.

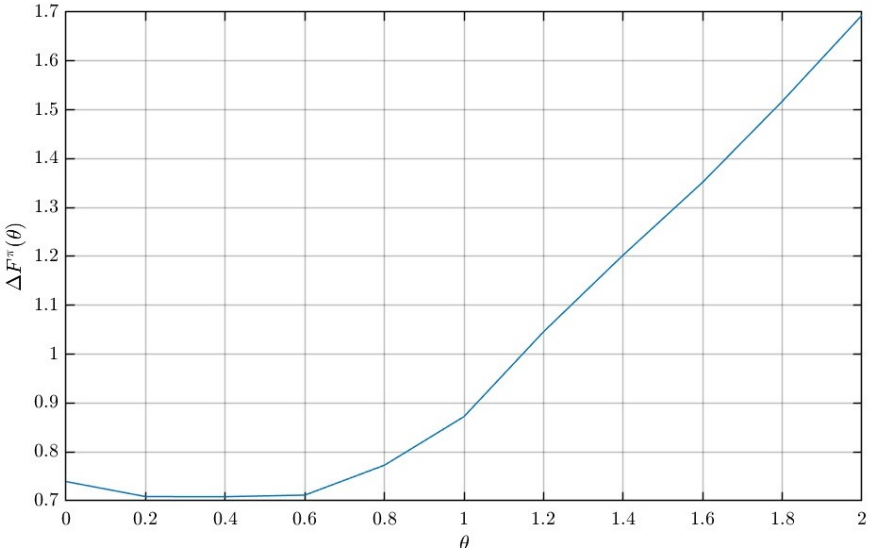

Figure 17: Average performance of lookup parameterization policy , $\sigma_E = 15$ for D,P^h,P^c and $\sigma_E = 1$ for T^{inf}, single scalar parameterization on transition function

Moreover, Fig. 18 and 19 show the results of monthly-based parameterization. In this case, the minimum obtained cost is 1,259,982 NOK and Policy Cost Index is 78.5 %. The Cost Reduction is calculated to be 4.6%.

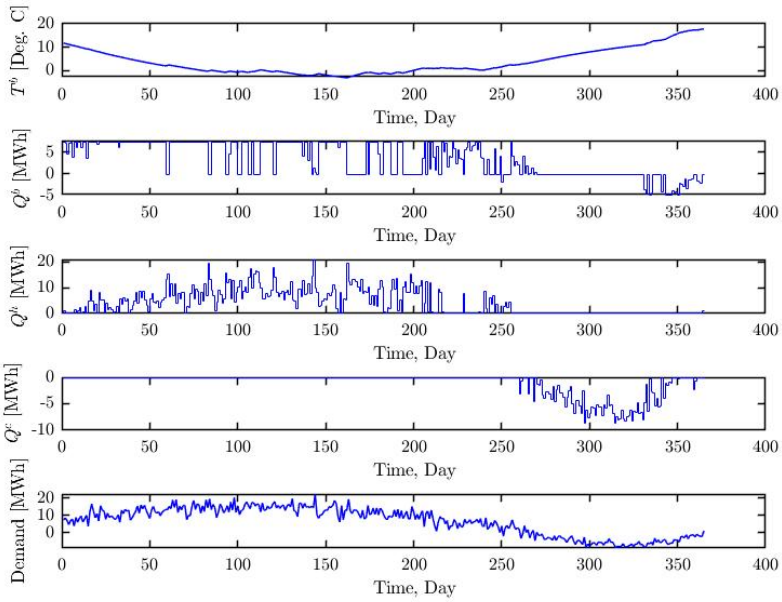

Figure 18: Profile of energy flows with error forecast $\sigma_E = 15$ for D,P^h,P^c and $\sigma_E = 1$ for T^{inf}, monthly-based parameterization on transition function

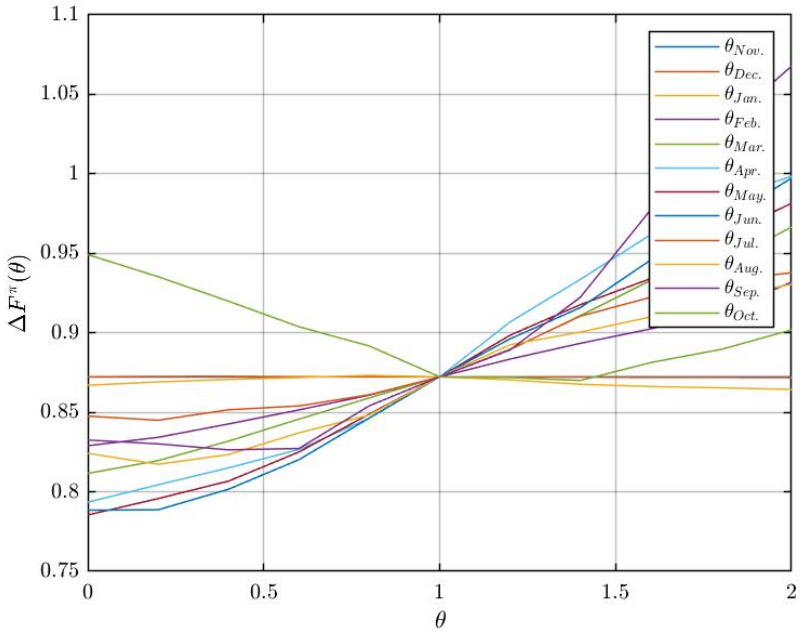

Figure 19: Average performance of lookup parameterization policy , $\sigma_E = 15$ for D,P^h,P^c and $\sigma_E = 1$ for T^{inf}, monthly-based parameterization on transition function

In another set of experiments we do a two-dimensional search over two arbitrary different coordinates of θ_{month} to check the change of cost. We select different pair of months. The lowest cost is obtained while we manipulate the coordinates of parameters in August and September. We presented just this result and ignore the others to avoid repetition. The Policy Cost Index can be seen in different ranges of θ_{Aug} and θ_{Sep} in Fig. 20 and 21. The minimum obtained cost is 1,269,832 NOK and the Policy Cost Index is 79.9 % which is almost same as previous value.

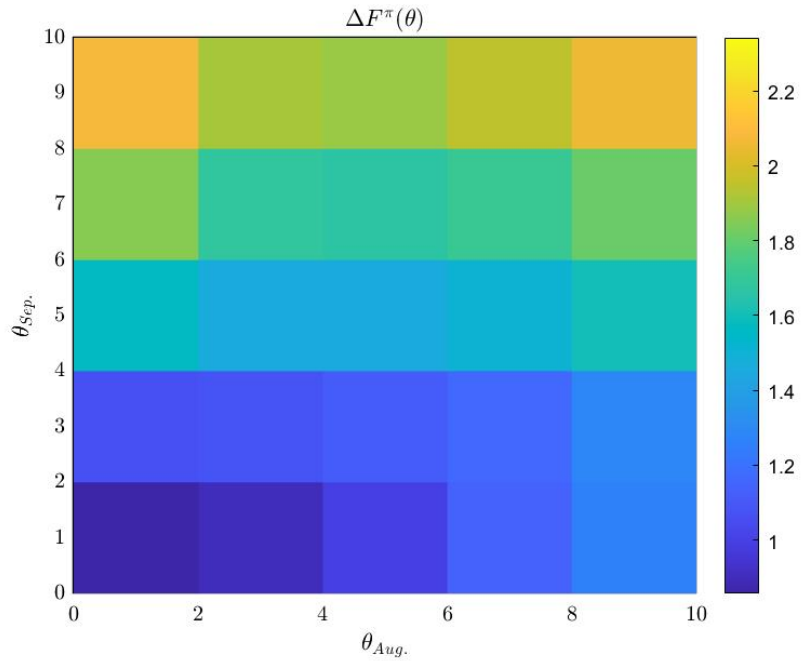

Figure 20: Two dimensional search over performance of parametrized policy under noisy forecast, $\sigma_E = 15$ for D,P^h,P^c and $\sigma_E = 1$ for T^{inf}, θ_{Aug} *and* θ_{Sep} ranges are $0 \sim 10$

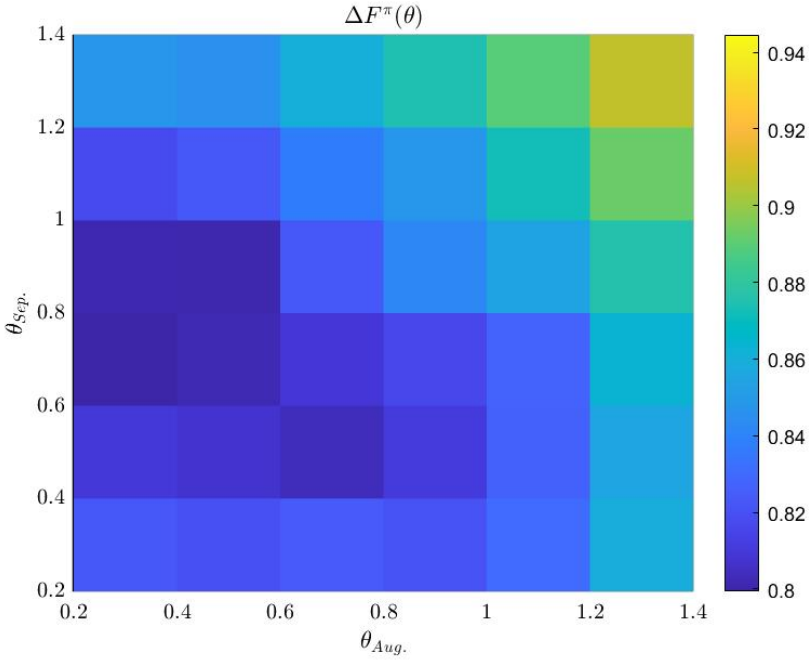

Figure 21: Two dimensional search over performance of parameterized policy under noisy forecast, $\sigma_E = 15$ for D,P^h,P^c and $\sigma_E = 1$ for T^{inf}, θ_{Aug} *and* θ_{Sep} ranges are $0 \sim 1.4$

6.4 Policy Performance of Parameterization on Demand Forecast

Now we shift to parameterization on forecast of the demand. Again we set $\sigma_E = 15$ for all D, P^h and P^c and $\sigma_E = 1$ for underground temperature while we use the formulation described in section 5.3.2. If we use single scalar parameterization, the minimum cost of 1,065,213 NOK is

obtained at $\theta = 80$. The Policy Cost Index is 51 % and the Cost Reduction is calculated to be as 24%. Profiles and parameterization curve are presented in Fig. 22 and 23, respectively. The results of monthly-based parameterization presented in Fig. 24 and 25, respectively. In this case, the minimum obtained cost is 1,220,939 NOK and Policy Cost Index is 73 %. The Cost Reduction is calculated to be as 7.6%.

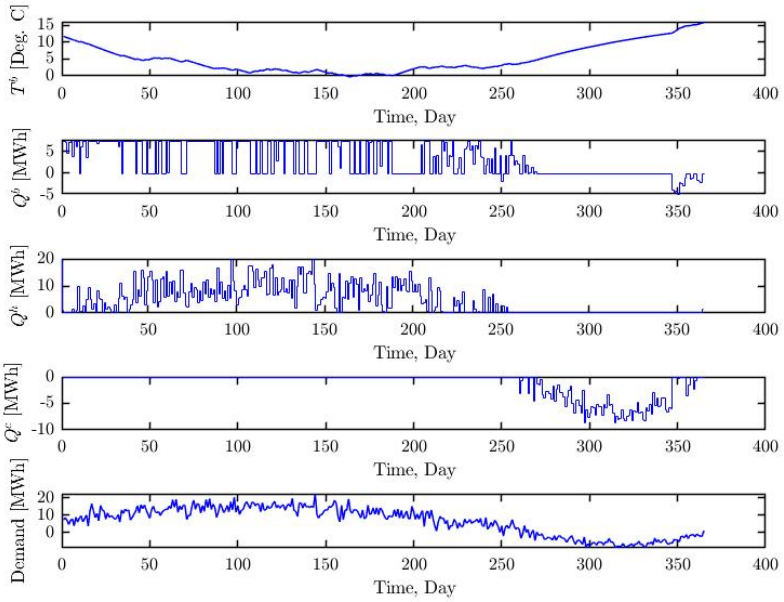

Figure 22: Profile of energy flows with error forecast $\sigma_E = 15$ for D,P^h,P^c and $\sigma_E = 1$ for T^{inf}, single scalar parameterization on demand forecast, $\theta = 80$.

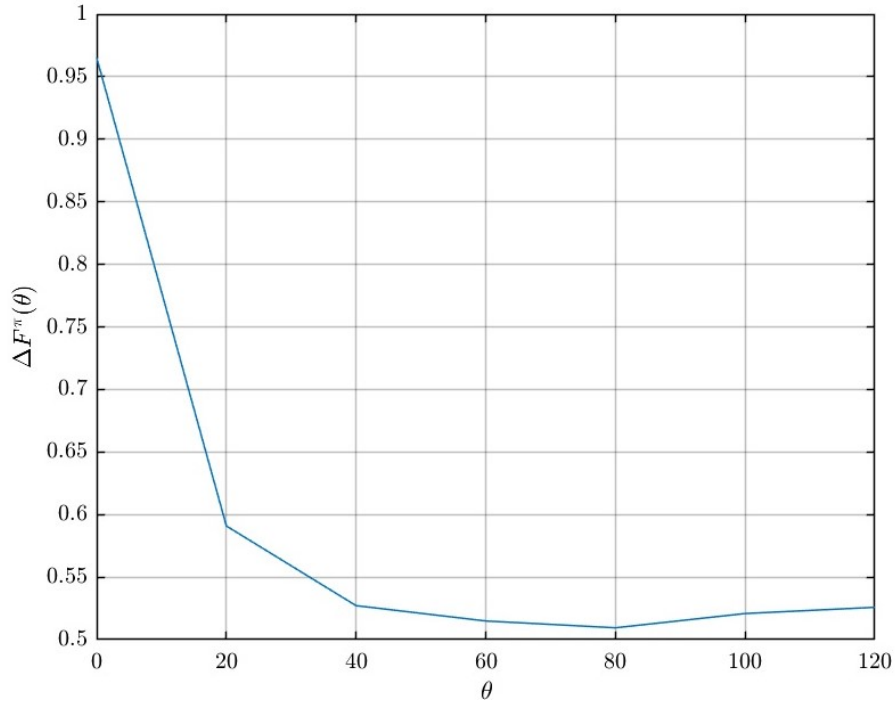

Figure 23: Average performance of lookup parameterization policy , $\sigma_E = 15$ for D,P^h,P^c and $\sigma_E = 1$ for T^{inf}, single scalar parameterization on demand forecast

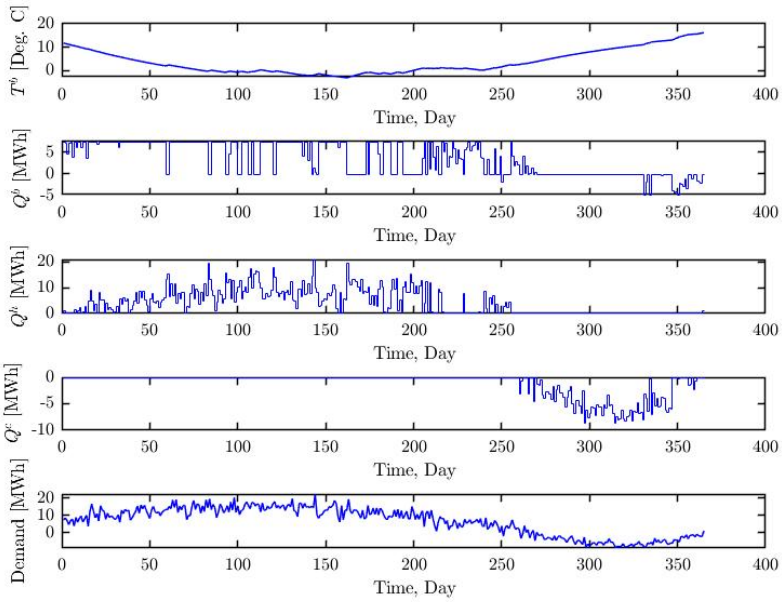

Figure 24: Profile of energy flows with error forecast $\sigma_E = 15$ for D,P^h,P^c and $\sigma_E = 1$ for T^{inf}, monthly based parameterization on demand forecast

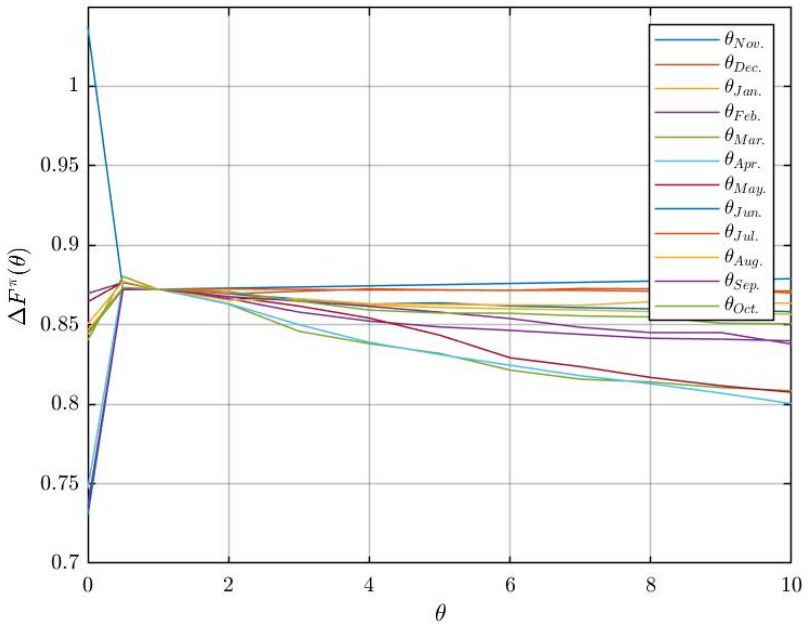

Figure 25: Average performance of lookup parameterization policy , $\sigma_E = 15$ for D,P^h,P^c and $\sigma_E = 1$ for T^{inf}, monthly-based parameterization on demand forecast

6.5 Comparison of the Results

We provide a summary of the results showing the absolute minimum value of cost and Policy Cost Index obtained in each type of the parameterized CFA Policy in Table 4.

Method	Cost	Policy Cost Index
Scalar Parameterization on Transition Function	1,205,647 NOK	70.8 %
Monthly-Based Parameterization on Transition Function	1,259,982 NOK	78.5 %
Scalar Parameterization on Demand Forecast	1,065,213 NOK	51 %
Monthly-Based Parameterization on Demand Forecast	1,220,939 NOK	73 %

Table 4: Summary of parameterization results

While the condition and sample paths of exogenous information were same for all above methods, the lowest possible cost obtained while we chose scalar parameterization on demand forecast and the highest one was parameterization of transition function while we set different coordinates of parameter θ.

7 Discussion

This chapter includes discussion about results of the parameterized CFA-DLA policy. We explain the difficulties with the model and some simplifications that we made and also the reality of this model and suggest possible future extension works of this case study.

7.1 Performance of Parameterized CFA

If we check the curves presented in Fig. 9, we see that in case of perfect forecast the optimized value for all coordinate of θ was equal to 1 as it is expected. When we set forecast error to zero and parameters to one, this is simply indicates the optimal cost. In the next set of experiments, we tested the importance of each type of exogenous information in value of the cost. In each experiment, we set 15% error in forecast of one type of exogenous information and set the forecast error of other exogenous information to zero and presented the results in section 6.2. we see that the error in demand forecast is sensibly more critical and results in higher cost. The reason is that we set the MPC horizon equal to control horizon in our problem. Therefore, forecast error in demand will cause excessive heat exchange between borehole and demand and, in consequence, more severe constraint violation.

When we used noisy forecast for all types of exogenous information in our model, the optimal values for each coordinates of θ was different than 1. In case of monthly-based parameterization on transition function, we obtained the minimum value of total cost while we set all coordinates of θ to 1 except θ_{May} which is 0. In this case, the total cost was equal to 1,259,982 NOK and percentage of Policy Cost Index was around 78.5%. The performance of parameterization in transition function was not so much high. When we used single scalar and monthly-based parameterization in transition function, we recorded just 8.8% and 4.6% lower cost compared to unparameterized lookahead policy, respectively. The reason is the low heat transfer between borehole and underground, because we were interested to keep the borehole temperature in a certain range. When we compare results of single scalar and monthly-based parameterization, we see that the first method yields higher performance. This is due to the fact that the control horizon is very long and parameterization over a part of horizon is not as effective as enough. In case of monthly-based parameterization, we generally see that the effect of this policy is very low when we parameterize the first coordinates of θ and cost barely change with change of parameters. The range of change of the cost increase when we implement parameterization in next coordinates of θ. The reason is that our system is subject to receiving continuous heat from underground. When we parameterize the first coordinates of θ, the optimizer can plan for protecting the system from violation of upper bound of borehole temperature at the end of the control period and keep the cost low when we change the parameters of lookahead model in first days of control; but when we shift to parameterization of next coordinates of θ, the borehole temperature is more and more closer to its upper limit. Therefore any change by parameters would affect constraint violation and consequently more impact on final cost. In this parameterization method we performed a two dimensional grid search over some arbitrary pairs of coordinates of θ but we did not obtain a lower cost than the one in monthly-based parameterization in section 6.3. We presented the shape of the response surface by performing the two-dimensional grid search for coordinate pair of θ_{Aug} and θ_{Sep}, while we kept the other coordinates fixed at one. Our results in Fig. 21 shows almost the same minimum value of cost similar to case of one dimensional search, but here coordinate's values are different, i.e. θ_{Aug} =0.2 and θ_{Sep} =0.6. Graphs can be drawn for any pair of coordinates of parameter and each one have limits on which changing the coordinates does not improve the the result anymore. The shape of the ridges can be quite different from one pair of coordinates to another one, while most of them share some kind of unimodularity.

In case of parameterization of demand forecast, we obtained even better results. It was expected due to sensitivity of the cost to the demand forecast which was mentioned earlier. We did not perform two dimensional grid search in this policy as it was very time consuming according to 12 coordinates of θ and we could not obtain lower cost by two dimensional grid search. The single scalar parameterization yielded lower cost which is justified due to long horizon again. We see that by increasing value of the scalar parameter, the optimizer acts more conservative about using

borehole in order to avoid penalty of constraint violation. The rate of change is sharp and then becomes moderate as it is seen in Fig. 23. After reaching a certain minimum, the cost rises again but at a slower rate. This is due to dominating the cost of thermal energy supply by heater and cooler in total cost.

Our results shows that the parameterization is better to be implemented on the uncertainty types which are more critical in our system and the quality of the parameterization depends to the model. Therefore, understanding the model characteristics and relationship between different subordinates of the model increase the performance of parameterization. In thiscase study, we obtained 24 % lower cost by single scalar parameterization of demand forecast compared to unparameterized lookahead model.

7.2 Simplifications and Challenges of the Model

Generally stochastic control of all energy storage case studies need some assumption to simplify the model as we did the same in this thesis. Nevertheless, it is very difficult and sometimes even not possible to solve this kind of problems because an energy storage problem is a kind of large state space problem and without simplifications the complexity of computations increase sharply. Here we assumed constant COP, but it depends on heat flow of hot / cold sides in reality and therefore it might undergo some variations based on the temperatures of both sides and flow rate of circulating medium. If we take COP variation into account in our model, our linear problem will convert to a nonlinear one. This matter not only make the optimization problem harder to solve, but also increase the cost of computation. However, our assumption is not far from reality because the mentioned variations are not huge. Parameterized CFA is an offline method and computation is performed before control and then best value of parameters is searched thorough all possible values. In our case study the speed of calculation in Matlab was low and time consuming and due to long control horizon, it took a lot of time to obtain the cost for possible values of parameters especially in case of monthly-based parameterizations. We recommend that it is better to adjust shorter control horizons and more powerful programming tools.

In terms of modelling, We assumed perfect mixing and ignored modelling of the heat exchange between returning water from heat pump and borehole. The recirculating water is a small faction of whole borehole in reality and borehole acts like as a container. Thus, this water might not affect the temperature of the borehole fast. However, perfect mixing is the worst case in our model and borehole temperature is changing somewhat slower than the model which is in favor of our system.

7.3 Suggestion for Further Studies

In this thesis, we checked the performance of parameterized CFA policy integrated with MPC method considering forecast error. We compared it with optimal policy and tested that how much the different values of parameters affect the expected cost. However, some improvements still can be made in terms of both modeling and policies in this case study. Some ideas are as below:

• The borehole can be modelled with more detail. Instead of heating / cooling values of energy, we can extend the model to flow rate and temperature variations of both sides of heat pump. However, we recommend lower control horizon in order to speed up the calculations.

• A good approach is to use machine learning methods to obtain the the best values of parameters online. Improvement can be made on rate of convergence of parameters to optimal value.

• Different price and demand forecast patterns can be implemented in this case study to test the performance of the policy in different situations.

8 Conclusion

We present the conclusion of our work in this chapter. We modeled an energy storage problem in which the required energy by a demand is supplied by a borehole district heating system as energy storage device in parallel to fire gas heater and electric cooler. We implemented some simplification to our model and formulated the model with constant parameters, states, actions and transition function. We also specified the objective function and constraints of the optimization problem in chapter 4. We modelled exogenous information in terms of heater price, cooler price, underground temperature and demand. We considered a noisy sinusoidal pattern for changes of demand and underground temperature by Eq. 44, 45, 46. We also simulated prices with first order Markov chain model via Eq. 43. In order to make prediction, we applied forecast error method within the control horizon and updated it after every single step as described in 5.2.

We designed and tested hybrid CFA-DLA policy with two different methods of parameterization of our model. This an offline method and computations are performed prior to control and best parsameters are obtained through search within a valid range. The optimal parameters then adjust when the control starts. In the first stage of experiments, we parameterized the transition function and in the second set we shifted to parameterization of the demand forecast. We performed both single scalar and monthly-based parameterization in both mentioned methods. Considering 15 % forecast error for demand and prices and setting this error to as maximum as possible for underground temperature, the results of the optimization show that annual cost is the lowest when we use single scalar parameter on demand forecast among our parameterization methods. In this case, the optimum value of the parameter is 80 and it yields a cost which is 51 % higher than optimal cost when we have perfect forecast. The cost is approximately 24 % lower than unparameterized lookahead policy. According to importance of the demand forecast which we tested earlier in section 6.2, it is justifiable that why this method indicates better performance. The second lowest cost was obtained from single scalar parameterization of transition function. The third and forth were monthly-based parameterization of demand forecast and transition function, respectively. The results prove that the performance in monthly-based parameterization is not as high as scalar parameterization.

Bibliography

Arnold, M. and G. Andersson (2011). 'Model predictive control of energy storage including uncertain forecasts'. In: *Power Systems Computation Conference (PSCC)* 23, pp. 24–29.

Cheng, B., T. samov and W. B. Powell (2018). 'Low-Rank Value Function Approximation for Co-Optimization of Battery Storage'. In: *IEEE Transactions on Smart Grid* 9.6, pp. 6590–6598.

Durante, Joseph, Juliana Nascimento and Warren Powell (2017). 'Backward Approximate Dynamic Programming with Hidden Semi-Markov Stochastic Models in Energy Storage Optimization'. In:

Ghadimi, Saeed, Raymond T. Perkins and Warren Powell (2019). 'Reinforcement Learning via Parametric Cost Function Approximation for Multistage Stochastic Programming'. In: pp. 1–40.

Han, J. and W. E (2016). 'Deep Learning Approximation for Stochastic Control Problems'. In: *CoRR*.

Jacobs, J. et al. (1995). 'SOCRATES: A system for scheduling hydroelectric generation under uncertainty'. In: *Operation Research* 59, pp. 99–133.

Kim, J. H. and W. Powell (2011). 'Optimal Energy Commitments with Storage and Intermittent Supply'. In: *Operations Research* 59, pp. 1347–1360.

Kou, P., D. Liang and L. Gao (2018). 'Stochastic Energy Scheduling in Microgrids Considering the Uncertainties in Both Supply and Demand'. In: *IEEE syst.* 12, pp. 2589–2600.

Lohndorf, N., D.Wozabal and S.Minner (2013). 'Optimizing trading decisions for hydro storage systems using approximate dual dynamic programming optimizing trading decisions for hydro storage systems using approximate dual dynamic programming'. In: *Operation Research* 61, pp. 810–823.

Lohndorf, N. and S. Minner (2010). 'Optimal day-ahead trading and storage of renewable energies an approximate dynamic programming approach'. In: *Energy Systems* 1, pp. 1–17.

Perkins, R. T. and W. Powell (2017). 'Stochastic Optimization with Parametric Cost Function Approximations'. In: pp. 1–42.

Powell, Warren B. (2019). *Stochastic Optimization and Learning*. John Wiley and Sons, Inc, Publication.

Rahmani, Mehdi (2017). 'Stochastic, adaptive, and dynamic control of energy storage systems integrated with renewable energy sources for power loss minimization'. In: *Renewable Energy* 113, pp. 1462–1471.

Ridder, Fjo De et al. (2011). 'An optimal control algorithm for borehole thermal energy storage systems'. In: *Energy and buildings* 43, pp. 2918–2925.

Rink, R.E. (1994). 'Optimal operation of solar heat-storage with off-peak energy price incentive'. In: *Journal of Optimization Theory and Applications* 15, pp. 251–266.

Rink, R.E., V. Gourishankar and M. Zaheeruddin (1988). 'Optimal-control of heat-pump heat-storage systems with time-of-day energy price incentive'. In: *Journal of Optimization Theory and Applications* 58, pp. 93–108.

Salas, D.F. and W. Powell (2018). 'Benchmarking a Scalable Approximate Dynamic Programming Algorithm for Stochastic Control of Grid-Level Energy Storage'. In: *Informs Journal on Computing* 30, pp. 106–123.

Simão, H.P. et al. (2017). 'The challenge of integrating offshore wind power in the U.S. electric grid. Part II: Simulation of electricity market operations'. In: *Renewable Energy* 103, pp. 418–431.

Sioshansi, R., S. H. Madaeni and P. Denholm (2014). 'A Dynamic Programming Approach to Estimate the Capacity Value of Energy Storage'. In: *IEEE Transactions on Power Systems* 29.1, pp. 395–403.

Takriti, S., J. R. Birge and E. Long (1996). 'A stochastic model for the unit commitment problem'. In: *Power Syst.* 11, pp. 810–823.

Wallace, S. W. and S. E. Fleten (2003). 'Stochastic Programming Models in Energy'. In: *Handbooks in Operations Research and Management Science* 10.1, pp. 637–677.

Warrington, J. et al. (2012). 'Robust reserve operation in power systems using affine policies. Decision and Control (CDC)'. In: *IEEE 51st Annual Conference on*, pp. 1111–1117.

Zhou, Y. et al. (2018). 'Managing wind-based electricity generation in the presence of storage and transmission capacity'. In: *Production and Operations Management* 28, pp. 970–989.

Appendix

A Static Parameters and Initial Conditions

```
function S0 = S0()
% Model parameters
S0.Tbmax = 285;       % Maximum allowable ground temp [Deg. K]
S0.Tbmin = 273;       % Minimum allowable ground temp [Deg. K]
S0.Rmax  = 5;         % Maximum heating/cooling rate  [MWh]
S0.m     = 13000000; % Water mass in the bore based on:
                        % 180 pipes
                        % Dia.    = 3 meters
                        % Length = 10 meters
S0.Cpw   = 1.167*10^-6   % Water Heat Capacity      [MWh/kG Deg. K]
S0.lambda= 0.2;          % Based on OHTC X Area     [MWh/Deg. K]
S0.COP   = 3;            %COP of heat pump
S0.Phmin = 20            % Minimum heating price
S0.Pcmin = 500           % Minimum cooling price
S0.P1std = 5             % Heating Price standard deviation [NOK/MWh]
S0.P2std = 300           % Cooling Price standard deviation [NOK/MWh]

% Initial states
S0.S.Tb  = 285;      % Borehole temp                  [Deg. K]
S0.S.Tinf= 291;      % Ground temp                    [Deg. K]
S0.S.D   = 7.5;      % Demand energy                  [MWh]
S0.S.Ph  = 70;       % Gas Boiler heateing price      [NOK/MWh]
S0.S.Pc  = 1200;     % Electricity-based cooling price[NOK/MWh]
S0.S.s_up   = 5000; % Constraint upper limit violation
S0.S.s_down = 5000; % Constraint lower limit violation
end
```

B Generation of Exogenous Information

B.1 Driver

```
close all
clear all
clc
addpath('C:\Users\win\Desktop\Master')
plot_settings
S0 = S0()

%% Exogenouse information generation
T        = 365;                    % Time horizon              [Day]
scenario = 30;                     % Number of sample paths

%Exogenous information storage.
sample_path.D    = [];
sample_path.Ph   = [];
sample_path.Pc   = [];
sample_path.Tinf= [];

for i = 1:scenario
    W = exogenous_information(S0,T);
    sample_path.D    = [sample_path.D; W.D];
    sample_path.Tinf = [sample_path.Tinf; W.Tinf];
    sample_path.Ph   = [sample_path.Ph; W.Ph];
    sample_path.Pc   = [sample_path.Pc; W.Pc];
end

%% Exogenous Information Plotting for a Sample Trial
Trial_plot=4;

subplot (4,1,1)
plot(sample_path.D(Trial_plot,:),'b')
xlabel('Time period, Day')
ylabel('Demand [MWh]')

subplot (4,1,2)
plot(sample_path.Tinf(Trial_plot,:)-273,'b')
xlabel('Time period, Day')
ylabel('$T_{underground}$, [Deg. C]')

subplot (4,1,3)
plot(sample_path.Ph(Trial_plot,:),'b')
xlabel('Time period, Day')
ylabel('Heating Price [NOK]')

subplot (4,1,4)
plot(sample_path.Pc(Trial_plot,:),'b')
xlabel('Time period, Day')
ylabel('Cooling Price [NOK]')

save('sample_path','sample_path');
```

B.2 Exogenous Information Function

```
function W = exoinfo(S0,T)
%% Ground Temp. Model
for N=1:T
muinf=289;
Ainf =2;
Tinf(N) =muinf-Ainf*sin ((2*pi*N/T)-(pi/2))+normrnd(0,0.3);     % [Deg. C]
end

%% Demand Model
T1=270;
for day=1:T1
    Ad1 = 15;                                   % Domain of demand energy [MWh]
    D1(day)  = -Ad1*sin((0.85*pi*day/T1)+7*pi/6)+normrnd(0,3);
end

T2=95;
for day=1:T2
    Ad2 = 7;                                    % Domain of demand energy [MWh]
    D2(day)  = -Ad2*sin(pi*day/T2)+ normrnd(0,1.4);
end

%% Price Model
for N=1:T
    Phn = normrnd(0,S0.P1std);
    Ph(N) = S0.S.Ph + Phn;
    Pcn = normrnd(0,S0.P2std);
    Pc(N) = S0.S.Pc + Pcn;
end
Ph=max(S0.Phmin,Ph);
Pc=max(S0.Pcmin,Pc);

W.D   = [S0.S.D];          % Resetting demand at first of each scenario
W.Ph  = [S0.S.Ph];         % Resetting heating price at first of each scenario
W.Pc  = [S0.S.Pc];         % Resetting cooling price at first of each scenario
W.Tinf= [S0.S.Tinf];       % Resetting underground temperature at first of each
↪  scenario

W.D   = [W.D D1 D2];
W.Ph  = [W.Ph Ph];
W.Pc  = [W.Pc Pc];
W.Tinf= [W.Tinf Tinf];
end
```

C Optimal Cost

```
close all
clear all
clc
tic
addpath('C:\Users\win\Desktop\Master\Exogenous Information')
load('sample_path.mat')

%% Settings
S0 = S0();
S = S0.S;
plot_settings
scenario = size(sample_path.D,1);     % Scenario number
N = size(sample_path.D,2)-1;          % Total time horizon              [Day]

%Extention of sample path
sample_path.D   = [sample_path.D];
sample_path.Ph  = [sample_path.Ph];
sample_path.Pc  = [sample_path.Pc];
sample_path.Tinf= [sample_path.Tinf];

%% Simulation
Par_vec = [1];                                % Vector of parameters multiplied by
↪  lambda
Days_number = [365];                          % Vector of month's day
Days_number_cumulative =[0, cumsum(Days_number)];
lambda_vec_ref = [S0.lambda * ones(1,N)];
rng('default')
Parametrized_Cost=zeros(size(Days_number,2), size(Par_vec,2));
for month=1:length(Days_number)
    for Parameter_sequence = 1:length(Par_vec)
        PathsCost =[];
        rng('default')
        for w = 1:scenario
            ss;
            %Initial setup
            S0 = S0();
            S = S0.S;
            %Plotting Arrays
            xSim = [];
            uSim = [];
            timeSim = [];
            %Setting parametrized demand at start of every simple path
            lambda_vec = lambda_vec_ref;
            lambda_vec(Days_number_cumulative(month)+1 :
            ↪  Days_number_cumulative(month+1)) = ...
            Par_vec(Parameter_sequence) *
            ↪  lambda_vec(Days_number_cumulative(month)+1 :
            ↪  Days_number_cumulative(month+1));
            Tinf_horizon = sample_path.Tinf(w,:);
            Demand_horizon = sample_path.D(w,:);
            Ph_horizon = sample_path.Ph(w,:);
            Pc_horizon = sample_path.Pc(w,:);
            for k = 1:N
                nd = length(sample_path.D)-(k);       % MPC control horizon
                % Finding indexes of the negative demands
                [~, col] = find(Demand_horizon<0);
```

```
% Creating "boolean" array
DProductArray = ones(nd,1);
% Indicating negative demands
DProductArray(col,:) = 0;
%% Equality constraint : Ax=b
% Demand constraint
Aeq_D = zeros(nd,7*nd);
%D_k = DPredictArray.*(3/2*U_k + Qheat_k) + (1 {
↪  DpredictArray).*(-U_k - Qcool_k)
for ii = 1:nd
    Aeq_D(ii,1 + 7*(ii - 1)) =
    ↪  (S0.COP/(S0.COP-1))*DProductArray(ii) + (-1*(1 -
    ↪  DProductArray(ii))); % for borehole daily energy flow
    ↪  (U_k)
    Aeq_D(ii,2 + 7*(ii - 1)) = (1)*DProductArray(ii) + (0)*(1 -
    ↪  DProductArray(ii));                  % for heater daily
    ↪  energy flow (Qheat_k)
    Aeq_D(ii,3 + 7*(ii - 1)) = (0)*DProductArray(ii) + (-1)*(1 -
    ↪  DProductArray(ii));                 % for cooler daily
    ↪  energy flow (Qcool_k)
end
Beq_D  = Demand_horizon(1:end-1)';
%Transition function
Aeq_T = zeros(nd,7*nd);
a=1-(lambda_vec/(S0.m*S0.Cpw));
a_cum=cumprod(a);
b=1/(S0.m*S0.Cpw);
for iii=1:nd
    Aeq_T(iii:end,1+7*(iii-1))= b*((+1)*DProductArray(iii) +
    ↪  (-1)*(1 - DProductArray(iii)));
    Aeq_T(iii:end,5+7*(iii-1))= (a(iii)-1);
    if (iii==nd)
        break;
    end
    for jjj=iii+1:nd
        Aeq_T(jjj,1+7*(iii-1))= a(jjj-1) *
        ↪  Aeq_T(jjj-1,1+7*(iii-1));
        Aeq_T(jjj,5+7*(iii-1))= a(jjj-1) *
        ↪  Aeq_T(jjj-1,5+7*(iii-1));
    end
end
Aeq_T=Aeq_T + kron(eye(nd),[0 0 0 1 0 0 0]);
Beq_T=zeros(nd,1);
for kkk=1:nd
    Beq_T(kkk,1) = [a_cum(kkk)*S.Tb];
end
% Integration of equality constraints
Aeq = [Aeq_D;Aeq_T];
Beq = [Beq_D;Beq_T];

%% Inequality constraints : Ax<=b
%Constraint handling : Tb+s_down >=Tbmin
Amin = kron(eye(nd), [0 0 0 -1 0 0 -1]);
Bmin = -S0.Tbmin*ones(nd,1);
%Constraint handling : Tb-s_up <=Tbmax
Amax = kron(eye(nd), [0 0 0 1 0 -1 0]);
Bmax = S0.Tbmax*ones(nd,1);
% Integration of inequality constraints
```

```
A = [Amin;Amax];
B = [Bmin;Bmax];

%% Objective function
f = zeros(1,7*nd);
for kk=1:nd-1
    f(1,2 + 7*(kk - 1)) = (Ph_horizon(kk));
    f(1,3 + 7*(kk - 1)) = (Pc_horizon(kk));
    f(1,6 + 7*(kk - 1)) = S0.S.s_up;              % Penalizing
    ↪  cost for violating upper constraint
    f(1,7 + 7*(kk - 1)) = S0.S.s_down;            % Penalizing
    ↪  cost for violating upper constraint
end

%% Boundaries - lower and upper
lb = zeros(7*nd,1);
lb(4:7:7*nd,1) = -inf;
for i=1:nd-1
    lb(5 + 7*(i - 1),1)=Tinf_horizon (i);
end
ub = repmat([S0.Rmax inf inf inf 0 inf inf]',nd,1);
for i=1:nd-1
    ub(5 + 7*(i - 1),1)=Tinf_horizon (i);
end

%% Optimization
w_opt = linprog(f,A,B,Aeq,Beq,lb,ub);
%  Data store from openloop optimization
u.u      = w_opt(1:7:7*nd);
u.qheat  = w_opt(2:7 :7*nd);
u.qcool  = w_opt(3:7:7*nd);
u.Tb     = w_opt(4:7:7*nd);
u.Tinf   = w_opt(5:7:7*nd);
u.s_up   = w_opt(6:7:7*nd);
u.s_down = w_opt(7:7:7*nd);

%% Take an action
uk = [u.u(1); u.qheat(1); u.qcool(1); u.s_up(1); u.s_down(1)];
sample_path.uk(1+7*(w-1):3+7*(w-1),k)=uk(1:3,:);     % For visual
↪  check
sample_path.uk(6+7*(w-1),k)=w_opt(6);                % For visual
↪  check
sample_path.uk(7+7*(w-1),k)=w_opt(7);                % For visual
↪  check

%% Transition during day
S.Tb = ((1-(S0.lambda/(S0.m*S0.Cpw))) * S.Tb) +
↪  [((-1)*DProductArray(1) + (+1)*(1 - DProductArray(1)))*b 0
↪  0]*uk(1:3,end) +
↪  (S0.lambda/(S0.m*S0.Cpw))*sample_path.Tinf(w,k);
sample_path.uk(4+7*(w-1),k)=S.Tb-273;                % For visual
↪  check

%% Realization
S.Ph = sample_path.Ph(w,k+1);
S.Pc = sample_path.Pc(w,k+1);
S.D = sample_path.D(w,k+1);
S.Tinf = sample_path.Tinf(w,k+1);
```

```
%% For plotting
xSim = [xSim, S.Tb];
uSim = [uSim, uk];

% Create sigma matrix
sigma_mat=zeros(length(Tinf_horizon));
alpha = 1;
for i = 1:size(sigma_mat,1)
    for j = 1:size(sigma_mat,1)
        sigma_mat(i,j) = exp(-alpha*abs(i-j));
    end
end

% Cholesky decomposition
L = chol(sigma_mat,'lower');

% Multivariate normal distribution of Tinf
sigma_E_Tinf = 0;
mu = zeros(1,length(Tinf_horizon));
Sigma_Tinf = sigma_E_Tinf*diag(Tinf_horizon.^0.5);
Z_Tinf = mvnrnd(mu,Sigma_Tinf);

% Multivariate normal distribution of Demand
sigma_E_D = 0;
mu = zeros(1,length(Demand_horizon));
Demand_P=abs(Demand_horizon);
Sigma_Demand = sigma_E_D*diag(Demand_P.^0.5);
Z_Demand = mvnrnd(mu,Sigma_Demand);

% Multivariate normal distribution of Heating Price
sigma_E_Ph = 0;
mu = zeros(1,length(Ph_horizon));
Sigma_Ph = sigma_E_Ph*diag(Ph_horizon.^0.5);
Z_Ph = mvnrnd(mu,Sigma_Ph);

% Multivariate normal distribution of Cooling Price
sigma_E_Pc = 0;
mu = zeros(1,length(Pc_horizon));
Sigma_Pc = sigma_E_Pc*diag(Pc_horizon.^0.5);
Z_Pc = mvnrnd(mu,Sigma_Pc);

%% Exogenous information model
% Ground temp horizon update
Tinf_error = L*Z_Tinf';
Tinf_horizon =  Tinf_horizon(2:end) + Tinf_error(2:end)';

% Demand update
Demand_error = L*Z_Demand';
Demand_forecast = Demand_horizon(3:end) + Demand_error(3:end)';
Demand_horizon = [sample_path.D(w,k+1),Demand_forecast];

% Heating Price update
Ph_error = L*Z_Ph';
Ph_horizon = Ph_horizon(2:end) + Ph_error(2:end)';
Ph_horizon = max(S0.Phmin,Ph_horizon);

% Coolimg Price update
```

```
        Pc_error = L*Z_Pc';
        Pc_horizon = Pc_horizon(2:end) + Pc_error(2:end)';
        Pc_horizon=max(S0.Pcmin,Pc_horizon);

        % Lambda Vector update
        lambda_vec = lambda_vec(2:end);
    end

    % Finding indexes of the negative demands
    [~, col] = find(sample_path.D(w,1:N)<0);
    % Creating "boolean" array
    %DemandProductArray = S0.COP/(S0.COP-1)*ones(N+1,1);
    DemandProductArray = ones(N,1);
    % Indicating negative demands
    DemandProductArray(col,:) = 0;

    % Finding indexes when demands are positive and calculated energy
    ↪  flows
    % from borehole will not satisfy the demand, The shortage have to be
    ↪  supplied by Heater
    [~, col] = find(sample_path.D(w,1:N)<0 | sample_path.D(w,1:N) -
    ↪  ((S0.COP/(S0.COP-1)*uSim(1,1:N) + uSim(2,1:N))+0.001)<=0);
    % Creating "boolean" array
    Remain_Qh_Array = ones(N,1);
    % Indicating negative demands
    Remain_Qh_Array(col,:) = 0;

    % Finding indexes when demands are negative and calculated energy
    ↪  flows
    % to borehole will not satisfy the demand, The shortage have to be
    ↪  supplied by Cooler
    [~, col] = find(sample_path.D(w,1:N)>0 | abs(sample_path.D(w,1:N)) -
    ↪  (uSim(1,1:N) + uSim(3,1:N)+0.001) <=0);
    % Creating "boolean" array
    Remain_Qc_Array = ones(N,1);
    % Indicating negative demands
    Remain_Qc_Array(col,:) = 0;

    % Penalty for constraint violation
    Penalty_up  = max(sample_path.uk(4+7*(w-1),:) + 273 - (S0.Tbmax *
    ↪  ones (1,N)) , 0);
    Penalty_down= abs(min(sample_path.uk(4+7*(w-1),:) + 273 - (S0.Tbmin *
    ↪  ones (1,N)) , 0));

    % Cost Over a sample path
    PathCost_i = sum(sample_path.Ph(w,1:N) .* uSim(2,1:N) + ...
                     + sample_path.Pc(w,1:N) .* uSim(3,1:N) + ...
                     + S0.S.s_up * Penalty_up ...
                     + S0.S.s_down * Penalty_down ...
                     + Remain_Qh_Array(1:N)' .* (sample_path.D(w,1:N) -
                     ↪  (S0.COP/(S0.COP-1)*uSim(1,1:N) + uSim(2,1:N)))
                     ↪  .* sample_path.Ph(w,1:N) + ...
                     + Remain_Qc_Array(1:N)' .*
                     ↪  (abs(sample_path.D(w,1:N)) - (uSim(1,1:N) +
                     ↪  uSim(3,1:N))) .* sample_path.Pc(w,1:N));
    % Making cost vector of different scenarios
    PathsCost = [PathsCost; PathCost_i];
end
```

```
        % Obtaining average cost
        Parametrized_Cost(month,Parameter_sequence) = mean(PathsCost);
        for i = 1:size(PathsCost,1)
            Avg_cost_cum(i) = mean(PathsCost(1:i));
        end
    end
end

% Save real optimal cost
sample_path.OpCo = Parametrized_Cost(1,1)
save('sample_path','sample_path');
```

D Code of Parameterized Cost Function Approximation: Single Scalar Parameterization on Transition Function

```
%%%%%%%%%%%%%%%%%%%%%%%%
%%%% NB: Before this code, you should run Optimal.m in
%%%%     Address: C:\Users\win\Desktop\Master\Optimal Cost
%%%%%%%%%%%%%%%%%%%%%%%%
close all
clear all
clc
tic
addpath( 'C:\Users\win\Desktop\Master\Exogenous Information')
addpath( 'C:\Users\win\Desktop\Master\Optimal Cost')
load('sample_path.mat')

%% Settings
S0 = S0();
S = S0.S;
plot_settings
scenario = size(sample_path.D,1);       % Scenario number
N = size(sample_path.D,2)-1;            % Total time horizon              [Day]

%Extention of sample path
sample_path.D   = [sample_path.D];
sample_path.Ph  = [sample_path.Ph];
sample_path.Pc  = [sample_path.Pc];
sample_path.Tinf= [sample_path.Tinf];

%% Simulation
Par_vec = [0 0.3 0.6 0.9 1 1.2 1.5];    % Vector of parameter values
rng('default')
for Parameter_sequence = 1:length(Par_vec)
        PathsCost =[];
        rng('default')
        for w = 1:scenario
            ss;
            %Initial setup
            S0 = S0();
            S = S0.S;
            %Defining plotting arrays
            xSim = [];
            uSim = [];
            timeSim = [];
            %Setting exogenous information at start of every simple path
            Tinf_horizon = sample_path.Tinf(w,:);
            Demand_horizon = sample_path.D(w,:);
            Ph_horizon = sample_path.Ph(w,:);
            Pc_horizon = sample_path.Pc(w,:);
            %MPC Control
            for k = 1:N
                nd = length(sample_path.D)-(k);     % MPC control horizon
                % Finding indexes of the negative demands
                [~, col] = find(Demand_horizon<0);
                % Creating "boolean" array
                DProductArray = ones(nd,1);
                % Indicating negative demands
                DProductArray(col,:) = 0;
```

```
%% Equality constraint : Ax=b
% Demand constraint
Aeq_D = zeros(nd,7*nd);
%D_k = DPredictArray.*(3/2*U_k + Qheat_k) + (1 {
↪  DpredictArray).*(-U_k - Qcool_k)
for ii = 1:nd
    Aeq_D(ii,1 + 7*(ii - 1)) =
    ↪  (S0.COP/(S0.COP-1))*DProductArray(ii) + (-1*(1 -
    ↪  DProductArray(ii))); % For borehole daily energy flow
    ↪  (U_k)
    Aeq_D(ii,2 + 7*(ii - 1)) = (1)*DProductArray(ii) + (0)*(1 -
    ↪  DProductArray(ii));                  % For heater daily
    ↪  energy flow (Qheat_k)
    Aeq_D(ii,3 + 7*(ii - 1)) = (0)*DProductArray(ii) + (-1)*(1 -
    ↪  DProductArray(ii));                 % For cooler daily
    ↪  energy flow (Qcool_k)
end
Beq_D  = Demand_horizon(1:end-1)';
%Transition function of states
Aeq_T = zeros(nd,7*nd);
a=1-(S0.lambda*Par_vec(Parameter_sequence)/(S0.m*S0.Cpw));
↪  % Parameterization on heat exchange between borehole and
↪  underground
b=1/(S0.m*S0.Cpw);
for iii=1:nd
   Aeq_T(iii:end,1+7*(iii-1))= b*((+1)*DProductArray(iii) +
   ↪  (-1)*(1 - DProductArray(iii)));
   Aeq_T(iii:end,5+7*(iii-1))= (a-1);
   for jjj=iii:nd
       Aeq_T(jjj,1+7*(iii-1))=a^(jjj-iii)*
       ↪  Aeq_T(jjj,1+7*(iii-1));
       Aeq_T(jjj,5+7*(iii-1))=a^(jjj-iii)*
       ↪  Aeq_T(jjj,5+7*(iii-1));
   end
end
Aeq_T=Aeq_T + kron(eye(nd),[0 0 0 1 0 0 0]);
Beq_T=zeros(nd,1);
for kkk=1:nd
    Beq_T(kkk,1) = [a^(kkk)*S.Tb];
end
% Integration of equality constraints
Aeq = [Aeq_D;Aeq_T];
Beq = [Beq_D;Beq_T];

%% Inequality constraints : Ax<=b
%Constraint handling : Tb+s_down >=Tbmin
Amin = kron(eye(nd), [0 0 0 -1 0 0 -1]);
Bmin = -S0.Tbmin*ones(nd,1);
%Constraint handling : Tb-s_up <=Tbmax
Amax = kron(eye(nd), [0 0 0 1 0 -1 0]);
Bmax = S0.Tbmax*ones(nd,1);
%Integration of inequality constraints
A = [Amin;Amax];
B = [Bmin;Bmax];

%% Objective function
f = zeros(1,7*nd);
```

```
for kk=1:nd-1
    f(1,2 + 7*(kk - 1)) = (Ph_horizon(kk));
    f(1,3 + 7*(kk - 1)) = (Pc_horizon(kk));
    f(1,6 + 7*(kk - 1)) = S0.S.s_up;       % Penalizing cost for
    ↪  violating upper constraint
    f(1,7 + 7*(kk - 1)) = S0.S.s_down;     % Penalizing cost for
    ↪  violating upper constraint
end

%% Boundaries - lower and upper
lb = zeros(7*nd,1);
lb(4:7:7*nd,1) = -inf;
for i=1:nd-1
    lb(5 + 7*(i - 1),1)=Tinf_horizon (i);
end
ub = repmat([S0.Rmax inf inf inf 0 inf inf]',nd,1);
for i=1:nd-1
    ub(5 + 7*(i - 1),1)=Tinf_horizon (i);
end

%% Optimization
w_opt = linprog(f,A,B,Aeq,Beq,lb,ub);
%  Data store from openloop optimization
u.u      = w_opt(1:7:7*nd);
u.qheat  = w_opt(2:7:7*nd);
u.qcool  = w_opt(3:7:7*nd);
u.Tb     = w_opt(4:7:7*nd);
u.Tinf   = w_opt(5:7:7*nd);
u.s_up   = w_opt(6:7:7*nd);
u.s_down = w_opt(7:7:7*nd);

%% Take an action
uk = [u.u(1); u.qheat(1); u.qcool(1); u.s_up(1); u.s_down(1)];
sample_path.uk(1+7*(w-1):3+7*(w-1),k)=uk(1:3,:);    % For visual
↪  check
sample_path.uk(6+7*(w-1),k)=w_opt(6);               % For visual
↪  check
sample_path.uk(7+7*(w-1),k)=w_opt(7);               % For visual
↪  check

%% Transition during day
S.Tb = ((1-(S0.lambda/(S0.m*S0.Cpw))) * S.Tb) +
↪  [((-1)*DProductArray(1) + (+1)*(1 - DProductArray(1)))*b 0
↪  0]*uk(1:3,end) +
↪  (S0.lambda/(S0.m*S0.Cpw))*sample_path.Tinf(w,k);
sample_path.uk(4+7*(w-1),k)=S.Tb-273;               % For visual
↪  check

%% Realization
S.Ph = sample_path.Ph(w,k+1);
S.Pc = sample_path.Pc(w,k+1);
S.D = sample_path.D(w,k+1);
S.Tinf = sample_path.Tinf(w,k+1);

%% For plotting
xSim = [xSim, S.Tb];
uSim = [uSim, uk];
```

```
    %% Creating forecast error
    % Create sigma matrix
    sigma_mat=zeros(length(Tinf_horizon));
    alpha = 1;
    for i = 1:size(sigma_mat,1)
            for j = 1:size(sigma_mat,1)
                sigma_mat(i,j) = exp(-alpha*abs(i-j));
            end
    end
    % Cholesky decomposition
    L = chol(sigma_mat,'lower');
    % Multivariate normal distribution of Tinf
    sigma_E_Tinf = 1;
    mu = zeros(1,length(Tinf_horizon));
    Sigma_Tinf = sigma_E_Tinf*diag(Tinf_horizon.^0.5);
    Z_Tinf = mvnrnd(mu,Sigma_Tinf);
    % Multivariate normal distribution of Demand
    sigma_E_D = 15;
    mu = zeros(1,length(Demand_horizon));
    Demand_P=abs(Demand_horizon);
    Sigma_Demand = sigma_E_D*diag(Demand_P.^0.5);
    Z_Demand = mvnrnd(mu,Sigma_Demand);
    % Multivariate normal distribution of Heating Price
    sigma_E_Ph = 15;
    mu = zeros(1,length(Ph_horizon));
    Sigma_Ph = sigma_E_Ph*diag(Ph_horizon.^0.5);
    Z_Ph = mvnrnd(mu,Sigma_Ph);
    % Multivariate normal distribution of Cooling Price
    sigma_E_Pc = 15;
    mu = zeros(1,length(Pc_horizon));
    Sigma_Pc = sigma_E_Pc*diag(Pc_horizon.^0.5);
    Z_Pc = mvnrnd(mu,Sigma_Pc);

    %% Exogenous information developement
    % Ground temp horizon update
    Tinf_error = L*Z_Tinf';
    Tinf_horizon =  Tinf_horizon(2:end) + Tinf_error(2:end)';

    % Demand update
    Demand_error = L*Z_Demand';
    Demand_forecast = Demand_horizon(3:end) + Demand_error(3:end)';
    Demand_horizon = [sample_path.D(w,k+1),Demand_forecast];

    % Heating Price update
    Ph_error = L*Z_Ph';
    Ph_horizon = Ph_horizon(2:end) + Ph_error(2:end)';
    Ph_horizon = max(S0.Phmin,Ph_horizon);

    % Coolimg Price update
    Pc_error = L*Z_Pc';
    Pc_horizon = Pc_horizon(2:end) + Pc_error(2:end)';
    Pc_horizon=max(S0.Pcmin,Pc_horizon);
end

%% Cost calculation over sample path
% Finding indexes of the negative demands
[~, col] = find(sample_path.D(w,1:N)<0);
% Creating "boolean" array
```

```
            DemandProductArray = ones(N,1);
            % Indicating negative demands
            DemandProductArray(col,:) = 0;

            % Finding indexes when demands are positive and calculated energy
            ↪  flows
            % from borehole will not satisfy the demand, The shortage have to be
            ↪  supplied by Heater
            [~, col] = find(sample_path.D(w,1:N)<0 | sample_path.D(w,1:N) -
            ↪  ((S0.COP/(S0.COP-1)*uSim(1,1:N) + uSim(2,1:N))+0.001)<=0);
            % Creating "boolean" array
            Remain_Qh_Array = ones(N,1);
            % Indicating negative demands
            Remain_Qh_Array(col,:) = 0;

            % Finding indexes when demands are negative and calculated energy
            ↪  flows
            % to borehole will not satisfy the demand, The shortage have to be
            ↪  supplied by Cooler
            [~, col] = find(sample_path.D(w,1:N)>0 | abs(sample_path.D(w,1:N)) -
            ↪  (uSim(1,1:N) + uSim(3,1:N)+0.001) <=0);
            % Creating "boolean" array
            Remain_Qc_Array = ones(N,1);
            % Indicating negative demands
            Remain_Qc_Array(col,:) = 0;

            % Penalty for constraint violation
            Penalty_up  = max(sample_path.uk(4+7*(w-1),:) + 273 - (S0.Tbmax *
            ↪  ones (1,N)) , 0);
            Penalty_down= abs(min(sample_path.uk(4+7*(w-1),:) + 273 - (S0.Tbmin *
            ↪  ones (1,N)) , 0));

            % Cost Over a sample path
            PathCost_i = sum(sample_path.Ph(w,1:N) .* uSim(2,1:N) + ...
                             + sample_path.Pc(w,1:N) .* uSim(3,1:N) + ...
                             + S0.S.s_up * Penalty_up ...
                             + S0.S.s_down * Penalty_down ...
                             + Remain_Qh_Array(1:N)' .* (sample_path.D(w,1:N) -
                             ↪  (S0.COP/(S0.COP-1)*uSim(1,1:N) + uSim(2,1:N)))
                             ↪  .* sample_path.Ph(w,1:N) + ...
                             + Remain_Qc_Array(1:N)' .*
                             ↪  (abs(sample_path.D(w,1:N)) - (uSim(1,1:N) +
                             ↪  uSim(3,1:N))) .* sample_path.Pc(w,1:N));
           % Making cost vector of different scenarios
           PathsCost = [PathsCost; PathCost_i];
        end
        % Obtaining average cost
        Parametrized_Cost(Parameter_sequence) = mean(PathsCost);
        for i = 1:size(PathsCost,1)
            Avg_cost_cum(i) = mean(PathsCost(1:i));
        end
end

%% Plotting Policy Cost Index against theta_{i}
hold on
plot(Par_vec, (Parametrized_Cost-sample_path.OpCo)/sample_path.OpCo)
grid('on');
box('on');
```

```
xlabel('$\theta$')
ylabel('$ \Delta F^{\pi} (\theta) $')
% save
save = ('P02_MPC')

%% Plotting the states and inputs for a given scenario
trial = 1;
sample_path.ukplot(1:4,:) = sample_path.uk(1+7*(trial-1):4+7*(trial-1),:);
sample_path.ukplot(5,:)= sample_path.D(trial,1:N);
for ind= 1:N
    if sample_path.ukplot(5,ind)>=0
       sample_path.ukplot(1,ind)=(S0.COP/(S0.COP-1))*sample_path.ukplot(1,ind)
    else
       sample_path.ukplot(1,ind)=-1*sample_path.ukplot(1,ind)
       sample_path.ukplot(3,ind)=-1*sample_path.ukplot(3,ind)
    end
end

figure
subplot(5,1,1)
plot(sample_path.ukplot(4,:),'b-')
xlabel('Time, Day')
ylabel('$T^{b}$ [Deg. C]')

subplot(5,1,2)
stairs(0:N,[sample_path.ukplot(1,:), nan],'b-')
xlabel('Time, Day')
ylabel('$Q^{b}$ [MWh]')

subplot(5,1,3)
stairs(0:N,[sample_path.ukplot(2,:), nan],'b-')
xlabel('Time, Day')
ylabel('$Q^{h}$ [MWh]')

subplot(5,1,4)
stairs(0:N,[sample_path.ukplot(3,:), nan],'b-')
xlabel('Time, Day')
ylabel('$Q^{c}$ [MWh]')

subplot(5,1,5)
plot(sample_path.ukplot(5,:), 'b')
xlabel('Time, Day')
ylabel('Demand [MWh]')
```

E Code of Parameterized Cost Function Approximation: Monthly-Based Parameterization on Transition Function

```
%%%%%%%%%%%%%%%%%%%%%%%%
%%%% NB: Before this code, you should run Optimal.m in
%%%%     Address: C:\Users\win\Desktop\Master\Optimal Cost
%%%%%%%%%%%%%%%%%%%%%%%%
close all
clear all
clc
tic
addpath( 'C:\Users\win\Desktop\Master\Exogenous Information')
addpath( 'C:\Users\win\Desktop\Master\Optimal Cost')
load('sample_path.mat')

%% Settings
S0 = S0();
S = S0.S;
plot_settings
scenario = size(sample_path.D,1);     % Scenario number
N = size(sample_path.D,2)-1;          % Total time horizon             [Day]

%Extention of sample path
sample_path.D   = [sample_path.D];
sample_path.Ph  = [sample_path.Ph];
sample_path.Pc  = [sample_path.Pc];
sample_path.Tinf= [sample_path.Tinf];

%% Simulation
Par_vec = [0 0.2 0.4 0.6 0.8 1 1.2 1.4 1.6 1.8 2];               % Vector of
↪  parameters will be multiplied by lambda
Days_number = [30 31 31 28 31 30 31 30 31 31 30 31];  % Vector of days in every
↪  month
Days_number_cumulative =[0, cumsum(Days_number)];
lambda_vec_ref = [S0.lambda * ones(1,N)];
rng('default')
Parametrized_Cost=zeros(size(Days_number,2), size(Par_vec,2));
for month=1:length(Days_number)
    for Parameter_sequence = 1:length(Par_vec)
        PathsCost =[];
        rng('default')
        for w = 1:1
            ss;
            %Initial setup
            S0 = S0();
            S = S0.S;
            %Defining plotting arrays
            xSim = [];
            uSim = [];
            timeSim = [];
            %Setting up lambda vecotor
            lambda_vec = lambda_vec_ref;
            lambda_vec(Days_number_cumulative(month)+1 :
            ↪  Days_number_cumulative(month+1)) = ...
                Par_vec(Parameter_sequence) *
                ↪  lambda_vec(Days_number_cumulative(month)+1 :
                ↪  Days_number_cumulative(month+1));
```

```
%Setting exogenous information at start of every simple path
Tinf_horizon = sample_path.Tinf(w,:);
Demand_horizon = sample_path.D(w,:);
Ph_horizon = sample_path.Ph(w,:);
Pc_horizon = sample_path.Pc(w,:);
%MPC Control
for k = 1:N
    nd = length(sample_path.D)-(k);     % MPC control horizon
    % Finding indexes of the negative demands
    [~, col] = find(Demand_horizon<0);
    % Creating "boolean" array
    DProductArray = ones(nd,1);
    % Indicating negative demands
    DProductArray(col,:) = 0;
    %% Equality constraint : Ax=b
    % Demand constraint
    Aeq_D = zeros(nd,7*nd);
    %D_k = DPredictArray.*(3/2*U_k + Qheat_k) + (1 {
    ↪  DpredictArray).*(-U_k - Qcool_k)
    for ii = 1:nd
        Aeq_D(ii,1 + 7*(ii - 1)) =
        ↪  (S0.COP/(S0.COP-1))*DProductArray(ii) + (-1*(1 -
        ↪  DProductArray(ii))); % For borehole daily energy flow
        ↪  energy (U_k)
        Aeq_D(ii,2 + 7*(ii - 1)) = (1)*DProductArray(ii) + (0)*(1 -
        ↪  DProductArray(ii));                  % For heater daily
        ↪  energy flow energy (Qheat_k)
        Aeq_D(ii,3 + 7*(ii - 1)) = (0)*DProductArray(ii) + (-1)*(1 -
        ↪  DProductArray(ii));                  % For cooler daily
        ↪  energy flow energy (Qcool_k)
    end
    Beq_D  = Demand_horizon(1:end-1)';

    %Transition function of states
    Aeq_T = zeros(nd,7*nd);
    a=1-(lambda_vec/(S0.m*S0.Cpw));
    a_cum=cumprod(a);
    b=1/(S0.m*S0.Cpw);
    for iii=1:nd
        Aeq_T(iii:end,1+7*(iii-1))= b*((+1)*DProductArray(iii) +
        ↪  (-1)*(1 - DProductArray(iii)));
        Aeq_T(iii:end,5+7*(iii-1))= (a(iii)-1);
        if (iii==nd)
            break;
        end
        for jjj=iii+1:nd
            Aeq_T(jjj,1+7*(iii-1))= a(jjj-1) *
            ↪  Aeq_T(jjj-1,1+7*(iii-1));
            Aeq_T(jjj,5+7*(iii-1))= a(jjj-1) *
            ↪  Aeq_T(jjj-1,5+7*(iii-1));
        end
    end
    Aeq_T=Aeq_T + kron(eye(nd),[0 0 0 1 0 0 0]);
    Beq_T=zeros(nd,1);
    for kkk=1:nd
        Beq_T(kkk,1) = [a_cum(kkk)*S.Tb];
    end
    % Integration of equality constraints
```

```
 Aeq = [Aeq_D;Aeq_T];
 Beq = [Beq_D;Beq_T];

 %% Inequality constraints : Ax<=b
 %Constraint handling : Tb+s_down >= Tbmin
 Amin = kron(eye(nd), [0 0 0 -1 0 0 -1]);
 Bmin = -S0.Tbmin*ones(nd,1);
 %Constraint handling : Tb-s_up <=Tbmax
 Amax = kron(eye(nd), [0 0 0 1 0 -1 0]);
 Bmax = S0.Tbmax*ones(nd,1);
 %Integration of inequality constraints
 A = [Amin;Amax];
 B = [Bmin;Bmax];

%% Objective function
f = zeros(1,7*nd);
for kk=1:nd-1
    f(1,2 + 7*(kk - 1)) = (Ph_horizon(kk));
    f(1,3 + 7*(kk - 1)) = (Pc_horizon(kk));
    f(1,6 + 7*(kk - 1)) = S0.S.s_up;       % Penalizing cost for
    ↪  violating upper constraint
    f(1,7 + 7*(kk - 1)) = S0.S.s_down;     % Penalizing cost for
    ↪  violating upper constraint
end

%% Boundaries - lower and upper
lb = zeros(7*nd,1);
lb(4:7:7*nd,1) = -inf;
for i=1:nd-1
    lb(5 + 7*(i - 1),1)=Tinf_horizon (i);
end
ub = repmat([S0.Rmax inf inf inf 0 inf inf]',nd,1);
for i=1:nd-1
    ub(5 + 7*(i - 1),1)=Tinf_horizon (i);
end

%% Optimization
w_opt = linprog(f,A,B,Aeq,Beq,lb,ub);
%  Data store from openloop optimization
u.u      = w_opt(1:7:7*nd);
u.qheat  = w_opt(2:7:7*nd);
u.qcool  = w_opt(3:7:7*nd);
u.Tb     = w_opt(4:7:7*nd);
u.Tinf   = w_opt(5:7:7*nd);
u.s_up   = w_opt(6:7:7*nd);
u.s_down = w_opt(7:7:7*nd);

%% Take an action
uk = [u.u(1); u.qheat(1); u.qcool(1); u.s_up(1); u.s_down(1)];
sample_path.uk(1+7*(w-1):3+7*(w-1),k)=uk(1:3,:);    % For visual
↪  check
sample_path.uk(6+7*(w-1),k)=w_opt(6);               % For visual
↪  check
sample_path.uk(7+7*(w-1),k)=w_opt(7);               % For visual
↪  check

%% Transition during day
```

```
S.Tb = ((1-(S0.lambda/(S0.m*S0.Cpw))) * S.Tb) +
↪  [((-1)*DProductArray(1) + (+1)*(1 - DProductArray(1)))*b 0
↪  0]*uk(1:3,end) +
↪  (S0.lambda/(S0.m*S0.Cpw))*sample_path.Tinf(w,k);
sample_path.uk(4+7*(w-1),k)=S.Tb-273;                % For visual
↪  check

%% Realization
S.Ph = sample_path.Ph(w,k+1);
S.Pc = sample_path.Pc(w,k+1);
S.D = sample_path.D(w,k+1);
S.Tinf = sample_path.Tinf(w,k+1);

%% For plotting
xSim = [xSim, S.Tb];
uSim = [uSim, uk];

%% Creating forecast error
% Create sigma matrix
sigma_mat=zeros(length(Tinf_horizon));
alpha = 1;
for i = 1:size(sigma_mat,1)
        for j = 1:size(sigma_mat,1)
            sigma_mat(i,j) = exp(-alpha*abs(i-j));
        end
end
% Cholesky decomposition
L = chol(sigma_mat,'lower');
% Multivariate normal distribution of Tinf
sigma_E_Tinf = 0;
mu = zeros(1,length(Tinf_horizon));
Sigma_Tinf = sigma_E_Tinf*diag(Tinf_horizon.^0.5);
Z_Tinf = mvnrnd(mu,Sigma_Tinf);
% Multivariate normal distribution of Demand
sigma_E_D = 0;
mu = zeros(1,length(Demand_horizon));
Demand_P=abs(Demand_horizon);
Sigma_Demand = sigma_E_D*diag(Demand_P.^0.5);
Z_Demand = mvnrnd(mu,Sigma_Demand);
% Multivariate normal distribution of Heating Price
sigma_E_Ph = 0;
mu = zeros(1,length(Ph_horizon));
Sigma_Ph = sigma_E_Ph*diag(Ph_horizon.^0.5);
Z_Ph = mvnrnd(mu,Sigma_Ph);
% Multivariate normal distribution of Cooling Price
sigma_E_Pc = 15;
mu = zeros(1,length(Pc_horizon));
Sigma_Pc = sigma_E_Pc*diag(Pc_horizon.^0.5);
Z_Pc = mvnrnd(mu,Sigma_Pc);

%% Exogenous information developement
% Ground temp horizon update
Tinf_error = L*Z_Tinf';
Tinf_horizon =  Tinf_horizon(2:end) + Tinf_error(2:end)';

% Demand update
Demand_error = L*Z_Demand';
Demand_forecast = Demand_horizon(3:end) + Demand_error(3:end)';
```

```
    Demand_horizon = [sample_path.D(w,k+1),Demand_forecast];

    % Heating Price update
    Ph_error = L*Z_Ph';
    Ph_horizon = Ph_horizon(2:end) + Ph_error(2:end)';
    Ph_horizon = max(S0.Phmin,Ph_horizon);

    % Coolimg Price update
    Pc_error = L*Z_Pc';
    Pc_horizon = Pc_horizon(2:end) + Pc_error(2:end)';
    Pc_horizon=max(S0.Pcmin,Pc_horizon);

    % Lambda Vector update
    lambda_vec = lambda_vec(2:end);
end

%% Cost calculation over sample path
% Finding indexes of the negative demands
[~, col] = find(sample_path.D(w,1:N)<0);
% Creating "boolean" array
DemandProductArray = ones(N,1);
% Indicating negative demands
DemandProductArray(col,:) = 0;

% Finding indexes when demands are positive and calculated energy
↪  flows
% from borehole will not satisfy the demand, The shortage have to be
↪  supplied by Heater
[~, col] = find(sample_path.D(w,1:N)<0 | sample_path.D(w,1:N) -
↪  ((S0.COP/(S0.COP-1)*uSim(1,1:N) + uSim(2,1:N))+0.001)<=0);
% Creating "boolean" array
Remain_Qh_Array = ones(N,1);
% Indicating negative demands
Remain_Qh_Array(col,:) = 0;

% Finding indexes when demands are negative and calculated energy
↪  flows
% to borehole will not satisfy the demand, The shortage have to be
↪  supplied by Cooler
[~, col] = find(sample_path.D(w,1:N)>0 | abs(sample_path.D(w,1:N)) -
↪  (uSim(1,1:N) + uSim(3,1:N)+0.001) <=0);
% Creating "boolean" array
Remain_Qc_Array = ones(N,1);
% Indicating negative demands
Remain_Qc_Array(col,:) = 0;

% Penalty for constraint violation
Penalty_up  = max(sample_path.uk(4+7*(w-1),:) + 273 - (S0.Tbmax *
↪  ones (1,N)) , 0);
Penalty_down= abs(min(sample_path.uk(4+7*(w-1),:) + 273 - (S0.Tbmin *
↪  ones (1,N)) , 0));

% Cost Over a sample path
PathCost_i = sum(sample_path.Ph(w,1:N) .* uSim(2,1:N) + ...
                 + sample_path.Pc(w,1:N) .* uSim(3,1:N) + ...
                 + S0.S.s_up * Penalty_up ...
                 + S0.S.s_down * Penalty_down ...
```

```
                                + Remain_Qh_Array(1:N)' .* (sample_path.D(w,1:N) -
                                ↪  (S0.COP/(S0.COP-1)*uSim(1,1:N) + uSim(2,1:N)))
                                ↪  .* sample_path.Ph(w,1:N) + ...
                                + Remain_Qc_Array(1:N)' .*
                                ↪  (abs(sample_path.D(w,1:N)) - (uSim(1,1:N) +
                                ↪  uSim(3,1:N))) .* sample_path.Pc(w,1:N));
            % Making cost vector of different scenarios
            PathsCost = [PathsCost; PathCost_i];
        end

        % Obtaining average cost
        Parametrized_Cost(month,Parameter_sequence) = mean(PathsCost);
        for i = 1:size(PathsCost,1)
            Avg_cost_cum(i) = mean(PathsCost(1:i));
        end
    end
end

%% Plotting policy improvement against theta_{i}
hold on
plot(Par_vec, (Parametrized_Cost(1,:)-sample_path.OpCo)/sample_path.OpCo,
↪  Par_vec, (Parametrized_Cost(2,:)-sample_path.OpCo)/sample_path.OpCo)
plot(Par_vec, (Parametrized_Cost(3,:)-sample_path.OpCo)/sample_path.OpCo,
↪  Par_vec, (Parametrized_Cost(4,:)-sample_path.OpCo)/sample_path.OpCo)
plot(Par_vec, (Parametrized_Cost(5,:)-sample_path.OpCo)/sample_path.OpCo,
↪  Par_vec, (Parametrized_Cost(6,:)-sample_path.OpCo)/sample_path.OpCo)
plot(Par_vec, (Parametrized_Cost(7,:)-sample_path.OpCo)/sample_path.OpCo,
↪  Par_vec, (Parametrized_Cost(8,:)-sample_path.OpCo)/sample_path.OpCo)
plot(Par_vec, (Parametrized_Cost(9,:)-sample_path.OpCo)/sample_path.OpCo,
↪  Par_vec, (Parametrized_Cost(10,:)-sample_path.OpCo)/sample_path.OpCo)
plot(Par_vec, (Parametrized_Cost(11,:)-sample_path.OpCo)/sample_path.OpCo,
↪  Par_vec, (Parametrized_Cost(12,:)-sample_path.OpCo)/sample_path.OpCo)
legend('$\theta_{Nov.}$','$\theta_{Dec.}$','$\theta_{Jan.}$','$\theta_{Feb.}$','$\theta_{Mar.}$',.

       ↪  '$\theta_{Apr.}$','$\theta_{May.}$','$\theta_{Jun.}$','$\theta_{Jul.}$','$\theta_{Aug.}
       '$\theta_{Sep.}$','$\theta_{Oct.}$')
grid('on');
box('on');
xlabel('$\theta$')
ylabel('$ \Delta F^{\pi} (\theta) $')
% save
save = ('P02_MPC')

%% Plotting the states and inputs for a given scenario
trial = 1;
sample_path.ukplot(1:4,:) = sample_path.uk(1+7*(trial-1):4+7*(trial-1),:);
sample_path.ukplot(5,:)= sample_path.D(trial,1:N);
for ind= 1:N
    if sample_path.ukplot(5,ind)>=0
       sample_path.ukplot(1,ind)=(S0.COP/(S0.COP-1))*sample_path.ukplot(1,ind)
    else
       sample_path.ukplot(1,ind)=-1*sample_path.ukplot(1,ind)
       sample_path.ukplot(3,ind)=-1*sample_path.ukplot(3,ind)
    end
end

figure
subplot(5,1,1)
```

```
plot(sample_path.ukplot(4,:),'b-')
xlabel('Time, Day')
ylabel('$T^{b}$ [Deg. C]')

subplot(5,1,2)
stairs(0:N,[sample_path.ukplot(1,:), nan],'b-')
xlabel('Time, Day')
ylabel('$Q^{b}$ [MWh]')

subplot(5,1,3)
stairs(0:N,[sample_path.ukplot(2,:), nan],'b-')
xlabel('Time, Day')
ylabel('$Q^{h}$ [MWh]')

subplot(5,1,4)
stairs(0:N,[sample_path.ukplot(3,:), nan],'b-')
xlabel('Time, Day')
ylabel('$Q^{c}$ [MWh]')

subplot(5,1,5)
plot(sample_path.ukplot(5,:), 'b')
xlabel('Time, Day')
ylabel('Demand [MWh]')
```

F Code of Parameterized Cost Function Approximation: Two Dimensional Search over Parameterized Transition Function

```
%%%%%%%%%%%%%%%%%%%%%%%%
%%%% NB: Before this code, you should run Optimal.m in
%%%%     Address: C:\Users\win\Desktop\Master\Optimal Cost
%%%%%%%%%%%%%%%%%%%%%%%%
close all
clear all
clc
tic
addpath( 'C:\Users\win\Desktop\Master\Exogenous Information')
addpath( 'C:\Users\win\Desktop\Master\Optimal Cost')
load('sample_path.mat')

%% Settings
S0 = S0();
S = S0.S;
plot_settings
scenario = size(sample_path.D,1);     % Scenario number
N = size(sample_path.D,2)-1;          % Total time horizon             [Day]

%Extention of sample path
sample_path.D   = [sample_path.D];
sample_path.Ph  = [sample_path.Ph];
sample_path.Pc  = [sample_path.Pc];
sample_path.Tinf= [sample_path.Tinf];

%% Simulation
Theta_Tinf = [1];
Par_vec = [0.2 0.4 0.6 0.8 1 1.2 1.4];                % Vector of parameters
↪  will be multiplied by lambda
Days_number = [30 31 31 28 31 30 31 30 31 31 30 31];   % Vector of days in every
↪  month
Days_number_cumulative =[0, cumsum(Days_number)];
lambda_vec_ref = [S0.lambda * ones(1,N)];
First_Month_Seq=10;                                    % Index of first paramter
↪  coordiantes (tetha_i) that we parametrized
Second_Month_Seq=11;                                   % Index of second paramter
↪  coordiantes (tetha_j) that we parametrized
rng('default')
Parametrized_Cost=zeros(size(Par_vec,1), size(Par_vec,2));
for Tetha_i=1:length(Par_vec)
    % Setting tetha_i
    lambda_vec_i = lambda_vec_ref;
    lambda_vec_i(Days_number_cumulative(First_Month_Seq)+1 :
    ↪  Days_number_cumulative(First_Month_Seq+1)) = ...
    Par_vec(Tetha_i) * lambda_vec_i(Days_number_cumulative(First_Month_Seq)+1 :
    ↪  Days_number_cumulative(First_Month_Seq+1));
    for Tetha_j=1:length(Par_vec)
        % Setting tetha_i
        lambda_vec_ij = lambda_vec_i;
        lambda_vec_ij(Days_number_cumulative(Second_Month_Seq)+1 :
        ↪  Days_number_cumulative(Second_Month_Seq+1)) = ...
        Par_vec(Tetha_j) *
        ↪  lambda_vec_i(Days_number_cumulative(Second_Month_Seq)+1 :
        ↪  Days_number_cumulative(Second_Month_Seq+1));
```

```
PathsCost =[];
rng('default')
for w = 1:30
ss;
%Initial setup
S0 = S0();
S = S0.S;
% Defining Plotting Arrays
xSim = [];
uSim = [];
timeSim = [];
lambda_vec_ijk=lambda_vec_ij;
%Setting exogenous information at start of every simple path
Tinf_horizon = sample_path.Tinf(w,:);
Demand_horizon = sample_path.D(w,:);
Ph_horizon = sample_path.Ph(w,:);
Pc_horizon = sample_path.Pc(w,:);
%MPC Control
for k = 1:N
    nd = length(sample_path.D)-(k);       % MPC control horizon
    % Finding indexes of the negative demands
    [~, col] = find(Demand_horizon<0);
    % Creating "boolean" array
    DProductArray = ones(nd,1);
    % Indicating negative demands
    DProductArray(col,:) = 0;

    %% Equality constraint : Ax=b
    % Demand constraint
    Aeq_D = zeros(nd,7*nd);
    %D_k = DPredictArray.*(3/2*U_k + Qheat_k) + (1 {
    ↪  DpredictArray).*(-U_k - Qcool_k)
    for ii = 1:nd
        Aeq_D(ii,1 + 7*(ii - 1)) = (S0.COP/(S0.COP-1))*DProductArray(ii)
        ↪  + (-1*(1 - DProductArray(ii))); % For borehole daily energy
        ↪  flow energy (U_k)
        Aeq_D(ii,2 + 7*(ii - 1)) = (1)*DProductArray(ii) + (0)*(1 -
        ↪  DProductArray(ii));                  % For heater daily
        ↪  energy flow energy (Qheat_k)
        Aeq_D(ii,3 + 7*(ii - 1)) = (0)*DProductArray(ii) + (-1)*(1 -
        ↪  DProductArray(ii));                 % For cooler daily energy
        ↪  flow energy (Qcool_k)
    end
    Beq_D  = Demand_horizon(1:end-1)';

    %Transition function of states
    Aeq_T = zeros(nd,7*nd);
    a=1-(lambda_vec_ijk/(S0.m*S0.Cpw));
    a_cum=cumprod(a);
    b=1/(S0.m*S0.Cpw);
    for iii=1:nd
        Aeq_T(iii:end,1+7*(iii-1))= b*((+1)*DProductArray(iii) + (-1)*(1
        ↪  - DProductArray(iii)));
        Aeq_T(iii:end,5+7*(iii-1))= (a(iii)-1);
        if (iii==nd)
            break;
        end
        for jjj=iii+1:nd
```

```
        Aeq_T(jjj,1+7*(iii-1))= a(jjj-1) * Aeq_T(jjj-1,1+7*(iii-1));
        Aeq_T(jjj,5+7*(iii-1))= a(jjj-1) * Aeq_T(jjj-1,5+7*(iii-1));
        end
    end
    Aeq_T=Aeq_T + kron(eye(nd),[0 0 0 1 0 0 0]);
    Beq_T=zeros(nd,1);
    for kkk=1:nd
        Beq_T(kkk,1) = [a_cum(kkk)*S.Tb];
    end
    % Aeq & Beq integration
    Aeq = [Aeq_D;Aeq_T];
    Beq = [Beq_D;Beq_T];

    %% Inequality constraints : Ax<=b
    %Constraint handling : Tb+s_down >=Tbmin
    Amin = kron(eye(nd), [0 0 0 -1 0 0 -1]);
    Bmin = -S0.Tbmin*ones(nd,1);
    %Constraint handling : Tb-s_up <=Tbmax
    Amax = kron(eye(nd), [0 0 0 1 0 -1 0]);
    Bmax = S0.Tbmax*ones(nd,1);
    % Amin & Amax, Bmin & Bmax integration
    A = [Amin;Amax];
    B = [Bmin;Bmax];

    %% Objective function
    f = zeros(1,7*nd);
    for kk=1:nd-1
        f(1,2 + 7*(kk - 1)) = (Ph_horizon(kk));
        f(1,3 + 7*(kk - 1)) = (Pc_horizon(kk));
        f(1,6 + 7*(kk - 1)) = S0.S.s_up;            % Penalizing cost for
        ↪  violating upper constraint
        f(1,7 + 7*(kk - 1)) = S0.S.s_down;          % Penalizing cost for
        ↪  violating upper constraint
    end
    %% Boundaries - lower and upper
    lb = zeros(7*nd,1);
    lb(4:7:7*nd,1) = -inf;
    for i=1:nd-1
        lb(5 + 7*(i - 1),1)=Tinf_horizon (i);
    end
   ub = repmat([S0.Rmax inf inf inf 0 inf inf]',nd,1);
   for i=1:nd-1
       ub(5 + 7*(i - 1),1)=Tinf_horizon (i);
   end

    %% Optimization
    w_opt = linprog(f,A,B,Aeq,Beq,lb,ub);

  %  Data store from openloop optimization
  u.u      = w_opt(1:7:7*nd);
  u.qheat  = w_opt(2:7 :7*nd);
  u.qcool  = w_opt(3:7:7*nd);
  u.Tb     = w_opt(4:7:7*nd);
  u.Tinf   = w_opt(5:7:7*nd);
  u.s_up   = w_opt(6:7:7*nd);
  u.s_down = w_opt(7:7:7*nd);

  %% Take an action
```

```
uk = [u.u(1); u.qheat(1); u.qcool(1); u.s_up(1); u.s_down(1)];
sample_path.uk(1+7*(w-1):3+7*(w-1),k)=uk(1:3,:);     % For visual check
sample_path.uk(6+7*(w-1),k)=w_opt(6);                % For visual check
sample_path.uk(7+7*(w-1),k)=w_opt(7);                % For visual check

%% Transition during day
S.Tb = ((1-(S0.lambda/(S0.m*S0.Cpw))) * S.Tb) + [((-1)*DProductArray(1)
↪  + (+1)*(1 - DProductArray(1)))*b 0 0]*uk(1:3,end) +
↪  (S0.lambda/(S0.m*S0.Cpw))*sample_path.Tinf(w,k);
sample_path.uk(4+7*(w-1),k)=S.Tb-273;                % For visual check

%% Realization
S.Ph = sample_path.Ph(w,k+1);
S.Pc = sample_path.Pc(w,k+1);
S.D = sample_path.D(w,k+1);
S.Tinf = sample_path.Tinf(w,k+1);

%% For plotting
xSim = [xSim, S.Tb];
uSim = [uSim, uk];

%% Creating forecast error
% Create sigma matrix
sigma_mat=zeros(length(Tinf_horizon));
alpha = 1;
for i = 1:size(sigma_mat,1)
    for j = 1:size(sigma_mat,1)
        sigma_mat(i,j) = exp(-alpha*abs(i-j));
    end
end
% Cholesky decomposition
L = chol(sigma_mat,'lower');
% Multivariate normal distribution of Tinf
sigma_E_Tinf = 1;
mu = zeros(1,length(Tinf_horizon));
Sigma_Tinf = sigma_E_Tinf*diag(Tinf_horizon.^0.5);
Z_Tinf = mvnrnd(mu,Sigma_Tinf);
% Multivariate normal distribution of Demand
sigma_E_D = 15;
mu = zeros(1,length(Demand_horizon));
Demand_P=abs(Demand_horizon);
Sigma_Demand = sigma_E_D*diag(Demand_P.^0.5);
Z_Demand = mvnrnd(mu,Sigma_Demand);
% Multivariate normal distribution of Heating Price
sigma_E_Ph = 15;
mu = zeros(1,length(Ph_horizon));
Sigma_Ph = sigma_E_Ph*diag(Ph_horizon.^0.5);
Z_Ph = mvnrnd(mu,Sigma_Ph);
% Multivariate normal distribution of Cooling Price
sigma_E_Pc = 15;
mu = zeros(1,length(Pc_horizon));
Sigma_Pc = sigma_E_Pc*diag(Pc_horizon.^0.5);
Z_Pc = mvnrnd(mu,Sigma_Pc);

%% Exogenous information model
% Ground temp horizon update
Tinf_error = L*Z_Tinf';
Tinf_horizon =  Tinf_horizon(2:end) + Tinf_error(2:end)';
```

```
  % Demand update
  Demand_error = L*Z_Demand';
  Demand_forecast = Demand_horizon(3:end) + Demand_error(3:end)';
  Demand_horizon = [sample_path.D(w,k+1),Demand_forecast];

  % Heating Price update
  Ph_error = L*Z_Ph';
  Ph_horizon = Ph_horizon(2:end) + Ph_error(2:end)';
  Ph_horizon = max(S0.Phmin,Ph_horizon);

  % Coolimg Price update
  Pc_error = L*Z_Pc';
  Pc_horizon = Pc_horizon(2:end) + Pc_error(2:end)';
  Pc_horizon=max(S0.Pcmin,Pc_horizon);

  % Lambda Vector update
  lambda_vec_ijk = lambda_vec_ijk(2:end);
end

 %% Cost calculation over sample path
% Finding indexes of the negative demands
[~, col] = find(sample_path.D(w,1:N)<0);
% Creating "boolean" array
DemandProductArray = ones(N,1);
% Indicating negative demands
DemandProductArray(col,:) = 0;

% Finding indexes when demands are positive and calculated energy flows
% from borehole will not satisfy the demand, The shortage have to be
↪  supplied by Heater
[~, col] = find(sample_path.D(w,1:N)<0 | sample_path.D(w,1:N) -
↪  ((S0.COP/(S0.COP-1)*uSim(1,1:N) + uSim(2,1:N))+0.001)<=0);
% Creating "boolean" array
 Remain_Qh_Array = ones(N,1);
 % Indicating negative demands
 Remain_Qh_Array(col,:) = 0;

 % Finding indexes when demands are negative and calculated energy flows
 % to borehole will not satisfy the demand, The shortage have to be
 ↪  supplied by Cooler
 [~, col] = find(sample_path.D(w,1:N)>0 | abs(sample_path.D(w,1:N)) -
 ↪  (uSim(1,1:N) + uSim(3,1:N)+0.001) <=0);
 % Creating "boolean" array
 Remain_Qc_Array = ones(N,1);
 % Indicating negative demands
 Remain_Qc_Array(col,:) = 0;

 % Penalty for constraint violation
 Penalty_up  = max(sample_path.uk(4+7*(w-1),:) + 273 - (S0.Tbmax * ones
 ↪  (1,N)) , 0);
 Penalty_down= abs(min(sample_path.uk(4+7*(w-1),:) + 273 - (S0.Tbmin *
 ↪  ones (1,N)) , 0));

 % Cost Over a sample path
 PathCost_i = sum(sample_path.Ph(w,1:N) .* uSim(2,1:N) + ...
                  + sample_path.Pc(w,1:N) .* uSim(3,1:N) + ...
                  + S0.S.s_up * Penalty_up ...
```

```
                        + S0.S.s_down * Penalty_down ...
                        + Remain_Qh_Array(1:N)' .* (sample_path.D(w,1:N) -
                        ↪  (S0.COP/(S0.COP-1)*uSim(1,1:N) + uSim(2,1:N))) .*
                        ↪  sample_path.Ph(w,1:N) + ...
                        + Remain_Qc_Array(1:N)' .* (abs(sample_path.D(w,1:N)) -
                        ↪  (uSim(1,1:N) + uSim(3,1:N))) .*
                        ↪  sample_path.Pc(w,1:N));
         % Making cost vector of different scenarios
         PathsCost = [PathsCost; PathCost_i];
        end
        % Obtaining average cost
        Parametrized_Cost(Tetha_i,Tetha_j) = mean(PathsCost);
        for i = 1:size(PathsCost,1)
            Avg_cost_cum(i) = mean(PathsCost(1:i));
        end
    end
end
%Policy Cost Index against changing 2 different coordiantes of tetha
policy_Cost_Index=(Parametrized_Cost - sample_path.OpCo) ./ sample_path.OpCo

%% Plotting
pcolor(Par_vec,Par_vec,policy_Cost_Index')
title('$ \Delta F^{\pi} (\theta) $','fontsi',10,'interpreter','latex')
shading flat
FaceColor = 'interp';
colorbar
xlabel('$\theta_{Aug.}$')
ylabel('$\theta_{Sep.}$')
```

G Code of Parameterized Cost Function Approximation: Single Scalar Parameterization on Demand Forecast

```
%%%%%%%%%%%%%%%%%%%%%%%%
%%%% NB: Before this code, you should run Optimal.m in
%%%%     Address: C:\Users\win\Desktop\Master\Optimal Cost
%%%%%%%%%%%%%%%%%%%%%%%%
close all
clear all
clc
tic
addpath('C:\Users\win\Desktop\Master\Exogenous Information')
addpath('C:\Users\win\Desktop\Master\Optimal Cost')
load('sample_path.mat')

%% Settings
S0 = S0();
S = S0.S;
plot_settings
scenario = size(sample_path.D,1);                % Scenario number
N = size(sample_path.D,2)-1;                     % Total time horizon      [Day]

%Extention of sample path
sample_path.D   = [sample_path.D];
sample_path.Ph  = [sample_path.Ph];
sample_path.Pc  = [sample_path.Pc];
sample_path.Tinf= [sample_path.Tinf];

%% Simulation
Par_vec = [0 1 20 40 60 80 100 120];              % Vector of parameter values
rng('default')
for Parameter_sequence = 1:length(Par_vec)
        PathsCost =[];
        rng('default')
        for w = 1:scenario
            ss;
            %Initial setup
            S0 = S0();
            S = S0.S;
            %Defining plotting arrays
            xSim = [];
            uSim = [];
            timeSim = [];
            %Setting exogenous information at start of every simple path
            Tinf_horizon = sample_path.Tinf(w,:);
            Demand_horizon = Par_vec(Parameter_sequence) * sample_path.D(w,:);
            ↪  %Parameterization on Demand Forecast
            Ph_horizon = sample_path.Ph(w,:);
            Pc_horizon = sample_path.Pc(w,:);
            %MPC Control
            for k = 1:N
                nd = length(sample_path.D)-(k);            % MPC control horizon
                % Finding indexes of the negative demands
                [~, col] = find(Demand_horizon<0);
                % Creating "boolean" array
                DProductArray = ones(nd,1);
                % Indicating negative demands
```

```
DProductArray(col,:) = 0;

%% Equality constraint : Ax=b
% Demand constraint
Aeq_D = zeros(nd,7*nd);
%D_k = DPredictArray.*(3/2*U_k + Qheat_k) + (1 {
↪  DpredictArray).*(-U_k - Qcool_k)
for ii = 1:nd
    Aeq_D(ii,1 + 7*(ii - 1)) =
    ↪  (S0.COP/(S0.COP-1))*DProductArray(ii) + (-1*(1 -
    ↪  DProductArray(ii))); % For borehole daily energy flow
    ↪  (U_k)
    Aeq_D(ii,2 + 7*(ii - 1)) = (1)*DProductArray(ii) + (0)*(1 -
    ↪  DProductArray(ii));                  % For heater daily
    ↪  energy flow (Qheat_k)
    Aeq_D(ii,3 + 7*(ii - 1)) = (0)*DProductArray(ii) + (-1)*(1 -
    ↪  DProductArray(ii));                 % For cooler daily
    ↪  energy flow (Qcool_k)
end
Beq_D  = Demand_horizon(1:end-1)';
%Transition function of states
Aeq_T = zeros(nd,7*nd);
a=1-(S0.lambda/(S0.m*S0.Cpw));
b=1/(S0.m*S0.Cpw);
for iii=1:nd
    Aeq_T(iii:end,1+7*(iii-1))= b*((+1)*DProductArray(iii) +
    ↪  (-1)*(1 - DProductArray(iii)));
    Aeq_T(iii:end,5+7*(iii-1))= (a-1);
   for jjj=iii:nd
       Aeq_T(jjj,1+7*(iii-1))=a^(jjj-iii)*
       ↪  Aeq_T(jjj,1+7*(iii-1));
       Aeq_T(jjj,5+7*(iii-1))=a^(jjj-iii)*
       ↪  Aeq_T(jjj,5+7*(iii-1));
   end
end
Aeq_T=Aeq_T + kron(eye(nd),[0 0 0 1 0 0 0]);
Beq_T=zeros(nd,1);
for kkk=1:nd
    Beq_T(kkk,1) = [a^(kkk)*S.Tb];
end
% Integration of equality constraints
Aeq = [Aeq_D;Aeq_T];
Beq = [Beq_D;Beq_T];

%% Inequality constraints : Ax<=b
%Constraint handling : Tb+s_down >=Tbmin
Amin = kron(eye(nd), [0 0 0 -1 0 0 -1]);
Bmin = -S0.Tbmin*ones(nd,1);
%Constraint handling : Tb-s_up <=Tbmax
Amax = kron(eye(nd), [0 0 0 1 0 -1 0]);
Bmax = S0.Tbmax*ones(nd,1);
%Integration of inequality constraints
A = [Amin;Amax];
B = [Bmin;Bmax];

%% Objective function
f = zeros(1,7*nd);
for kk=1:nd-1
```

```
    f(1,2 + 7*(kk - 1)) = (Ph_horizon(kk));
    f(1,3 + 7*(kk - 1)) = (Pc_horizon(kk));
    f(1,6 + 7*(kk - 1)) = S0.S.s_up;       % Penalizing cost for
    ↪  violating upper constraint
    f(1,7 + 7*(kk - 1)) = S0.S.s_down;     % Penalizing cost for
    ↪  violating lower constraint
end

%% Boundaries - lower and upper
lb = zeros(7*nd,1);
lb(4:7:7*nd,1) = -inf;
for i=1:nd-1
    lb(5 + 7*(i - 1),1)=Tinf_horizon (i);
end
ub = repmat([S0.Rmax inf inf inf 0 inf inf]',nd,1);
for i=1:nd-1
    ub(5 + 7*(i - 1),1)=Tinf_horizon (i);
end

%% Optimization
w_opt = linprog(f,A,B,Aeq,Beq,lb,ub);
%  Data store from openloop optimization
u.u      = w_opt(1:7:7*nd);
u.qheat  = w_opt(2:7:7*nd);
u.qcool  = w_opt(3:7:7*nd);
u.Tb     = w_opt(4:7:7*nd);
u.Tinf   = w_opt(5:7:7*nd);
u.s_up   = w_opt(6:7:7*nd);
u.s_down = w_opt(7:7:7*nd);

%% Take an action
uk = [u.u(1); u.qheat(1); u.qcool(1); u.s_up(1); u.s_down(1)];
sample_path.uk(1+7*(w-1):3+7*(w-1),k)=uk(1:3,:);    % For visual
↪  check
sample_path.uk(6+7*(w-1),k)=w_opt(6);               % For visual
↪  check
sample_path.uk(7+7*(w-1),k)=w_opt(7);               % For visual
↪  check

%% Transition during day
S.Tb = ((1-(S0.lambda/(S0.m*S0.Cpw))) * S.Tb) +
↪  [((-1)*DProductArray(1) + (+1)*(1 - DProductArray(1)))*b 0
↪  0]*uk(1:3,end) +
↪  (S0.lambda/(S0.m*S0.Cpw))*sample_path.Tinf(w,k);
sample_path.uk(4+7*(w-1),k)=S.Tb-273;               % For visual
↪  check

%% Realization
S.Ph = sample_path.Ph(w,k+1);
S.Pc = sample_path.Pc(w,k+1);
S.D = sample_path.D(w,k+1);
S.Tinf = sample_path.Tinf(w,k+1);

%% For plotting
xSim = [xSim, S.Tb];
uSim = [uSim, uk];

%% Creating forecast error
```

```
    % Create sigma matrix
    sigma_mat=zeros(length(Tinf_horizon));
    alpha = 1;
    for i = 1:size(sigma_mat,1)
            for j = 1:size(sigma_mat,1)
                sigma_mat(i,j) = exp(-alpha*abs(i-j));
            end
    end
    % Cholesky decomposition
    L = chol(sigma_mat,'lower');
    % Multivariate normal distribution of Tinf
    sigma_E_Tinf = 1;
    mu = zeros(1,length(Tinf_horizon));
    Sigma_Tinf = sigma_E_Tinf*diag(Tinf_horizon.^0.5);
    Z_Tinf = mvnrnd(mu,Sigma_Tinf);
    % Multivariate normal distribution of Demand
    sigma_E_D = 15;
    mu = zeros(1,length(Demand_horizon));
    Demand_P=abs(Demand_horizon);
    Sigma_Demand = sigma_E_D*diag(Demand_P.^0.5);
    Z_Demand = mvnrnd(mu,Sigma_Demand);
    % Multivariate normal distribution of Heating Price
    sigma_E_Ph = 15;
    mu = zeros(1,length(Ph_horizon));
    Sigma_Ph = sigma_E_Ph*diag(Ph_horizon.^0.5);
    Z_Ph = mvnrnd(mu,Sigma_Ph);
    % Multivariate normal distribution of Cooling Price
    sigma_E_Pc = 15;
    mu = zeros(1,length(Pc_horizon));
    Sigma_Pc = sigma_E_Pc*diag(Pc_horizon.^0.5);
    Z_Pc = mvnrnd(mu,Sigma_Pc);

    %% Exogenous information developement
    % Ground temp horizon update
    Tinf_error = L*Z_Tinf';
    Tinf_horizon =  Tinf_horizon(2:end) + Tinf_error(2:end)';

    % Demand update
    Demand_error = L*Z_Demand';
    Demand_forecast = Demand_horizon(3:end) + Demand_error(3:end)';
    Demand_horizon = [sample_path.D(w,k+1),Demand_forecast];

    % Heating Price update
    Ph_error = L*Z_Ph';
    Ph_horizon = Ph_horizon(2:end) + Ph_error(2:end)';
    Ph_horizon = max(S0.Phmin,Ph_horizon);

    % Coolimg Price update
    Pc_error = L*Z_Pc';
    Pc_horizon = Pc_horizon(2:end) + Pc_error(2:end)';
    Pc_horizon=max(S0.Pcmin,Pc_horizon);
end

%% Cost calculation over sample path
% Finding indexes of the negative demands
[~, col] = find(sample_path.D(w,1:N)<0);
% Creating "boolean" array
DemandProductArray = ones(N,1);
```

```
            % Indicating negative demands
            DemandProductArray(col,:) = 0;

            % Finding indexes when demands are positive and calculated energy
            ↪  flows
            % from borehole will not satisfy the demand, The shortage have to be
            ↪  supplied by Heater
            [~, col] = find(sample_path.D(w,1:N)<0 | sample_path.D(w,1:N) -
            ↪  ((S0.COP/(S0.COP-1)*uSim(1,1:N) + uSim(2,1:N))+0.001)<=0);
            % Creating "boolean" array
            Remain_Qh_Array = ones(N,1);
            % Indicating negative demands
            Remain_Qh_Array(col,:) = 0;

            % Finding indexes when demands are negative and calculated energy
            ↪  flows
            % to borehole will not satisfy the demand, The shortage have to be
            ↪  supplied by Cooler
            [~, col] = find(sample_path.D(w,1:N)>0 | abs(sample_path.D(w,1:N)) -
            ↪  (uSim(1,1:N) + uSim(3,1:N)+0.001) <=0);
            % Creating "boolean" array
            Remain_Qc_Array = ones(N,1);
            % Indicating negative demands
            Remain_Qc_Array(col,:) = 0;

            % Penalty for constraint violation
            Penalty_up  = max(sample_path.uk(4+7*(w-1),:) + 273 - (S0.Tbmax *
            ↪  ones (1,N)) , 0);
            Penalty_down= abs(min(sample_path.uk(4+7*(w-1),:) + 273 - (S0.Tbmin *
            ↪  ones (1,N)) , 0));

            % Cost Over a sample path
            PathCost_i = sum(sample_path.Ph(w,1:N) .* uSim(2,1:N) + ...
                             + sample_path.Pc(w,1:N) .* uSim(3,1:N) + ...
                             + S0.S.s_up * Penalty_up ...
                             + S0.S.s_down * Penalty_down ...
                             + Remain_Qh_Array(1:N)' .* (sample_path.D(w,1:N) -
                             ↪  (S0.COP/(S0.COP-1)*uSim(1,1:N) + uSim(2,1:N)))
                             ↪  .* sample_path.Ph(w,1:N) + ...
                             + Remain_Qc_Array(1:N)' .*
                             ↪  (abs(sample_path.D(w,1:N)) - (uSim(1,1:N) +
                             ↪  uSim(3,1:N))) .* sample_path.Pc(w,1:N));
           % Making cost vector of different scenarios
           PathsCost = [PathsCost; PathCost_i];
        end
        % Obtaining average cost
        Parametrized_Cost(Parameter_sequence) = mean(PathsCost);
        for i = 1:size(PathsCost,1)
            Avg_cost_cum(i) = mean(PathsCost(1:i));
        end
end

%% Plotting Policy Cost Index against theta_{i}
hold on
plot(Par_vec, (Parametrized_Cost-sample_path.OpCo)/sample_path.OpCo)
grid('on');
box('on');
xlabel('$\theta$')
```

```
ylabel('$ \Delta F^{\pi} (\theta) $')
% save
save = ('P02_MPC')

%% Plotting the states and inputs for a given scenario
trial = 1;
sample_path.ukplot(1:4,:) = sample_path.uk(1+7*(trial-1):4+7*(trial-1),:);
sample_path.ukplot(5,:)= sample_path.D(trial,1:N);
for ind= 1:N
    if sample_path.ukplot(5,ind)>=0
       sample_path.ukplot(1,ind)=(S0.COP/(S0.COP-1))*sample_path.ukplot(1,ind)
    else
       sample_path.ukplot(1,ind)=-1*sample_path.ukplot(1,ind)
       sample_path.ukplot(3,ind)=-1*sample_path.ukplot(3,ind)
    end
end

figure
subplot(5,1,1)
plot(sample_path.ukplot(4,:),'b-')
xlabel('Time, Day')
ylabel('$T^{b}$ [Deg. C]')

subplot(5,1,2)
stairs(0:N,[sample_path.ukplot(1,:), nan],'b-')
xlabel('Time, Day')
ylabel('$Q^{b}$ [MWh]')

subplot(5,1,3)
stairs(0:N,[sample_path.ukplot(2,:), nan],'b-')
xlabel('Time, Day')
ylabel('$Q^{h}$ [MWh]')
axis([1 400 0  20])


subplot(5,1,4)
stairs(0:N,[sample_path.ukplot(3,:), nan],'b-')
xlabel('Time, Day')
ylabel('$Q^{c}$ [MWh]')

subplot(5,1,5)
plot(sample_path.ukplot(5,:), 'b')
xlabel('Time, Day')
ylabel('Demand [MWh]')
```

H Code of parameterized cost function approximation: Monthly-Based Parameterization on Demand forecast constraint

```
%%%%%%%%%%%%%%%%%%%%%%%%
%%%% NB: Before this code, you should run Optimal.m in
%%%%     Address: C:\Users\win\Desktop\Master\Optimal Cost
%%%%%%%%%%%%%%%%%%%%%%%%
close all
clear all
clc
tic
addpath( 'C:\Users\win\Desktop\Master\Exogenous Information')
addpath( 'C:\Users\win\Desktop\Master\Optimal Cost')
load('sample_path.mat')

%% Settings
S0 = S0();
S = S0.S;
plot_settings
scenario = size(sample_path.D,1);     % Scenario number
N = size(sample_path.D,2)-1;          % Total time horizon              [Day]

%Extention of sample path
sample_path.D    = [sample_path.D];
sample_path.Ph   = [sample_path.Ph];
sample_path.Pc   = [sample_path.Pc];
sample_path.Tinf= [sample_path.Tinf];

%% Simulation
Par_vec = [0 0.5 1 2 3 4 5 6 7 8 9 10];                % Vector of parameters
↪  will be multiplied by demand
Days_number = [30 31 31 28 31 30 31 30 31 31 30 31];   % Vector of days in every
↪  month
Days_number_cumulative =[0, cumsum(Days_number)];
rng('default')
Parametrized_Cost=zeros(size(Days_number,2), size(Par_vec,2));
for month=1:length(Days_number)
    for Parameter_sequence = 1:length(Par_vec)
        PathsCost =[];
        rng('default')
       for w = 1:scenario
           ss;
           %Initial setup
           S0 = S0();
           S = S0.S;
           %Defining Plotting Arrays
           xSim = [];
           uSim = [];
           timeSim = [];

           %Setting parametrized demand at start of every simple path
           Demand_horizon = sample_path.D(w,:);
           Demand_horizon(Days_number_cumulative(month)+1 :
           ↪  Days_number_cumulative(month+1)) = ...
           Par_vec(Parameter_sequence) *
           ↪  Demand_horizon(Days_number_cumulative(month)+1 :
           ↪  Days_number_cumulative(month+1));
```

```
%Setting other exogenous information at start of every simple path
Tinf_horizon = sample_path.Tinf(w,:);
Ph_horizon = sample_path.Ph(w,:);
Pc_horizon = sample_path.Pc(w,:);
%MPC Control
for k = 1:N
    nd = length(sample_path.D)-(k);       % MPC control horizon
    % Finding indexes of the negative demands
    [~, col] = find(Demand_horizon<0);
    % Creating "boolean" array
    DProductArray = ones(nd,1);
    % Indicating negative demands
    DProductArray(col,:) = 0;
    %% Equality constraint : Ax=b
    % Demand constraint
    Aeq_D = zeros(nd,7*nd);
    %D_k = DPredictArray.*(3/2*U_k + Qheat_k) + (1 {
    ↪  DpredictArray).*(-U_k - Qcool_k)
    for ii = 1:nd
        Aeq_D(ii,1 + 7*(ii - 1)) =
        ↪  (S0.COP/(S0.COP-1))*DProductArray(ii) + (-1*(1 -
        ↪  DProductArray(ii))); % for borehole daily energy flow
        ↪  (U_k)
        Aeq_D(ii,2 + 7*(ii - 1)) = (1)*DProductArray(ii) + (0)*(1 -
        ↪  DProductArray(ii));                  % for heater daily
        ↪  energy flow (Qheat_k)
        Aeq_D(ii,3 + 7*(ii - 1)) = (0)*DProductArray(ii) + (-1)*(1 -
        ↪  DProductArray(ii));                 % for cooler daily
        ↪  energy flow (Qcool_k)
    end
    Beq_D  = Demand_horizon(1:end-1)';
    %Transition function
    Aeq_T = zeros(nd,7*nd);
    a=1-(S0.lambda/(S0.m*S0.Cpw));
    b=1/(S0.m*S0.Cpw);
    for iii=1:nd
        Aeq_T(iii:end,1+7*(iii-1))= b*((+1)*DProductArray(iii) +
        ↪  (-1)*(1 - DProductArray(iii)));
        Aeq_T(iii:end,5+7*(iii-1))= (a-1);
        for jjj=iii:nd
            Aeq_T(jjj,1+7*(iii-1))= a^(jjj-iii) *
            ↪  Aeq_T(jjj,1+7*(iii-1));
            Aeq_T(jjj,5+7*(iii-1))= a^(jjj-iii) *
            ↪  Aeq_T(jjj,5+7*(iii-1));
        end
    end
    Aeq_T=Aeq_T + kron(eye(nd),[0 0 0 1 0 0 0]);
    Beq_T=zeros(nd,1);
    for kkk=1:nd
        Beq_T(kkk,1) = [a^kkk*S.Tb];
    end
    % Integration of equality constraints
    Aeq = [Aeq_D;Aeq_T];
    Beq = [Beq_D;Beq_T];

    %% Inequality constraints : Ax<=b
    %Constraint handling : Tb+s_down >=Tbmin
    Amin = kron(eye(nd), [0 0 0 -1 0 0 -1]);
```

```
Bmin = -S0.Tbmin*ones(nd,1);
%Constraint handling : Tb-s_up <=Tbmax
Amax = kron(eye(nd), [0 0 0 1 0 -1 0]);
Bmax = S0.Tbmax*ones(nd,1);
%Integration of inequality constraints
A = [Amin;Amax];
B = [Bmin;Bmax];

%% Objective function
f = zeros(1,7*nd);
for kk=1:nd-1
    f(1,2 + 7*(kk - 1)) = (Ph_horizon(kk));
    f(1,3 + 7*(kk - 1)) = (Pc_horizon(kk));
    f(1,6 + 7*(kk - 1)) = S0.S.s_up;            % Penalizing cost
    ↪  for violating upper constraint
    f(1,7 + 7*(kk - 1)) = S0.S.s_down;          % Penalizing cost
    ↪  for violating upper constraint
end

%% Boundaries - lower and upper
lb = zeros(7*nd,1);
lb(4:7:7*nd,1) = -inf;
for i=1:nd-1
    lb(5 + 7*(i - 1),1)=Tinf_horizon (i);
end
ub = repmat([S0.Rmax inf inf inf 0 inf inf]',nd,1);
for i=1:nd-1
    ub(5 + 7*(i - 1),1)=Tinf_horizon (i);
end

%% Optimization
w_opt = linprog(f,A,B,Aeq,Beq,lb,ub);
%  Data store from openloop optimization
u.u      = w_opt(1:7:7*nd);
u.qheat  = w_opt(2:7:7*nd);
u.qcool  = w_opt(3:7:7*nd);
u.Tb     = w_opt(4:7:7*nd);
u.Tinf   = w_opt(5:7:7*nd);
u.s_up   = w_opt(6:7:7*nd);
u.s_down = w_opt(7:7:7*nd);

%% Take an action
uk = [u.u(1); u.qheat(1); u.qcool(1); u.s_up(1); u.s_down(1)];
sample_path.uk(1+7*(w-1):3+7*(w-1),k)=uk(1:3,:);    % For visual
↪  check
sample_path.uk(6+7*(w-1),k)=w_opt(6);               % For visual
↪  check
sample_path.uk(7+7*(w-1),k)=w_opt(7);               % For visual
↪  check

%% Transition during day
S.Tb = ((1-(S0.lambda/(S0.m*S0.Cpw))) * S.Tb) +
↪  [((-1)*DProductArray(1) + (+1)*(1 - DProductArray(1)))*b 0
↪  0]*uk(1:3,end) +
↪  (S0.lambda/(S0.m*S0.Cpw))*sample_path.Tinf(w,k);
sample_path.uk(4+7*(w-1),k)=S.Tb-273;               % For visual
↪  check
```

```
%% Realization
S.Ph = sample_path.Ph(w,k+1);
S.Pc = sample_path.Pc(w,k+1);
S.D = sample_path.D(w,k+1);
S.Tinf = sample_path.Tinf(w,k+1);

%% For plotting
xSim = [xSim, S.Tb];
uSim = [uSim, uk];

%% Creating forecast error
% Create sigma matrix
sigma_mat=zeros(length(Tinf_horizon));
alpha = 1;
for i = 1:size(sigma_mat,1)
    for j = 1:size(sigma_mat,1)
        sigma_mat(i,j) = exp(-alpha*abs(i-j));
    end
end
% Cholesky decomposition
L = chol(sigma_mat,'lower');
% Multivariate normal distribution of Tinf
sigma_E_Tinf = 1;
mu = zeros(1,length(Tinf_horizon));
Sigma_Tinf = sigma_E_Tinf*diag(Tinf_horizon.^0.5);
Z_Tinf = mvnrnd(mu,Sigma_Tinf);
% Multivariate normal distribution of Demand
sigma_E_D = 15;
mu = zeros(1,length(Demand_horizon));
Demand_P=abs(Demand_horizon);
Sigma_Demand = sigma_E_D*diag(Demand_P.^0.5);
Z_Demand = mvnrnd(mu,Sigma_Demand);
% Multivariate normal distribution of Heating Price
sigma_E_Ph = 15;
mu = zeros(1,length(Ph_horizon));
Sigma_Ph = sigma_E_Ph*diag(Ph_horizon.^0.5);
Z_Ph = mvnrnd(mu,Sigma_Ph);
% Multivariate normal distribution of Cooling Price
sigma_E_Pc = 15;
mu = zeros(1,length(Pc_horizon));
Sigma_Pc = sigma_E_Pc*diag(Pc_horizon.^0.5);
Z_Pc = mvnrnd(mu,Sigma_Pc);

%% Exogenous information developement
% Ground temp horizon update
Tinf_error = L*Z_Tinf';
Tinf_horizon =  Tinf_horizon(2:end) + Tinf_error(2:end)';
% Demand update
Demand_error = L*Z_Demand';
Demand_forecast = Demand_horizon(3:end) + Demand_error(3:end)';
Demand_horizon = [sample_path.D(w,k+1),Demand_forecast];

% Heating Price update
Ph_error = L*Z_Ph';
Ph_horizon = Ph_horizon(2:end) + Ph_error(2:end)';
Ph_horizon = max(S0.Phmin,Ph_horizon);

% Coolimg Price update
```

```
            Pc_error = L*Z_Pc';
            Pc_horizon = Pc_horizon(2:end) + Pc_error(2:end)';
            Pc_horizon=max(S0.Pcmin,Pc_horizon);
        end

        %% Cost calculation over sample path
        % Finding indexes of the negative demands
        [~, col] = find(sample_path.D(w,1:N)<0);
        % Creating "boolean" array
        %DemandProductArray = S0.COP/(S0.COP-1)*ones(N+1,1);
        DemandProductArray = ones(N,1);
        % Indicating negative demands
        DemandProductArray(col,:) = 0;

        % Finding indexes when demands are positive and calculated energy
        ↪  flows
        % from borehole will not satisfy the demand, The shortage have to be
        ↪  supplied by Heater
        [~, col] = find(sample_path.D(w,1:N)<0 | sample_path.D(w,1:N) -
        ↪  ((S0.COP/(S0.COP-1)*uSim(1,1:N) + uSim(2,1:N))+0.001)<=0);
        % Creating "boolean" array
        Remain_Qh_Array = ones(N,1);
        % Indicating negative demands
        Remain_Qh_Array(col,:) = 0;

        % Finding indexes when demands are negative and calculated energy
        ↪  flows
        % to borehole will not satisfy the demand, The shortage have to be
        ↪  supplied by Cooler
        [~, col] = find(sample_path.D(w,1:N)>0 | abs(sample_path.D(w,1:N)) -
        ↪  (uSim(1,1:N) + uSim(3,1:N)+0.001) <=0);
        % Creating "boolean" array
        Remain_Qc_Array = ones(N,1);
        % Indicating negative demands
        Remain_Qc_Array(col,:) = 0;

        % Penalty for constraint violation
        Penalty_up  = max(sample_path.uk(4+7*(w-1),:) + 273 - (S0.Tbmax *
        ↪  ones (1,N)) , 0);
        Penalty_down= abs(min(sample_path.uk(4+7*(w-1),:) + 273 - (S0.Tbmin *
        ↪  ones (1,N)) , 0));

        % Cost Over a sample path
        PathCost_i = sum(sample_path.Ph(w,1:N) .* uSim(2,1:N) + ...
                        + sample_path.Pc(w,1:N) .* uSim(3,1:N) + ...
                        + S0.S.s_up * Penalty_up ...
                        + S0.S.s_down * Penalty_down ...
                        + Remain_Qh_Array(1:N)' .* (sample_path.D(w,1:N) -
                        ↪  (S0.COP/(S0.COP-1)*uSim(1,1:N) + uSim(2,1:N)))
                        ↪  .* sample_path.Ph(w,1:N) + ...
                        + Remain_Qc_Array(1:N)' .*
                        ↪  (abs(sample_path.D(w,1:N)) - (uSim(1,1:N) +
                        ↪  uSim(3,1:N))) .* sample_path.Pc(w,1:N));
        % Making cost vector of different scenarios
        PathsCost = [PathsCost; PathCost_i];
    end
    % Obtaining average cost
    Parametrized_Cost(month,Parameter_sequence) = mean(PathsCost);
```

```
        for i = 1:size(PathsCost,1)
            Avg_cost_cum(i) = mean(PathsCost(1:i));
        end
    end
end

%% Plotting policy improvement against theta_{i}
hold on
plot(Par_vec, (Parametrized_Cost(1,:)-sample_path.OpCo)/sample_path.OpCo,
↪  Par_vec, (Parametrized_Cost(2,:)-sample_path.OpCo)/sample_path.OpCo)
plot(Par_vec, (Parametrized_Cost(3,:)-sample_path.OpCo)/sample_path.OpCo,
↪  Par_vec, (Parametrized_Cost(4,:)-sample_path.OpCo)/sample_path.OpCo)
plot(Par_vec, (Parametrized_Cost(5,:)-sample_path.OpCo)/sample_path.OpCo,
↪  Par_vec, (Parametrized_Cost(6,:)-sample_path.OpCo)/sample_path.OpCo)
plot(Par_vec, (Parametrized_Cost(7,:)-sample_path.OpCo)/sample_path.OpCo,
↪  Par_vec, (Parametrized_Cost(8,:)-sample_path.OpCo)/sample_path.OpCo)
plot(Par_vec, (Parametrized_Cost(9,:)-sample_path.OpCo)/sample_path.OpCo,
↪  Par_vec, (Parametrized_Cost(10,:)-sample_path.OpCo)/sample_path.OpCo)
plot(Par_vec, (Parametrized_Cost(11,:)-sample_path.OpCo)/sample_path.OpCo,
↪  Par_vec, (Parametrized_Cost(12,:)-sample_path.OpCo)/sample_path.OpCo)
legend('$\theta_{Nov.}$','$\theta_{Dec.}$','$\theta_{Jan.}$','$\theta_{Feb.}$','$\theta_{Mar.}$',.

        ↪  '$\theta_{Apr.}$','$\theta_{May.}$','$\theta_{Jun.}$','$\theta_{Jul.}$','$\theta_{Aug.}
        '$\theta_{Sep.}$','$\theta_{Oct.}$')
grid('on');
box('on');
xlabel('$\theta$')
ylabel('$ \Delta F^{\pi} (\theta) $')
% save
save = ('P02_MPC')

%% Plotting the states and inputs for a given scenario
trial = 1;
sample_path.ukplot(1:4,:) = sample_path.uk(1+7*(trial-1):4+7*(trial-1),:);
sample_path.ukplot(5,:)= sample_path.D(trial,1:N);
for ind= 1:N
    if sample_path.ukplot(5,ind)>=0
   sample_path.ukplot(1,ind)=(S0.COP/(S0.COP-1))*sample_path.ukplot(1,ind)
    else
    sample_path.ukplot(1,ind)=-1*sample_path.ukplot(1,ind)
    sample_path.ukplot(3,ind)=-1*sample_path.ukplot(3,ind)
    end
end

figure
subplot(5,1,1)
plot(sample_path.ukplot(4,:),'b-')
xlabel('Time, Day')
ylabel('$T^{b}$ [Deg. C]')

subplot(5,1,2)
stairs(0:N,[sample_path.ukplot(1,:), nan],'b-')
xlabel('Time, Day')
ylabel('$Q^{b}$ [MWh]')

subplot(5,1,3)
stairs(0:N,[sample_path.ukplot(2,:), nan],'b-')
xlabel('Time, Day')
```

```
ylabel('$Q^{h}$ [MWh]')

subplot(5,1,4)
stairs(0:N,[sample_path.ukplot(3,:), nan],'b-')
xlabel('Time, Day')
ylabel('$Q^{c}$ [MWh]')

subplot(5,1,5)
plot(sample_path.ukplot(5,:), 'b')
xlabel('Time, Day')
ylabel('Demand [MWh]')
```

Master's thesis

2021

Alireza Mirzaei

NTNU
Norwegian University of
Science and Technology