Master's thesis

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

Ding Nan

Optimal Control of TES System by Using Nonlinear Model Predictive Control

Master's thesis in Chemical Process Technology

Supervisor: Johannes Jäschke (Associate Professor), Zawadi Ntengua Mdoe (PhD Candidate)

June 2020

NTNU
Norwegian University of Science and Technology

TKP4900 - Chemical Process Technology, Master's Thesis

Optimal Control of TES System by Using Nonlinear Model Predictive Control

Ding Nan

Submission date: June 07.2020
Supervisor: Johannes Jäschke, Associate Professor IKP
Co-Supervisor: Zawadi Ntengua Mdoe, PhD Candidate IKP

Norwegian University of
Science and Technology
Norwegian University of Science and Technology
Department of Chemical Engineering

Abstract

Industrial waste heat (WE) recovery process reuses heat energy that would otherwise be expelled and wasted to ensure and enhance resource conservation, waste as well as costs reduction. Under this circumstance, thermal energy storage (TES) system can be used to store WE for meeting future peak energy demand. With the presence of TES tank, the performance of simple thermal energy system has been significantly increased. In order to realize considerable energy saving and cost reduction, one of the significant optimal operation strategies, nonlinear model predictive control (NMPC) is implemented. The performance and stability of NMPC depend strongly on the accuracy of the model which is utilized in the optimization. However, in many practical application domains, uncertainty like plant-model mismatch is present and severely effects its robustness. As a conventional method, standard NMPC works well without considering uncertainty such as plant-model mismatch. However, after taking into account uncertainty, standard NMPC offers few robustness against it, where energy supply and demand profile may vary considerably from their predicted profiles. So as to significantly reject uncertainty, one of the robust version of NMPCs, scenario-based multistage NPMC is implemented, where the evolution of uncertainties is modeled as a propagating scenario tree along prediction horizon.

The numerical method for solving optimal control problem in this thesis is selected as direct collocation method. Then NLP problem is defined in CasADi framework and solved by IPOPT within the MATLAB programming environment. Simulation studies have been carried out for three major and one minor cases, namely optimal control of a simple TES system by standard NMPC with and without direct solar heating without plant-model mismatch, optimal control of a simple TES system by multistage NMPC and optimal control of a simple thermal energy system by standard NMPC without storage tank without plant-model mismatch.

Preface

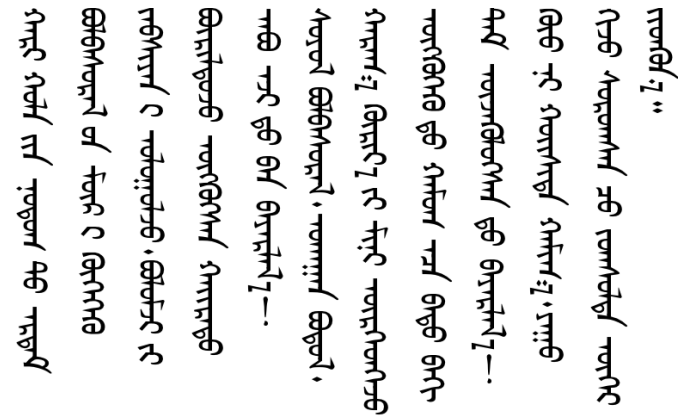
This master thesis is written for the course **TKP4900 Chemical Process Technology, Master's Thesis** at Norwegian University of Science and Technology (NTNU). The work of this thesis was carried out during the Spring semester of 2020 at Process Systems Engineering group of the Department of Chemical Engineering.

I would like to express my deepest gratitude towards my supervisor Associate Professor Johannes Jäschke, for giving me the valuable opportunity to work on this challenging project and also for his excellent guidance and support during this process.

I am also extremely thankful to my co-supervisor PhD Candidate Zawadi Ntengua Mdoe for his constant guidance and close assistance throughout the project, for being available to me even while being away on paternity leave, for sharing knowledge and providing technical background on this work and for reviewing the thesis. It was quite inconvenient to get frequent contact with each other during this serious corona-virus spreading period, but sincerely thanks to him for providing me normal weekly online meeting and even more. With his guidance, support and discussion, I became familiar with this quite new work very soon. Thanks to his family and bless their newborn baby.

I would also like to thank other PhD candidates and my fellow master students at the Process Systems Engineering Group. The open and sharing environment with you has helped me get through this thesis. And my friends who are in Trondheim or not for supporting me.

Finally, I would like to thank my parents for supporting, guiding and loving me. Since my parents are not familiar with English, it is better for me to express my gratitude in Mongolian.



Declaration of Compliance

I declare that this is an independent work according to the exam regulations of the Norwegian University of Science and Technology (NTNU).

Ding Nan
Trondheim, Norway
June 07, 2020

Table of Contents

List of Tables

List of Figures

Abbreviations

AD	Algorithmic Differentiation	
CAS	Computer Algebra System	
CasADi	Computer Algebra System for Automatic Differentiation	
DAE	Differential Algebraic Equation	
ES	Energy Storage	
HEX-i	i-th Heat Exchanger	
HTF	Heat Transfer Fluid	
iNMPC	A Certain Type of Nonlinear Model Predictive Control	
IPOPT	Interior Point Optimization	
LMTD	Logarithmic Mean Temperature Difference	ΔT_{lm}
msNMPC	Multistage Nonlinear Model Predictive Control	
MIMO	Multiple-Input-Multiple-Output	
MPC	Model predictive control	
MTD	Mean Temperature Difference	ΔT_m
NMPC	Nonlinear Model Predictive Control	
ODE	Ordinary Differential Equation	
OCP	Optimal Control Problem	
PCMs	Phase Change Materials	
PDE	Partial Differential Equations	
sNMPC	Standard Nonlinear Model Predictive Control	
SQP	Sequential Quadratic Programming	
TES	Thermal Energy Storage	
WE	Waste Heat	
A	Surface Area	$[m^2]$
c_p	Specific Heat Capacity	$[kJ/kgK]$
P	Energy Source Cost	
Q	Heat Flow Rate	$[kW]$
q	Volumetric Flow Rate	$[m^3/s]$
T	Temperature	$[^\circ C]$
U	Overall heat transfer coefficient	$[kW/m^2K]$
V	Volume	$[m^3]$
ρ	Density	$[kg/m^3]$

Chapter 1

Introduction

Energy is needed for us, no matter in electrical or thermal energy form. But in the most cases not where or when energy is available. For example, someone is listening music while jogging outside, it is not possible to stand beside the socket or even carry with. In order to solve this problem, battery is charged with electrical energy in advance or even power bank is charged for later necessary use. These two examples are the most common ways of electrical energy storage. Was it possible for people to drink cold beverage before the invention of fridge or freezer? The answer is yes. At that time people cut and collect ice from the lakes during the winter time and stored it in deep cellars, which is an example of storing cold from the winter to the summer. Moreover, summer solar heat can also be stored and then used in winter for the heating of buildings. These applications are the reflections of long-term thermal energy storage and renewable energy utilization. However, if there is no energy storage medium, it is not possible to achieve the above-mentioned applications. Therefore, it is necessary to apply energy storage to store energy, which are especially waste heat from industrial processes, electricity from photo-voltaic panels and etc, and then use these stored energy extensively.

Renewable energy like solar thermal energy, wind energy and photovoltaics is the emerging energy sources to substitute traditional fossil fuels. In some fields, the application and technique of renewable energy are already mature. It is necessary to mention that this substitution can be fulfilled by energy storage systems. The implementation of energy storage system can lead to CO_2 emission reduction and substantial energy conservation. In addition, energy storage systems like thermal and electrical energy storage system are able to more efficiently use renewable energy, which is intermittent, by matching the energy supply with the demand by storing energy during off-peak times and dispatch during peak times. Thus, it is clear to know that energy storage technologies can resolve problems caused by the intermittent energy supply sources. The energy sources can come from conventional industrial plant which is constant and also from the industry surplus energy which is the waste heat.

In this thesis, the surplus heat is applied as the energy source to supply to a thermal energy

storage (TES) unit and then extracted to the demand side. The reusing of surplus industrial heat can reduce the energy consumption, energy costs and increase energy efficiency. In TES system, TES unit plays an important role to act as a buffer between the supply and demand of surplus heat to further reduce the requirement of extra external peak-heating energy. While after implementing TES unit in the energy system, it is crucial to take into account its dynamic operation for example the storage process itself is transient. Thus, an appropriate control strategy is critical. Taking into account the system will be complex, nonlinear, dynamic and there will also be consideration for process operating constraints, therefore it is better to select the advanced control methods for TES optimal control. These advanced control methods often use optimization in order to handle the additional complexity and constraints (Edgar TF et al. (2001)). There are a wide variety of control techniques have been applied for advanced process control, among these techniques model predictive control (MPC) is the spotlight, which uses system model to forecast the system performance and then determine the control inputs which are the result from optimization problem. Since the system model is nonlinear model, one of the MPC algorithm, NMPC is then applied as the main core of the two different NMPC algorithm in this thesis.

However, in practice, the operation of TES system encounters various system uncertainties. For example, the temperature fluctuation arises at the supply and demand side, which causes influence on the efficient application of TES unit, daily varying commercial energy price, which influences the operating costs, and so on. If these uncertainties are neglected in the system, mismatch between the real plant and the derived model will occur and further lead to constraint violations, sub-optimal and even infeasible solution. Thus, it will cause the large energy cost to satisfy the consumer demands, and even lead to the high CO_2 emissions if the commercial energy purchased from energy market for peak-heating is coming from the fossil fuel. In order to reject large disturbances arise in supply and demand side along the control procedure, standard NMPC is modified to obtain robust NMPC control strategy. In this thesis, scenario-based multistage NMPC is employed to achieve robust operation against uncertainty. Because multistage NMPC is much less conservative than other robust MPC methods such as min-max MPC and tube-based NMPC according to Campo,P.J.and Morari,M.(1987). Then in order to avoid the exponential increase of problem size with the prediction horizon, robust horizon is taking to seize the maximum information of uncertainty with fewer different scenarios.

In order to solve the optimal control problem of a simple thermal energy storage system, one of the most common numerical method, direct collocation method is applied to convert the continuous time dynamic problem into finite horizon dynamic optimization problem.

1.1 Structure of the thesis

The remainder of this thesis is structured as follows:

1. **Chapter 2:** The background information and theories of energy storage (ES) systems, thermal energy storage (TES) system, TES basic thermodynamics, water storge TES system, heat exchanger and its heat transfer mechanisms, logarithmic mean temperature difference (LMTD) and its important approximations are introduced in

order to have a deep insight into the simple TES system which is the basic and major framework of this thesis.

2. **Chapter 3:** The basic concepts of optimal control problem (OCP) and numerical methods for solving this problem are introduced to solve the corresponding open loop optimization problem of a simple TES system. In addition, several types of model predictive control (MPC) algorithms are covered to achieve the optimal control purpose to reduce the cost caused by external commercial energy purchase and further reduce the extra energy consumption and carbon emission .

3. **Chapter 4:** Different model types and methods as well as tools used in this thesis both in modeling and simulation of a simple TES system are covered in this chapter.

4. **Chapter 5:** In order to correctly model the simple TES system, the model descriptions and necessary assumptions are held in advance to derive dynamic system model by using governing equations. In addition, energy demand modeling is covered in this chapter to meet the different energy demand profiles in diverse case studies.

5. **Chapter 6:** The modeling of a simple TES system is used in three main case studies and one minor case study to study and explore the effects of different energy demand profiles and performances of different NMPC algorithms on optimal controlling. Two of the case studies considered here are about standard NMPC on non-plant-model mismatch dynamic with and without direct solar heating; Multistage NMPC with uncertainty consideration; And an extra minor case study to explore the effect and benefit of using TES unit.

6. **Chapter 7:** Simulation result of each case and result comparisons are presented in this chapter, as well as the discussions.

7. **Chapter 8:** Conclusions of this thesis work and recommendations for future improvements are stated in this chapter.

Chapter 2

Thermal energy storage (TES) systems

The main objective of this chapter is to provide a comprehensive introduction on the basic theories and concepts which closely related to thermal energy storage, since the work of this thesis is based on the TES system and it constructs the main overall structure of the system. As a method for storing heat or cold in its most primitive forms for future demand, TES is typically cost-effective when compared to other storage technologies which strongly rely on expensive and even exotic materials such as storage by battery. Then, the heat transfer mechanism of heat exchanger is introduced, which is the essential part of the energy balance of the system. Above all, before introducing TES, a brief introduction of energy storage system is given in the first section of this chapter.

2.1 Energy storage systems

Energy storage (ES) systems are becoming an essential support for modern living and it has a significant impact on modern technology. For simplicity, energy storage is the capture of energy produced at one time and for the use at a later time. Particularly, energy storage is very essential to the success of energy, which is intermittent, in satisfying the demand. Thus, ES systems dedicate significantly to satisfying the energy requirements by more efficient and environmentally benign energy in many utilization fields like heating and cooling of the building etc.

The application of ES systems leads to significant benefits in practice according to Dinçer, I. and Rosen, M.A.(2010):

- Decreased energy costs.
- Decreased energy consumption.
- Ameliorated indoor air quality.

- Increased operational flexibility.
- Decreased initial and maintenance costs.

Moreover, there are also some further advantages of ES systems which had been pointed out by Dinçer, I.(1997)):

- Decreased equipment size.
- More efficient and effective equipment utilization.
- Promoting more efficient energy use and fuel substitution, which leads to increased conservation of fossil fuels.
- Decreased CO_2 and some other pollutant emissions.

Energy demand is always seasonal, weekly and even daily varying in the sectors like commercial, industrial, residential and so on. In many cases, energy demand is satisfied by synergistically operated energy-conversion systems. However, in meeting energy demand purpose, meeting peak-hour energy demand is the most troublesome and costly to supply. In general, peak-demand is always met by conventional fossil fuels, which are relatively expensive and scarce. Thus, the utilization of ES system is an alternative way for supplying peak energy demand. Nevertheless, the price of fossil fuels have been fallen to the bottom in this abnormal covid-19 crisis period, and it seems like the cost of these are much lower than novel and eco-friendly energy production but the goal of building more sustainable ecosystem cannot be terminated by this unusual phenomena. There are given some details of ES system applications:

- The energy source of ES system usually comes from relatively cheap base-load electricity, wind and run-of-river hydro, industrial process high-temperature waste heat and sunny-day solar energy etc.
- Theses energy sources are stored in ES system for the future peak energy demand such as during nighttime or cloudy days.
- The application of ES system can increase the capacity factor of corresponding devices and also increase its economic value.

However, when the energy stored is no more adequate to meet the peak energy demand, an external exterior energy have to be purchased as emergency measure to fulfill the demand. Therefore, it is necessary to comprehend the energy demand trend in advance to make better decisions for the energy supply profile and ensure the energy stored in ES system is sufficient to satisfy energy demand as much as possible. This comprehension requires the forecasting of energy demand. But in practice, the forecasting can not be guaranteed to equal to the actual demand trend, since it is always varying.

For energy technologies, storage is an important aspect, and various ES techniques have been developed or are under development now. The categories of these techniques which are used to store energy is shown in Figure 2.1.

Each method has its own corresponding applied fields and pros and cons, but the concrete

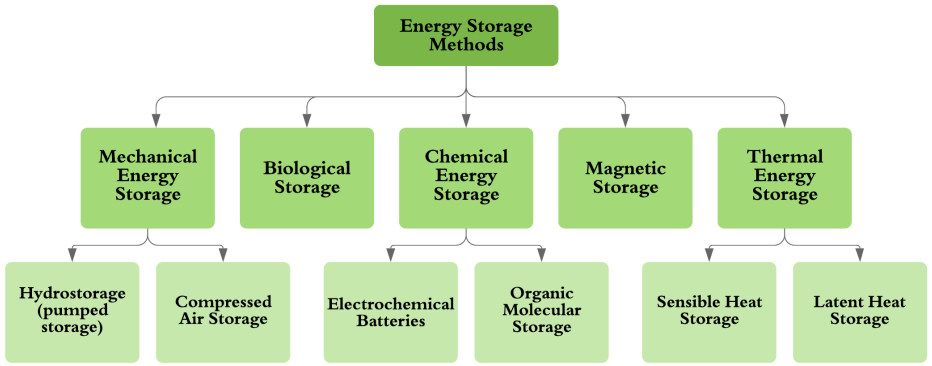


Figure 2.1: Classification of energy storage methods

introductions are not being mentioned in this thesis. The detailed information of each method can be found in Dinçer, I. and Rosen, M.A.(2010). In this thesis, thermal energy storage method is selected as the energy storage method to establish the system and achieve optimal control objective.

2.2 Thermal energy storage systems

2.2.1 Introduction

Thermal energy storage systems, which generally involves an interim storage medium, are temporarily storing energy like heat or cold for the later use. Thus, TES is also known as heat or cold storage (Mehling, H. and Cabeza, L.F. (2008)). It is very easy to find out some examples related to the utilization of TES. For example, solar thermal energy is stored during the daytime for the overnight use. Before the fridge was invented, lake and river ice were collected during the winter and stored in the deep cellar for space or beverage cooling in the summer time. Nowadays, the main use of TES systems is to offset the mismatch between energy supply and demand in commercial, industrial and daily-life utility parts, since the rapid growth of energy demand and also its seasonal, weekly and even daily variations. Hence, TES systems are needed to satisfy the future peak-demand when the extra energy sources besides surplus energy are coming from renewable energy, which is intermittent, and energy supply rate is small compared with the instantaneous high demand during the peak-demand period. TES is being promoted, since it can decrease the energy consumption and can further reduce the use of traditional fossil fuels. Therefore TES has a vital meaning to energy conservation. According to Mehling, H. and Cabeza, L.F. (2008), TES systems have following potential advantages:

- Better economics: reduced investment and operational costs.
- Better efficiency: energy is used more efficiently.
- Environmental friendly: less pollution to the environment and less CO_2 emissions.

- Better system performance and reliability.

In TES systems, surplus energy is supplied to a storage system for the later necessary use. The complete storage cycle of TES system is shown in Figure 2.2, which involves at least three steps: *charging, storing, and discharging*. In practical applications, it is possible for some of the steps take place simultaneously, and each step can arise more than once in each storage cycle (Gil,A. and Medrano,M. et al.(2011)).

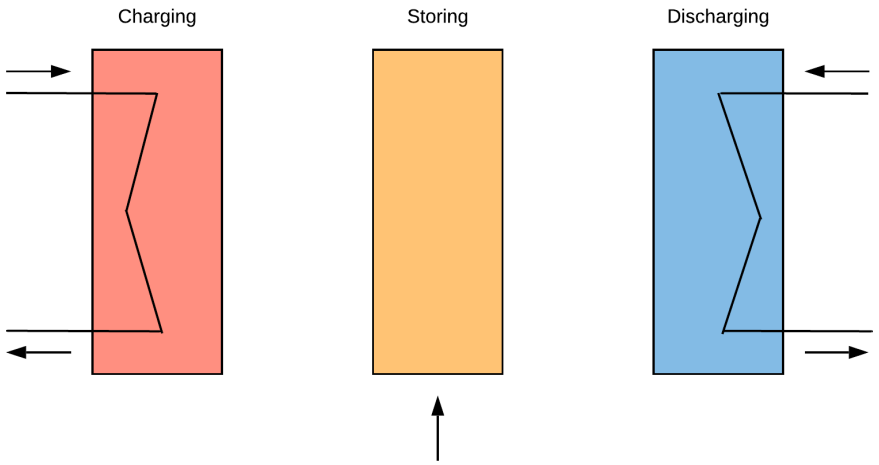


Figure 2.2: Complete storage cycle of a TES system

2.2.2 Basic thermodynamics of energy storage

Thermal energy storage systems can be classified into three storage types by the mechanism employed for the storing: *sensible heat storage*, *latent heat storag*, *thermo-chemical storage*. The concrete definitions and explanations of these three types are given below.

- **Sensible heat storage:** Energy is stored by changing the temperature of the storage material or medium, without changing its phase. In general, water, rocks or ground is used as the storage material in sensible storage system, with water being the cheapest option and has good thermal capacity (ρc_p). The effectiveness of storage material mainly depends on its specific heat. Main advantages of sensible heat storage are cost-effective and without the risks associated with the use of toxic materials according to Sarbu, I. and Sebarchievici, C. (2018). Furthermore, it is simple and has good heat transfer performance. The amount of energy stored in a storage

material can be expressed as:

$$Q = mc_p\Delta T = \rho c_p V \Delta T \tag{2.1}$$

Where, Q is the amount of heat stored in the storage material, m is the mass of storage material, c_p is the specific heat of storage material, ΔT is temperature change, V is the volume of storage material, and ρ is the density of the storage material. Thus, the amount of energy introduced to the storage system is proportional to m, c_p and ΔT. As the most common method for heat and cold storage, a typical sensible TES system consists of a storage material, a storage tank, and inlet–outlet devices. For storage tank, it must has the ability to keep the storage material and prevent the loss of thermal energy to the surrounding. Thermal gradient is also required to transfer heat to and from the sensible TES system.

- **Latent heat storage:** Energy is stored by changing the phase of storage material with no change in temperature. The transition of phase change materials (PCMs) is commonly from solid to liquid or from liquid to vapour, in which process energy is released or absorbed. Compared with sensible heat storage system, the main drawbacks of laten heat storage system is the higher investment costs, and higher risks due to leaks of stability and erosion of material encapsulating PCMs according to Sarbu, I. and Sebarchievici, C. (2018). However, higher energy density than sensible heat storage implies smaller storage size. The amount of energy stored in the storage material can be expressed as:

$$Q = m\Delta h \tag{2.2}$$

Where, Q is the amount of heat stored in the storage material, m is the mass of storage material, Δh is phase change enthalpy, which is also known as melting enthalpy or heat of fusion. The best known and mostly used PCM is water, which has a representative example - cold storage.

- **Thermo-chemical thermal storage:** Reversible endothermic/exothermic reactions are used to store and release thermal energy in thermo-chemical thermal storage process. During the charging process, heat is injected to conduct endothermic chemical reaction. The chemical products can be used to store thermal energy and restore until discharging is required by exothermic reverse reaction. According to M. Orosz and R. Dickes (2017), due to the large change of enthalpy in chemical reactions, thermo-chemical storage has the highest energy density among the other different storage technologies. However, high costs and technical complexity are obstacles to commercial utilization.

Recently, Cabeza, L.F. (2015) classified the storage concept as active and passive storage system. The concept of each storage system is given as below, and the scheme of classification is illustrated in Figure 2.3.

- **Active storage system:** Heat transferred into the storage material by the forced convection heat flow. Active storage systems can be direct system, in which heat transfer fluid (HTF) is also the storage medium, or indirect system, in which storage medium is other medium instead of HTF.

- **Passive storage system:** Dual-medium storage system, where HTF passes through the storage only for charging and discharging a solid material.

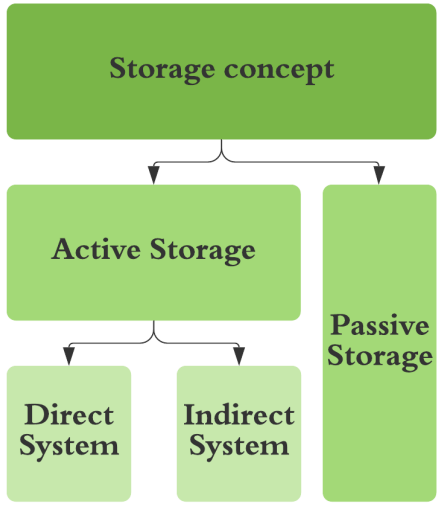


Figure 2.3: Scheme of classification of different storage systems according to the storage concept

2.2.3 Sensible TES system using water storage

The heat storage of sensible TES systems achieved by increasing or reducing the temperature of the storage material, which can be water, concrete, brick, air, etc. According to Mehling, H. and Cabeza, L.F. (2008), each material has its own pros and cons, but the storage material is selected according to its heat capacity and the available space for storage. Hence, in order to achieve high heat storage density per volume, high thermal capacity (ρc_p) is the primary choice for heat storage material selection. Table 2.1 gives specific heat and heat content per volume of some representative heat storage materials.

As shown in Table 2.1, water has the highest specific heat and heat content per volume, which implies that water has the highest heat storage density in both per weight and per volume compared with other typical heat storage materials. Moreover, water is inexpensive, safe, nontoxic, easy available and easy to handle. As a suitable heat storage material, water is also easy to store in the temperature interval from its freezing point to its boiling point ($0 - 100°C$). Thus, water is the most common material used in sensible heat storage system and hot water tank is the best known thermal energy storage technologies.

Nowadays, the water storage tanks are made of steel, stainless steel or concrete or watertight materials which are used to envelope the water volume. During the heat storage process, tank material or water volume envelope material while equipment inside the storage tank will be heated to the same temperature. In order to reduce the heat loss of storage

Material	Specific heat (kJ/(kgK))	Heat content per volume (MJ/(m3K))
Water	4.2	4.2
Oil	2.0	1.7
Ice	2.0	1.8
Paraffin	2.9	2.6
Wood	1.8	0.9
Concrete	0.8	2.1
Brick	0.8	1.2
Glass	0.8	2.2
Steel	0.5	3.6
Magnetite	0.8	4.1
Copper	0.4	3.5
Gold	0.1	2.5

Table 2.1: Heat storage density per volume for different materials (S. Furbo (2015))

tank, insulation materials with low heat conductivity are used to insulate storage tank. The storage performance of hot water tank influenced by these thermal characteristics according to S. Furbo (2015): *heat storage capacity, heat loss, heat exchange capacity rates to the hot water storage, heat exchange capacity rates from the hot water storage and temperature stratification.*

- **Heat storage capacity:** It can determine the heat content of hot water storage when temperature interval is from T_1 to T_2.

$$Heat\ content = heat\ storage\ capacity \times (T_2 - T_1) \tag{2.3}$$

- **Heat loss:** Heat loss have to be small to ensure better performance of energy system with hot water storage. It can be guaranteed by adopting good insulation materials.
- **Heat exchange capacity rates to the hot water storage:** During the charging period, it must be set high to keep the efficiency of energy system heating the heat storage is not decreased fairly owing to the increased temperature of heat transfer fluid.
- **Heat exchange capacity rates from the hot water storage:** During the discharging period, it must also be set high in order to get rid of high heat power sufficiently from the storage tank when high temperature is required in the demand side.
- **Temperature or thermal stratification:** Large thermal stratification in both charging and discharging period can increase the efficiency of energy system with hot water storage. Moreover, it is necessary to maintain temperature stratification during periods without charge and discharge. The temperature stratification of water is the result of water temperature difference which causes the layers that act as barriers to prevent water mixing. Warm water moves to the top of the tank due to its low density, and cold water vice versa. In order to improve the temperature stratification,

water tank can be designed as thin and tall. Moreover, inlet and outlet sites can be installed in a manner which shown in Figure 2.4 to avoid mixing by the uniform flows.

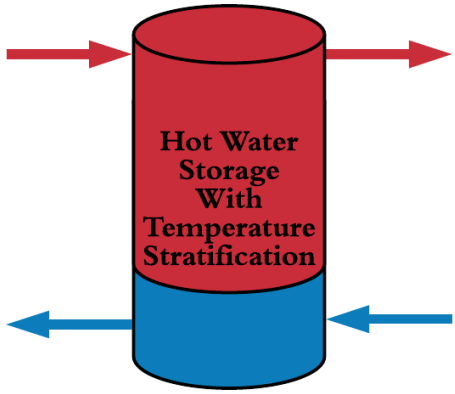


Figure 2.4: Stratified hot water tank

Hence, the efficiency of hot water tank can be further improved by guaranteeing optimal tank thermal stratification and more effective insulation tank material. These abovementioned thermal characteristics are highly influenced by the material properties like density, kinematic viscosity, thermal conductivity and specific heat of water.

- **Water density:** It is decreased as the water temperature increases. As a result, hot water with low density rise upwards and cold water with high density move downwards, which is strong thermal stratification with large temperature difference in storage tank.
- **Water kinematic viscosity:** It is decreased as the water temperature increases. This leads to easy water movement in the storage tank.
- **Water thermal conductivity:** It is increased as the water temperature increases. Thus, the temperature differences will be equalized more quickly than the temperature is low in storage tank.
- **Water specific heat:** In the temperature interval from 0 $^{\circ}C$ to 100$^{\circ}C$, specific heat capacity of water is nearly constant.

TES has a number of different technologies, which with their own specific performances, applications, and costs. Since, sensible heat storage using hot water tank is rather inexpensive due to its simple tank structure, simple equipment to charge and discharge and available cheap storage medium, in this thesis it is adopted as the storage method for a simple thermal energy storage system.

2.3 Heat transfer modeling in heat exchangers

2.3.1 Brief introduction of heat exchanger

Heat exchanger is a device built for efficient heat transfer from one medium to another, whether the media are separated by a wall so that they never mix, or they are in direct contact (Maurice I. Stewart, Jr. (2014)). Heat exchanger has wide specific applications from upstream to downstream and from space heating to chemical processing.

The categorization of heat exchangers can be taken into two approaches. The first is to consider the flow configuration within the heat exchanger, while the second is classified by the equipment constructions. Since, the heat exchanger construction design is not the main topic and study of this thesis, so only the flow configurations will be considered and discussed. There are four basic flow configurations. The graphical illustrition of these four flow configurations are shown in Figure 2.5.

Classification of heat exchangers by flow configuration:

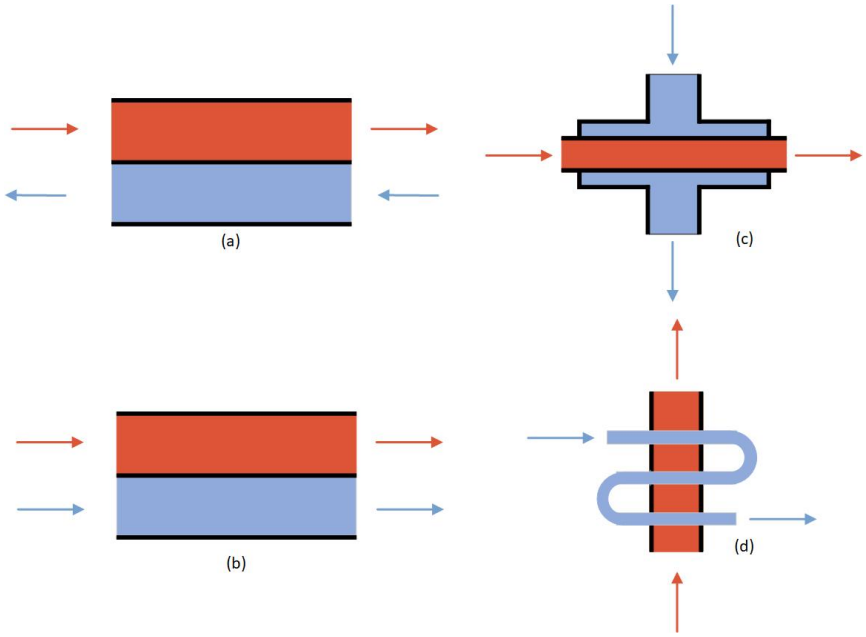


Figure 2.5: Heat exchanger flow configurations (or arrangements): (a) counter (b) parallel (c) cross (d) hybrid: cross counter

- **Counter Flow:** In counter-flow heat exchanger, two fluids flow parallel to each other but enter at opposite ends, flow in opposite directions and leave at opposite ends. This type of flow arrangement allows the largest change in temperature of both fluids and is therefore most efficient.
- **Parallel flow:** In parallel-flow or co-current flow heat exchanger, two fluids flow

parallel to each other and enter at the same ends, flow in the same directions and leave at the same ends. This is less efficient than counter-current (counter) flow but does provide more uniform wall temperatures.

- **Cross flow:** In cross-flow heat exchanger, two fluids flow perpendicular to each other.
- **Hybrids flow:** In industrial heat exchangers, the hybrid of above mentioned flow types are often used, such as cross-counter flow and multi-pass flow.

The design of heat exchanger is not the main topic of this thesis, but it is necessary to understand which type of the simplest heat exchanger, like counter- and co-current heat exchanger, is good enough in the practical applications. The temperature profiles, Figure 2.6 indicate the main disadvantages and advantages two types of the simplest heat exchangers.

Parallel-flow heat exchanger:

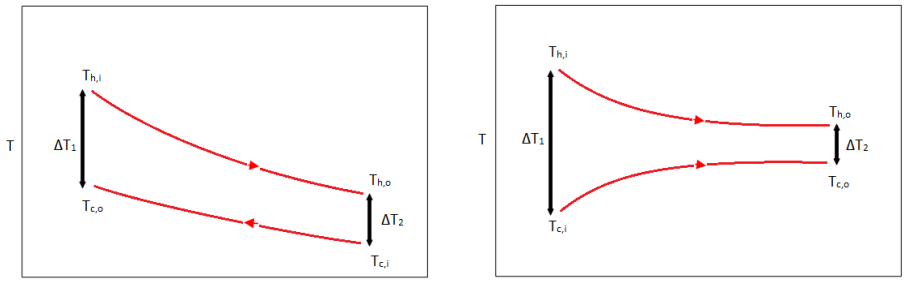


Figure 2.6: Simple temperature profiles of heat exchangers

- Disadvantages: The large temperature difference at the end, such as ΔT_1, can cause large thermal stress, which is a stress created by any change in temperature, like temperature gradients, thermal expansion or contraction and thermal shocks, to a material. Eventually, it leads to material failure, such as fracture and cracking etc. Then if the purpose is to raise the temperature of the cold fluid, the distinct disadvantage is the exit temperature of cold fluid can never exceed the lowest temperature of hot fluid.
- Advantages: If it is required to get nearly the same temperatures at the exit, parallel-flow heat exchanger is advantageous.

Counter-flow heat exchanger:

- Advantages: The temperature differences between two fluids are more uniform, it can reduce the thermal stresses and also can lead to more uniform heat transfer

rate along the heat exchanger. The exit temperature of cold fluid can exceed the temperature of hot fluid, and even can approach the inlet temperature of hot fluid.

Since the one of the purpose of this thesis is to increase the temperature of thermal storage tank and store as much as thermal energy in it, and then discharge stored energy to meet the requirement of the demand. Therefore, according to the discussed pros and cons, counter-flow heat exchanger will be suitable to achieve this goal.

2.3.2 Heat transfer mechanisms

Due to the temperature difference between the fluids flowing oppositely with each other inside the heat exchanger, sensible heat is successfully transferred from the hot side to the cold side. Heat is mainly transferred inside the heat exchanger by one or several of the following mechanisms (Maurice I. Stewart, Jr. (2014)).

- **Conduction:** The transfer of heat from one molecule to an adjacent molecule while the particles remain in fixed positions relative to each other. It is the principal heat transfer mechanism in solids and slowly flowing or even stagnant fluids. The formulation can be expressed as:

$$q = k\frac{A}{L}\Delta T \tag{2.4}$$

- **Convection:** The transfer of heat by the physical movement of molecules from place to place. It is the principal heat transfer mechanism in fluids. The formulation can be expressed as:

$$q = hA\Delta T \tag{2.5}$$

- **Radiation:** The heat waves emitted from a hot body are absorbed, reflected, or transmitted through a colder body. For fluid-fluid heat exchangers, the temperatures are not hot enough to let radiation become a main mechanism. The formulation can be expressed as:

$$q = \sigma A\Delta T^4 \tag{2.6}$$

Where, q is heat transfer rate or heat flow rate, A is heat transfer area or flow rate area, ΔT is temperature difference, k is thermal conductivity, h is film coefficient or heat transfer coefficient, L is heat conducted distance, σ is Stefan-Boltzmann constant.

If heat is being transferred through different layers from hot side to cold side in heat exchanger, two following conclusions can be obtained according to Maurice I. Stewart, Jr. (2014):

- Heat transfer rate (q) is equal through all layers.

$$q_1 = q_2 = q_3 = ... = q_n \tag{2.7}$$

- Heat transfer rate equals temperature difference divided by total thermal resistance.

$$q = \frac{\Delta T}{\sum R} \tag{2.8}$$

  Where, R is the thermal resistance of each layer. For conduction and convection it can be expressed as follows respectively.

$$R = \frac{L}{kA} \tag{2.9a}$$

$$R = \frac{1}{hA} \tag{2.9b}$$

On the whole, heat transfer mechanisms like conduction or convention or the combination of these two are mostly applied in the facilities. Therefore, the heat transfer completed in heat exchangers can be commonly seen as the result of these three procedures:

- **Convective procedure:**
  - Heat transferred from hot fluid to the tube of heat exchanger.
  - Heat transferred from heat exchanger tube to the cold fluid.
- **Conductive procedure:**
  - Heat transferred through the wall of tube from hot side to cold.

2.3.3 Overall heat transfer coefficient, U

According to above-mentioned introductions, it is easy to know that the driving force which leads heat transfer between two fluids inside the heat exchanger is the temperature difference. If define $\sum R$ in equation (2.8) as $\frac{1}{UA}$, this can result into:

$$q = UA\Delta T \tag{2.10}$$

Where, q is the total heat transfer rate between the hot and cold fluids, U is overall heat transfer coefficient, A is heat transfer area or flow rate area, ΔT is temperature difference between the hot and cold fluids, which can be expressed as:

$$\Delta T = T_h - T_c \tag{2.11}$$

Where, the subscripts h and c represent hot fluid and cold fluid respectively. Thus, according to redefinition of $\sum R$, the definition of overall heat transfer coefficient can be expressed in formula form as:

$$U = \frac{1}{A\sum R} \tag{2.12}$$

Therefore, overall heat transfer coefficient, U, for a heat exchanger is the sum of thermal resistances per unit area. The determination of overall heat transfer coefficient is important and also uncertain for heat exchanger. Since, the derivation of specific *logarithmic mean temperature difference (LMTD)*, which is discussed in the next subsection, uses constant U assumption, so the quantity of U in this thesis is seen as constant throughout the heat exchanger.

2.3.4 Logarithmic mean temperature difference – LMTD

As mentioned above, equation (2.10) is the fundamental formulation used in heat transfer calculations. However, the temperature of process fluids will change as it flows through the heat exchanger. Hence, the temperature difference between the hot and cold fluids, ΔT, will also continuously vary with location, especially when the flow pattern is in counter-flow arrangement. So, it is necessary to work with *mean temperature difference (MTD)* in total heat transfer rate equation as a form like:

$$q = UA\Delta T_m \tag{2.13}$$

Where, ΔT_m is *appropriate mean temperature difference*. Hence, equation (2.13) can be used to perform a heat exchanger analysis. However, before directly applying equation (2.13), the specific form of ΔT_m must be established and determined.

According to Bergman,T.L et al.(2011), the specific form of ΔT_m can be found by applying an energy balance to differential elements in the hot and cold fluids. Those elements are length, dx, and heat transfer area, dA, respectively. After applying necessary assumptions and derivation steps, the determined form of appropriate mean temperature difference is *logarithmic mean temperature difference (LMTD)*. The assumptions used in the derivation of LMTD are listed below:

- The overall heat transfer coefficient, U, is constant throughout heat exchanger.
- Heat exchanger is insulated from the surrounding. Hence, the only heat transfer approach is from hot fluid to cold.
- Axial direction or x-coordinate in Figure 2.5 is convection dominated, so heat transferred by conduction through the tube wall can be neglected.
- Potential and kinetic energy changes are negligible.
- Specific heat capacities of fluids are constant.

Hence the concrete form of equation (2.13) follows that:

$$q = UA\Delta T_{lm} \tag{2.14}$$

Where,

$$\Delta T_{lm} = \frac{\Delta T_2 - \Delta T_1}{\ln(\Delta T_2/\Delta T_1)} = \frac{\Delta T_1 - \Delta T_2}{\ln(\Delta T_1/\Delta T_2)} \tag{2.15}$$

Where, endpoint temperature difference ΔT_1 is the largest terminal temperature difference and ΔT_2 is smallest terminal temperature difference. Equation (2.15) works for both parallel-flow and counter-flow heat exchangers. Due to the hot and cold fluid temperature distributions associated with parallel-flow and counter-flow heat exchangers are different, so the endpoint temperature differences, ΔT_1 and ΔT_2, are also not alike. Hence, it leads to two diverse sub-forms of ΔT_{lm} in equation (2.15). The graphic illustrations of both two

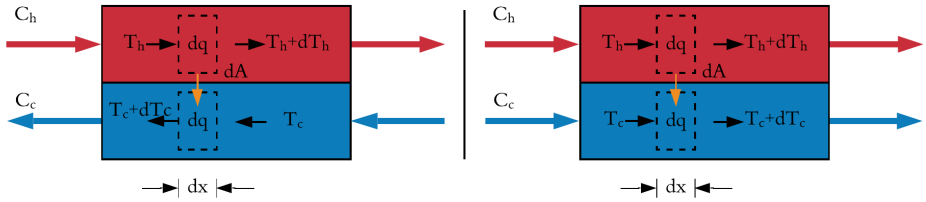


Figure 2.7: Counter-flow and parallel-flow heat exchangers

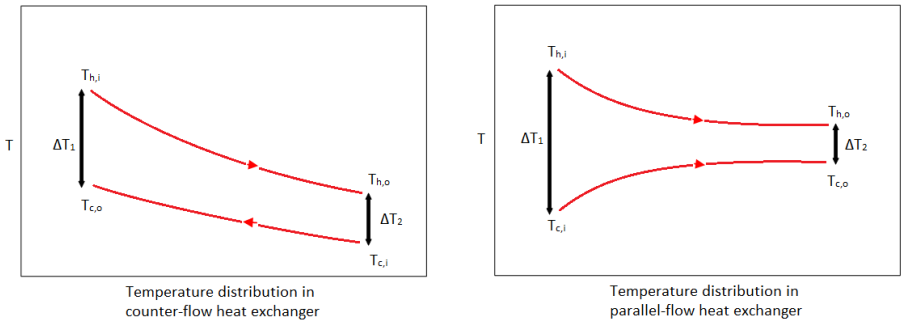


Figure 2.8: Counter-flow and parallel-flow heat exchangers: temperature distributions

flow patterns and temperature distributions are shown in Figure 2.7 and Figure 2.8.

Therefore, according to Figure 2.8, the endpoint temperature differences, ΔT_1 and ΔT_2 for both two flow patterns can be expressed as follows. Also have to note that, the outlet temperature of cold fluid, $T_{c,o}$ can exceed the outlet temperature of hot fluid, $T_{h,o}$ in counter-flow arrangement, but it can not appear for parallel-flow arrangement.

Counter-flows:

$$\Delta T_1 = T_{h,i} - T_{c,o} \tag{2.16a}$$
$$\Delta T_2 = T_{h,o} - T_{c,i} \tag{2.16b}$$

Parallel-flows:

$$\Delta T_1 = T_{h,i} - T_{c,i} \tag{2.17a}$$
$$\Delta T_2 = T_{h,o} - T_{c,o} \tag{2.17b}$$

The larger the LMTD, the more heat is transferred. In addition, when inlet and outlet temperatures are the same, the LMTD for counter-flow is larger than that for parallel-flow. Thus, when the same rate of heat is transferred with the same value of overall heat transfer coefficient, U, the requirement of heat transfer area A for counter-flow is smaller than that for parallel-flow (Bergman,T.L et al.(2011)). It means that, with the constant U, in order

to achieve more heat transfer, the optimal operation is to take counter-flow arrangement to use less heat transfer area, A, to further decrease the construction cost of heat exchanger.

According to Maurice I. Stewart, Jr. (2014), the relation between MTD and LMTD is:

$$MTD = F(LMTD) \tag{2.18}$$

Where, F is the correction factor for heat exchanger geometry. For example, for pipe-in pipe and counter-flow heat exchanger, $F = 1.0$, which means for these two types of heat exchangers MTD equals the calculated LMTD.

2.3.5 Approximation of LMTD

The logarithmic mean temperature difference (LMTD) has significant meaning to the both theoretical and practical aspects of the heat exchangers from design to modelling and control. However, according to Zavala-Río et al.(2005), LMTD has caused inconveniences in several applications such as chemcial engineering programs.

W.R.Paterson (1984) pointed out that in practice, adopting starting values as the equality of stream temperatures for iterative equation solving procedures, can reduce the definition of the logarithmic mean temperature difference into *indeterminate form of the logarithmic mean*. Furthermore, the problems of inconveniences are caused this indeterminate form which is given in equation (2.19).

$$\Delta T_{lm} = \Delta T_1 = \Delta T_2 \tag{2.19}$$

However, the derivatives of ΔT_{lm} w.r.t ΔT_1 and ΔT_2, which are needed in the Newton iterative solution of the equations, has value at the limit when ΔT_1=ΔT_2, but they are not defined at that limit. Hence, it means that the practical applications of heat exchangers which using the LMTD as driving force (fluid mean temperature difference) may suffer from inconveniences. Moreover, indeterminate form is a result of the incomplete model derivation. As a well-defined replacement expression, a new mean has been developed as an approximation to the logarithmic mean by W.R.Paterson (1984), in order to overcome the difficulties caused by indeterminate form of the logarithmic mean, which is shown in equation (2.20).

$$\Delta T_{nm} = \frac{2}{3}\Delta T_{gm} + \frac{1}{3}\Delta T_{am} \approx \Delta T_{lm} \tag{2.20}$$

Where, ΔT_{gm} is geometric mean, and ΔT_{am} is arithmetic mean. And sometimes they are preferred to be used to approximate the mean temperature difference along the heat exchanger.

$$\Delta T_{gm} = \sqrt{\Delta T_1 \Delta T_2} \tag{2.21a}$$

$$\Delta T_{am} = \frac{\Delta T_1 + \Delta T_2}{2} \tag{2.21b}$$

By approximating ΔT_{lm} over an acceptable range of ΔT_1 and ΔT_2, Underwood, A.J.V (1970) and Chen J. (1987) obtained modifications of the approximation with well-defined

replacement expressions.

Underwood approximation:

$$\Delta T_{um}^{\frac{1}{3}} = \frac{1}{2}(\Delta T_1^{\frac{1}{3}} + \Delta T_2^{\frac{1}{3}}) \tag{2.22}$$

Chen approximation:

$$\Delta T_{cm}^{0.3275} = \frac{1}{2}(\Delta T_1^{0.3275} + \Delta T_2^{0.3275}) \tag{2.23}$$

As shown in equation (2.22) and (2.23), these two new means have the same form and are a polynomial of endpoint temperature differences of heat exchanger, ΔT_1 and ΔT_2. According to Chen J. (1987), Underwood approximation gives not only a very simple solution, but also a superior results comparing with results of original LMTD. Moreover, as a slight modification of Underwood approximation, Chen approximation is almost exactly the same as the results of original LMTD. Hence, these two approximated means are both suitable to be applied in this thesis.

For the sake of simplicity, the general form of equation (2.22) and (2.23) is expressed as:

$$\Delta T_m^n = \frac{1}{2}(\Delta T_1^n + \Delta T_2^n) \tag{2.24a}$$

$$\Delta T_m = \left[\frac{1}{2}(\Delta T_1^n + \Delta T_2^n)\right]^{\frac{1}{n}} \tag{2.24b}$$

The values of n for different types of approximations are summarised in Table 2.2.

n	**Mean approximation**	ΔT_m
1	Arithmetic mean	ΔT_{am}
$1/3$	Underwood mean	ΔT_{um}
0.3275	Chen mean	ΔT_{cm}

Table 2.2: Values of n for different types of approximations

Chapter 3

Optimal control problem and nonlinear model predictive control

The main objective of this chapter is to introduce the basic concept of optimal control problem (OCP), nonlinear model predictive control (NMPC) and the numerical method which is used to solve OCP. For NMPC, its standard and robust form are both introduced to meet the problems of this thesis.

3.1 Introduction to optimization problem

In chemical engineering applications, especially in the process systems engineering, optimization has an extensive utilization. Optimization is an important tool in decision science and in the analysis of physical systems (Nocedal,J. and Wright,S.J.(2006)). In order to take advantage of this tool, an objective must be identified which is the performance reflection of the system under certain studies. For example, this objective could be the profit of a company which needs to be maximized or energy consumption of industry which needs to be minimized. The goal of optimization problem is to find out the values of the variables or unknowns to maximize or minimize objective, where the objective is determined by variables or unknowns which are known as the system characteristics. Regularly, these variables or unknowns are constrained to some certain bounds. Thus, a typical optimization problem involves three main elements: *objective function*, *decision variables* and *constraints*. The concrete explanation of each element is given in the list below.

- **Objective function:** Scalar function which is used to describe the function's property that needed to be optimized. Optimization can be minimization or maximization problem.

- **Decision variables:** Can be real numbers, integers or binary variables, or function spaces like a vector whose values can be changed until find an optimal solution x*.

- **Constraints:** Bounds on functions of the decision variables, which means decision variables must satisfy, and can define which solutions are feasible after optimization. Constraints normally have two types, which are equality constraints and inequality constraints respectively.

The identification process of these three elements of optimization problem is known as *modeling*. It is important to note that the preciseness of model is quite significant to the optimization problem.

3.1.1 Mathematical formulation

According to the above-mentioned introduction, *optimization is the minimization or maximization process of the objective function subject to constraints on its variables*. After converting these descriptions into mathematical formulation, the optimization problem can be written as:

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & f(x) \\ \text{s.t.} \quad & g_i(x) = 0, \quad i \in \mathcal{E} \\ & h_j(x) \leq 0, \quad j \in \mathcal{I} \\ & x_{min} \leq x \leq x_{max} \end{aligned} \tag{3.1}$$

Where, $f(x)$ is objective function and optimization problem above is stated as minimization problem. x are decision variables which are constrained by lower bounds x_{min} and upper bounds x_{max}. $g_i(x)$ is the set of equality constraints, $h_i(x)$ is the set of inequality constraints, $\mathcal{E}$ and $\mathcal{I}$ are sets of indices for equality and inequality constraints, respectively. Note that $\mathcal{E}$ and $\mathcal{I}$ are disjoined and distinct from each other, ie. $\mathcal{E} \bigcap \mathcal{I} = \emptyset$. Furthermore, $\mathcal{E}$ and $\mathcal{I}$ can also determine the *feasible set* or *feasible region*, which is the subset of $\mathbb{R}^n$.

Once the model of optimization problem has been formulated, a specific optimization algorithm have to be used to solve this problem and find out the solutions. After getting the solutions, it is necessary for operator to check whether this set of variables is the appropriate solution or not. This mathematical expressions are known as optimality conditions, where there is a condition namely *Karush- Kuhn-Tucker (KKT) conditions*, which are used to check the first order necessary conditions of optimization problem are satisfied or not. The detailed discussions can be found from Nocedal,J. and Wright,S.J.(2006) and Foss,B. and Heirung,T.A.N.(2013). What should the operator do if the optimality conditions are not satisfied? If the satisfaction of optimality conditions are failed, techniques like *sensitivity analysis* can be applied to improve the model.

The classification of optimization problem depends on properties like linearity and convexity of the objective function and constraints, and also the size of the decision variables in the optimization problem. According to Biegler L.T.(2010), *nonlinear programming problem (NLP)* plays an important role in process system engineering applications. Thus, NLP is briefly introduced in the next subsection and the definitions of rest of the problems are available in Nocedal,J. and Wright,S.J.(2006).

3.1.2 Nonlinear programming problem

Nonlinear programming problem (NLP) is an optimization problem where the objective function or the constraints are defined by nonlinear functions. The general form of NLP is the variant of optimization problem (3.1), where functions are just nonlinear functions. NLP can be solved by active set SQP (sequential quadratic programming) methods or the interior-point methods, which are the most prospective methods. The specific details of SQP method can be found in Nocedal,J. and Wright,S.J.(2006), and it is not involved in this thesis. The method implemented in this thesis is focused on interior-point methods, which solve NLP problem by employing Newtons method to a sequence of modified KKT conditions. In addition, the objective of interior-point method is to approximately formulate the inequality constrained problem as an equality constrained problem where Newtons method can be used. Interior-point methods can be implemented by several software packages, in which IPOPT is applied in this thesis to solve NLP problem (A. Wächter and L.T.Biegler (2006)).

3.2 Optimal control problem

Optimal control problem (OCP) is also known as dynamic optimization problem. The purpose of OCP is to optimize an objective function with respect to some constraints, in which dynamic system model is included, by finding out optimal control inputs u. Thus, the dynamic system can be optimally controlled by a suitable choice of control inputs u and the process of finding the optimal control inputs requires numerical methods which will be discussed in following sections. However, it is necessary to briefly introduce what is dynamic system in the next subsection.

3.2.1 Dynamic systems and optimization

As a time dependent system, the processes of dynamic system are evolving in time and it can be characterized by the states x that allow operators to forecast system's future behavior. Dynamic system and its mathematical model have numerous variants, and two of the important classes are continuous time and discrete time systems, which are introduced below. The rest types of dynamic systems are available in Moritz Diehl and Sébastien Gros (2017).

The main characteristic of dynamic system is time-evolving, but the time has two different variants, which finally lead to the difference between continuous and discrete time system.

- **Continuous time systems:** The time is physical time, which are an interval of real numbers, $[0, T]$. Time and the states can be expressed as $t \in \mathbb{R}$ and $x(t)$ respectively. The system is described in a form of differential equations, $\frac{dx_n}{dt} = f(x_1...x_n)$. It can be transformed into discrete time systems. The work in this thesis is exclusively concerned with continuous time systems.

- **Discrete time systems:** The time only takes values on a predefined time grid, which are assumed as integers. The states can be expressed as x_k, which means state at

time point k. After the time variable is discretised, the differential system equations are replaced by difference equations, which can be expressed as $\frac{\Delta x}{\Delta t}$.

There are many different ways to model dynamic systems. For example, they can be modeled by ordinary differential equations (ODEs) or differential algebraic equations (DAEs), which are used to describe the evolution of finite dimensional continuous-state-spaces systems in continuous time. By contrast, infinite dimensional continuous-state-spaces systems in continuous time is described by partial differential equations (PDE).

The optimization of dynamic systems can also be performed by different optimization methods like quasi dynamic optimization and dynamic optimization.

- **Quasi dynamic optimization:** Reoptimize frequently on a static model, when the system is slowly varying or is mostly under steady state, to optimize a dynamic system.

- **Dynamic optimization:** Optimize on a dynamic model when time varying dynamics play a major role.

The optimization section in this thesis will be focused on dynamic optimization or optimal control problem (OCP). Hence, the general form of dynamic optimization problem can be written as follows:

$$\begin{aligned} \min_{x,z,u} \quad & \phi(x(t_f)) \\ \text{s.t.} \quad & \dot{\mathbf{x}} = f(x(t), z(t), u(t), t, p) \\ & 0 = g(x(t), z(t), u(t), t, p) \\ & x_l \leq x(t) \leq x_u \\ & z_l \leq z(t) \leq z_u \\ & u_l \leq u(t) \leq u_u \\ & p_l \leq p(t) \leq p_u \\ & x_{t_0} = x_0 \end{aligned} \tag{3.2}$$

Where, x, z, u represent differential variables, algebraic variables, control inputs respectively, which are going to be optimized. x_0 is the given initial state value, t is time, t_f is final time, p is time independent parameters. f and g are differential equations and algebraic equations, respectively, which are additional set of equality constraints and represent the phenomena arises in the system. The remaining inequality constraints are variable bounds. As mentioned in the preceding subsection, the dynamic optimization problem (3.2) becomes NLP problem when the objective function or constraint functions are nonlinear. The objective of this dynamic optimization problem is to minimize objective $\phi()$ in a specified time interval $[t_0, t_f]$.

The formulating process of nonlinear optimal control problem (3.2) is straightforward, however, the difficulty is concentrated in problem solving. For example, analytic solution of linear optimal control problem can be computed by solving Riccati differential equation. But NLP generally do not have analytic solution. Thus, it is essential to seek after numerical methods used for solving NLP to find out the numerical solutions.

3.2.2 Numerical methods for solving dynamic optimization problem

The exact (analytic) solution of dynamic optimization problem is difficult to find out because this type of problem contains the dynamics of the system , which is usually described by Ordinary Differential Equations (ODEs) or Differential Algebraic Equations (DAEs), and the computation of control inputs at each sampling time within a given time interval and then finally leads to the solution of an infinite-dimensional optimization problem. Therefore it is necessary to enforce numerical methods to approximate the solution of the problem. There are two realms of numerical methods which are used to solve dynamic optimization problems: *indirect method* and *direct method*.

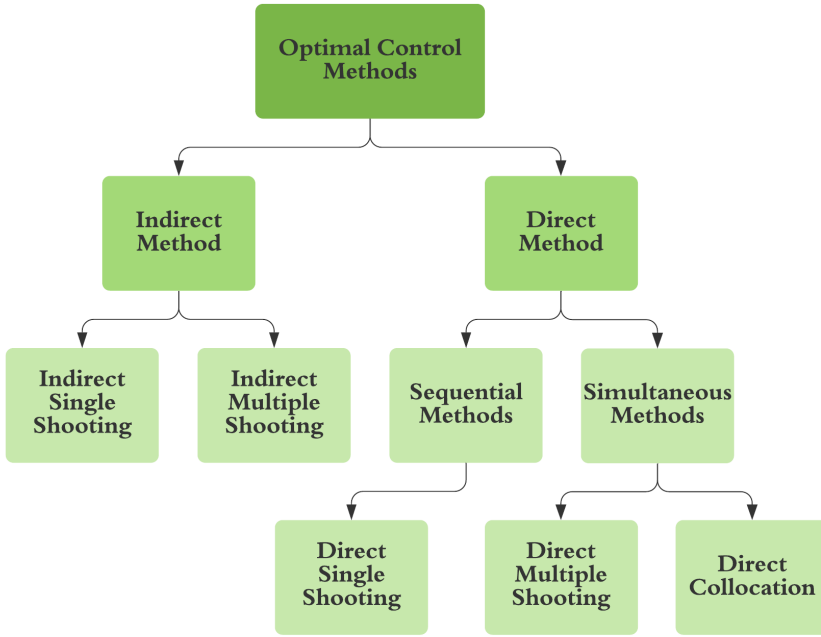


Figure 3.1: Numerical methods for solving dynamic optimization problem

As illustrated in Figure 3.1, the relations between the methods are easy to be known. Where,

1. **Indirect method or variational approach** obtains a solution numerically by formulating the optimality conditions - the first order necessary condition, that are obtained from Pontryagin's Maximum Principle (Pontryagin VG et al. (1962)) and then discretize. Indirect method can be divided into two main groups: *indirect single shooting* and *indirect multiple shooting*. As an optimize-then-discretize approach, the optimality conditions of OCP in indirect method are considered before the control trajectory is parameterized.

2. **Direct method or direct optimal control method** can convert a continuous optimal control problem into a discrete nonlinear programming problem (NLP) by discretizing the system dynamics and the control inputs and then NLP will be solved numerically. The reason of discretising continuous dynamic is integrals or differential equations cannot be directly processed by NLP solver. Direct method can be generally divided into two categories: *sequential approach* or *sequential NLP strategy* and *simultaneous approach* or *simultaneous NLP strategy*. After applying one of the direct methods, dynamic optimization problem can be directly solved by NLP solver in terms of control inputs and state variables. As a discretize-then-optimize approach, the control trajectory is parameterized, which is the converse approach of indirect method.

3.2.3 Sequential approach (direct single shooting)

- **Discretization:** Only the control inputs u are discretized. Hence, only the control inputs are decision variables for the optimizer, $w = [u_0, u_1, u_2 ... u_{N-1}]^\top$. After the discretization, control inputs are denoted by piece-wise constants or piece-wise polynomials.

- **Process:** Give initial guesses of control inputs and a set of control parameters or get these by implementing real-time optimization. Then the system is integrated by DAE solver (integrator) at each iteration to get states, $x_t = f(w, x_0, t)$. In optimizer (NLP solver), system's DAE models are replaced by its gradient information with respect to the control inputs. Note that, gradient information contains either the direct or adjoint sensitivity equations. After evaluating the error in the variable's boundary conditions, optimizer (NLP solver) is executed to get a new guess of control inputs. The iteration of solving NLP for a control trajectory and solving DEA with new guess from optimization is kept until the boundary condition is satisfied. The graphic illustration is shown in Figure 3.2.

- Large integration time step t causes high non-linearity. It can be avoided by implementing quite small t. When the problem is only stable or linear, the direct single shooting approach becomes robust. t is also the check point, and constraints on states are enforced at check point t.

- The construction of sequential method is relatively easy and straightforward. However, if the solved problem has large-scale system, the implementation of sequential method leads to repeated numerical integration with the DAE solver and this process is time-consuming. In addition, it has also been reported to have difficulties of handling stiff or unstable systems (J.T.Betts (2010)).

3.2.4 Simultaneous approach

- **Discretization:** Complete discretization of the states and control inputs. Hence, decision variables for optimizer are $w = [u_0, u_1, u_2, ..., u_{N-1}, x_0, x_1, x_2, ..., x_N]^\top$

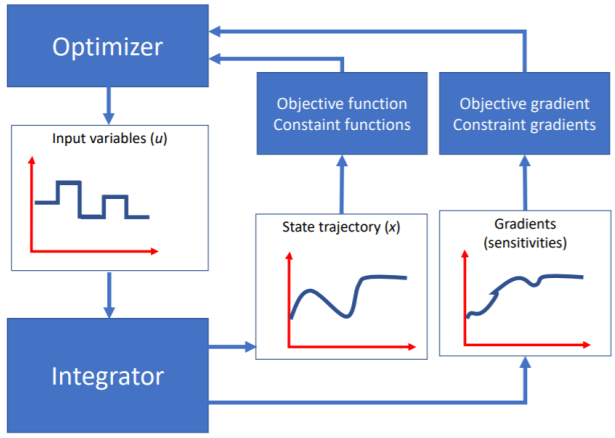


Figure 3.2: Sequential approach process (Johannes Jäschke (2019))

- Compared with direct single shooting method, the computationally expensive numerical integration of the differential DEAs can be avoided according to O.von Stryk (1992).
- **Two main subapproaches:** Direct multiple shooting and direct collocation.

Direct multiple shooting:

- To reduce infinite dimensional optimization problem into a finite dimensional optimization problem by partitioning the whole time domain $[t_0, t_f]$ into finite time elements, $[t_0, t_1], [t_1, t_2], ..., [t_{N-1}, t_f]$. Thus, shooting is no more on the whole time domain.
- Integrate DAEs model at each time interval $[t_k, t_{k+1}]$ to obtain states, $x_k = f(x_k, u_k)$.
- In order to ensure the continuity of states between two consecutive time intervals along the integration process, shooting gap $x_{k+1} - f(x_k, u_k) = 0 \, \forall \, k \in 1, 2, ..., N)$ is given as equality constraints to the NLP.
- Can handle unstable and nonlinear optimization problems. Thus, the numerical unstability of single shooting method can be reduced by implementing direct multiple shooting method.
- Finite time element index k is also the sample time, and constraints on states are enforced at sample times.

Direct collocation:

- Also known as *direct transcription*. The complete discretization is implemented by adopting collocation on finite elements which leading to large-scale but sparse NLP.

- Unlike the shooting methods, the integration of direct collocation is done inside the optimizer, so that system's DAEs model is solved as constraint in the NLP. Therefore, direct collocation is a fully simultaneous approach, which means the integration of DAEs and the dynamic optimization problem are performed together in the NLP solver.
- Control inputs $u(t)$ is piecewise-constant, which means that $u(t)$ is constant in each collocation time interval $[t_k, t_{k+1}]$.

The concrete introduction of direct collocation method is given in the next subsection.

3.2.5 Direct collocation method

In this thesis, the numerical method for solving dynamic optimization problem is focused on direct collocation method, where transcription and collocation are used interchangeably.

Introduction

1. As a variant of Runge–Kutta methods (RK methods), the solution of the differential state $x(t)$ is approximated by using a K^{th} order polynomial $p(\theta_k, t)$ on the collocation time interval $[t_k, t_{k+1}] \subseteq [t_0, t_f]$.
2. The interpolation polynomials $p(\theta_k, t)$, which is the parametrized state trajectory used in orthogonal collocation, is usually constructed as *Lagrange polynomials*. Within the each collocation time interval $[t_k, t_{k+1}]$, K number of interpolation points can be selected, and the differential states x_t is approximated by using Lagrange interpolation polynomials.

$$p(\theta_k, t) = \sum_{i=0}^{K} \theta_{k,i} L_{k,i}(t) \tag{3.3}$$

   Where, $L_{k,i}(t)$ are Lagrange interpolation polynomials and which is expressed as:

$$L_{k,i}(t) = \prod_{j=0, j\neq i}^{K} \frac{t - t_{k,j}}{t_{k,i} - t_{k,j}} \in \mathbb{R} \tag{3.4}$$

   Where, θ is polynomial coefficient, k implies time step , j implies collocation points, i is collocation time index and K is the order of interpolation polynomial and it determines the number of collocation points taken inside the time interval.

3. One of the property of $L_{k,i}(t)$ can be observed from the Figure 3.3, which is:

$$L_{k,i}(t_{k,j}) = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases}$$

   Furthermore, it indicates the interpolation polynomial $p(\theta_k, t)$ passes through the

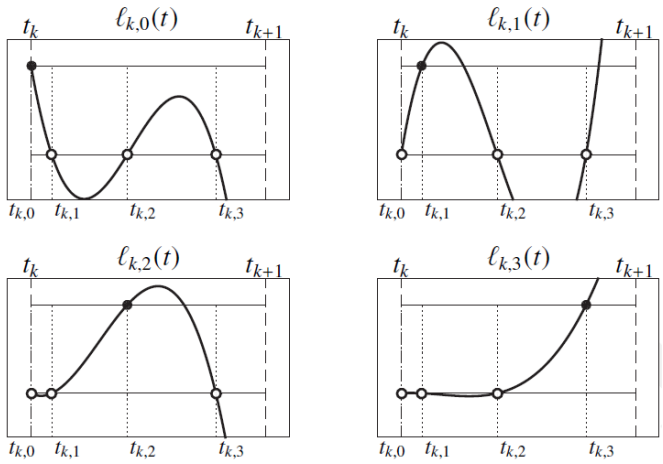


Figure 3.3: Lagrange polynomial $L_{k,i}(t)$ on the interval $[t_k, t_{k+1}]$ (Moritz Diehl and Sébastien Gros (2017))

interpolation points $\theta_{k,j}$.

$$p(\theta_k, t_{k,j}) = \theta_{k,j} \tag{3.5}$$

An additional property of Lagrange polynomial $L_{k,i}(t)$ is orthogonal.

$$\int_{t_k}^{t_{k+1}} L_{k,i}(t) L_{k,j}(t)\, \mathrm{d}t = 0, \ i \neq j \tag{3.6}$$

4. The integration of the system dynamics ODE $\dot{\mathbf{x}} = \mathbf{F}(\mathbf{x}, t) = 0$ over the collocation time interval $[t_k, t_{k+1}]$ can be performed by solving the *collocation equations*:

$$c(\theta_k, t_{k,i}, x_k) = \begin{bmatrix} \theta_{k,0} - x_k \\ \dot{p}_k(t_{k,1}, \theta_k) - \mathbf{F}(\theta_{k,1}, t_{k,i}) \\ \cdot \\ \cdot \\ \cdot \\ \dot{p}_k(t_{k,K}, \theta_k) - \mathbf{F}(\theta_{k,K}, t_{k,i}) \end{bmatrix} = 0 \tag{3.7}$$

If the collocation equations (3.7) are satisfied, the interpolation polynomial $p(\theta_k, t)$ can accurately capture the the state trajectory over the collocation time interval. And the end state $x(t_{k+1})$ can be accurately approximated by $p(\theta_k, t_{k+1})$.

5. In order to satisfy $c(\theta_k, t_{k,i}, x_k) = 0$, two constraints for collocation equations also have to be satisfied. First of all, at collocation time $t_k = t_{k,0}$, the interpolation polynomial $p(\theta_k, t_k) = x_k$, where x_k is the given initial value of the state at time t_k. Then according to the property in equation (3.5), this relation can be simplified as

$\theta_{k,0} = x_k$. Secondly, $p(\theta_k, t_{k,i})$ have to meet the system dynamic on the remaining collocation times from $t_{k,1}$ to $t_{k,K}$.

$$\dot{p}(\theta_k, t_{k,i}) = \mathbf{F}(p(\theta_k, t_{k,i}), \mathbf{u_k} t_{k,i}) \tag{3.8}$$

Where, $p(\theta_k, t_{k,i})$ in equation (3.8) can be expressed as $\theta_{k,i}$ according to equation (3.5). Thus, equation (3.8) can be simplified as:

$$\dot{p}(\theta_k, t_{k,i}) = \mathbf{F}(\theta_{k,i}, \mathbf{u_k}, t_{k,i}) \tag{3.9}$$

Equation (3.9) is the system dynamic model after implementing direct collocation method.

Selection of the collocation times $t_{k,i}$

1. The sufficient selection of collocation times or collocation points can lead to high orders integration.

2. If the order of polynomial **$\dot{\mathbf{x}}$=F(x,u,**t) is up to $(2K-1)$ or mathematically $< (2K-1)$, the integration will be exact by using *Gauss-Radau* collocation points, which includes the end point of the collocation interval as collocation point. Hence the integration order of *Gauss-Radau* is $(2K-1)$ and it is best suited for DAEs system.

3. If the order of polynomial **$\dot{\mathbf{x}}$=F(x,u,**t) is up to $(2K)$ or mathematically $< (2K)$, the integration will be exact by using *Gauss-Legendre* collocation points. Hence the integration order of *Gauss-Legendre* is $(2K)$ and it is best suited for stiff ODEs system.

4. Note that, an extra point $t_{k,0}$ is added into the collocation points $t_{k,1}$,...,$t_{k,K}$, in order to enforce the initial value constraint $\theta_{k,0} = x_k$ or continuity in collocation equations.

5. The interval length of each collocation point have to be scaled by the length of collocation time interval $h = t_{k+1} - t_k$. Moreover, when collocation time interval is too long, the approximation of interpolation polynomial will be not good.

Error and Stability

1. Like many other collocation methods, the Gauss-Legendre methods are A-stable, which can be used to can handle stiff equations. According to Biegler L.T.(2010), in the presence of very fast dynamics, larger time steps $h = t_{k+1} - t_k$ can be used to predict steady state and slow dynamics accurately. According to Sébastien Gros (2016), it can handle eigenvalues at ∞.

2. Gauss-Radau collocation methods are L-stable, which is the special case of the A-stability. And can often be used for stiff equations.

3. Integration error: on collocation time interval as $h = t_{k+1} - t_k$, the integration error depends on the order K of the interpolation polynomial. For Gauss-Legendre is $O(h^{2K})$ and for Gauss-Radau is $O(h^{2K-1})$. Have to note that the integration error only appears at the end-state of the integrator, but not at the intermediate points (Biegler L.T.(2010)).

4. The accuracy of direct collocation method can be increased by adopting following two methods.

   - Increase the order K of interpolation polynomial.
   - Decrease the size of the collocation time intervals $[t_k, t_{k+1}]$.

   However, according to Moritz Diehl and Sébastien Gros (2017), the result of NLP becomes worse as K increases beyond relatively small orders. Thus it is better to select K up to 4.

3.2.6 Nonlinear optimization

After implementing direct collocation method, a dynamic optimization problem is transformed into a NLP problem, which have to be solved by one of the nonlinear optimization algorithms. Before seeking en efficient NLP solver, a generic NLP formulation is given as:

$$\begin{aligned} \min_{\mathbf{w}} \quad & \phi(\mathbf{w}) \\ \text{s.t.} \quad & g(\mathbf{w}) = \begin{bmatrix} \theta_{0,0} - \bar{\mathbf{x}}_\mathbf{0} \\ p(\theta_0, t_1) - \theta_{1,0} \\ \mathbf{F}(\theta_{0,i}, \mathbf{u}_0) - \sum_{j=0}^{K} \theta_{0,j} \dot{L}_{0,j}(t_{0,i}) \\ \cdot \\ \cdot \\ \cdot \\ p(\theta_k, t_{k+1}) - \theta_{k+1,0} \\ \mathbf{F}(\theta_{k,i}, \mathbf{u}_k) - \sum_{j=0}^{K} \theta_{k,j} \dot{L}_{k,j}(t_{k,i}) \\ \cdot \\ \cdot \\ \cdot \end{bmatrix} = 0 \end{aligned} \tag{3.10}$$

Where, decision variables are:

$$\mathbf{w} = [\theta_{0,0}, ..., \theta_{0,K}, u_0, ..., \theta_{N-1,0}, ..., \theta_{N-1,K}, u_{N-1}]^\top \tag{3.11}$$

With $k = 0, 1, ..., N-1$. The dimension of $\mathbf{w}$ in NLP is $N(n_x(K+1) + n_u)$. Where, N represents the number of discrete time elements. n_x and n_u represent the number of states and control inputs respectively. Then for equality constraint $g(\mathbf{w}) = 0$, the meaning of each equality constraint is:

- $\theta_{0,0} - \bar{\mathbf{x}}_\mathbf{0} = 0$ is fixed initial value. Where, $\bar{\mathbf{x}}_\mathbf{0}$ is the given initial condition and $\theta_{0,0}$ is interpolated state approximation.

- $p(\theta_k, t_{k+1}) - \theta_{k+1,0}$ is continuity constraint required across the interval boundaries, which is similar with shooting gap in multiple shooting method.

- $\mathbf{F}(\theta_{0,i}, \mathbf{u}_0) - \sum_{j=0}^{K} \theta_{0,j} \dot{L}_{0,j}(t_{0,i})$ is integration constraints for $k = 0$, and the rest are remaining integration constraints for $k = 1, ..., N-1$. Have to note that the integration of the system dynamics over collocation time interval is performed by solving collocation equations as discussed in **section 3.2.5**. Hence these can also be seen as collocation equation conditions except from continuity constraint and can also enforce continuity constraint directly within the collocation equations.

- The high integration order of collocations holds only at the the main time grid t_k, because interpolations at finer time grids $t_{k,i}$ causes a poor numerical accuracy.(Moritz Diehl and Sébastien Gros (2017))

Definition and Properties of DAEs

If a system of equations with the following form, $\mathbf{F}\ (\dot{\mathbf{x}}, \mathbf{x}, t)$**=0** and the jacobian matrix, $\frac{\alpha F}{\alpha \dot{\mathbf{x}}}$, is singular, then this form of equations is called as differential algebraic equations (DAEs). Where singular matrix is a square matrix which is non-invertible. Furthermore, if and only if a matrix has a determinant of 0, then this matrix is a singular matrix. Then the fully-implicit form of DAEs, $\mathbf{F}\ (\dot{\mathbf{x}}, \mathbf{x}, t)$**=0**, can be transformed into a semi-explicit form as expressed in equation (3.12):

$$\dot{\mathbf{x}} = \mathbf{F}(\mathbf{x}, \mathbf{z}, \mathbf{u}) \ \ (differential\ equation) \tag{3.12a}$$

$$\mathbf{0} = \mathbf{G}(\mathbf{x}, \mathbf{z}, \mathbf{u}) \ \ (algebraic\ equation) \tag{3.12b}$$

Where, **x** is differential variables, **z** is algebraic variables which derivatives do not appear in the DAEs , **u** are input variables respectively. In semi-explicit DAEs, differential and algebraic equations have their own specific functions:

- **Differential equations:** Describe the dynamic behavior of the system, such as mass and energy balances.

- **Algebraic equations:** By describing conservation laws of mass, energy etc, balances of mass, energy, mole, etc and also the constraints of the dynamic process to ensure physical and thermodynamic relations.

Therefore DAEs are extensively applied in large system which consists of many subsystems, due to the aims of above-mentioned two separate subequations. Furthermore, the system described by DAEs is easier to develop, maintain and modify. The DAE-

constrained NLP solved by direct collocation method can be expressed as:

$$\begin{aligned} \min_{\mathbf{w}} \quad & \phi(\mathbf{w}) \\ \text{s.t.} \quad & g(\mathbf{w}) = \begin{bmatrix} \theta_{0,0} - \bar{\mathbf{x}}_{\mathbf{0}} \\ p(\theta_0, t_1) - \theta_{1,0} \\ \mathbf{F}\Big(\frac{\alpha}{\alpha t}\mathbf{x}(\theta_k, t_{k,0}), (\theta_{k,0}), (\mathbf{z}_{k,0}), \mathbf{u}_k\Big) \\ . \\ . \\ . \\ p(\theta_k, t_{k+1}) - \theta_{k+1,0} \\ \mathbf{F}\Big(\frac{\alpha}{\alpha t}\mathbf{x}(\theta_k, t_{k,K}), (\theta_{k,i}), (\mathbf{z}_{k,K}), \mathbf{u}_k\Big) \\ . \\ . \\ . \end{bmatrix} = 0 \end{aligned} \tag{3.13}$$

Where, decision variables $\mathbf{w}$ are:

$$\mathbf{w} = [..., \theta_{k,0}, \theta_{k,1}, \mathbf{z}_{k,1}, ..., \theta_{k,K}, \mathbf{z}_{k,K}, \mathbf{u}_K, ...]^\top \tag{3.14}$$

With, $k = 0, 1, ..., N-1$.

Hence, for system dynamic described by DAEs, the interpolation polynomial used in orthogonal collocation can be expressed for both differential and algebraic states.

Interpolation polynomials:

$$\mathbf{x}(\theta_k, t) = \sum_{i=0}^{K} \theta_{k,i} L_{k,i}(t) \tag{3.15a}$$

$$\mathbf{z}(z_k, t) = \sum_{i=1}^{K} z_{k,i} L_{k,i}(t) \tag{3.15b}$$

Where, have to notice that for algebraic state interpolation polynomial i starts from 1, that is because algebraic states z appear only in the dynamic constraint and it leads to the difference of degree of freedom for two states. The degree of freedom for per differential state is $K+1$ and per algebraic state is K. However, algebraic state z does not actually integrated during interpolation process in the collocation equations.

3.3 Model predictive control

Due to the effectiveness in respecting constraints and multiple-input-multiple-output systems, the application of model predictive control (MPC) has had great success in chemical processes optimization and control field. MPC , which was initially proposed by Richalet,J. et al. (1978) and Cutler,C.R. and Ramaker,B.L. (1980), calculates a sequence

of optimal control inputs at every sampling time of the controller by solving an optimization problem which minimizes a pre-defined cost function over a finite prediction horizon subject to input and state constraints. Hence the basic concept of MPC can also be briefly descried as using a dynamic model to predict system behavior, and then optimize the prediction to get the best decision. As a kind of closed-loop optimal control strategy, the advantage of MPC is that MPC couples open loop optimization with feedback control. The main idea and strategies of MPC are discussed in the following sections.

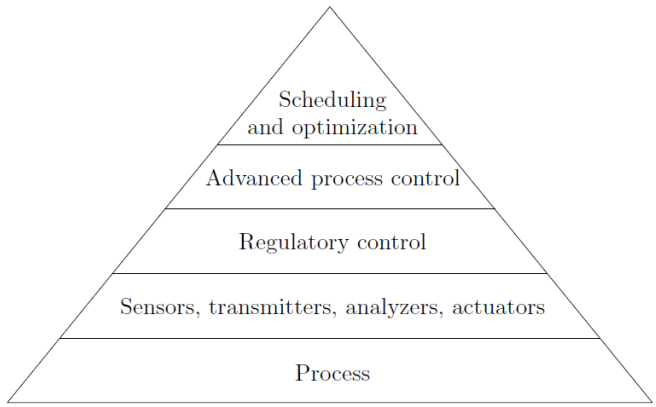


Figure 3.4: Typical control hierarchy (Foss,B. and Heirung,T.A.N.(2013))

MPC and control hierarchy

The typical control system is constructed according to the control hierarchy given in Figure 3.4.

- **Process:** Often known as the controlled system which provides realtime data to the regulatory control layer through sensors, transmitters, analyzers and actuators.
- **Regulatory control layer:** Conventional controllers such as PID controllers are applied to control properties like flow rate, pressure and temperature etc. Set points for those properties are required in regulatory control layer, which are provided by advanced control layer.
- **Advanced control layer:** MPC is implemented in advanced control layer to optimally control a process through regulatory control layer. The measurements y_t are passed from the lower control layer, at the same time set points and constraints are supplied from scheduling and optimization layer. After MPC implementation, the computed control inputs u_t which are the outputs from MPC will be used as set points for regulatory control layer for low level control.

- **Scheduling and optimization:** Optimization problem supplies set points and constraints which will be adopted in the advanced control layer.

3.3.1 MPC algorithm

At first, the block diagram of a model predictive control system is given in Figure 3.5.

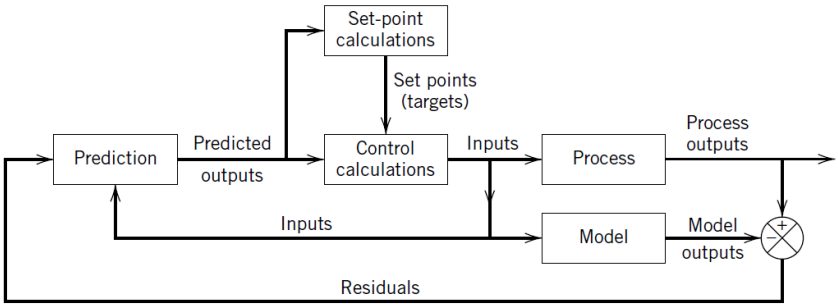


Figure 3.5: Block diagram for model predictive control (Seborg et al. (2011))

1. *Model*, which is dynamic model described by ODEs or DAEs, is used to predict the current values of the output variables. Models are also the central of the MPC, because the optimal control move Δu depends on the initial state of the dynamic system and the most likely initial state is determined by using the past record of measurements. According to J.B.Rawlings et al. (2018), *the state estimation problem* in MPC is to examine the record of past data, and reconcile these measurements with the model to determine the most likely value of the state at the current time.

2. *Residuals*, which is the differences between the actual outputs (*Process outputs,* y) and predicted outputs (*Model outputs,*$\hat{y}$) of the previous step, is sent to the *Prediction* block as the feedback signal along the prediction horizon. Hence the feedback information, which is necessary to control an uncertain system, enters the control loop.

3. The MPC calculations, which is consist of *Set-point calculations* and *Control calculations*, depends on current measurements and predictions of the outputs. Note that, the boundaries of inputs and outputs can be included in any of these two calculations. The MPC calculation procedures at each sampling time can be explained by calcualtions at the most current sampling time k.

    - First, the set points or targets, which later used in control calculations, are calculated by solving an economic optimization problem (steady-state optimisation) at upper layer. In addition, the calculation of set points are performed at each sampling time when the control calculations are executed.

- Then, M number of control moves $\Delta u_{(k)}, ..., \Delta u_{(k+M-1)}$ are generated by control calculations of MPC, where the current control moves Δu_k and $M-1$ future control moves are incorporated. And only the first control move is implemented to the plant. As shown in Figure 3.6, the control moves are usually considered as piece-wise constant. Note that, the control trajectory after the control horizon M are kept constant in order to reduce the computational complexity of the resulting optimization problems.
- As the calculation of control moves, P number of predicted outputs $\hat{y}_{(k)}, ..., \hat{y}_{(k+P)}$ optimally reaches the set points.

4. Moreover, *Set-point calculations* and *Control calculations* are cooperated in MPC calculation and it is the unique characteristic of MPC.

5. Why is a $M-$step control moves calculated if only the first control move is implemented? (Seborg D.E.et al.(2011))

   - This strategy is called *receding horizon approach* and it is executed in order to avoid the adverse effect of unmeasured disturbances which is caused by multistep, like $P-$step predictions and $M-$step control moves are based on the old information.
   - Therefore, the advantage of adopting receding horizon approach is that new information in the form of the most recent measurement $y_{(k)}$ is used immediately instead of being ignored for the next M sampling times by moving the prediction horizon P one sampling time forward at every new round of calculation.

Therefore, the basic principle of MPC can be summarised as the following principle given by D.Q.Mayne and J.B.Rawlings et al. (2000). As a form of control, the current control action of MPC is obtained by solving a finite horizon open loop optimal control problem at each sampling instant by using the current state of the plant or process as the initial state. The optimization yields an optimal control sequence and the first control in this sequence is applied to the plant. The illustration of the MPC principle is shown in Figure 3.6 and the basic MPC algorithm can be expressed as given in Table 3.1.

Algorithm State feedback MPC procedure
for $t = 0, 1, 2, ...$ **do**
Get the current state x_t.
Solve a dynamic optimization problem on the prediction horizon from t to $t + N$ with x_t as the initial condition.
Apply the first control move u_t form the solution above.
end for

Table 3.1: Basic MPC algorithm (Foss,B. and Heirung,T.A.N.(2013))

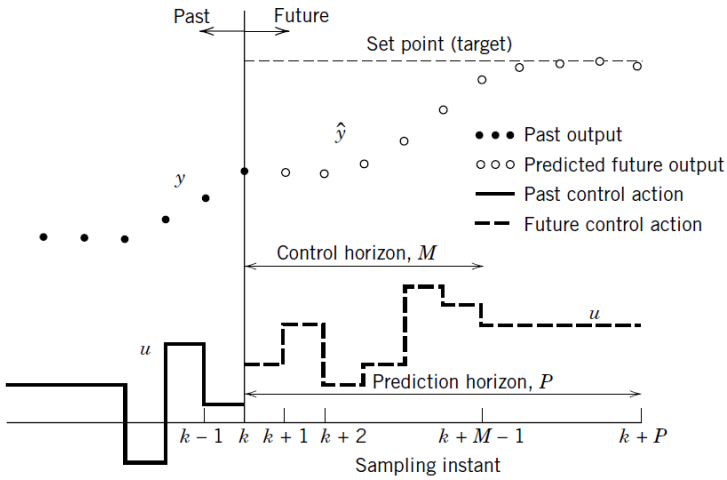


Figure 3.6: Illustration of the MPC principle (Seborg D.E.et al.(2011))

3.3.2 Nonlinear model predictive control

Nonlinear model predictive control (NMPC) has been popular in many applications, especially when constraint satisfaction is critical (Zhou Joyce Yu and Biegler,L.T (2019)). NMPC is an optimization method for the nonlinear system's feedback control, which means the system model $x_{t+1} - f(x_t, z_t, u_t) = 0$ as a constraint in MPC is nonlinear. Hence, it is also the main difference between linear MPC and NMPC. As a result, NLP solver is needed instead of QP solver to solve nonlinear and nonconvex problem.

$$\begin{aligned} \min_{\mathbf{w}} \quad & \phi(\mathbf{w}) \\ \text{s.t.} \quad & x_0 - \bar{x}_0 = 0 \\ & x_{t+1} - f(x_t, z_t, u_t) = 0 \\ & g(x_t, z_t, u_t) = 0 \\ & h(x_t, z_t, u_t) \leq 0 \\ & r(x_N) \leq 0 \end{aligned} \tag{3.16}$$

Where, $t = 0, ..., (N-1)$. Note that for direct collocation, collocation equations (3.7) on a collocation time interval $[t_k, t_{k+1}]$ would be included within the function $g(x_t, z_t, u_t) = 0$ and the collocation node values in the variables z_t. (Moritz Diehl and Sébastien Gros (2017))

As the promising technique in the industry, the performance and stability of NMPC method is strongly affected by the model quality used in the predictions, therefore it is sensitive to uncertainties which generally caused by noise, unknown disturbances and plant-model mismatch due to inaccurate modeling. Although standard MPC, which without taking into consideration uncertainties, possesses an inherent robustness under some strong assump-

Algorithm Nonlinear MPC with state feedback
for $t = 0, 1, 2, ...$ **do**
Get the current state x_t.
Solve a dynamic optimization problem (3.16) on the prediction horizon from t to $t + N$ with x_t as the initial condition.
Apply the first control move u_t form the solution above.
end for

Table 3.2: Basic NMPC algorithm (Foss,B. and Heirung,T.A.N.(2013))

tions by Grimm,G. et al. (2004), it is still not sufficient for general nonlinear constrained systems. Consequently, the robustification of MPC is necessary in the reality to solve above-mentioned issues. Over the development of last few decades, several MPC controller robustification approaches have been fulfilled, which are given in following list.

1. **Min-Max MPC:** As the first effort of MPC robustification, min-max MPC (Campo,P.J.and Morari,M.(1987)) uses ideas of robust optimization to obtain a sequence of control inputs that minimizes the cost of the worst-case realization of the uncertainty meanwhile satisfying the constraints for all the cases of the uncertainty. However, the result obtained by implementing min-max MPC may be excessively conservative and may lead to infeasible optimization problems due to the inconsideration of futural new information. This problem has been solved by taking into account the feedback into the optimization problem, which is known as feedback min-max MPC (J.H.Lee and Z.H.Yu (1997)). Meanwhile the resulting problem has infinite dimension and which leads to the difficulty to solve.

2. **Tube-based MPC methods:** As an alternative to min-max method, the application of tube-based MPC was conducted in both linear systems (D.Q. Mayne and M.M. Seron et al. (2005)) and nonlinear systems (D.Q. Mayne and E.C. Kerrigan (2007)). Tube-based MPC solves the nominal control problem, and in addition it contains an ancillary controller which ensures the evolution of the real uncertain system stays in a tube that is centered around the nominal solution. In order to develop stability and recursive feasibility guarantees, the cross section of the tube is selected as positive invariant set according to Lucia,S. and Engell,S. (2013).

3. **Multistage NMPC:** Also known as scenario-based robust NMPC. As a robust NMPC approach, multistage NMPC is able to provide the best possible solution for the robust NMPC problem by computing the optimal closed loop feedback policy over a finite prediction horizon under the assumption that scenario tree can perfectly model the uncertainties. The specific details discussed in the following section.

3.3.3 Multistage NMPC

The design of multistage NMPC is based on describing (modeling) the evolution of the uncertainty by a discrete scenario tree, where the uncertainty is resolved subsequently

at each node and the control inputs are separated into stages according to Sergio Lucia et al.(2015). In other words, M different models (scenarios) where each model has a different value for the uncertain parameter to show how the uncertainty influences state evolution over prediction horizon. Furthermore, in multistage NMPC the modeling of uncertainty by a tree of discrete scenarios requires that scenario tree can perfectly model uncertainty.

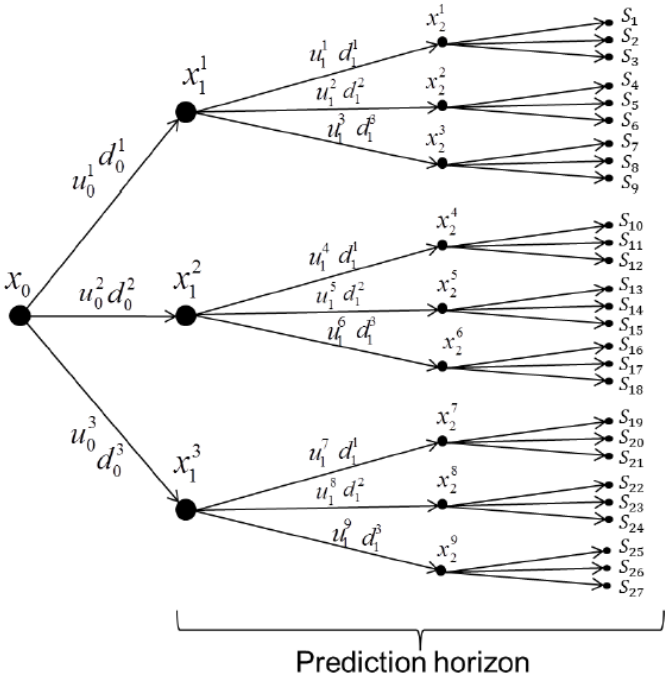


Figure 3.7: Scenario tree representation of the evolution of the uncertainty for multistage NMPC. (Lucia,S. and Engell,S. (2015))

The scenario tree that is used in multistage NMPC shown in Figure 3.7 describes the uncertainty evolution. The scenario tree develops toward future by branching from node to node.

- Each node in the discrete scenario tree represents an unknown uncertain event which affects the system besides the applied control input. x_0 is root node which can be also seem as the initial condition of state.
- Each branch from the root node to every leaf node is called a scenario. Each branching at a node represents the effect of an unknown uncertain influence (disturbance and model error) together with the chosen control input. (Lucia,S. and Engell,S. (2013))
- Under perfect estimation assumption or full state measurement, the structure of scenario tree can represent how future control inputs depends on the previous values of

the uncertainty. Thus, future control inputs can act as recourse variables to eliminate the effect of the future uncertainties.

- At each node of the scenario tree, a decision or control input is computed based on the information up to that time with taking into consideration clearly the uncertainty's future evolution as well as the future decisions or recourse variables on these branches.
- Multistage NMPC is a closed-loop robust NMPC approach. Note that, if the uncertainty is not known at one sampling time, it will remain unknown at the next sampling time. Thus, new scenario tree will have to be solved at the next sampling time.

The discrete-time formulation for an uncertain nonlinear system scenario tree can be written as:

$$x_{k+1}^{j} = f(x_k^{p(j)}, u_k^j, d_k^{r(j)}) \tag{3.17}$$

Where, k is the stage, j is the position of node and r is the realization of uncertainty in the scenario tree. Hence the position of each node can be expressed as (j,k) and it also can be simplified as I. Each state $x_{k+1}^{j} \in \mathbb{R}^{n_x}$ depends on the the values of the previous stage, like parent state $x_k^{p(j)} \in \mathbb{R}^{n_x}$, the control input $u_k^j \in \mathbb{R}^{n_u}$ and the corresponding uncertainty $d_k^{r(j)} \in \mathbb{R}^{n_d}$. For the sake of simplicity, it is convenient to assume the tree has the same number of branches at all nodes. The scenario from root node x_0 to the leaf nodes can be denoted by S as expressed in equation (3.18).

$$S = \{x_0, x_1^1, x_1^2 ..., x_{N_P}^N, u_1^1, u_1^2, ..., u_{N_P}^N\} \tag{3.18}$$

Where, N is the number of scenarios or leaf nodes $\forall i = 1, ..., N$. N_P is the length of prediction horizon. Hence, all states (nodes) and control inputs in the scenario tree can be written respectively as:

$$X = \{x_0, x_1^1, x_1^2 ..., x_{N_P}^N\} \tag{3.19a}$$

$$U = \{u_1^1, u_1^2, ..., u_{N_P}^N\} \tag{3.19b}$$

Therefore, the optimization problem resulting from multistage NMPC can be written as:

$$\min_{x_k^j, u_k^j} \sum_{j=1}^{S} \omega_j \sum_{i=1}^{N} J_i(x_{k,j}, u_{k,j}) \tag{3.20a}$$

subject to:

$$x_{0,j} = x_{init} \tag{3.20b}$$

$$x_{k+1,j} = f\big(x_{k,j}, u_{k,j}, p_j\big) \tag{3.20c}$$

$$g\big(x_{k+1}^j, u_k^j\big)\big) \leq 0 \tag{3.20d}$$

$$u_{k,j} = u_{k,l} \; if \; x_{k,j} = x_{k,l} \tag{3.20e}$$

Where, J_i is the cost function of each scenario on the scenario tree, which is weighted by ω_i denotes the probability of each scenario S_i. Equation (3.20b) is the initial condition constraint, where x_{init} is the vector of starting points for the states. Equation (3.20c) is the model of nonlinear dynamic system. Equation (3.20d) is the constraints on control inputs and states. Equation (3.20e) is called as *non-anticipativity constraints*, which indicate that decisions or control inputs based on the same parent state $x_k^{p(j)}$ or information have to be equal to correctly model the real-time decision problem. This is because uncertainties can be realised after applying the control inputs. In other words, before the control inputs or decisions are taken at one node, the future branching of the scenario tree at a node is unanticipatable. Furthermore, the multistage NMPC converts into the standard NMPC when the number of scenarios is $S = 1$.

As a summary, what multistage scenario based NMPC doing is at each sampling time, the uncertain parameters can take any discrete value from the subset of M different models. Then different control input profiles are designed for all scenarios. Thus, by doing this procedure new information will be available in the future and the decision variables can offset the effect of the uncertainty.

Selection of scenarios

Selection of scenarios will build the scenario tree. The scenario tree is often generated using finite realizations of the uncertainty sampled from an uncertainty set or a probability distribution function.

Thus the first step to formulate a scenario-based multistage NMPC is to chose the discrete realizations of the uncertainty from the uncertainty set, which can be denoted as P. Then, a combination of maximum, minimum and nominal values of the different uncertain parameters are selected as the scenarios to assure the robust constraint satisfaction for any realization of the uncertainty from the uncertainty set P. At the end, a scenario tree as shown in Figure 3.7 is generated by these M amounts of discrete realizations of the uncertain parameters which are sampled from the uncertainty set.

However, this approach, which is similar to random sampling methods, ignores the correlations between uncertain parameters.

Robust horizon of the problem

According to Lucia,S. and Engell,S. (2013), the main drawback of multistage NMPC is that the size of the NLP that has to be solved at each time step grows exponentially with the prediction horizon, with the number of uncertainties and with the levels of the uncertainty considered in the design of the scenario tree. $Robust\ horizon$ can be taken to avoid this phenomenon, where to limit the branching of the scenario tree only up to certain stages. This simple strategy to deal with the growth of the tree with the prediction horizon is known as robust horizon, which states that after exceeding this new limited prediction horizon, the uncertainty is assumed to be constant until the end of the prediction horizon. The main idea of this simplification is coming from the receding horizon nature of NMPC, since the control inputs will be recomputed at the next sampling time, accurate modeling

of far future is not important anymore. Thus, the example of multistage NMPC with robust horizon is given in Figure 3.8. Where prediction horizon $N_P = 4$, robust horizon $N_R = 2$ and the total number of scenarios $S = M^{N_R}$.

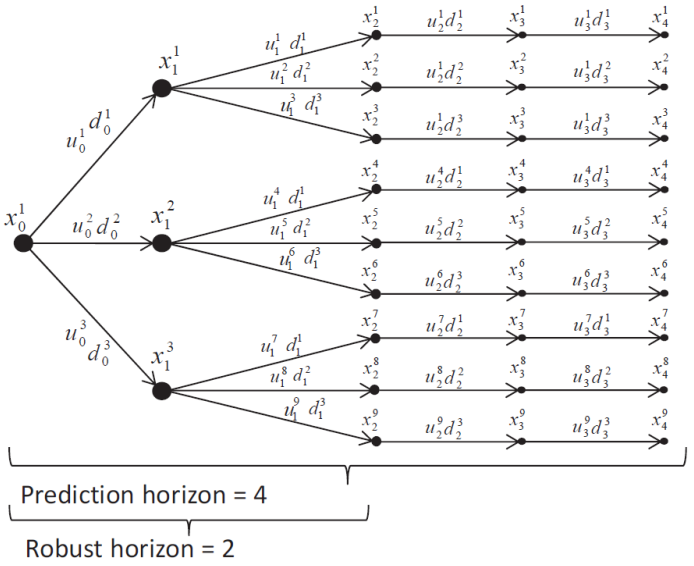


Figure 3.8: Scenario tree representation of the uncertainty evolution with robust horizon for multistage NMPC (Lucia,S. and Engell,S. (2015))

Direct collocation in multistage NMPC

Since the discrete scenario tree requires time-discrete model of the system, direct collocation method is selected to discretize both the states and control inputs and set them as optimization variables in the resulting nonlinear programming problem (NLP). The specific details of direct collocation method has been discussed in **Chapter 3.2.5**. The purpose of this subsection is to fit the direct collocation method into multistage NMPC. Since each control interval is divided into finite elements, the state trajectory can be parametrized using Lagrange polynomials:

$$x^j_{k,\gamma}(t) = \sum_{i=0}^{K} x^j_{k,\gamma,i} L_i(t) \tag{3.21}$$

Where, $x^j_{k,\gamma}(t)$ are the state variables at position (j,k) of the scenario tree when time is t for the finite element. $x^j_{k,\gamma,i} L_i(t)$ are the state variables at stage k for the finite element γ at the collocation point i.

Chapter 4

Implementation

4.1 Implementation of modeling

In order to reduce the requirement for real experimentation, and further promote the achievement of cost, risk and time reduction, model is chosen to be used. Therefore, any chemical or physical process in practical can be described by a model of its own process. A good model is required to contain information, which is enable to predict the results when process operating conditions have been changed. Depending on the certain task, process model can be classified as shown in Figure 4.1. Since the model used in this process is mathematical model, the introduction here is focused in this type of model as well.

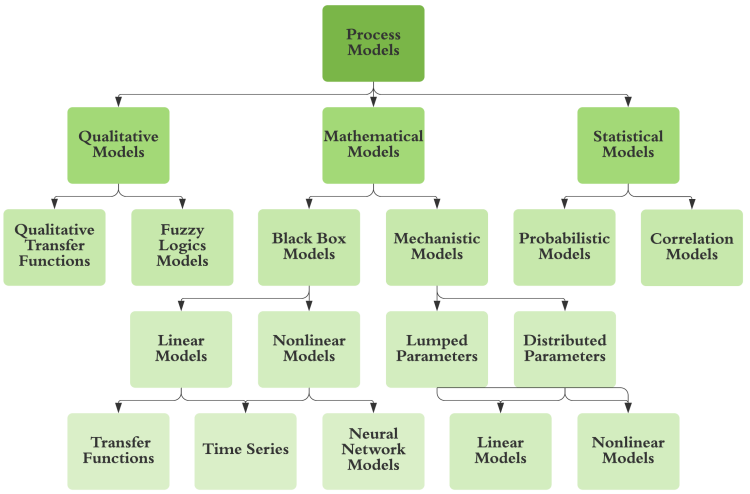


Figure 4.1: Process model classification

1. **Mathematical models:** As specified in Figure 4.1, mathematical model involves two different sub-types, which are knowledge-based mechanistic model or data-based black box model.

    - *Mechanistic models:* Also known as *white box model*, is derived from the well defined process which is governed by physics and chemistry. Hence a set of differential equations, which is based on conservation laws, can be adopted to describe its dynamic behavior. Furthermore, the structures of mechanistic models are distinct due to the different systems. Where lumped parameter model is described by a set of ODEs, while distributed parameter model requires partial differential equations. Notwithstanding, with certain assumptions distributed systems can be approximated by a set of ODEs. Both these two models can be further described by linear or nonlinear descriptions. However, when the information about the process is vague or the process is too complicated, the resulting model equations will be too difficult to solve. In this situation, black box model may be established to use.

    - *Black box models:* It is a lumped model with parameter model, which describes the functional relationships between system inputs and outputs. These functional parameters have no physical significance and may potentially result unreasonable consequences, thus it is the disadvantage of black box model compared to mechanistic model. Moreover, when the system is MIMO system, the modelling with black box pattern needs huge data in order to lead to reliable predictions.

    - *Hybrid models:* The combination of mechanistic models with black box model is known as hybrid model or grey box model. The detailed implementation have already been fulfilled in specialization project Ding,N. (2019).

2. **Qualitative models:** The concrete introduction of this model type can be found from J. Hendler et al. (2010) and not introduced in this thesis.

3. **Statistical models:** The process is described in statistical terms and system dynamics are not captured in statistical models, but it plays an important role in assisting in higher-level decision making, process monitoring, data analysis and statistical process control according to Peng Zhang (2010). Where probabilistic model combines random variables and probability distributions into the model of an event or phenomenon and then provide likelihood of these variables which taking on certain values. For correlation models, by measuring the variations of any two variables to quantify the extent of similarity and further build correlation models.

In this thesis, modeling of the simple thermal energy storage system is based on the conservation laws. Thus, it is mechanistic mathematical model and its specific derivations are discussed in **Chapter 5**.

4.2 Implementation of simulation

Once obtaining the system model, which is a set of DAEs in this thesis, simulations for specific control scenarios can be implemented. There are numerous simulation tools for various purposes and applications, for instantce MATLAB, LabVIEW, SIMULINK and ModelSim have been mostly applied. In this thesis, the simulation will be perfomred in MATLAB (R2019b), which is an integrated technical computing environment that involves numerical computation, advanced graphics and visualization, and a high-level programming language. Furthermore, CasADi, which is an open-source software package, is used to improve computational performance and convenient code implementation in MATLAB environment.

4.2.1 Background information of CasADi

CasADi is developed as a tool for algorithmic differentiation (AD) using a syntax similar to a computer-algebra system (CAS), and it is also the origin of its name (Joel A.E.Andersson et al.(2018)).

OCP can be efficiently solved by the direct method. In this thesis, the direct collocation method is selected as the numerical method for solving OCP with Gauss-Radau collocation method and the 3rd order ($K = 3$) interpolation polynomials. By implementing direct method, OCP is converted into a nonlinear programming problem (NLP). CasADi has many open-source and commercial NLP solver plugins, and they all have their own corresponding application fields. In this thesis NLP solver IPOPT, which is the abbreviation of *Interior Point OPTimizer* pronounced I-P-Opt, is used as the NLP solver. IPOPT implements a primal-dual interior point method, and the specific details can be found in A. Wächter and L.T.Biegler (2006).

According to Joel A.E.Andersson et al.(2018), the core of CasADi is symbolic framework, and it allows users to construct expressions and use these to define automatically differentiable functions. Therefore, NLP can be expressed in a highly symbolic expression and meanwhile with the help from ODE or DAE integrators it can be expressed as graph representation as scalar expression or matrix expression type in the background. As a tool, which is executing algorithmic differentiation, CasADi can efficiently and automatically calculate derivative information of functions to any order. Consequently, necessary first-order and second-order derivative information can be acquired for NLP solver and derivative calculation errors can be efficiently avoided.

By writing dynamic system model codes in MATLAB environment for the determined sampling time and with function ode15s which used in optimizer to get optimal inputs to simulate real plant and further present the mismatch of plant and model. This function is a solver applied to solve stiff differential equations and DAEs by using variable order method. So, ode15s is implemented inside NLP solver to find out optimal inputs.

Therefore, the resulting iNMPC in this thesis is implemented in MATLAB by using CasADi algorithmic differentiation tool, and IPOPT solver is used to solve the resulting nonlinear programming problem. The optimal control problem of this thesis are then classified into three main cases and one minor case to achieve the objective of minimizing commercial

energy cost. Where, standard NMPC is implemented for cases with and without storage tank with no presence of uncertainty, and with storage tank with direct solar heating. Multistage NMPC is implemented for case with the presence of uncertainty. Furthermore, in order to study the robustness of multistage NMPC, the standard NMPC also implemented with consideration of uncertainty to make comparison.

Chapter 5

Modeling of thermal energy storage system

The main purpose of this chapter is to derive the model equations of the simple thermal energy storage system. In this thesis, a simple two-plant thermal energy storage system is considered. The illustration of this simple system is shown in Figure 5.1.

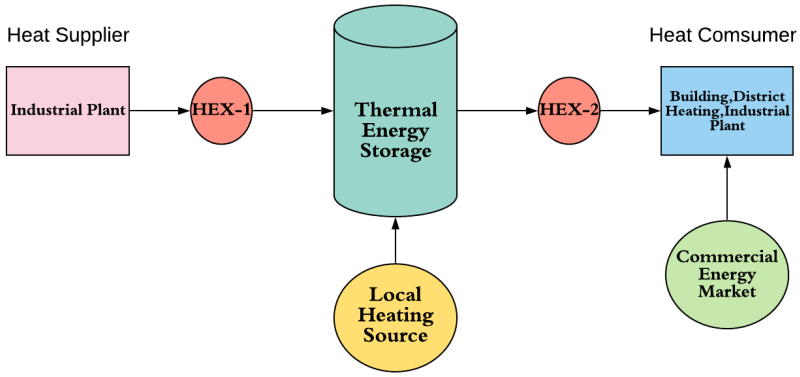


Figure 5.1: Illustration of a simple thermal energy storage system

This thermal energy storage system is consist of two plants which are heat supplier and heat consumer. Heat supplier can be various industrial plants which provide surplus (excess) heat to heat up the tank. On the other hand, the heat consumer (heat sink) can be commercial building, district heating unit and also industrial plants, which extracts energy from the TES tank to satisfy their energy demands. The thermal energy storage (TES) tank works as a buffer between these two plants to promote the energy exchange and it is con-

sidered as a hot water tank. The interaction of tank with two plants is fulfilled by two heats exchangers. Local heating source can be inexpensive energy source like solar energy, and TES tank can also be directly heated up by cheap local heating source. Expensive commercial energy will be purchased when the energy stored in the TES tank is not sufficient to fulfill the requirements on the consumer side. In practice, the cost of commercial energy is much more expensive than the local heating source energy, hence the objective of this thesis is to minimize the cost of purchased commercial energy.

In order to derive the system model, mass and energy conservation laws are used with some necessary assumptions to the system. As a result, the numerical case studies are performed under the obtained mathematical model in this thesis.

5.1 Model description and assumptions

In order to model the two-plant thermal energy storage system correctly, some necessary assumptions have to be held, which are considered as follows:

1. Consider the heat source and sink as reservoirs, which are infinitely large sources of extensive quantity with constant intensive properties. So, this implies that the temperature of the source and sink streams, which is an intensive property not depending on the system size or the amount of material in the system, are only depending on the source and sink plants.

2. Consider two heat exchangers as devices with two chambers (lumps) which represent the hot side and the cold side respectively. Consider the temperature of both chambers throughout their volumes are the same.

3. Assume the supplier and consumer plant are close enough to each other so that the heat losses caused along the pipes due to time delays can be neglected. Moreover, assume heat exchangers are insulated from its surroundings and such that no heat losses from two heat exchangers to the surrounding environment.

4. Assume heat exchanger overall heat transfer coefficient ,U_{hex}, is constant.

5. Assume two heat exchangers have the same dimensions and parameters, which means they are identical in the system.

6. The thermal stratification is not considered in this thesis. Assume the tank is uniformly mixed throughout its volume so that the temperature inside the tank is identical with the tank outlet temperature. Assume holdup of the tank is perfectly controlled and constant at V_{tank}.

7. Assume the heat storage fluid is incompressible and has a constant heat capacity c_p. Then, the storage fluid has the physical properties of water, since thermal energy storage material is selected as water.

8. Assume the temperature range of storage fluid is below the boiling temperature of water, $100°C$. Therefore, TES can further be assumed as an unpressurized tank.

9. The energy flow is from the source side to the sink side conventionally.

In the following sections, the procedure of deriving system's mathematical model based on the forenamed assumptions are presented step by step.

5.2 Process modeling

Before deriving the system model, it is better to convert the abstract system illustration into more detailed type to deeply understand what is going on this TES system. Thus, the topology is illustrated in the following subsection for better understating.

5.2.1 Topology illustration

The illustration of simple TES system with one supplier and one consumer, along with TES tank is given by a topology in Figure 5.2 and the specific explanations of each part and model parameters are given in Table 5.1 and 5.2 . The supplier is the source of energy, which can supply excess thermal energy to the TES tank. The consumer is a heat sink, which extracts thermal energy from the TES tank to satisfy the energy demands. Because of the presence of two heat exchangers, the heat can be exchanged with hot water as storage material between the supplier and consumer and also TES tank.

In practice, in order to deal with the possibly happened sharp fluctuations in heat demand at the consumer side, purchasing extra commercial energy from market is one of the common method to compensate the mismatch caused by insufficient energy supply from the TES tank. However, these commercial energy is expensive and may further increase the emissions of greenhouse gases.

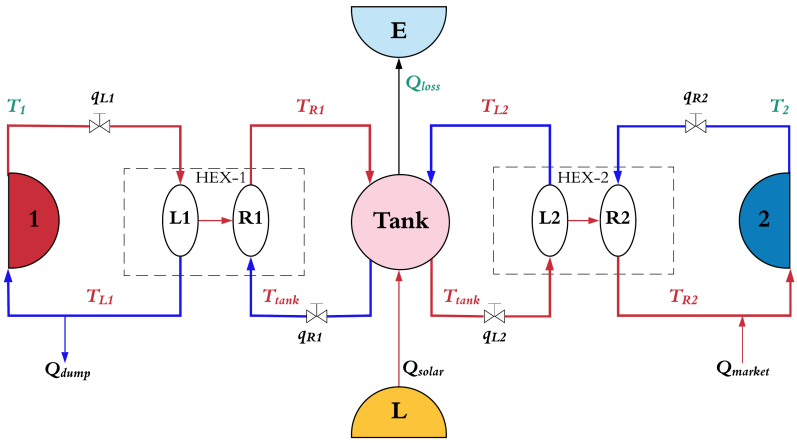


Figure 5.2: Topology of the system. The states, inputs and disturbances are shown in red, black, and green respectively. The red lines represent the hot streams and the blue ones cold streams.

Term	Explanation
1	Energy supply: Sources
2	Energy demand: Sinks
L	Local heating source
E	Environment
Tank	Thermal energy storage tank
HEX-1	Heat exchanger 1
HEX-2	Heat exchanger 2
L1,L2	Hot side of HEX-1,2
R1,R2	Cold side of HEX-1,2

Table 5.1: Topology term explanations

Term	Explanation
V_{hex}	Heat exchanger volume
A_{hex}	Heat exchanger heat transfer area
U_{hex}	Heat exchanger heat transfer coefficient
V_{tank}	Tank volume
A_{tank}	Tank surface area
U_{tank}	Tank heat loss coefficient
ρ	Storage fluid density
c_p	Storage fluid specific heat capacity
Q_{market}	Purchased commercial energy source
Q_{dump}	Dumped surplus heat
Q_{solar}	Direct solar heat
Q_{loss}	Tank lost heat
P_m	Commercial heating cost
P_t	Direct solar heating cast
P_u	Quadratic penalty cost

Table 5.2: Model parameter explanations

5.2.2 Energy balances and Mass balances

The general energy conservation law or also called as the first law of thermal dynamics and mass conservation law are used to derive the system's model. First of all, the overall

description of dynamic balance can be expressed as:

$$\underbrace{Change\ Inventory}_{\text{accumulated in the system}} = \underbrace{In - Out}_{\text{through the system's boundary}} + \underbrace{Generated - Loss}_{\text{internally in the system}} \tag{5.1}$$

Mathematically, the general balance equation per unit time is expressed as:

$$\frac{dB}{dt} = B_{in} - B_{out} + B_{generated} - B_{loss} \tag{5.2a}$$

Thus, general energy balance law can be expressed as: *rate of energy accumulation = rate of energy in by convection - rate of energy out by convection + net rate of heat addition to the system from the surroundings + net rate of work performed on the system by the surroundings* . Mathematically, it can be expessed as:

$$\frac{dU}{dt} = H_{in} - H_{out} + Q + W - p_{ex}\frac{dV}{dt} \tag{5.3a}$$

Since no additional work added to the thermal energy storage system, W can be neglected. Furthermore, $-p_{ex}\frac{dV}{dt}$ which is the work supplied to the system when its volume changes can also be neglected. After assuming constant pressure and constant volume, $H = U + PV$, which can lead equation (5.3a) to become:

$$\frac{dH}{dt} = H_{in} - H_{out} + Q \tag{5.4}$$

Then, conservation of mass can also be expressed as following the expression of general dynamic balance: *rate of mass accumulation = rate of mass in - rate of mass out*. Mathematically, it can be expressed as follows:

$$\frac{dm}{dt} = w_{in} - w_{out} \tag{5.5}$$

By introducing the density, equation (5.5) becomes:

$$\frac{d\rho V}{dt} = \rho_{in} q_{in} - \rho_{out} q_{out} \tag{5.6}$$

In this thesis, the main focus is on modeling of the dynamics of heat exchange between the TES tank, and one supplier and also one consumer plant. Therefore, it includes the energy balances over the heat exchangers and over the TES tank. The dynamic of whole system can be derived by applying enthalpy balances over the components of system shown in Figure 5.2.

Reservoirs:

Since the heat source and heat sink have been assumed as reservoirs in assumption 1, the temperatures of these two reservoirs, T_1 and T_2, are considered as given. So they are independent of the system's dynamic, for example the change of system's size can not affect these two variables. Then the energy balance of heat source and heat sink can be derived

according to equation (5.4).

Source and sink

$$T_1 = constant, \quad \frac{dH_1}{dt} = 0 \tag{5.7a}$$

$$T_2 = constant, \quad \frac{dH_2}{dt} = 0 \tag{5.7b}$$

Lumps:

1. **Flow simplification:** Before deriving the corresponding model equations for each sub-part of the thermal energy storage system, the simplifications of flow streams are necessary. During the whole operation period, the heat exchanger 1 and 2 are always filled with fluid, so the volumes of the heat exchangers are always constant. Thus, $\frac{dV_{hex}}{dt} = 0$. Thence, the volumetric inlet and outlet flows of heat exchanger 1 and 2 are equal. Then, the volume of the storage tank is also fully filled and perfectly controlled during the entire operation period. So, the flow in and out of the TES tank are also identical. Thus, the simplifications of flows can be expressed as shown in equation (5.8), which are already marked out in Figure 5.2.

$$q_{1|L1} = q_{L1|1} = q_{L1} \tag{5.8a}$$

$$q_{2|R2} = q_{R2|2} = q_{R2} \tag{5.8b}$$

$$q_{R1|tank} = q_{tank|R1} = q_{R1} \tag{5.8c}$$

$$q_{L2|tank} = q_{tank|L2} = q_{L2} \tag{5.8d}$$

2. **Temperature indications:** As shown in Figure 5.2, temperatures in this simple system is indicated in Table 5.3. According to the assumptions of the system, the temperatures exiting HEX-1, HEX-2 and TES tank are considered to be same as those inside the their corresponding volumes.

Term	Explanation
T_1	Supplier side inlet temperature
T_2	Consumer side inlet temperature
T_{L1}	HEX-1 hot side temperature
T_{R1}	HEX-1 cold side temperature
T_{tank}	TES tank temperature
T_{L2}	HEX-2 hot side temperature
T_{R2}	HEX-2 cold side temperature
T_{surr}	Ambient temperature

Table 5.3: Temperature indications

3. **Heat exchanger 1:** The temperature difference of two chambers is the driving force for heat exchangers. Further, the fluid for heat exchange is considered as hot water. By applying the mass balance equation (5.6) across the heat exchanger 1 gives the mass balance as:

$$\frac{d\rho V_{hex}}{dt} = \rho_{in} q_{1|L1} - \rho_{out} q_{L1|1} \tag{5.9a}$$

$$\frac{dV_{hex}}{dt} = q_{1|L1} - q_{L1|1} \tag{5.9b}$$

$$\frac{dV_{hex}}{dt} = q_{L1} - q_{L1} = 0 \tag{5.9c}$$

Inside the heat exchanger, energy is transferred from the hot side (L_1) to the cold side (R_1). The energy balance equation for this phenomena can be expressed as:

$$\frac{dH_{L1}}{dt} = \sum H_{in} - \sum H_{out} + Q_{net} - W_s \tag{5.10}$$

As already assumed, heat exchangers are insulated from its surroundings, so there is no heat loss from heat exchangers to the environment, $Q_{loss,HEX-i} = 0$. Therefore the only heat exchange in heat exchanger is between the hot and cold fluids. Then equation (5.10) becomes:

$$\frac{d(\rho c_p V_{hex} T_{L1})}{dt} = \rho_{in} c_{pin} q_{L1} T_1 - \rho_{out} c_{pout} q_{L1|1} T_{L1} - Q_{L1|R1} \tag{5.11a}$$

$$\rho c_p V_{hex} \frac{dT_{L1}}{dt} = \rho c_p q_{L1}(T_1 - T_{L1}) - Q_{L1|R1} \tag{5.11b}$$

Where, $Q_{L1|R1}$ implies energy transfer happened among heat exchanger's two chambers. After rearranging equation (5.11b), the final form can be obtained.

Energy balance equation for $\mathbf{L}_1$:

$$\frac{dT_{L1}}{dt} = \frac{1}{V_{hex}} \Big\{ q_{L1}(T_1 - T_{L1}) - \frac{Q_{L1|R1}}{\rho c_p} \Big\} \tag{5.12}$$

As following the same procedures done in L_1, the energy balance equation for the cold side, R_1 can also be obtained.

Energy balance equation for $\mathbf{R}_1$:

$$\frac{dT_{R1}}{dt} = \frac{1}{V_{hex}} \Big\{ q_{R1}(T_{tank} - T_{R1}) + \frac{Q_{L1|R1}}{\rho c_p} \Big\} \tag{5.13}$$

4. **Heat exchanger 2:** Since the design parameters of both heat exchangers are the same, so the derivations of energy balance equations for heat exchanger 2 follow the same procedures done in heat exchanger 1. In addition, the results are similar except

from different variables.

Energy balance equation for $\mathbf{L}_2$:

$$\frac{dT_{L2}}{dt} = \frac{1}{V_{hex}}\Big\{q_{L2}(T_{tank} - T_{L2}) - \frac{Q_{L2|R2}}{\rho c_p}\Big\} \tag{5.14}$$

Energy balance equation for $\mathbf{R}_2$:

$$\frac{dT_{R2}}{dt} = \frac{1}{V_{hex}}\Big\{q_{R2}(T_2 - T_{R2}) + \frac{Q_{L2|R2}}{\rho c_p}\Big\} \tag{5.15}$$

5. **Heat transferred from hot side to cold side in heat exchanger:** As discussed in **Chapter 2.3**: Heat transfer modeling in heat exchangers, the heat transfer model from hot fluid to cold fluid can be expressed as following equation (2.13) as:

$$Q = U_{hex}A_{hex}\Delta T_m \tag{5.16}$$

Where, subscript *hex* implies heat exchanger and $(UA)_{hex}$ implies the heat-transfer conductance of the heat exchangers.

Mean temperature difference, its specific form logarithmic mean temperature difference and approximations of LMTD have been already clearly discussed in **Chapter 2.3**. Chen J. (1987) shows that both Underwood approximation and Chen approximation are performing quite close to the results of original LMTD. In this thesis, Underwood approximation will be used as the approximation of LMTD. Hence, heat transfer from hot fluid to cold fluid in two heat exchangers can be expressed respectively as:

$$Q_{L1|R1} = U_{hex}A_{hex}\Delta T_{m,1} \tag{5.17a}$$
$$Q_{L2|R2} = U_{hex}A_{hex}\Delta T_{m,2} \tag{5.17b}$$

Following the endpoint temperature differences of current-flow heat exchanger which is given in equation (2.16), temperature differences $\Delta T_{m,1}$ and $\Delta T_{m,2}$ in equation (5.17) become:

$$\Delta T_{m,1} = \left[\frac{1}{2}\big((T_1 - T_{R1})^n + (T_{L1} - T_{tank})^n\big)\right]^{\frac{1}{n}}, n = \frac{1}{3} \tag{5.18a}$$

$$\Delta T_{m,2} = \left[\frac{1}{2}\big((T_{tank} - T_{R2})^n + (T_{L2} - T_2)^n\big)\right]^{\frac{1}{n}}, n = \frac{1}{3} \tag{5.18b}$$

6. **Storage tank:** By applying the mass balance equation (5.6) across the storage tank gives the mass balance as:

$$\frac{d(\rho V_{tank})}{dt} = (\rho q_{R1|Tank} + \rho q_{L2|tank}) - (\rho q_{Tank|R1} + \rho q_{Tank|L2}) = 0 \tag{5.19a}$$

$$\frac{d(\rho V_{tank})}{dt} = (\rho q_{R1} + \rho q_{L2}) - (\rho q_{R1} + \rho q_{L2}) = 0 \tag{5.19b}$$

The energy balance equation for storage tank can be expressed as:

$$\frac{dH_{tank}}{dt} = \sum H_{in} - \sum H_{out} + Q_{net} - W_s \tag{5.20}$$

Where, enthalpies can be expressed as:

$$\sum H_{in} = \rho c_p (q_{R1} T_{R1} + q_{L2} T_{L2}) \tag{5.21a}$$

$$\sum H_{out} = \rho c_p (q_{R1} T_{tank} + q_{L2} T_{tank}) \tag{5.21b}$$

And the net heat flow, Q_{net}:

$$Q_{net} = Q_{D|tank} - Q_{tank|E} \tag{5.22a}$$

$$Q_{D|tank} = Q_{tank}, \;\; Q_{tank|E} = Q_{loss} \tag{5.22b}$$

Hence, the energy balance equation for storage tank becomes:

$$\frac{dT_{tank}}{dt} = \frac{1}{V_{tank}} \Big\{ q_{R1}(T_{R1} - T_{tank}) + q_{L2}(T_{L2} - T_{tank}) + \frac{Q_{tank} - Q_{loss}}{\rho c_p} \Big\} \tag{5.23}$$

5.2.3 Model equations

The model equations of two plants thermal energy storage system are a set of ordinary differential equations (ODEs). For the sake of simplicity and for the easy checking purpose, this set of ODEs is expressed in equation (5.24).

$$\frac{dT_{L1}}{dt} = \frac{1}{V_{hex}} \Big\{ q_{L1}(T_1 - T_{L1}) - \frac{U_{hex} A_{hex}}{\rho c_p} \Delta T_{m,1} \Big\} \tag{5.24a}$$

$$\frac{dT_{R1}}{dt} = \frac{1}{V_{hex}} \Big\{ q_{R1}(T_{tank} - T_{R1}) + \frac{U_{hex} A_{hex}}{\rho c_p} \Delta T_{m,1} \Big\} \tag{5.24b}$$

$$\frac{dT_{L2}}{dt} = \frac{1}{V_{hex}} \Big\{ q_{L2}(T_{tank} - T_{L2}) - \frac{U_{hex} A_{hex}}{\rho c_p} \Delta T_{m,2} \Big\} \tag{5.24c}$$

$$\frac{dT_{R2}}{dt} = \frac{1}{V_{hex}} \Big\{ q_{R2}(T_2 - T_{R2}) + \frac{U_{hex} A_{hex}}{\rho c_p} \Delta T_{m,2} \Big\} \tag{5.24d}$$

$$\frac{dT_{tank}}{dt} = \frac{1}{V_{tank}} \Big\{ q_{R1}(T_{R1} - T_{tank}) + q_{L2}(T_{L2} - T_{tank}) + \frac{Q_{tank} - Q_{loss}}{\rho c_p} \Big\} \tag{5.24e}$$

Where,

$$Q_{loss} = (UA)_{tank}(T_{tank} - T_{surr}) \tag{5.25a}$$

$$\Delta T_{m,1} = \left[\frac{1}{2} \big((T_1 - T_{R1})^n + (T_{L1} - T_{tank})^n \big) \right]^{\frac{1}{n}}, n = \frac{1}{3} \tag{5.25b}$$

$$\Delta T_{m,2} = \left[\frac{1}{2} \big((T_{tank} - T_{R2})^n + (T_{L2} - T_2)^n \big) \right]^{\frac{1}{n}}, n = \frac{1}{3} \tag{5.25c}$$

Where, $(UA)_{tank}$ is the heat-loss conductance of the tank. According to the model equation (5.24), the two plant thermal energy storage system's model has five distinct states (**X**) which are the exit temperatures from the chambers of heat exchangers and the tank. Inputs (**u**) are flow rates on either sides of the heat exchanger and with the local heat supply, market heat supply and dumped heat. Disturbances (**d**) are inlet temperatures of the two plants and the ambient temperature.

- **States (X):**

$$\mathbf{X} = [T_{L1} \;\; T_{R1} \;\; T_{L2} \;\; T_{R2} \;\; T_{tank}]^\top \tag{5.26}$$

- **Inputs (u):**

$$\mathbf{u} = [q_{L1} \;\; q_{R1} \;\; q_{L2} \;\; q_{R2} \;\; Q_{solar} \;\; Q_{market} \;\; Q_{dump}]^\top \tag{5.27}$$

- **Disturbances (d):**

$$\mathbf{d} = [T_1 \;\; T_2 \;\; T_{surr}]^\top \tag{5.28}$$

Then, the remaining variables are model parameters like design and physical parameters, which are independent of time during the system's whole process but dependent of storage material and heat exchanger material properties, the definition is already given in Table 5.2. After defining the states, control inputs and disturbances, the system model formulation can also be obtained, which is given in equation (5.30).

- **Model parameters:**

$$\mathbf{P} = [V_{hex} \;\; V_{tank} \;\; U_{hex} \;\; A_{hex} \;\; A_{tank} \;\; U_{tank} \;\; \rho_{water} \;\; c_{pwater} \;\; n] \tag{5.29}$$

- **System's model:**

$$\dot{\mathbf{x}} = \mathbf{F}(\mathbf{x}, \mathbf{u}, \mathbf{d}) \tag{5.30}$$

As shown in model ordinary differential equations (5.24), the Underwood's approximations of LMTD $\Delta T_{m,1}$ and $\Delta T_{m,2}$, which shown in equation (5.25b) and (5.25c), are exponential equations with $n = \frac{1}{3}$. Have to mention that the derivatives of root expressions are rational, which means the states of model would be en element of fractional denominator. Therefore, when calculating gradients by using CasADi at each time step with state values approaching zero, rational derivatives would cause singularity issues which let the derivatives explode to ∞ and further become non-defined. In order to avoid model equations having exponential or root expression terms, it is necessary to convert ODEs form into DAEs form, which the latter can eliminate the derivatives of algebraic equations.

DAEs system's model

The aforementioned issues can be solved by defining new algebraic states **z** as:

$$\mathbf{z} = [a \;\; b \;\; c \;\; d] \tag{5.31}$$

Where, let algebraic states be as:

$$a = (T_1 - T_{R1})^n \tag{5.32a}$$
$$b = (T_{L1} - T_{tank})^n \tag{5.32b}$$
$$c = (T_{tank} - T_{R2})^n \tag{5.32c}$$
$$d = (T_{L2} - T_2)^n \tag{5.32d}$$

Moreover, equation (5.32) can be written in the general form of algebraic equations as **0=G(x,z,u,d)** as in equation (5.33).

$$0 = a^{\frac{1}{n}} - (T_1 - T_{R1}) \tag{5.33a}$$
$$0 = b^{\frac{1}{n}} - (T_{L1} - T_{tank}) \tag{5.33b}$$
$$0 = c^{\frac{1}{n}} - (T_{tank} - T_{R2}) \tag{5.33c}$$
$$0 = d^{\frac{1}{n}} - (T_{L2} - T_2) \tag{5.33d}$$

Therefore, equation (5.25b) and (5.25c) can be written as:

$$\Delta T^n_{m,1} = \frac{1}{2}(a+b), \quad n = \frac{1}{3} \tag{5.34a}$$
$$\Delta T^n_{m,2} = \frac{1}{2}(c+d), \quad n = \frac{1}{3} \tag{5.34b}$$

After combining differential equations (5.24) and algebraic equations (5.33), the former ODEs model becomes a differential algebraic equations (DAEs) model as expressed in equation (5.35):

$$\dot{\mathbf{x}} = \mathbf{F}(\mathbf{x}, \mathbf{z}, \mathbf{u}, \mathbf{d}) \tag{5.35a}$$
$$\mathbf{0} = \mathbf{G}(\mathbf{x}, \mathbf{z}, \mathbf{u}, \mathbf{d}) \tag{5.35b}$$

Where, **x** is differential states, **z** is algebraic states , **u** and **d** are inputs and disturbances respectively. Moreover, the explicit form of $\dot{\mathbf{x}}$ can be expressed as:

$$\dot{\mathbf{x}} = [\frac{T_{L1}}{dt} \quad \frac{T_{R1}}{dt} \quad \frac{T_{L2}}{dt} \quad \frac{T_{R2}}{dt} \quad \frac{T_{tank}}{dt}]^\top \tag{5.36}$$

Thus, the model described by DAEs has total 9 states instead of only 5 differential states like the former ODEs model. As a result, DEAs model makes optimal control problem (OCP) and nonlinear programming (NLP) larger, which is somewhat slower to be solved compared with ODEs model. However, DEAs model has its own corresponding advantages which makes it better to be used prior to ODEs. For example, after taking DAEs model, the relevant NLP problem can be resolved from any possible initial state conditions. Furthermore, the optimal solution convergence is significantly increased when the gradient information of NLP can not be obtained by CasADi.

5.2.4 Energy demand modeling

In order to simulate thermal energy storage optimal control problem in this thesis, an energy demand prediction model is necessarily needed. Hence, the prediction of energy demand requires a fine historical energy demand data, and it is also depends on the factors like year's season and day's time.

One of the energy demand used in this thesis is a simple assumed changeable energy demand profile, which is shown in Figure 5.3. It is easy to recognise that the expected hourly energy demand, Q_{demand}, varies with the time period, which is $1500\ kW$ at the first 12 hours and increases to $3500\ kW$ for the rest of the day. Meanwhile the expected hourly supply, Q_{supply}, always keeps constant at $2500\ kW$. Since this simple assumed energy demand profile is not realistic in practice, one of the practical energy demand profile is also modeled and implemented in this thesis.

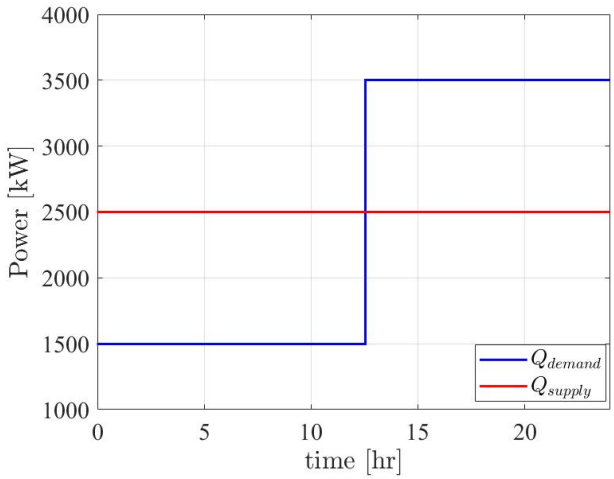


Figure 5.3: Simple supply-demand mismatch profile used in TES system

In practice, the intermittent renewable energy sources, primarily wind and solar energy can be used as the supplementary energy source besides conventional plants generated power or its waste energy to meet the supply-demand mismatch caused by the peak demand requirement. For example, solar energy is available during the daytime, thus solar power generation also occurs at this period, and conventional energy plant doesn't need to generate as much power as during the nighttime. Therefore, for energy demand consideration, the net load, which is the subtraction of solar generation from the total electric demand, drops during the midday. As aforementioned, it is caused by the large power input provided from solar resources. Thus the terminology "duck curve" is originated from this intermittent property of renewable energy, which is high available at the midsection and low at the right and left. The California "duck curve", which is the estimated net load values for 2020, is shown in Figure 5.4.

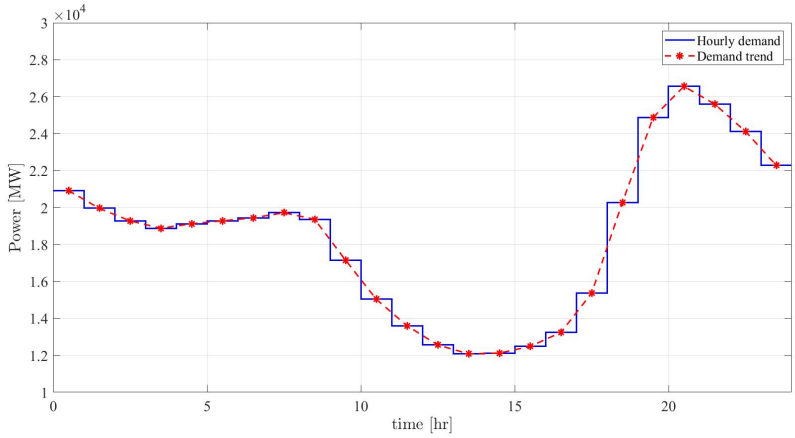


Figure 5.4: Estimated energy net load values for 2020 from CAISO Burnett,M. (2016)

In addition, the estimated energy net load is scaled down to study the requirement for commercial energy source from market when solar generation is no longer available during the nighttime. Scaling down of the net hourly demand data is fulfilled by implementing *min-max normalization*, which is one of the most common ways to normalize data. The normalization is done by converting the minimum vale of hourly demand data into 0, the maximum value into 100, and every other value into a number between 0 and 100. Then the normalized data set is re-scaled into the final data set, which has the range [1000,9000]. The formulation of min-max normalization is expressed as:

Min-max normalization:

$$Q_{d,n,i} = \frac{Q_{d,i} - Q_{d,min}}{Q_{d,max} - Q_{d,min}} \times 100 \tag{5.37a}$$

$$Q_{d,rs,i} = 1000 + Q_{d,n,i} \times 0.01 \times 8000 \tag{5.37b}$$

Where, $Q_{d,i}$ is the i-th hourly demand data. $Q_{d,min}$ and $Q_{d,max}$ is minimum and maximum value of hourly demand data respectively. $Q_{d,n,i}$ is the i-th normalized data within the range [0,100]. $Q_{d,rs,i}$ is the i-th re-scaled data within the range [1000,9000].

A hypothetical solar thermal energy supply profile is also assumed to directly heat the TES unit which is strictly following the practical solar energy availability of a certain sunny day in 2020. Thus, the expected energy demand based on California "duck curve" and expected direct solar heating profile is shown in Figure 5.5. Have to note that, under this profile, direct solar heating is just a supplementary energy source and the main supply energy is still coming from the industrial waste heat, where $Q_{supply} = 2500\ kW$.

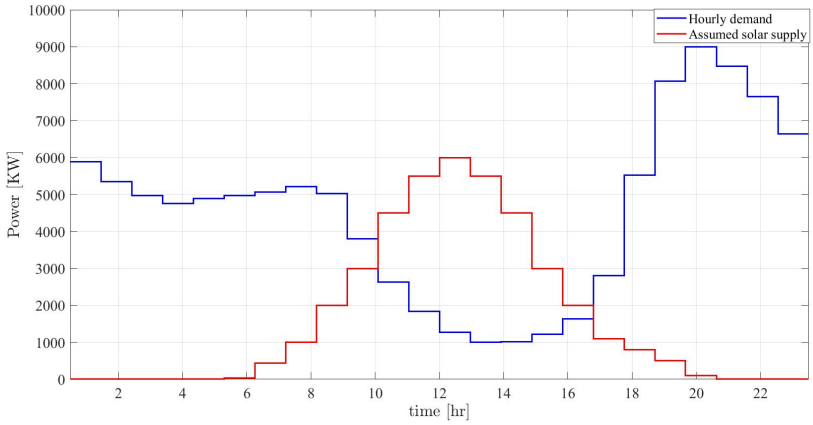


Figure 5.5: Scaled hourly demand and hourly hypothetical solar supply

Chapter 6

Optimal control of TES system by iNMPC

In this chapter, the optimal control objective of thermal energy storage system is fulfilled by implementing standard NMPC without considering uncertainties and multistage NMPC with considering uncertainties. The results are given in each case by figures and discussions for comparing thermal energy system with and without storage tank, with and without uncertainties.

6.1 Implementation details

The parameters will be used in the simulation of a simple thermal energy storage system are given in the Table 6.1, which arranged by specific areas for the sake of the simplicity.

6.2 Standard NMPC on a simple TES system without direct solar heating

In order to optimally control the simple thermal energy storage system with one supplier, one consumer and one TES tank, a simple assumed supply-demand mismatch scenario is adopted (Figure 6.1) and the concrete details are already discussed in **Chapter 5.2.4** . In this thesis, in order to achieve hourly control action of the finite-dimensional NLP problem within one certain day, prediction horizon is selected as $N_P = 24\ h$. Thus, NLP is formulated by implementing 24 finite elements.

A standard NMPC is applied to adjust the the flow of q_{R2} as shown in Figure 5.2 which is the demand-side stream flowing into the cold side of heat exchanger R_2. Moreover, supply-side inflow q_{L1}, storage outlet flows q_{R1} and q_{L2} are kept constant and assigned

Term	Explanation	Value	Unit
U_{hex}	Overall heat loss coefficient	0.5	$[KW/m^2K]$
A_{hex}	Heat transfer area	300	$[m^2]$
V_{hex}	HEX volume	0.5	$[m^3]$
U_{tank}	Overall heat loss coefficient	$5{\times}10^{-4}$	$[Kw/m^2K]$
A_{tank}	Surface area	100	$[m^2]$
V_{tank}	Storage tank volume	1000	$[m^3]$
ρ	Density	1000	$[kg/m^3]$
c_p	Specific heat capacity	4.186	$[kJ/kgK]$
T_1	Supply stream temperature	95	$[^\circ C]$
T_2	Demand stream temperature	20	$[^\circ C]$
T_{sur}	Surrounding temperature	15	$[^\circ C]$
P_m	Peak heating source cost	10^{-3}	...
P_t	Direct tank heating cost	5×10^{-6}	...

Table 6.1: System model parameters

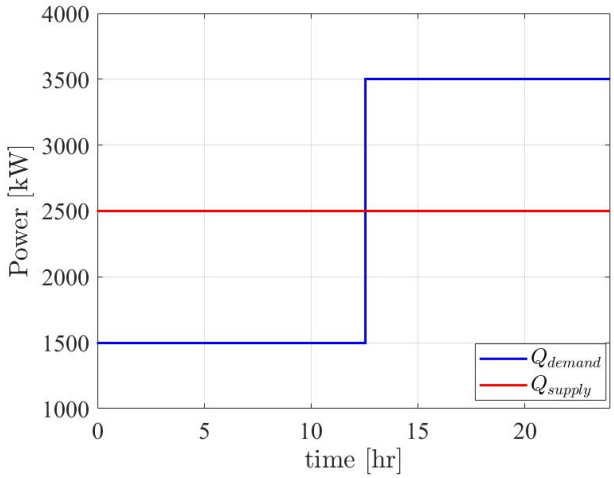


Figure 6.1: Simple supply-demand mismatch profile used in TES system

values as $50\ L/s$. The direct heating Q_{tank} or Q_{solar} is neglected here. Therefore, within inputs u just q_{R2}, Q_{market} and Q_{dump} have to be manipulated. When energy demand exceeds energy supply, there will be a necessary need to purchase extra commercial energy from the market. Hence, the optimal control objective of thermal energy storage system is to minimize the use of expensive peak heating sources and further minimize the cost caused by this expensive commercial energy.

6.2.1 Optimization problem

The objective function of this optimization purpose can be formulated as:

$$\min_{\mathbf{x},\mathbf{u}} \ \phi(\mathbf{x},\mathbf{u}) = P_m Q_{market} \tag{6.1}$$

In order to avoid ill-conditioned singular control problems, the additional term can be added to the original linear objective (6.1), which is a quadratic cost with a penalty parameter $P_u \ll 1$ and known as *regularization term*. The purpose of adding regularisation term is to penalize the other two control inputs. Thus, the optimization problem becomes:

$$\min_{\mathbf{x},\mathbf{u}} \ \phi(\mathbf{x},\mathbf{u}) = P_m Q_{market} + P_u(q_{R2}^2 + Q_{dump}^2) \tag{6.2}$$

In order to fulfill this optimization purpose, constraints which may relate to physical limitations on certain variables, considerations of safety and some performances have to be identified. The identification of system constraints is completed in these two categories, equality and inequality constraint.

1. **Equality constraints:** These are hard constraints that the system has to strictly satisfy for optimal and stable operation.

   - **System model:** In this TES system, DAEs system model is en equality constraint, and this equality constraint can not be violated in the optimization process. If it is violated, the stability of system will be broken and become unstable. The fully-implicit form of system model can be expressed as:

     $$\mathbf{F}(\dot{\mathbf{x}},\mathbf{x},\mathbf{z},\mathbf{u},\mathbf{d}) = 0 \tag{6.3}$$

   - **Consumer demand satisfaction:** In practice, the energy required at the demand-side is not constant and the variations of demand depend on factors like time and season. Hence, the energy demand profile has to be specified at the concrete time. So as to achieve the satisfaction for energy from the consumer, this can be considered as the sum of enthalpy gain of demand stream across the heat exchanger and additional peak hour energy from the market.

     $$Q_{demand} = Q_{market} + \rho c_p q_{R2}(T_{R2} - T_2) \tag{6.4}$$

     Where, Q_{market} is the commercial energy purchased from the market to be used as peak heating source.

   - **Supply side satisfaction:** On the return stream after exiting the heat exchanger, it is necessary to dump heat in order to decrease the return temperature to a demanded level, which is denoted as Q_{dump}. Therefore, the supply side satisfaction can be expressed as:

     $$Q_{supply} = Q_{dump} + \rho c_p q_{L1}(T_1 - T_{L1}) \tag{6.5}$$

- **Variable limitations:** As above-mentioned control input conditions, the rest of them expect from q_{R2}, Q_{market} and Q_{dump} are kept constant.

$$\begin{bmatrix} q_{L1} \\ q_{R1} \\ q_{L2} \\ Q_{tank} \end{bmatrix} = \begin{bmatrix} 50 \\ 50 \\ 50 \\ 0 \end{bmatrix} \tag{6.6}$$

Therefore, the equality constraint can be formulated as:

$$c_{eq} = \begin{bmatrix} \mathbf{F}(\dot{\mathbf{x}}, \mathbf{x}, \mathbf{z}, \mathbf{u}, \mathbf{d}) \\ Q_{demand} - Q_{market} - \rho c_p q_{R2}(T_{R2} - T_2) \\ Q_{supply} - Q_{dump} - \rho c_p q_{L1}(T_1 - T_{L1}) \\ q_{L1} - 50 \\ q_{R1} - 50 \\ q_{L2} - 50 \\ Q_{tank} \end{bmatrix} = 0 \tag{6.7}$$

2. **Inequality constraints:**

- **State bounds:** According to assumption 9 in **section 5.1**, the temperature of TES tank is upper bounded as $100\ ^\circ C$ in order to avoid boiling. Then it is lower bounded as $30\ ^\circ C$ so as to prevent bacterial growth inside the tank. The rest of temperatures can also be bounded as following the above-mentioned way, but the lower bounds of them are set as $0\ ^\circ C$.

$$\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 30 \end{bmatrix} \leq \begin{bmatrix} T_{L1} \\ T_{R1} \\ T_{L2} \\ T_{R2} \\ T_{tank} \end{bmatrix} \leq \begin{bmatrix} 100 \\ 100 \\ 100 \\ 100 \\ 100 \end{bmatrix} \tag{6.8}$$

- **Input bounds:** For volumetric flow rates, they have to be constrained within their physical limits which are valve opening from 0% to 100%. And it is necessary to avoid bounding flow rates so low in order to prevent fast fouling of heat exchangers. Since manipulated variables have saturation limits, so it is not expected to obtain optimality when inputs exceed their saturation. In this thesis, the volumetric flow rate of q_{R2} is manipulated in the range from 0 to $50\ L/s$.

$$0 \leq q_{R2} \leq 50 \tag{6.9}$$

By adjusting q_{R2}, how much heat can be extracted from the TES tank can be determined. Furthermore, Q_{market} and Q_{dump} are bound from 0 to positive infinity.

$$\begin{bmatrix} 0 \\ 0 \end{bmatrix} \leq \begin{bmatrix} Q_{market} \\ Q_{dump} \end{bmatrix} \leq \begin{bmatrix} +\infty \\ +\infty \end{bmatrix} \tag{6.10}$$

Therefore, the inequality constraint can be formulated as:

$$\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 30 \\ 0 \\ 0 \\ 0 \end{bmatrix} \leq \begin{bmatrix} T_{L1} \\ T_{R1} \\ T_{L2} \\ T_{R2} \\ T_{tank} \\ q_{R2} \\ Q_{marlet} \\ Q_{dump} \end{bmatrix} \leq \begin{bmatrix} 100 \\ 100 \\ 100 \\ 100 \\ 100 \\ 50 \\ +\infty \\ +\infty \end{bmatrix} \tag{6.11}$$

Finally, the dynamic optimization problem of this case can be described as minimizing the objective function (6.1), which subject to the system dynamics represented by the model equations (5.24) -(5.25) and the operating constraints (6.7) and (6.11). The complete formulation of the dynamic optimization problem is given in **Appendix C** (8.2). In addition, in order to study the effect of storage tank, thermal energy system without a storage tank is also implemented in this thesis. The topology illustration, system description and optimization problem of thermal energy system without storage tank are given in **Appendix F** (8.2).

6.2.2 Results

In this thesis, the assumed expectation is to charge the TES tank during the first 12 hours and completely discharge during the rest of the day. However, there is uncertainty about q_{R2}, Q_{dump} and Q_{market} to achieve this expected goal. Hence, the TES tank is initially assumed under optimal steady state, and by solving the steady state optimization problem the initial values of states and inputs can be obtained. The initial states obtained from steady state optimization will be used as current states to trigger the standard NMPC procedure. After implementing standard NMPC, the optimal control problem (OPC) results are given in Figure 6.2.

1. **Period of $Q_{demand} < Q_{supply}$:**

   - It is the charging period for TES tank. Its temperature, T_{tank} rises until consumer demand excesses the supply.
   - As the presence of TES unit (TES tank), surplus thermal energy can be stored in the storage tank when energy demand is low. However, TES storage capacity decreases with more and more energy is stored which leads to reduction of energy transfer from supply side to TES unit and the increase of surplus heat dumping. The highest Q_{dump} occurs at when TES unit is fully charged. Although Q_{dump} increases, it is still much lower than thermal energy grid without storage tank, because most of the surplus energy transferred can be stored in the tank and reallocated. The surplus heat dumping profile of thermal energy system without energy storage tank is shown in Figure 6.3.

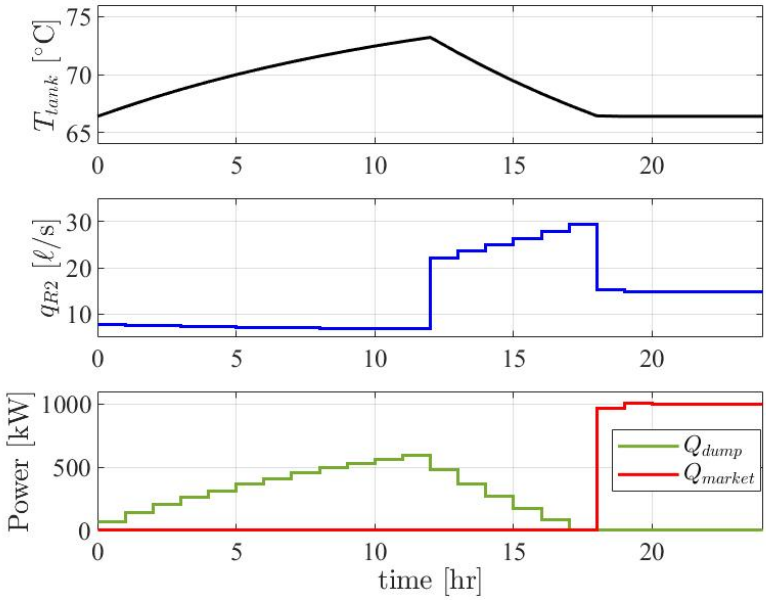


Figure 6.2: OCP result of one supply one demand TES system

- q_{R2} is manipulated by standard NMPC to keep low value to fulfill and ensure low consumer demand.
- The energy stored in TES tank is sufficient to meet energy demand in this period. Thus, there is no need to purchase extra commercial energy from energy market.

2. **Period of $Q_{demand} > Q_{supply}$:**

   - Consumer demand suddenly increases at 12th hour, which leads to the discharge of TES unit and storage tank starts to cool down. The discharging rate of TES tank is higher than its charging rate. Q_{dump} also decreases as the cooling down of TES tank.
   - Control input q_{R2} is manipulated by the standard NMPC controller, and its variation is also the reflection of Q_{demand} variation. After the future sudden increase of Q_{demand} is detected by the controller, q_{R2} is manipulated to increase to satisfy the increased result of Q_{demand} until the TES tank is completely nearly discharged. Before the TES tank is completely discharged, q_{R2} increases gradually with a step-wise mode, as the decreasing T_{tank}. When TES tank is completely discharged, there is no more energy transferred from heat exchanger hot side to cold side. Thus, q_{R2} is decreased but not equal to 0, since Q_{demand} is stil exist. In general, the purpose of q_{R2} is to satisfy peak energy demand and try as much as possible to fill out the mismatch by using stored TES energy to avoid purchasing commercial energy.

- The complete discharge of TES tank occurs at 19th hour when the T_{tank} decreases to an equilibrium temperature which equals to initial TES temperature before charging.
- Although the complete discharge happens at 19th hour, stored TES energy is no more able to satisfy consumer demand at 18th hour. This is the reason why q_{R2} sharply decreased at this moment. Since, the remaining stored TES energy no longer can satisfy the consumer demand, standard NMPC takes action to increase purchased commercial energy Q_{market} to fill out the mismatch. When TES tank is completely discharged, a step increase in Q_{market} is taken to match the mismatch caused by the consumed last remaining stored TES energy.
- The TES system purchases commercial energy Q_{market} only for the last 6 hours when the stored TES energy is not sufficient to fulfill the consumer demand. The amount of purchased Q_{market} is also lower than the thermal energy system without storage tank. The result of the latter case is given in Figure 6.3.

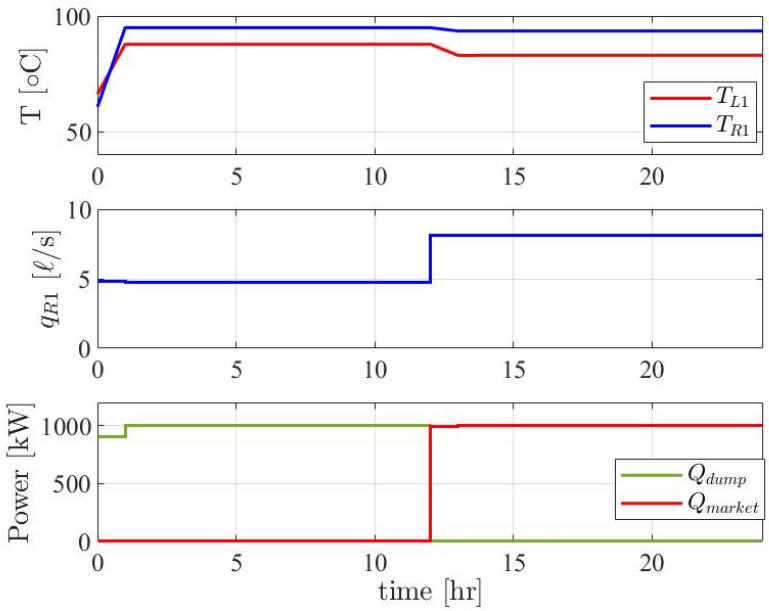


Figure 6.3: OCP result of one supply one demand thermal energy system without storage tank

In order to study the effects on storage charging and discharging rate, the multiple standard NMPC simulations with diverse storage sizes such as $500m^3$, $1000m^3$ and $1500m^3$ are taken. The results of T_{tank} with three different V_{tank} are shown in Figure 6.4 and Figure 6.5.
By analysing the results shown in these two figures:

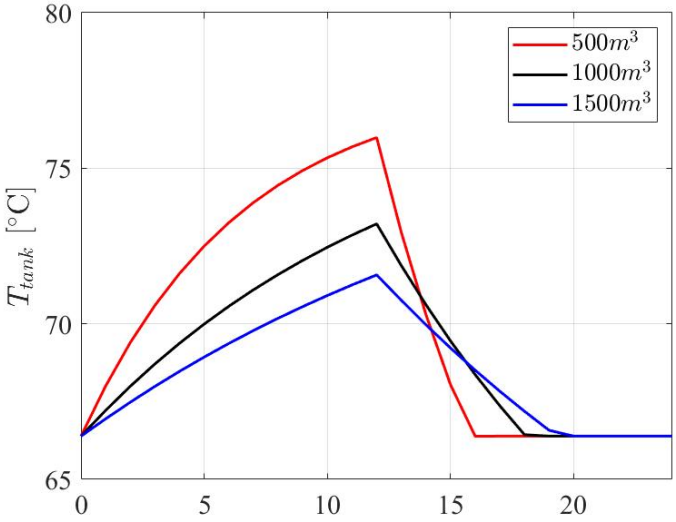


Figure 6.4: Comparison of T_{tank} with diverse V_{tank}

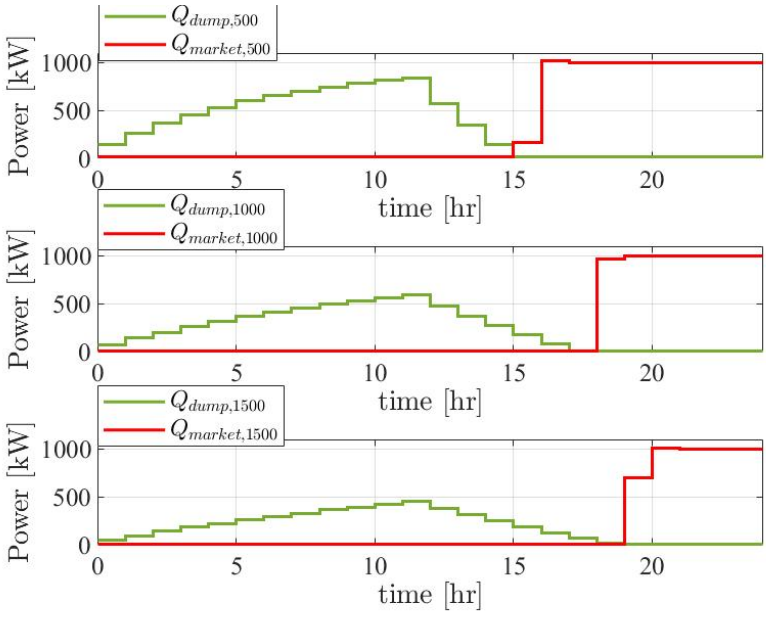


Figure 6.5: Comparison of Q_{dump} and Q_{market} with diverse V_{tank}

1. TES storage volume, V_{tank} indeed has effect on tank temperature profile and dis-

charging rate.

2. Under sufficient supply, smaller V_{tank} results in higher T_{tank} and faster charging rate.

3. Under peak energy demand, smaller V_{tank} also results in faster discharging rate.

4. Larger V_{tank} provides larger storage capacity and thus less heat is dumped during the low demand period. Furthermore, since the discharging rate of larger V_{tank} is slower compared with smaller V_{tank}, the TES can transfer heat to the consumer for longer period. Thus the period of commercial energy demanding is short for large V_{tank}.

5. Since the design of thermal energy storage tank is not the topic of this thesis, there is no consideration for comparison of most economic case under different tank volumes with tank construction cost. However, it can be expected that the tank cost will play a significant role under large storage volume, when the design optimization is performed.

6.3 Standard NMPC on a simple TES system with direct solar heating

Since the cost of solar energy is quite cheap and the enforcing of renewable energy is the trend for protecting ecosystem, in this case solar energy is used as a complementary heating source for TES system during the available period to reduce the requirement of expensive commercial energy. In this case the direct solar heating source is denoted as Q_{tank} and it can also be interchangeably expressed as Q_{solar}, since the Q_{tank} is coming from direct solar heating. The energy demand profile follows the re-scaled California "duck curve" which is shown in Figure 5.5.

6.3.1 Optimization problem

Therefore, the cost function which has to be minimized in this case has an additional term of direct solar heating cost $P_t Q_{tank}$.

$$\min_{\mathbf{x},\mathbf{u}} \ \phi(\mathbf{x},\mathbf{u}) = P_m Q_{market} + P_t Q_{tank} \tag{6.12}$$

The standard NMPC is enforced to control not only q_{R2}, Q_{market} and Q_{dump}, but also Q_{tank}. Thus, the control input of this case is $u = [q_{R2}, Q_{market}, Q_{dump}, Q_{tank}]$. Since, there is an additional degree of freedom, the input constraints have been changed, which is shown in equation (6.13). The state constraints are still identical with which used in the case one in equation (6.8).

$$\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \leq \begin{bmatrix} q_{R2} \\ Q_{market} \\ Q_{dump} \\ Q_{tank} \end{bmatrix} \leq \begin{bmatrix} 50 \\ +\infty \\ +\infty \\ Q_{solar,k} \end{bmatrix} \tag{6.13}$$

Where, $Q_{solar,k}$ is the predicted hypothetical solar energy supply given in Figure 5.5. The entire form of dynamic optimization problem of this case is given in **Appendix D** (8.2). The initial values of states are obtained by executing steady-state optimization problem, and these initial values are used as the starting points of standard NMPC. After implementing simulation of this simple case with direct solar heating, the following results are obtained as shown below.

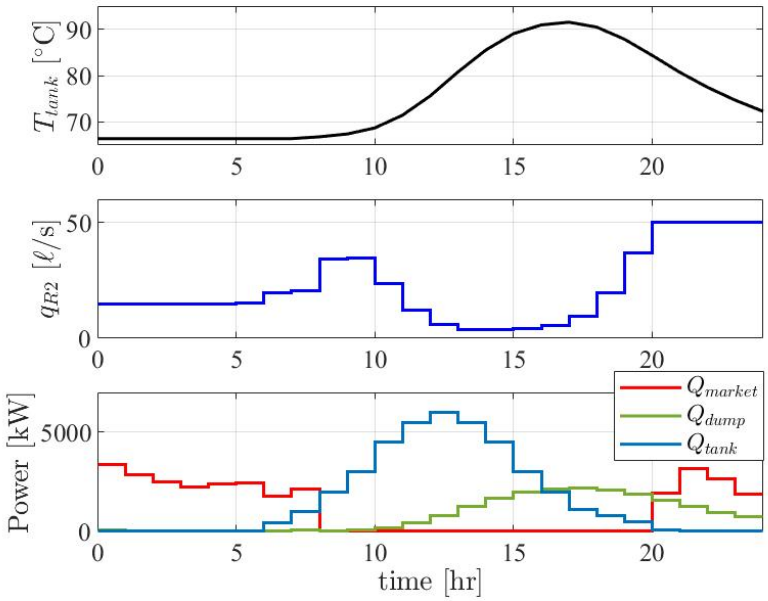


Figure 6.6: OCP result of one supply one demand TES system with direct solar heating

6.3.2 Results

By analysing the result given in Figure 6.6, it is easy to know that the result is varying with respect to the supplied direct solar source availability:

1. T_{tank} is constant during the first 7 hours and the solar source availability is quite low in this period. Thus, a gap arises between the total energy demand and the total energy supplied into the TES tank. In order to cater this gap, commercial energy is purchased from the energy market at the beginning.

2. T_{tank} starts to rise rapidly, when the hourly demand decreases and it is no more high as which during the nighttime and solar source availability increases with the daytime. Since, the cost of solar energy is much lower than that of commercial

energy ($P_t \ll P_m$), TES tank stores solar energy which is then used for the nighttime's peak-heating demand. Moreover, the total energy supply has exceeded the total energy demand, so there is no need to purchase commercial market energy.

3. T_{tank} reaches its maximum value at 17th hour. Then TES tank starts to discharge to fulfill the increasing demand requirement. When the peak demand period comes, commercial energy is purchased again to meet the demand. However, the total requirement of commercial energy after 20th hour is less than that before the daytime.

4. $Q_{dump} = 0$ at the beginning, since the total energy supply cannot fulfill the total energy demand. However, it starts to increase along the daytime to ensure much cheaper solar energy can be stored as much as possible in TES tank. Because the surplus energy supply Q_{supply} is set as constant and supplied into the tank during the entire day. Then it decreases during the night peak demand period.

Since standard NMPC recomputes dynamic optimization problem at every time step along the prediction horizon $N_P = 24h$, the future peak demand period can be anticipated by controller and take necessary actions to fulfill the upcoming peak demand.

6.4 Multistage NMPC on a simple TES system with uncertainty

In practice, the execution of standard NMPC does not take into consideration the uncertainty at the optimization level. For example, the uncertainty in the heat supply and demand profile, which may be caused by unperdicted variations in each of the profiles, will lead to problems like plant–model mismatch, operating or safety constraints violation and infeasible optimization problem. In this thesis and especially in reality, the uncertainty in heat supply and demand profile with expected corresponding profiles may cause problems like consumer demand mismatch, and further lead to extra energy purchasing or unnecessary energy wasting. Moreover, it also can cause unstable operation and even safety issues.

In this thesis, the uncertainty of TES system may be caused by:

1. Inaccurate modeling or approximated modeling, which leads to plant-model mismatch.

2. Unknown disturbances which interfere supply and demand temperature T_1 and T_2, ambient temperature T_{sur}, and also supply stream flow rate q_{L1}.

3. The actual supply-demand profile is deviated from the profile implemented in the NMPC control process.

4. The changes in design parameters which for example caused by the fouling of heat exchanger leads to the change in overall heat transfer coefficient.

6.4.1 Modeling of the uncertainties

Therefore, in this case a scenario-based multistage NMPC which uses a discrete scenario tree to formulate the future propagation of the uncertainty in the prediction horizon $N_P = 24\ h$ is implemented. Before implementation it is necessary to model the uncertainties to set up the scenario tree, which can represent how the uncertainty in this case influences the propagation of states over the prediction horizon and the optimization problem is solved over the entire scenario tree.

As discussed in **Chapter 5.2**, the uncertainty in this case is considered in the form of disturbances in supply inlet temperature T_1, demand inlet temperature T_2 and ambient temperature T_{sur}. Before implementing multistage NMPC, it is important to mention that all the states of the TES system can be exactly measured. In this case, only the uncertainty in the supply side and demand side inlet temperatures are taken into account, ($\mathbf{p} = [T_1, T_2]$), since the ambient temperature can be measured. Let P denote the uncertainty set to which the uncertain parameter is known to belong. In this case, the uncertainty in $T_i \in P_i$ to be considered equally distributed within $T_1 \in [90, 100]$ and $T_2 \in [15, 25]$. Therefore, the unsertainty set $P = P_1 \times P_2$ is given as box uncertainty set. Then $M = 9$ discrete uncertainty realizations are considered that corresponds to the combination of maximum, minimum and nominal values of supply and demand inlet temperature to cover the complete uncertainty space. The uncertain parameter realization is given in Table 6.2.

	Nominal value	Min nominal max	$[^\circ C]$
T_1	95	90 95 100	$[^\circ C]$
T_2	20	15 20 25	$[^\circ C]$

Table 6.2: Discrete realizations of **p** used in the simulation

Moreover, the range of each actual inlet temperature is identical with corresponding uncertainty set range P_i. Thus, it can be expressed as:

$$T_1, actual \in [90, 100] \;\; T_2, actual \in [15, 25] \tag{6.14}$$

The dynamic scenario selection is done by implementing conventional BOX method which plotting the two parameters against each other and it is built as `scenpara()` in MATLAB and automatically returns the scenarios. The illustration of uncertainty space is shown in Figure 6.7. Scenario tree, which has $S = M^{N_R} = 9$ total scenarios, with $N_R = 1$ robust horizon and $M = 9$ models is illustrated in Figure 6.8.

6.4.2 Optimization problem

The optimization problem solved for ith scenario and jth time step can then be stated as:

$$\min_{\mathbf{x_{i,j}}, \mathbf{u_{i,j}}} \phi(\mathbf{x}, \mathbf{u}) = P_m(Q_{market})_{i,j} + P_t(Q_{tank})_{i,j} + P_u\big((q_{L2})^2_{i,j} + (q_{R2})^2_{i,j}\big) \tag{6.15}$$

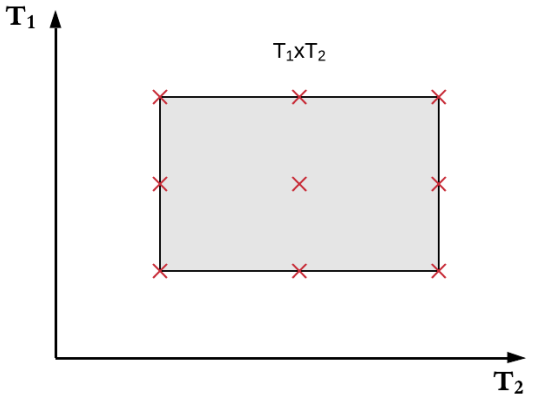


Figure 6.7: Uncertainty subspace and the possible M models for the scenario tree denoted as ×

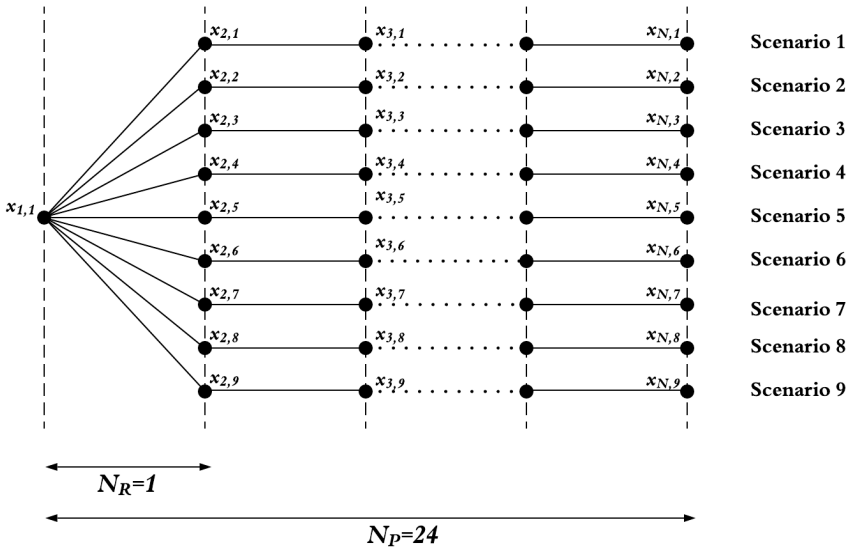


Figure 6.8: Scenario tree of this case

Where,

- ith scenario: $i \in \{1, ..., M\}$, jth time step: $j \in \{0, ..., N-1\}$.
- Dynamic system model and state boundaries are identical with a simple TES system

without direct solar heating .

- Control input is $u = [q_{L2}, q_{R2}, Q_{tank}, Q_{market}]$ and bounds are given in below.

$$\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \leq \begin{bmatrix} q_{L2} \\ q_{R2} \\ Q_{tank} \\ Q_{market} \end{bmatrix} \leq \begin{bmatrix} 50 \\ 50 \\ 10 \\ +\infty \end{bmatrix} \tag{6.16}$$

  Where, direct solar heating source Q_{tank} is set as a certain small number, in this case is $10kW$, to ensure there is a possibility to purchase commercial energy Q_{market} since the main core of this thesis is to minimize its cost, when the energy stored in the TES unit is not sufficient to meet the peak demand.

- $P_u(q_{L2}^2 + q_{R2}^2)$ is the additional regularization term added to the linear objective $P_m Q_{Market} + P_t Q_{tank}$ to avoid ill-conditioned singular control problems. $P_u \ll 1$ is the quadratic cost which penalizes the two other sets of control variables.

- In order to avoid the the resulting large optimization problem, robust horizon is taken in this case and which is set as $N_R = 1$ and $N_R \ll N_P$. Thus, this problem has total $S = M^{N_R} = 9^1$ discrete scenarios in the scenario tree, which is already given in Figure 6.8.

Note that each node in the scenario tree contains all the states and control inputs in the model $x = [T_{L1}, T_{R1}, T_{L2}, T_{R2}, T_{tank}]$ and $u = [q_{L2}, q_{R2}, Q_{tank}, Q_{market}]$. The complete form of optimal control problem of scenario-based multistage NMPC is given in **Appendix E** (8.2).

In order to compare the performance of standard NMPC and multistage NMPC when taking into account plant-model mismatch, the actual values of T_1 and T_2 are considered as $T_{1,actual} = 99°C$ and $T_{2,actual} = 18°C$ for the plant, while the NLP optimizer or controller is still programmed for expected mean temperatures $T_1 = 95°C$ and $T_2 = 20°C$.

6.4.3 Results

The simulation result of a simple TES system with consideration of uncertainties, $\mathbf{p} = [T1, T2]$ is illustrated in Figure 6.9. Note that, the supply-demand profile used in this case is the simple assumed supply-demand profile, which is given in Figure 5.3.

By analysing the result, it is easy to know that:

1. **Period of $Q_{demand} < Q_{supply}$:**
   - This period is the charging period, the surplus energy from the industrial waste heat is stored in the storage tank until $12h$.
   - Since energy stored in the storage tank is sufficient to meet the current low demand, control inputs q_{L2} and q_{R2} are manipulated within their operational bounds $[0, 50l/s]$ to ensure energy can successfully transferred to the demand side.

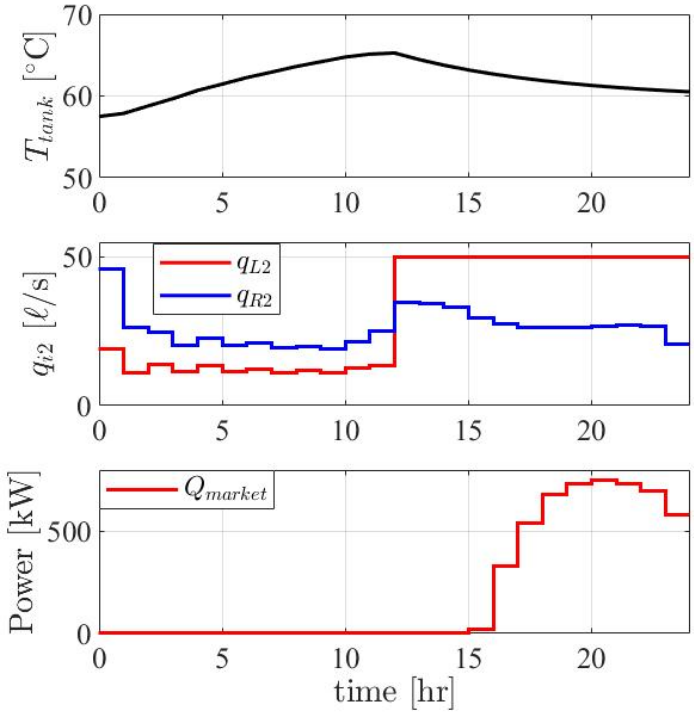


Figure 6.9: OCP result of one supply one demand TES system with uncertainty

- Therefore, there is no need to purchase extra commercial energy Q_{market} from energy market.

2. **Period of $Q_{demand} > Q_{supply}$:**

   - This period is the discharging period, the energy stored in the storage tank have to be extracted to meet the suddenly increased energy demand at $12h$.
   - The sudden increase of Q_{demand} can be detected by the controller, then in order to satisfy this peak demand control input q_{L2} is fully opened to ensure energy stored in the storage tank can be promptly transferred to the demand side.
   - Before the energy stored in the storage tank is completely discharged, extra commercial energy Q_{demand} is purchased from the energy market to fill out the supply-demand mismatch since available Q_{tank} is quite low and controller can anticipate the total energy stored in the TES tank is not sufficient to meet the demand.

6.4.4 Standard NMPC on a simple TES system with uncertainty

In order to compare the performance of scenario-based multistage NMPC with standard NMPC with the consideration of uncertainties, the optimal control problem solved by multistage NMPC is completed again by using standard NMPC.

As the presence of uncertainties along the whole prediction horizon, $N_P = 24h$ and if the disturbances are very large, the state constraint limit may be violated for all feasible control inputs especially for T_{tank}, and no feasible point may exist and the standard NMPC would easily run into infeasible problems. This is an unacceptable situation. In order to avoid this phenomena, the state constraints (bounds) have to be soften by using *slack variables* (Foss,B. and Heirung,T.A.N.(2013)). After implementing slack variables, the optimization problem solved by standard NMPC becomes:

$$\min_{\mathbf{x},\mathbf{u}} \ \phi(\mathbf{x},\mathbf{u}) = P_m Q_{market} + P_t Q_{tank} + P_u\big(q_{L2}^2 + q_{R2}^2\big) + \rho^\top \epsilon \tag{6.17}$$

Where, ϵ is the positive slack variable, ρ is the tuning parameter (penalty) and the exact value of the penalty depends on the problem, but in general it selected as very large. Thus in this sub-case, it is assigned as $\rho = 10^6$. The complete form of optimal control problem of this sub-case is given in **Appendix F** (8.2).

$$\epsilon \in \mathbb{R}^{n_x} \geq 0, \quad \rho \in \mathbb{R}^{n_x} \geq 0 \tag{6.18}$$

The result comparison of using standard NMPC and multistage NMPC is given in next chapter.

Chapter 7

Results and discussion

In this thesis, the infinite-dimensional problems are converted into finite-dimensional NLP problems by implementing direct collocation method where the third-order Radau collocation ($K = 3$) is applied to approximate the state equations. The states, control inputs and the uncertain parameters (which is only considered in case three) are all discretized into finite elements. However, these discretized control inputs and uncertain parameters are piece-wise constant in each finite element. The detailed information of the collocation-based discretization method has been discussed in **Chapter 3** already, for more information **Chapter 10** of Biegler,L.T. (2010) is recommend.

In this thesis, prediction horizon is chosen as $N_P = 24h$. The finite-dimensional NLP problem is formulated by adopting 24 finite elements. The purpose is to take the control action every one hour within one single day $T = 24h$. In addition, for multistage NMPC, which takes into account uncertainty, the evolution of uncertain parameters is also every one hour. The scenario of uncertainty is selected by using the conventional BOX scenario selection method, which is based on three different discrete realizations of the uncertainty for each uncertain parameter of two uncertainties for every stage of the multistage NMPC problem. In order to reduce the computational expense, robust horizon of $N_R = 1$ is assumed, which finally gives 9 total scenarios in the multistage NMPC problem.

In order to study the effect of thermal energy storage tank in the simple thermal energy storage system, standard NMPC is implemented to compare the effect caused by TES tank, where in this two cases with and without storage tank no uncertainty is considered. All parameters and supply demand profile are identical in this two cases except from the system itself and its corresponding dynamic model. Supply demand profile used in this two cases is a common assumed mismatched profile. TES system with direct solar heating is also considered to make the case closer to the reality condition, since the solar supply is intermittent. The demand profile used in this case is the re-scaled California ”duck curve” from CAISO (Burnett,M. (2016)) with assumed solar energy supply based on real solar availability tendency of a random day of year 2020. After taking into account uncertainty, the performance of standard NMPC and multistage NMPC is compared by using actual

values of uncertainties, T_1 and T_2 in the the plant simulation, for both the standard and multistage NMPC methods.

The application of iNMPC strategies are only considered for dynamic operation of TES and not for edge-cases such as TES is empty at the beginning. Moreover, initial conditions in this thesis are not given directly. Thus, the steady-state optimization problems are solved to find out the initial conditions for each case.

The NLP problem in this thesis is modeled by using MATLAB (version R2019b). IPOPT is used as the NLP solver, which uses interior-point algorithms to solve the NLP problems.

7.1 Storage vs. No storage

Since the case with TES storage tank is one of the main studied case of this thesis, the specific result discussions have already been discussed in **Chapter 6.2**. Before comparing the results, it is necessary to introduce the system and the system's model of thermal energy system without TES tank. These necessary information is given and able to find in **Appendix G** (8.2). The case without storage tank uses a single heat exchanger to directly couple the supply and demand plant, the illustration which is given in Figure 8.1. The results of a standard NMPC formulation applied to both cases—with and without storage tank under the same supply-demand profile (Figure 7.1) are shown in Figure 7.2. In addition, there is no plant–model mismatch is considered in these two cases.

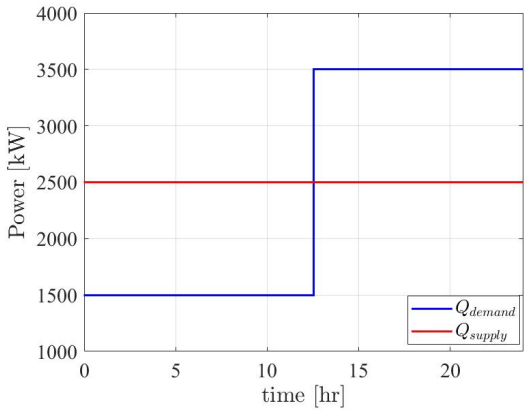


Figure 7.1: Simple supply-demand mismatch profile used in TES system

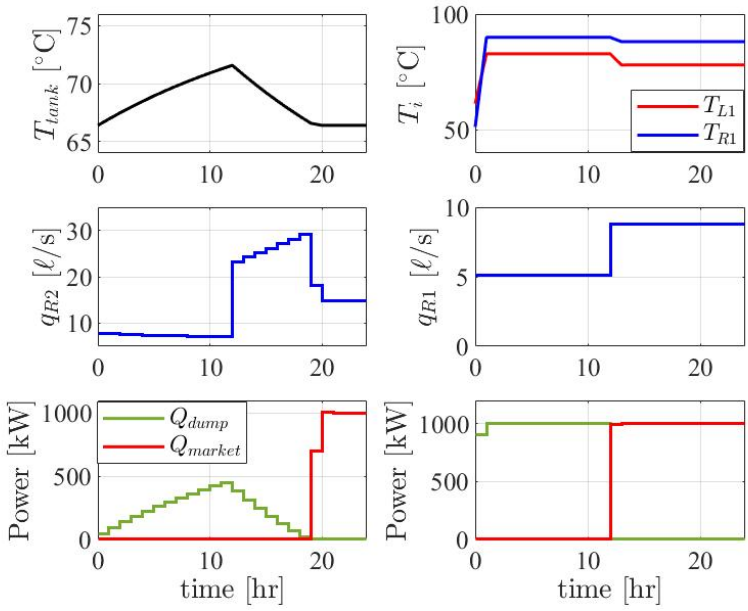


Figure 7.2: Results of the standard NMPC applied to the cases with and without TES tank

In the case with no storage:

- When the energy demand is lower than the energy supply, the surplus heat is dumped and when the energy demand is higher than the supply, the extra heat is required to be purchased for peak-heating subject to the operating constraints.

- The amount of dumped heat and purchased commercial energy is equal to the difference between supply and demand profile, which is $1000kW$.

- Since the supply energy rate q_{L1} is set as constant, ($q_{L1} = 50\ [l/s]$) and overall heat transfer rate through the heat exchanger is determined by q_{L1}, the rate of demand side q_{R1} is impossible to exceed q_{L1}. Until $12h$, q_{R1} is constant due to the demand can be easily satisfied by supply. After $12h$, q_{R1} increased rapidly to take the action to fulfill demand requirement.

Comparison:

- The application of TES tank significantly decreases the amount of dumped heat when the demand is low. These surplus heat is used to charge up the TES tank, which is then used for peak-demand period. This is apparent from tank temperature variation result T_{tank} as shown in Figure 7.2, T_{tank} rises during off-peak periods and falls when the demand exceeds the supply.

- Only after the energy charged inside the TES tank is completely discharged, the expensive peak-heating commercial market energy will be purchased to use. Thus, the dependence on commercial market energy is much lower for the simple thermal energy system with storage tank.
- Overall, TES tank leads to significant reduction in the dumped heat and peak-heating requirements.

Therefore, applying TES storage tank is a good method for meeting peak-demand, since it has the storing capability to significantly reduce Q_{dump} and Q_{market}. However, the maintenance and the construction of TES tank may cause partially cost.

7.2 Without direct solar heating vs. with direct solar heating

Since the increasing interest and application requirements of renewable energy, and also in order to study the case relevant to real energy supply condition in one day period, direct solar heating is added to the case with TES storage tank under standard NMPC control algorithm. The only two differences between these two cases are TES tank is directly heated by solar energy and the energy demand and assumed solar supply profile, which is already given in **Chapter 5.2.4** in Figure 5.5. Note that, the surplus energy supply, Q_{supply}, is still applied in this case with direct solar heating and which is also constant ($Q_{supply} = 2000[kW]$) in the 24 hours. The results of these two cases are shown in the following Figure 7.3.

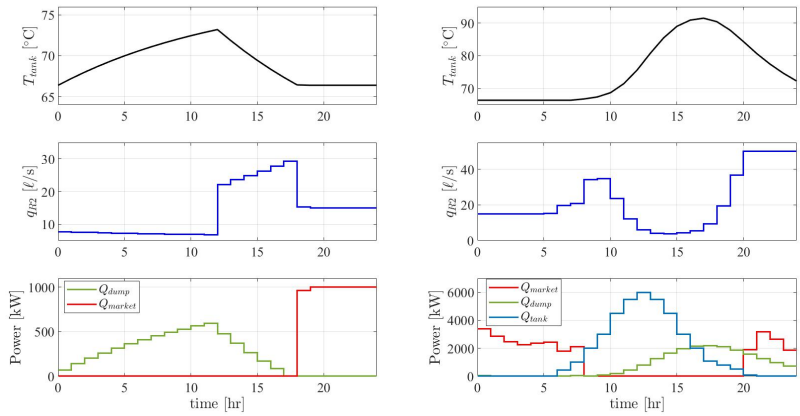


Figure 7.3: Results of the standard NMPC applied to the cases with and without thermal direct solar heating to TES tank

The specific discussions of two case's results have been already discussed in **Chapter 6.2**

and **Chapter 6.3**. Thus, the remainder is to make comparison between the results of these two cases.

Comparison:

- As the presence of direct solar heating energy source Q_{tank} or Q_{solar}, the requirement of Q_{market} follows the real availability of the solar energy. Q_{market} is highly needed at the nighttime when solar energy is unavailable. Therefore, the result of power variation is the reflection of intermittent property of solar energy.
- In addition, the required amount of Q_{market} is much higher than that is without direct solar heating, which makes sense since the re-scaled California duck curve is still much higher than the assumed demand in Figure 7.1.
- At the nighttime, the demand is higher than the total supply, thus Q_{market} is purchased to meet the demand requirement. The result of this part is the same as that of case without direct solar heating after $20h$. When the day is coming, T_{tank} increases rapidly where its gradient increases with the availability of solar energy. Before the night is coming again, TES tank stores solar energy so much as possible for the future peak-heating use. This reflects that with the solar energy supply, the variation of T_{tank} is closer to the reality.
- It is clear to recognise from the upper right result of T_{tank}, the stored solar energy in TES tank is still available at $24h$ and it can be expected that the solar energy stored in TES tank today can supply energy to the demand side for the nighttime of next day. Thus, stored solar energy is able to reduce the requirement of Q_{market} before tomorrow's daytime in practice.
- The variation of Q_{dump} is following the intermittent property of solar energy and it can also be seen as the direct result of T_{tank} variation. Since the total energy supply exceeds energy demand during the daytime, TES tank needs to dump some amount of heat to satisfy the operation constraints.

7.3 Multistage NMPC vs. standard NMPC

The results of the standard NMPC and multistage NMPC applied to the TES system are compared in this section. In both controllers, the TES tank temperature , T_{tank}, are limited below $70°C$. The results of standard NMPC without additional slack variables and multistage NMPC are shown in Figure 7.4.

Comparison:

- Much more commercial energy Q_{market} is purchased in multistage NMPC. Thus, it leads to higher peak-heating cost than the standard NMPC. This phenomena can be seen as robustness cost required by multistage NMPC. For standard NMPC, the

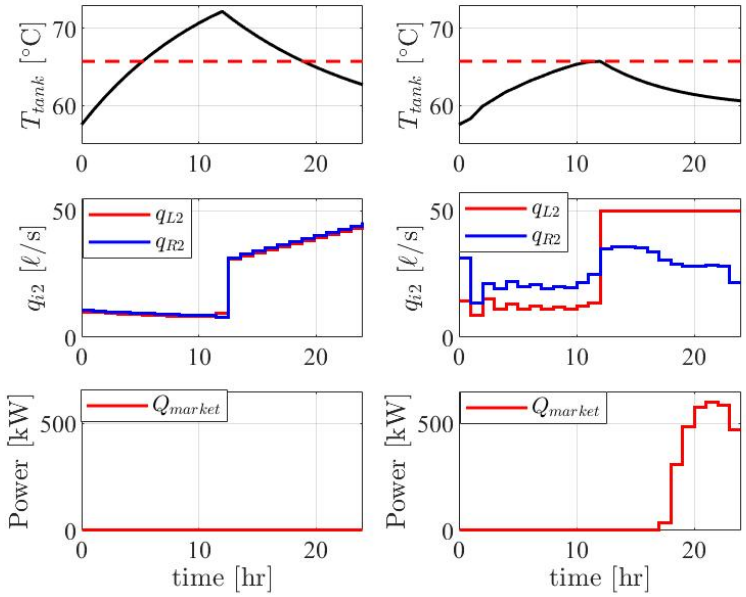


Figure 7.4: Results of the standard NMPC and multistage NMPC with consideration of uncertainties

surplus heat stored in the TES tank is available for meeting the peak demand due to the actual supply is higher than the expected supply. Therefore, there is no need to purchase commercial energy from the market in standard NMPC.

- The redefined upper constraint of T_{tank} is not violated in multistage NMPC. This result can be seen as the proof of high conservativeness of multistage NMPC, which respects the constraint of tank temperature to keep it lower than $70°C$. By contrast, it is violated in standard NMPC closely during $10h$ to $15h$, since the violation is to ensure the economic objective of the system.

However, if the upper bound of T_{tank} is chosen as lower than $65.72°C$, the optimal control problem solved by using standard NMPC would not converge to a feasible solution. Thus, in order to avoid this phenomena, *slack variables* are introduced to soften the constraints and the concept is already discussed in **Chapter 6.4.4**. The result of introducing slack variables for optimization problem of the standard NMPC is shown in Figure 7.5. Note that, in order to avoid the conflict arises in control inputs optimal control procedure, one of the manipulated flow rates is set as constant.

According to the demand satisfaction constraint which is given in equation (7.1), it is apparent to know that, the meeting of demand satisfaction constraint is directly fulfilled by optimal control of q_{R2}. In addition, the control action of q_{L2} also effects the heat transfer from hot side of HEX-2 to the cold and further effect T_{R2}. Thus, the strict control of q_{L2} may let demand satisfaction constraint under two control regimes and finally leads

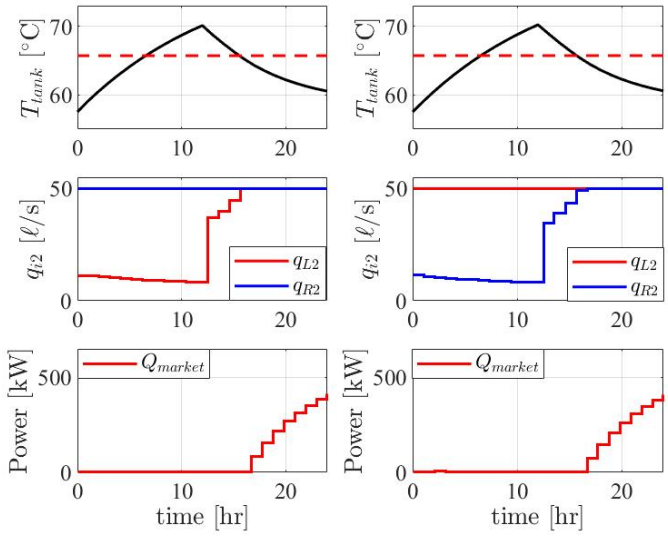


Figure 7.5: Results of the standard NMPC including slack variables with consideration of uncertainties

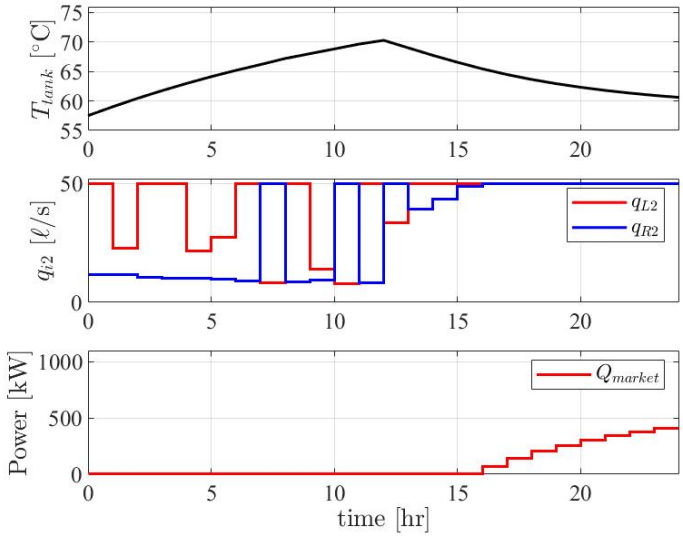


Figure 7.6: Results of the standard NMPC including slack variables with consideration of uncertainties (q_{L2}, q_{R2} are inequality constraint)

to the conflicts between q_{L2} and q_{R2} which is shown in Figure 7.6. Therefore, in order to avoid this phenomena, only q_{R2} is set as inequality constraint and this decision is made

after taking various trials in variable bounds and parameters.

$$Q_{demand} = Q_{market} + \rho c_p q_{R2}(T_{R2} - T_2) \tag{7.1}$$

Although the robustness provided by multistage NMPC results a much higher peak heating cost, it is still the better choice than standard NMPC, which is not robust and violates constraint during a period of operation. However, if there are no considerations of uncertainties and operational constraint in the system, the standard NMPC would be a better choice to control a simple TES system.

Chapter 8

Conclusion

8.1 Conclusion

In this thesis, the application of standard NMPC and scenario-based multistage NMPC strategy was investigated for a simple thermal energy storage system with a hot water TES tank. The case studies performed a system with one supplier, one consumer of surplus heat and the heat is stored and exchanged through the TES tank. The optimal control problem was formulated for three main and one minor cases and solved by using direct collocation method in CasADi framework within MATLAB programming environment. The nonlinear dynamic model in this thesis, which was formulated as a semi-explicit DAE system, was derived for heat exchanger outlet temperatures and TES tank outlet temperatures by applying energy balances over the different system components.

By comparing cases with and without TES tank, it is apparent to know that as the presence of a TES tank, the system becomes much more cost-effective. The size of a TES tank also effects the performance and the cost of the thermal energy storage system which directly reflected by discharge rate, energy storage capability and stored energy temperature T_{tank}. In practice, the intermittent renewable energy like solar thermal energy is available during some period and vise versa. Thus, adding additional eco-friendly and cost-effective energy source to the thermal energy storage system would considerably reduce the need for commercial energy when have to meet the peak demand. Uncertainty is the challenge to accurately control the TES system. Therefore, the uncertainties arise in the supply and demand side temperatures were effectively handled by using scenario-based multistage NMPC controller, which provides robustness and the better choice for rejecting disturbances than the standard NMPC. Therefore, the simulations conducted in this thesis were to study the effect of utilising TES tank, the performance of system with direct solar heating and the different NMPC controller performances toward the presence of uncertainties. The result of each case was demonstrated the above-mentioned conclusions were correct.

Consequently, the optimal control objectives to minimize the cost caused by extra commercial energy for peak-heating of a simple thermal energy storage system with one supplier,

one consumer and one TES tank with and without the presence of uncertainty by using iNMPC were successfully achieved in this thesis.

8.2 Further work

For scenario-based multistage NMPC, the improvements are considered in following sections:

1. The selection of scenarios have the potential to be improved. In this thesis, the selection of scenarios were done by implementing basic and conventional BOX method, which ignores the correlations between uncertain parameters. In the future work, more advanced scenario selection method, such as *data-driven scenario selection* can be chosen to implement.

2. Other uncertain parameters could also be the choice to develop robust NMPC. The system model could be reformulated to include supply and demand and further considering these two parameters as uncertainties.

3. The problem solved by multistage NMPC increases exponentially along the propagation of scenario tree and it increases the challenge of solving the control problem online. The resultant delayed availability of control actions might contribute to system instability or sub-optimal control performance according to R. Findeisen and F.Allgöwer.(2004). Therefore, in order to solve this nontrivial delays of multistage NMPC, *advanced-step multistage NMPC* could be proposed.

Bibliography

Biegler L.T.(2010), . Nonlinear programming: Concepts, algorithms, and applications to chemical processes , *https://doi.org/10.1137/1.9780898719383*.

Biegler,L.T. (2010), . Nonlinear programming: Concepts, algorithms, and applications to chemical processes. SIAM: Philadelphia, PA, USA 10.

Burnett,M. (2016), . Energy storage and the california "duck curve". Submitted as coursework for PH240, Stanford University, Fall 2015 , http://large.stanford.edu/courses/2015/ph240/burnett2/.

Cutler,C.R. and Ramaker,B.L. (1980), . Dynamic matrix control - a computer control algorithm. Joint Automatic Control Conference Preprints Paper WP5-B, San Francisco.

D.Q.Mayne and J.B.Rawlings et al. (2000), . Constrained model predictive control: Stability and optimality. Automatica 36, 789–814, *https://doi.org/10.1016/S0005–1098(99)00214–9*.

Edgar TF et al. (2001), . Optimization of chemical processes. SIAM: Philadelphia, PA, USA , New York, NY: McGraw–Hill.

Grimm,G. et al. (2004), . Examples when nonlinear model predictive control is nonrobust. Automatica 40, 1729–1738, *https://doi.org/10.1016/j.automatica.2004.04.014*.

J.H.Lee and Z.H.Yu (1997), . Worst-case formulations of model predictive control for systems with bounded parameters. Automatica 33, 763– 781, *https://doi.org/10.1016/S0005–1098(96)00255–5*.

Lucia,S. and Engell,S. (2013), . Robust nonlinear model predictive control of a batch bioreactor using multi-stage stochastic programming. IEEE 33, 763– 781, *10.23919/ECC.2013.6669521*,.

Lucia,S. and Engell,S. (2015), . Potential and limitations of multi-stage nonlinear model predictive control. IFAC-PapersOnLine 48, 1015–1020 , *https://doi.org/10.1016/j.ifacol.2015.09.101*,.

Richalet,J. et al. (1978), . Model predictive heuristic control. Application to industrial processes. Automatica 14, 413–428.

Sébastien Gros (2016), . Numerical optimal control lecture 6: Direct collocation, NTNU PhD course. lecture slide .

A. Wächter and L.T.Biegler (2006), . On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. Mathematical programming 106, 25–57.

Bergman,T.L et al.(2011), . Heat exchangers, in: Fundamentals of Heat and Mass Transfer, 7th Edition.

Cabeza, L.F. (2015), . Introduction to thermal energy storage (TES) systems, in: Advances in Thermal Energy Storage Systems Methods and Applications, *https://doi.org/10.1016/C2013-0-16453-7*.

Campo,P.J.and Morari,M.(1987), . Robust model predictive control. In Proceedings of the 1987 American Control Conference, Minneapolis, MN, USA , 1021–1026.

Chen J. (1987), . Comments on improvements on a replacement for the logarithmic mean. Chemical Engineering Science 42, 2488–2489, *https://doi.org/10.1016/0009–2509(87)80128–8*.

Dinçer, I. and Rosen, M.A.(2010), . Energy storage systems. Thermal energy storage. System and applications, 2nd edition .

Dinçer, I.(1997)), . Heat transfer in food cooling applications. Taylor Francis, Washington, DC .

Ding,N. (2019), . Hybrid modeling with machine learning and first principles models, in: TKP4580 - Specialization Project, Process Systems Engineering. NTNU.

D.Q. Mayne and E.C. Kerrigan (2007), . Tube-based robust nonlinear model predictive control. In Proc. of the 7th IFAC Symposium on Nonlinear Control Systems , 110–115.

D.Q. Mayne and M.M. Seron et al. (2005), . Robust model predictive control of constrained linear systems with bounded disturbances. Automatica 41, 219–224.

Foss,B. and Heirung,T.A.N.(2013), . Optimization, in: Merging optimization and control. Lecture Notes.

Gil,A. and Medrano,M. et al.(2011), . State of the art on high temperature thermal energy storage for power generation. part 1 – concepts, materials and modellization. Renewable and Sustainable Energy Reviews .

J. Hendler et al. (2010), . Foundations of artificial intelligence, handbook of knowledge representation .

J.B.Rawlings et al. (2018), . Getting started with model predictive control, in: Model Predictive Control: Theory, Computation, and Design 2nd Edition.

Joel A.E.Andersson et al.(2018), . Casadi – a software framework for nonlinear optimization and optimal control. Mathematical Programming Computation 11, 1015–1020 , *10.1007/s12532–018–0139–4*,.

Johannes Jäschke (2019), . Dynamic optimization for mpc, orthogonal collocation on finite elements, in: TKP4555 Process system engineering, specialization course,MPC.

J.T.Betts (2010), . Practical methods for optimal control and estimation using nonlinear programming 19, Siam,*https://doi.org/10.1137/1.9780898718577*.

M. Orosz and R. Dickes (2017), . Solar thermal powered organic rankine cycles, in: Organic Rankine Cycle (ORC) Power Systems, Technologies and Applications, *https://doi.org/10.1016/C2014-0-04239-6*.

Maurice I. Stewart, Jr. (2014), . Heat transfer theory, in: Surface Production Operations, Volume 2: Design of Gas-Handling Systems and Facilities, *https://doi.org/10.1016/C2009-0-64501-3*.

Mehling, H. and Cabeza, L.F. (2008), . Solid-liquid phase change materials, in: Heat and cold storage with PCM, An up to date introduction into basics and applications, *https://doi.org/10.1007/978-3-540-68557-9*.

Moritz Diehl and Sébastien Gros (2017), . Numerical simulation, orthogonal collocation, in: Numerical Optimal Control (preliminary and incomplete draft).

Nocedal,J. and Wright,S.J.(2006), . Theory of constrained optimization, in: Numerical optimization Second Edition.

O.von Stryk (1992), . Numerical solution of optimal control problems by direct collocation, in: Optimal Control. Calculus of Variations, Optimal Control Theory and Numerical Methods.

Peng Zhang (2010), . Advanced industrial control technology. Elsevier Inc .

Pontryagin VG et al. (1962), . The mathematical theory of optimal processes .

R. Findeisen and F.Allgöwer.(2004), . Computational delay in nonlinear model predictive control. IFAC Proc 37, 427–432.

S. Furbo (2015), . Using water for heat storage in thermal energy storage (TES) systems, in: Advances in Thermal Energy Storage Systems Methods and Applications, *https://doi.org/10.1533/9781782420965.1.31*.

Sarbu, I. and Sebarchievici, C. (2018), . A comprehensive review of thermal energy storage. Sustainability 10, *https://doi.org/10.3390/su10010191*.

Seborg D.E.et al.(2011), . Model predictive control, in: Process Dynamics and Control. 3rd Edition.

Underwood, A.J.V (1970), . Simple formula to calculate mean temperature difference. Chemical Engineering 77.

W.R.Paterson (1984), . A replacement for the logarithmic mean. Chemical Engineering Science 39, 1635–1636.

Zavala-Río et al.(2005), . An analytical study of the logarithmic mean temperature difference. Revista Mexicana de Ingenier´ıa Qu´ımica 4, 201–212.

Zhou Joyce Yu and Biegler,L.T (2019), . Advanced-step multistage nonlinear model predictive control: Robustness and stability. Journal of Process Control 84, 192–206. *https://doi.org/10.1016/j.jprocont.2019.10.006*.

Appendix A

ODEs System Model:

$$\frac{dT_{L1}}{dt} = \frac{1}{V_{hex}}\Big\{q_{L1}(T_1 - T_{L1}) - \frac{U_{hex}A_{hex}}{\rho c_p}\Delta T_{m,1}\Big\} \quad (8.1a)$$

$$\frac{dT_{R1}}{dt} = \frac{1}{V_{hex}}\Big\{q_{R1}(T_{tank} - T_{R1}) + \frac{U_{hex}A_{hex}}{\rho c_p}\Delta T_{m,1}\Big\} \quad (8.1b)$$

$$\frac{dT_{L2}}{dt} = \frac{1}{V_{hex}}\Big\{q_{L2}(T_{tank} - T_{L2}) - \frac{U_{hex}A_{hex}}{\rho c_p}\Delta T_{m,2}\Big\} \quad (8.1c)$$

$$\frac{dT_{R2}}{dt} = \frac{1}{V_{hex}}\Big\{q_{R2}(T_2 - T_{R2}) + \frac{U_{hex}A_{hex}}{\rho c_p}\Delta T_{m,2}\Big\} \quad (8.1d)$$

$$\frac{dT_{tank}}{dt} = \frac{1}{V_{tank}}\Big\{q_{R1}(T_{R1} - T_{tank}) + q_{L2}(T_{L2} - T_{tank}) + \frac{Q_{tank} - Q_{loss}}{\rho c_p}\Big\} \quad (8.1e)$$

Where,

$$Q_{loss} = (UA)_{tank}(T_{tank} - T_{surr}) \quad (8.2a)$$

$$\Delta T_{m,1} = \left[\frac{1}{2}\big((T_1 - T_{R1})^n + (T_{L1} - T_{tank})^n\big)\right]^{\frac{1}{n}}, n = \frac{1}{3} \quad (8.2b)$$

$$\Delta T_{m,2} = \left[\frac{1}{2}\big((T_{tank} - T_{R2})^n + (T_{L2} - T_2)^n\big)\right]^{\frac{1}{n}}, n = \frac{1}{3} \quad (8.2c)$$

Appendix B

DAEs System Model:

$$\frac{dT_{L1}}{dt} = \frac{1}{V_{hex}}\Big\{q_{L1}(T_1 - T_{L1}) - \frac{U_{hex}A_{hex}}{\rho c_p}\Delta T_{m,1}\Big\} \tag{8.3a}$$

$$\frac{dT_{R1}}{dt} = \frac{1}{V_{hex}}\Big\{q_{R1}(T_{tank} - T_{R1}) + \frac{U_{hex}A_{hex}}{\rho c_p}\Delta T_{m,1}\Big\} \tag{8.3b}$$

$$\frac{dT_{L2}}{dt} = \frac{1}{V_{hex}}\Big\{q_{L2}(T_{tank} - T_{L2}) - \frac{U_{hex}A_{hex}}{\rho c_p}\Delta T_{m,2}\Big\} \tag{8.3c}$$

$$\frac{dT_{R2}}{dt} = \frac{1}{V_{hex}}\Big\{q_{R2}(T_2 - T_{R2}) + \frac{U_{hex}A_{hex}}{\rho c_p}\Delta T_{m,2}\Big\} \tag{8.3d}$$

$$\frac{dT_{tank}}{dt} = \frac{1}{V_{tank}}\Big\{q_{R1}(T_{R1} - T_{tank}) + q_{L2}(T_{L2} - T_{tank}) + \frac{Q_{tank} - Q_{loss}}{\rho c_p}\Big\} \tag{8.3e}$$

$$0 = a^{\frac{1}{n}} - (T_1 - T_{R1}) \tag{8.4a}$$

$$0 = b^{\frac{1}{n}} - (T_{L1} - T_{tank}) \tag{8.4b}$$

$$0 = c^{\frac{1}{n}} - (T_{tank} - T_{R2}) \tag{8.4c}$$

$$0 = d^{\frac{1}{n}} - (T_{L2} - T_2) \tag{8.4d}$$

Where,

$$Q_{loss} = (UA)_{tank}(T_{tank} - T_{surr}) \tag{8.5a}$$

$$\Delta T^n_{m,1} = \frac{1}{2}(a + b), \;\; n = \frac{1}{3} \tag{8.5b}$$

$$\Delta T^n_{m,2} = \frac{1}{2}(c + d), \;\; n = \frac{1}{3} \tag{8.5c}$$

Appendix C

Dynamic optimization problem for the TES system without direct solar heating:

$$\min_{\mathbf{x},\mathbf{u}} \; \phi(\mathbf{x}, \mathbf{u}) = P_m Q_{market} \tag{8.6a}$$

subject to:

$$x_0 = x_{init} \tag{8.6b}$$

DAEs system model (8.6c)

Operating constraints—states (8.6d)

Operating constraints—inputs (8.6e)

Demand satisfaction constraint (8.6f)

Supply satisfaction constraint (8.6g)

Where,

Operating constraints—states:

$$\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 30 \end{bmatrix} \leq \begin{bmatrix} T_{L1} \\ T_{R1} \\ T_{L2} \\ T_{R2} \\ T_{tank} \end{bmatrix} \leq \begin{bmatrix} 100 \\ 100 \\ 100 \\ 100 \\ 100 \end{bmatrix} \tag{8.7}$$

Operating constraints—inputs:

$$\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \leq \begin{bmatrix} q_{R2} \\ Q_{market} \\ Q_{dump} \end{bmatrix} \leq \begin{bmatrix} 50 \\ +\infty \\ +\infty \end{bmatrix} \tag{8.8}$$

Demand satisfaction constraint:

$$Q_{demand} - Q_{market} - \rho c_p q_{R2}(T_{R2} - T_2) = 0 \tag{8.9}$$

Supply satisfaction constraint:

$$Q_{supply} - Q_{dump} - \rho c_p q_{L1}(T_1 - T_{L1}) = 0 \tag{8.10}$$

Appendix D

Dynamic optimization problem for the TES system with direct solar heating:

$$\min_{\mathbf{x},\mathbf{u}} \ \phi(\mathbf{x},\mathbf{u}) = P_m Q_{market} + P_t Q_{tank} \tag{8.11a}$$

subject to:

$$x_0 = x_{init} \tag{8.11b}$$

DAEs system model (8.11c)

Operating constraints—states (8.11d)

Operating constraints—inputs (8.11e)

Demand satisfaction constraint (8.11f)

Supply satisfaction constraint (8.11g)

Where,

Operating constraints—states:

$$\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 30 \end{bmatrix} \leq \begin{bmatrix} T_{L1} \\ T_{R1} \\ T_{L2} \\ T_{R2} \\ T_{tank} \end{bmatrix} \leq \begin{bmatrix} 100 \\ 100 \\ 100 \\ 100 \\ 100 \end{bmatrix} \tag{8.12}$$

Operating constraints—inputs:

$$\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \leq \begin{bmatrix} q_{R2} \\ Q_{market} \\ Q_{dump} \\ Q_{tank} \end{bmatrix} \leq \begin{bmatrix} 50 \\ +\infty \\ +\infty \\ Q_{solar,k} \end{bmatrix} \tag{8.13}$$

Demand satisfaction constraint:

$$Q_{demand} - Q_{market} - \rho c_p q_{R2}(T_{R2} - T_2) = 0 \tag{8.14}$$

Supply satisfaction constraint:

$$Q_{supply} - Q_{dump} - \rho c_p q_{L1}(T_1 - T_{L1}) = 0 \tag{8.15}$$

Appendix E

Dynamic optimization problem for the TES system with plant-model mismatch, msN-MPC:

$$\min_{\mathbf{x_{i,j}}, \mathbf{u_{i,j}}} \ \phi(\mathbf{x}, \mathbf{u}) = P_m(Q_{market})_{i,j} + P_t(Q_{tank})_{i,j} + P_u\big((q_{L2})_{i,j}^2 + (q_{R2})_{i,j}^2\big) \tag{8.16a}$$

subject to:

$$x_0 = x_{init} \tag{8.16b}$$

DAEs system model (8.16c)

Operating constraints—states (8.16d)

Operating constraints—inputs (8.16e)

Demand satisfaction constraint (8.16f)

Non-anticipativity constraint (8.16g)

Where,

Scenario and time step index:

$$i \in \{1, ..., M\};\ j \in \{0, ..., N-1\} \tag{8.17}$$

Operating constraints—states:

$$\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 30 \end{bmatrix} \leq \begin{bmatrix} T_{L1} \\ T_{R1} \\ T_{L2} \\ T_{R2} \\ T_{tank} \end{bmatrix} \leq \begin{bmatrix} 100 \\ 100 \\ 100 \\ 100 \\ 70 \end{bmatrix} \tag{8.18}$$

Operating constraints—inputs:

$$\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \leq \begin{bmatrix} q_{L2} \\ q_{R2} \\ Q_{tank} \\ Q_{market} \end{bmatrix} \leq \begin{bmatrix} 50 \\ 50 \\ 10 \\ +\infty \end{bmatrix} \tag{8.19}$$

Demand satisfaction constraint:

$$Q_{demand} - Q_{market} - \rho c_p q_{R2}(T_{R2} - T_2) = 0 \tag{8.20}$$

Non-anticipativity constraint:

$$u_{j,i} = u_{j,l} \ \ if \ \ x_{j,i} = x_{j,l} \tag{8.21}$$

Appendix F

Dynamic optimization problem for the TES system with plant-model mismatch, sNMPC with slack variables:

$$\min_{\mathbf{x},\mathbf{u}} \ \phi(\mathbf{x}, \mathbf{u}) = P_m Q_{market} + P_t Q_{tank} + P_u\left(q_{L2}^2 + q_{R2}^2\right) + \rho^\top \epsilon \tag{8.22a}$$

subject to:

$$x_0 = x_{init} \tag{8.22b}$$

DAEs system model (8.22c)

Operating constraints—states (8.22d)

Operating constraints—inputs (8.22e)

Demand satisfaction constraint (8.22f)

Where,

Operating constraints—states:

$$\begin{bmatrix} 0-\epsilon \\ 0-\epsilon \\ 0-\epsilon \\ 0-\epsilon \\ 30-\epsilon \end{bmatrix} \leq \begin{bmatrix} T_{L1} \\ T_{R1} \\ T_{L2} \\ T_{R2} \\ T_{tank} \end{bmatrix} \leq \begin{bmatrix} 100+\epsilon \\ 100+\epsilon \\ 100+\epsilon \\ 100+\epsilon \\ 70+\epsilon \end{bmatrix} \tag{8.23}$$

Operating constraints—inputs:

$$\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \leq \begin{bmatrix} q_{L2} \\ q_{R2} \\ Q_{tank} \\ Q_{market} \end{bmatrix} \leq \begin{bmatrix} 50 \\ 50 \\ 10 \\ +\infty \end{bmatrix} \tag{8.24}$$

Demand satisfaction constraint:

$$Q_{demand} - Q_{market} - \rho c_p q_{R2}(T_{R2} - T_2) = 0 \tag{8.25}$$

Appendix G

Thermal energy system without storage tank:

DAEs system mode

$$\frac{dT_{L1}}{dt} = \frac{1}{V_{hex}}\Big\{q_{L1}(T_1 - T_{L1}) - \frac{U_{hex}A_{hex}}{\rho c_p}\Delta T_m\Big\} \tag{8.26a}$$

$$\frac{dT_{R1}}{dt} = \frac{1}{V_{hex}}\Big\{q_{R1}(T_2 - T_{R1}) + \frac{U_{hex}A_{hex}}{\rho c_p}\Delta T_m\Big\} \tag{8.26b}$$

$$0 = a^{\frac{1}{n}} - (T_1 - T_{R1}) \tag{8.26c}$$

$$0 = b^{\frac{1}{n}} - (T_{L1} - T_2) \tag{8.26d}$$

Where,

$$\Delta T_m^n = \frac{1}{2}(a+b), \quad n = \frac{1}{3} \tag{8.27}$$

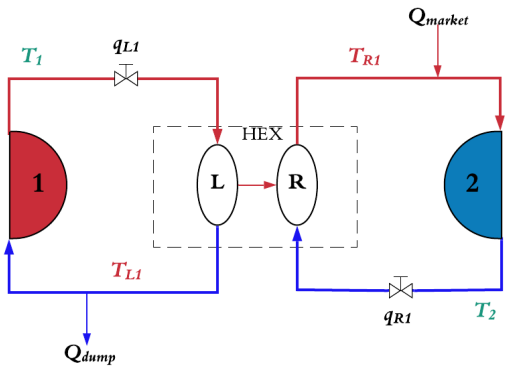


Figure 8.1: Topology of a simple thermal energy system without thermal storage tank

Variables classification

- **Differential states x:**

$$\mathbf{x} = [T_{L1}, T_{R1}] \tag{8.28}$$

- **Algebraic states z:**

$$\mathbf{z} = [a, b] = [(T_1 - T_{R1})^n, (T_{L1} - T_2)^n] \tag{8.29}$$

- **Inputs u:**

$$\mathbf{u} = [q_{L1}, q_{R1}, Q_{market}, Q_{dump}] \tag{8.30}$$

- **Disturbances d:**

$$\mathbf{d} = [T_1, T_2, T_{surr}] \tag{8.31}$$

Dynamic optimization problem for the thermal energy system without storage tank

$$\min_{\mathbf{x},\mathbf{u}} \ \phi(\mathbf{x}, \mathbf{u}) = P_m Q_{market} \tag{8.32a}$$

subject to:

$$x_0 = x_{init} \tag{8.32b}$$

DAEs system model (8.32c)

Operating constraints—states (8.32d)

Operating constraints—inputs (8.32e)

Demand satisfaction constraint (8.32f)

Supply satisfaction constraint (8.32g)

Where,

Operating constraints—states:

$$\begin{bmatrix} 0 \\ 0 \end{bmatrix} \leq \begin{bmatrix} T_{L1} \\ T_{R1} \end{bmatrix} \leq \begin{bmatrix} 100 \\ 100 \end{bmatrix} \tag{8.33}$$

Operating constraints—inputs:

$$\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \leq \begin{bmatrix} q_{R1} \\ Q_{market} \\ Q_{dump} \end{bmatrix} \leq \begin{bmatrix} 50 \\ +\infty \\ +\infty \end{bmatrix} \tag{8.34}$$

Where, $q_{L1} = 50\ [l/s]$, which is set as constant in the whole process.

Demand satisfaction constraint:

$$Q_{demand} - Q_{market} - \rho c_p q_{R1}(T_{R1} - T_2) = 0 \tag{8.35}$$

Supply satisfaction constraint:

$$Q_{supply} - Q_{dump} - \rho c_p q_{L1}(T_1 - T_{L1}) = 0 \tag{8.36}$$

Matlab code

Case one: Standard NMPC on simple TES with storage tank

Listing 8.1: Simple TES system model

```
function dxdt = twoPlantModelChen(~,x,p)
%% Description of the states:
T_L1 = x(1);
T_R1 = x(2);
T_L2 = x(3);
T_R2 = x(4);
T_tank = x(5);
%% Description of parameters
q_L1 = p(1);
q_R1 = p(2);
q_L2 = p(3);
q_R2 = p(4);
Q = p(5);
T1 = p(6);
T2 = p(7);
V_hex = p(8);
V_tank = p(9);
U_hex = p(10);
A_hex = p(11);
rho = p(12);
cp = p(13);
h_s = p(14);
A_tank = p(15);
T_s = p(16);
n = p(17);

%% ODEs
dxdt = zeros(5,1);

dxdt = [(1/V_hex)*(q_L1*(T1-x(1))-sign(T1-x(5))*(U_hex*A_hex*0.5^(1/n)/...
         (rho*cp))*((abs(T1-x(2)))^n+(abs(x(1)-x(5)))^n)^(1/n));

        (1/V_hex)*(q_R1*(x(5)-x(2))+sign(T1-x(5))*(U_hex*A_hex*0.5^(1/n)/...
        (rho*cp))*((abs(T1-x(2)))^n+(abs(x(1)-x(5)))^n)^(1/n));

        (1/V_hex)*(q_L2*(x(5)-x(3))-sign(x(5)-T2)*(U_hex*A_hex*0.5^(1/n)/...
        (rho*cp))*((abs(x(3)-T2))^n+(abs(x(5)-x(4)))^n)^(1/n));

        (1/V_hex)*(q_R2*(T2-x(4))+sign(x(5)-T2)*(U_hex*A_hex*0.5^(1/n)/...
        (rho*cp))*((abs(x(3)-T2))^n+(abs(x(5)-x(4)))^n)^(1/n));

        (1/V_tank)*(q_R1*(x(2)-x(5)) + q_L2*(x(3)-x(5)) + (Q-h_s*A_tank*...
        (x(5)-T_s))/(rho*cp))];
```

```
end
```

Listing 8.2: TES parameters

```
%%  Parameters_TES.m file
% Defenition of all constant parameters used in TES
%% TES Parameters
V_tank =1000;       %Storage tank volume [m^3]
A_tank =100;        %Tank heat loss area [m^2]
h_s = 5e-4;         %Tank heat loss coefficient [kW/m^2K]
%% HEX Parameters
V_hex = 0.5;        %Heat exchanger volume (shell or tube side) [m^3]
A_hex = 300;        %HEX heat transfer area [m^2]
U_hex = 0.5;        %Overall heat transfer coefficient [kW/m^2K]
%% Storage fluid parameters
rho= 1000;          %Density [kg/m^3]
cp = 4.186;         %Specific heat capacity of fluid [kJ/kgK]
%% Common Parameters
n = 1/3;
T1 = 95;            %Supply side intet temperature [C]
T2 = 20;            %Consumer side intet temperature [C]
T_s = 15;           %Ambient temperature [C]
Pm = 1e-3;          %Commercial energy cost
% Pt = 5e--6;       %Direct solar heating cost
Pu = 50e-6;         %Regularization weight
%% Energy supply and demand profile
Qdemand = 5000;
Qsupply = 2500;
%% NLP solver parameters
tol = 10e-8;       % Desired convergence tolerance (relative)
maxiter = 5000;
%% Declaration of constant inputs
q_L1=0.05;          % [m^3/s]
q_R1=0.05;
q_L2=0.05;
Q_tank=0;
%% Other combined terms
h_dot = (U_hex*A_hex)/(rho*cp);
h_t_dot = (h_s*A_tank)/(rho*cp);
```

Listing 8.3: Steady state optimization

```
addpath('C:\Users\dingn\Desktop\Thesis\casadi-windows-matlabR2016a-v3.5.1')
import casadi.*
%% Run model parameters
run Parameters_TES.m
%% Declare model variables
% Diffential states
x1 = SX.sym('x1');            % T_L1
x2 = SX.sym('x2');            % T_R1
x3 = SX.sym('x3');            % T_L2
x4 = SX.sym('x4');            % T_R2
x5 = SX.sym('x5');            % T_tank
```

```
x = [x1; x2; x3; x4; x5];
% Algebraic states
x6 = SX.sym('x5');              % a=(T1-x2)^n
x7 = SX.sym('x5');              % b=(x1-x5)^n
x8 = SX.sym('x5');              % c=(x3-T2)^n
x9 = SX.sym('x5');              % d=(x5-x4)^n
z = [x6; x7; x8; x9];           % z=[a,b,c,d]
% Control inputs
u1 = SX.sym('u1');              % qR2
u2 = SX.sym('u2');              % Q_Market
u3 = SX.sym('u3');              % Q_Dump
u = [u1; u2; u3];
%% Model equations (DEAs)

xdot = [(1/V_hex)*(q_L1*(T1-x1) - h_dot*(0.5*(x6  + x7 ))^(1/n));

        (1/V_hex)*(q_R1*(x5-x2) + h_dot*(0.5*(x6  + x7 ))^(1/n));

        (1/V_hex)*(q_L2*(x5-x3) - h_dot*(0.5*(x8 + x9 ))^(1/n));

        (1/V_hex)*(u1*(T2-x4) + h_dot*(0.5*(x8  + x9 ))^(1/n));

        (1/V_tank)*(q_R1*(x2-x5) + q_L2*(x3-x5) - h_t_dot*(x5-T_s));

         T1-x2-x6^(1/n);

         x1-x5-x7^(1/n);

         x3-T2-x8^(1/n);

         x5-x4-x9^(1/n)];

diff = xdot(1:5);
alg = xdot(6:9);

%% Lower and upper bounds of x,z,u
% LB Differential states
x1_lb = 0;
x2_lb = 0;
x3_lb = 0;
x4_lb = 0;
x5_lb = 30;
% LB Algebraic states
x6_lb = -Inf;
x7_lb = -Inf;
x8_lb = -Inf;
x9_lb = -Inf;
% LB Control inputs
u1_lb=0;
u2_lb=0;
u3_lb=0;
% LB Vertical concatenation
xlb = vertcat(x1_lb,x2_lb,x3_lb,x4_lb,x5_lb);
zlb = vertcat(x6_lb,x7_lb,x8_lb,x9_lb);
ulb = vertcat(u1_lb,u2_lb,u3_lb);
% UB Differential states
x1_ub = 100;
```

```
x2_ub = 100;
x3_ub = 100;
x4_ub = 100;
x5_ub = 100;
% UB Algebraic states
x6_ub = +Inf;
x7_ub = +Inf;
x8_ub = +Inf;
x9_ub = +Inf;
% UB Control inputs
u1_ub = 0.05;                          % [m^3/s]
u2_ub = +Inf;
u3_ub = +Inf;
% UB Vertical concatenation
xub = vertcat(x1_ub,x2_ub,x3_ub,x4_ub,x5_ub);
zub = vertcat(x6_ub,x7_ub,x8_ub,x9_ub);
uub = vertcat(u1_ub,u2_ub,u3_ub);
%% Declare two satisfaction equality constraint as symbolic variables
C1 = Qdemand - u2 - u1*rho*cp*(x4-T2);
C2 = Qsupply - u3 -q_L1*rho*cp*(T1-x1);
C1_lb = 0;
C1_ub = 0;
C2_lb = 0;
C2_ub = 0;

%% Initial guess
% Differential states
x10 = 90;
x20 = 80;
x30 = 60;
x40 = 50;
x50 = 70;
% Algebraic states
x60 = 1;
x70 = 1;
x80 = 1;
x90 = 1;
X0=[x10; x20; x30; x40; x50];
Z0=[x60; x70; x80; x90];

% Control inputs
u10 = 0.02;
u20 = 1500;
u30 = 200;
U0 = [u10; u20; u30];
%% Steady-state optimization
% Preparing symbolic variables
w = {};
% Preparing numeric variables and bounds
w0 = [];
lbw = [];
ubw = [];
% Preparing symbolic constraints
g = {};
% Preparing numeric bounds
lbg = [];
ubg = [];
```

```
% Declaring them symbolic
w = {w{:},x,z,u};
lbw = [lbw;xlb;zlb;ulb];
ubw = [ubw;xub;zub;uub];
w0 = [w0;X0;Z0;U0];

% Add the system model as constraints
g = {g{:},vertcat(diff,alg),C1,C2};
lbg = [lbg;zeros(9,1);C1_lb;C2_lb];   % Steady-state optimisation, dx/dt=0
ubg = [ubg;zeros(9,1);C1_ub;C2_ub];
% Cost function
% Adding additional regularisation term!!!
L = Pm*u2 + Pu*(u1^2+u3^2);
% Economic objective
J = L;
nlp = struct('x',vertcat(w{:}),'f',J,'g',vertcat(g{:}));
solver = nlpsol('solver','ipopt',nlp); % NLP solver IPOPT
sol = solver('x0',w0,'lbx',lbw,'ubx',ubw,'lbg',lbg,'ubg',ubg);
%% Extracting solutions
w_opt_SS = full(sol .x);
x_init=w_opt_SS(1:5);
z_init=w_opt_SS(6:9);
u_init=w_opt_SS(10:end);
```

Listing 8.4: Standard NMPC

```
clear;
clc;
close all;
addpath('C:\Users\dingn\Desktop\Thesis\casadi-windows-matlabR2016a-v3.5.1')
import casadi.*
%% TES parameters
run Parameters_TES.m
%% Energy supply and demand profile
Q_demand = [1500*ones(12,1); 3500*ones(12,1)];
Q_supply = 2500*ones(24,1);
%% Degree of interpolating polynomial
d = 3;      % Gauss-Radua
%% Get collocation points
tau_root = [0 collocation_points(d, 'radau')];
%% Coefficients of the collocation equation
C = zeros(d+1,d+1);
% Coefficients of the continuity equation
D = zeros(d+1, 1);
% Coefficients of the quadrature function
B = zeros(d+1, 1);

%% Construct polynomial basis
for j=1:d+1
% Construct Lagrange polynomials to get the polynomial basis
% at the collocation point
 coeff = 1;
  for r=1:d+1
     if r ~= j
       coeff = conv(coeff, [1, -tau_root(r)]);
       coeff = coeff / (tau_root(j)-tau_root(r));
```

```
    end
  end
% Evaluate the polynomial at the final time to get the
% coefficients of the continuity equation
D(j) = polyval(coeff, 1.0);
% Evaluate the time derivative of the polynomial at all collocation
% points to get the coefficients of the continuity equation
pder = polyder(coeff);
  for r=1:d+1
    C(j,r) = polyval(pder, tau_root(r));
  end
% Evaluate the integral of the polynomial to get the coefficients
% of the quadrature function
pint = polyint(coeff);
B(j) = polyval(pint, 1.0);

end
%% Time horizon
T = 24*60*60;               % Prediction horizon:24h
ndiff = 5;                  % Number of differential states, x
nalg = 4;                   % Number of algebraic states, z
nu = 3;                     % Number of control inputs, u
nx = nalg + ndiff;          % Total number of states
%% Bounds of algebraic states, z
zub = Inf*ones(nalg,1);
zlb = -Inf*ones(nalg,1);
%% Run RTO to get ssopt results
run RTO.m
x_0 = x_init;
%% Declare model variables
% Differential states
x1 = SX.sym('x1');              % T_L1
x2 = SX.sym('x2');              % T_R1
x3 = SX.sym('x3');              % T_L2
x4 = SX.sym('x4');              % T_R2
x5 = SX.sym('x5');              % T_tank
x = [x1; x2; x3; x4; x5];
% Algebraic states
x6 = SX.sym('x6');              % a=(T1-x2)^n
x7 = SX.sym('x7');              % b=(x1-x5)^n
x8 = SX.sym('x8');              % c=(x3-T2)^n
x9 = SX.sym('x9');              % d=(x5-x4)^n
z = [x6; x7; x8; x9];
% Control inputs
u1 = SX.sym('u1');              % q_R2
u2 = SX.sym('u2');              % Q_Market
u3 = SX.sym('u3');              % Q_dump
u = [u1; u2; u3];
%% Model equations (DEAs)

xdot = [(1/V_hex)*(q_L1*(T1-x1) - h_dot*(0.5*(x6 + x7))^(1/n));

        (1/V_hex)*(q_R1*(x5-x2) + h_dot*(0.5*(x6 + x7))^(1/n));

        (1/V_hex)*(q_L2*(x5-x3) - h_dot*(0.5*(x8 + x9))^(1/n));

        (1/V_hex)*(u1*(T2-x4) + h_dot*(0.5*(x8 + x9))^(1/n));
```

```
        (1/V_tank)*(q_R1*(x2-x5) + q_L2*(x3-x5) - h_t_dot*(x5-T_s));

         T1-x2-x6^(1/n);

         x1-x5-x7^(1/n);

         x3-T2-x8^(1/n);

         x5-x4-x9^(1/n)];

%% Objective term
L = Pm*u2 + Pu*(u1^2+u3^2);
%% Continuous time dynamics
f = Function('f', {x, z, u}, {xdot, L});
%% Control discretization
N = 24;              % number of control intervals, 24 hours
M = 24;              % number of MPC loops, reoptimizing times
h = T/N;
period = N;
%% Variable counting
NXD = N*ndiff*(d+1);         %Total Number of differential state variables
NXA = N*nalg*d;              %Total Number of algebraic state variables
NU = N*nu;                   %Total Number of input variables
NXF = ndiff;                 %Total Number of end point variables
NV = NXD + NXA + NU + NXF;   %Total number of NLP variables
%% Prepare output variables
x_opt = zeros(ndiff,M);
u_opt = zeros(nu,M);
i_infeasible = zeros(M,1);
solver_return = {};
w_stored = zeros(NV,1);
%% Predicted Demand
Q_demand = [Q_demand; Q_demand];
Q_supply = [Q_supply; Q_supply];
%% MPC loop
for i=1:M
   if i==1
     % Start with an empty NLP
     w={};
     w0 = [];
     lbw = [];
     ubw = [];
     J = 0;
     g={};
     lbg = [];
     ubg = [];
     % Initial conditions
     % Differential states
     Xk = MX.sym('X0', ndiff);
     w = [w(:)', {Xk}];
     lbw = [lbw; x_init];
     ubw = [ubw; x_init];
     w0 = [w0; x_init];
     % Formulate the NLP
     for k=0:N-1
       % New NLP variable for the control
```

```
Uk = MX.sym(['U_' num2str(k)], nu);
w = [w(:)', {Uk}];
lbw = [lbw; ulb];
ubw = [ubw; uub];
w0 = [w0; u_init];
% State at collocation points
Xkj = cell(1,d);
Zkj = cell(1,d);
for j=1:d
  % Differential states
  Xkj{j} = MX.sym(['X_' num2str(k) '_' num2str(j)],ndiff);
  w = [w(:)', Xkj(j)];
  lbw = [lbw; xlb];
  ubw = [ubw; xub];
  w0 = [w0; x_init];
  % Algebraic states
  Zkj{j} = MX.sym(['Z_' num2str(k) '_' num2str(j)],nalg);
  w = [w(:)', Zkj(j)];
  lbw = [lbw; zlb];
  ubw = [ubw; zub];
  w0 = [w0; z_init];
end
% Loop over collocation points
Xk_end = D(1)*Xk;

for j=1:d
   % Expression for the state derivative at the collocation point
   xp = C(1,j+1)*Xk;
   zp = tol*ones(nalg,1);
   for r=1:d
      xp = xp + C(r+1,j+1)*Xkj{r};
   end
   % Append collocation equations
   [fj, qj] = f(Xkj{j}, Zkj{j}, Uk);
   g = [g(:)', {h*fj - [xp;zp]}];
   lbg = [lbg; zeros(nx,1)];
   ubg = [ubg; zeros(nx,1)];
   % Add contribution to the end state
   Xk_end = Xk_end + D(j+1)*Xkj{j};
   % Add contribution to quadrature function
   J = J + B(j+1)*qj*h;
end
% New NLP variable for state at end of interval
% Differential states
Xk = MX.sym(['X_' num2str(k+1)], ndiff);
w = [w(:)', {Xk}];
lbw = [lbw; xlb];
ubw = [ubw; xub];
w0 = [w0; x_init];
% Add inequality constraint
g = [g(:)', {Uk(2) + Uk(1)*rho*cp*(Xk(4)-T2)}];
lbg = [lbg; Q_demand((i-1)+k+1)];
ubg = [ubg; Q_demand((i-1)+k+1)];

g = [g(:)', {Uk(3) + q_L1*rho*cp*(T1-Xk(1))}];
lbg = [lbg; Q_supply(i)];
ubg = [ubg; Q_supply(i)];
```

```
      % Add equality constraint
      g = [g(:)', {Xk_end-Xk}];
      lbg = [lbg; zeros(ndiff,1)];
      ubg = [ubg; zeros(ndiff,1)];
    end

  else
    % Start with an empty NLP
    w={};
    w0 = w_stored;
    lbw = [];
    ubw = [];
    J = 0;
    g={};
    lbg = [];
    ubg = [];
    % Apply the first control from the previous solution
    w0(1:ndiff+nu) = [x_init; u_init];

    % "Lift" initial conditions
    % Differential states
    Xk = MX.sym('X0', ndiff);
    w = [w(:)', {Xk}];
    lbw = [lbw; w0(1:ndiff)];
    ubw = [ubw; w0(1:ndiff)];
    % Formulate the NLP
    for k=0:N-1
       % New NLP variable for the control
       Uk = MX.sym(['U_' num2str(k)],nu);
       w = [w(:)', {Uk}];
       lbw = [lbw; ulb];
       ubw = [ubw; uub];
       % State at collocation points
       Xkj = {};
       Zkj = {};
       for j=1:d
           % Differential states
           Xkj{j} = MX.sym(['X_' num2str(k) '_' num2str(j)],ndiff);
           w = [w(:)', Xkj(j)];
           lbw = [lbw; xlb];
           ubw = [ubw; xub];
           % Algebraic states
           Zkj{j} = MX.sym(['Z_' num2str(k) '_' num2str(j)],nalg);
           w = [w(:)', Zkj(j)];
           lbw = [lbw; zlb];
           ubw = [ubw; zub];
       end

       % Loop over collocation points
       Xk_end = D(1)*Xk;
       for j=1:d
           % Expression for the state derivative at the collocation point
           xp = C(1,j+1)*Xk;
           zp = tol*ones(nalg,1);
           for r=1:d
              xp = xp + C(r+1,j+1)*Xkj{r};
```

```
            end

            % Append collocation equations
            [fj, qj] = f(Xkj{j},Zkj{j},Uk);
            g = [g(:)', {h*fj - [xp;zp]}];
            lbg = [lbg; zeros(nx,1)];
            ubg = [ubg; zeros(nx,1)];

            % Add contribution to the end state
            Xk_end = Xk_end + D(j+1)*Xkj{j};
            % Add contribution to quadrature function
            J = J + B(j+1)*qj*h;
        end
        % New NLP variable for state at end of interval
        % Only for differential states
        Xk = MX.sym(['X_' num2str(k+1)], ndiff);
        w = [w(:)', {Xk}];
        lbw = [lbw; xlb];
        ubw = [ubw; xub];

        % Add inequality constraint
        g = [g(:)', {Uk(2) + Uk(1)*rho*cp*(Xk(4)-T2)}];
        lbg = [lbg; Q_demand((i-1)+k+1)];
        ubg = [ubg; Q_demand((i-1)+k+1)];
        g = [g(:)', {Uk(3) + q_L1*rho*cp*(T1-Xk(1))}];
        lbg = [lbg; Q_supply(i)];
        ubg = [ubg; Q_supply(i)];
        % Add equality constraint (only differential states)
        g = [g(:)', {Xk_end-Xk}];
        lbg = [lbg; zeros(ndiff,1)];
        ubg = [ubg; zeros(ndiff,1)];
    end
end
%% Create an NLP solver
opts = struct;
opts.ipopt.max_iter = maxiter;
opts.ipopt.print_level = 3;
opts.print_time = 1;
opts.ipopt.tol = tol;
opts.ipopt.acceptable_tol =100*tol; % Optimality convergence tolerance

prob = struct('f', J, 'x', vertcat(w{:}), 'g', vertcat(g{:}));
solver = nlpsol('solver', 'ipopt', prob, opts);
%% Solve the NLP
sol = solver('x0', w0, 'lbx', lbw, 'ubx', ubw,'lbg', lbg, 'ubg', ubg);

if solver.stats.success == 0
    error('Error: Optimal Solution Not Found')
end

i_infeasible(i) = solver.stats.success;
solver_return{i} = solver.stats.return_status;
w_opt = full(sol.x);

%% Store the open loop solution
w_stored = [w_opt((ndiff+nu)+d*nx+1:end); ...
w_opt(end+1-((ndiff+nu)+d*nx):end)];
```

```
%% Simulator
x0 = x_init;
tspan = [0 3600];
p = [q_L1; q_R1; q_L2; w_opt(ndiff+1:ndiff+nu-2); ...
Q_tank; T1; T2; V_hex; V_tank; U_hex; A_hex; rho;...
cp; h_s; A_tank; T_s; n];

options = odeset('RelTol', 1e-8,'Stats','off','OutputFcn', @odeplot);
[t,x] = ode15s(@(t,x) twoPlantModelChen(t,x,p), tspan, x0, options);
u_opt(:,i) = w_opt(ndiff+1:ndiff+nu);
u_init = w_opt(ndiff+1:ndiff+nu);
x_m = x(end,:);
x_opt(:,i) = x_m;
x_init = x_m';
x_init(5)
u_init
i
end
%% Adding additional regularisation term!!!
Cost = Pm*u_opt(2,:) + Pu*(u_opt(1,:).^2+u_opt(3,:).^2);
Cost = [Cost, NaN];

%% Store data
storageNMPC = struct('Demand',Q_demand,'OptimalStates',...
x_opt,'OptimalInputs',u_opt,'Measurement',...
x_opt,'Cost',Cost);
%% Plot results
T = T/3600;
tgrid = linspace(0, T, N+1);
clf;

x_opt = [x_0, x_opt];

%% Plot optimal controls
set(0,'DefaultTextFontName','Times',...
'DefaultTextFontSize',15,...
'DefaultAxesFontName','Times',...
'DefaultAxesFontSize',15,...
'DefaultLineLineWidth',1.5,...
'DefaultLineMarkerSize',7.75,...
'DefaultStairLineWidth',1.5);
set(findall(gcf,'Type','text'),'FontSize',15,'Interpreter','latex');
set(gcf,'color','white');
figure(1)
subplot(311)
plot(tgrid, x_opt(5,:), 'k-')
ylabel('$T_{tank}$ [$^{\circ}$C]','Interpreter','latex')
hold off
grid on
xlim([0 24])
ylim([64 76])

subplot(312)
stairs(tgrid, [u_opt(1,:)*1e3, nan], 'b-')
ylabel('$q_{R2}$ [$\ell$/s]','Interpreter','latex')
grid on
```

```
xlim([0 24])
ylim([5 35])

subplot(313)
stairs(tgrid, [u_opt(3,:), nan], 'color','#77AC30')
hold on
stairs(tgrid, [u_opt(2,:), nan], 'r-')
hold off
xlabel('time [hr]','Interpreter','latex')
ylabel('Power [kW]','Interpreter','latex')
legend('$Q_{dump}$','$Q_{market}$','Interpreter','latex')
grid on
xlim([0 24])
ylim([0 1100])
```

Case two: Standard NMPC on simple TES without storage tank

Listing 8.5: Non-storage system model

```
function dxdt = twoPlantModelDirect(~,x,p)
%% Description of the states
T_L1 = x(1);
T_R1 = x(2);
%% Assignment of inputs and disturbances
% Input variables u
q_L1 = p(1);
q_R1 = p(2);
% Distubances d
T1 = p(3);
T2 = p(4);
% Design and physical parameters
V_hex = p(5);
U_hex = p(6);
A_hex = p(7);
rho = p(8);
cp = p(9);
n = p(10);
%% ODEs
dxdt = [(1/V_hex)*(q_L1*(T1-T_L1) - (U_hex*A_hex/(rho*cp))*...
        (0.5*((abs(T1 - T_R1))^n + (abs(T_L1 - T2))^n))^(1/n));

        (1/V_hex)*(q_R1*(T2-T_R1) + (U_hex*A_hex/(rho*cp))*...
        (0.5*((abs(T1 - T_R1))^n + (abs(T_L1 - T2))^n))^(1/n))];
end
```

Listing 8.6: Non-storage parameters

```
%  Parameters_TES.m file
% Defenition of all constant parameters used in TES
%% HEX Parameters
V_hex = 0.5;        %Heat exchanger volume (shell or tube side) [m^3]
A_hex = 300;        %HEX heat transfer area [m^2]
U_hex = 0.5;        %Overall heat transfer coefficient [kW/m^2K]
%% Storage fluid parameters
```

```
rho= 1000;          %Density [kg/m^3]
cp = 4.186;         %Specific heat capacity of fluid [kJ/kgK]
%% Common Parameters
n = 1/3;
T1 = 95;            %Supply side intet temperature [C]
T2 = 20;            %Consumer side intet temperature [C]
T_s = 15;           %Ambient temperature [C]
Pm = 1e-3;          %Commercial energy cost
% Pt = 5e-6;        %Direct solar heating cost
Pu = 50e-6;         %Regularization weight
%% Energy supply and demand profile
Qdemand = 6000;
Qsupply = 6000;
%% NLP solver parameters
tol = 10e-8;        % Desired convergence tolerance (relative)
maxiter = 5000;
%% Declaration of constant inputs
q_L1=0.05;          % [m^3/s]
Q_tank=0;
h_dot = (U_hex*A_hex)/(rho*cp);
```

Listing 8.7: Non-storage steady state optimization

```
addpath('C:\Users\dingn\Desktop\Thesis\casadi-windows-matlabR2016a-v3.5.1')
import casadi.*
%% Model parameters
run Parameters_NoStorage.m
%% Declare model variables
% Diffential states
x1 = SX.sym('x1');      % T_L1
x2 = SX.sym('x2');      % T_R2
x = [x1; x2];
% Algebraic states
x3 = SX.sym('x3');
x4 = SX.sym('x4');
z = [x3; x4];
% Control inputs
u1 = SX.sym('u1');      %q_R2
u2 = SX.sym('u2');      %Q_Market
u3 = SX.sym('u3');      %Q_dump
u = [u1; u2; u3];
%% Model equations (DEAs)

xdot = [(1/V_hex)*(q_L1*(T1-x1) - (h_dot)*(0.5*(x3 + x4))^(1/n));

        (1/V_hex)*(u1*(T2-x2) + (h_dot)*(0.5*(x3 + x4))^(1/n));

        T1 - x2 - x3^(1/n);

        x1 - T2 - x4^(1/n)];

diff = xdot(1:2);
alg = xdot(3:4);

%% Lower and upper bounds of x,z,u
% LB Differential states
```

```
x1_lb = 0;
x2_lb = 0;
% LB Algebraic states
x3_lb = -Inf;
x4_lb = -Inf;
% LB Control inputs
u1_lb=0;
u2_lb=0;
u3_lb=0;
% LB Vertical concatenation
xlb = vertcat(x1_lb,x2_lb);
zlb = vertcat(x3_lb,x4_lb);
ulb = vertcat(u1_lb,u2_lb,u3_lb);
% UB Differential states
x1_ub = 100;
x2_ub = 100;
% UB Algebraic states
x3_ub = +Inf;
x4_ub = +Inf;
% UB Control inputs
u1_ub = 0.05;                          % [m^3/s]
u2_ub = +Inf;
u3_ub = +Inf;
% UB Vertical concatenation
xub = vertcat(x1_ub,x2_ub);
zub = vertcat(x3_ub,x4_ub);
uub = vertcat(u1_ub,u2_ub,u3_ub);
%% Declare two satisfaction equality constraint as symbolic variables
C1 = Qdemand - u2 - u1*rho*cp*(x2-T2);
C2 = Qsupply - u3 -q_L1*rho*cp*(T1-x1);
C1_lb = 0;
C1_ub = 0;
C2_lb = 0;
C2_ub = 0;
%% Initial guess
% Differential states 80 60 45 30 50
x10 = 60;
x20 = 30;
% Algebraic states
x30 = 1;
x40 = 1;
X0=[x10; x20];
Z0=[x30; x40];
% Control inputs
u10 = 0;
u20 = 0;
u30 = 0;
U0 = [u10; u20; u30];
%% Steady-state optimization
% preparing symbolic variables
w = {};
% preparing numeric variables and bounds
w0 = [];
lbw = [];
ubw = [];
% preparing symbolic constraints
g = {};
```

```
% preparing numeric bounds
lbg = [];
ubg = [];
% declaring them symbolic
w = {w{:},x,z,u};
lbw = [lbw;xlb;zlb;ulb];
ubw = [ubw;xub;zub;uub];
w0 = [w0;X0;Z0;U0];

% Add the system model as constraints
g = [g(:)',{vertcat(diff,alg)},{C1},{C2}];
lbg = [lbg;zeros(4,1);C1_lb;C2_lb];    % Steady-state optimisation, dx/dt=0
ubg = [ubg;zeros(4,1);C1_ub;C2_ub];

% Cost function
L = Pm*u2;
% Economic objective
J = L;
nlp = struct('x',vertcat(w{:}),'f',J,'g',vertcat(g{:}));
solver = nlpsol('solver','ipopt',nlp); % NLP solver IPOPT
sol = solver('x0',w0,'lbx',lbw,'ubx',ubw,'lbg',lbg,'ubg',ubg);

%% Extracting solutions
w_opt_SS = full(sol .x);
x_init=w_opt_SS(1:2);
z_init=w_opt_SS(3:4);
u_init=w_opt_SS(5:end);
```

Listing 8.8: Non-storage standard NMPC

```
clear;
clc;
close all;
addpath('C:\Users\dingn\Desktop\Thesis\casadi-windows-matlabR2016a-v3.5.1')
import casadi.*
%% Parameters
run Parameters_NoStorage.m
%% Demand supply profile
Q_demand = [1500*ones(12,1); 3500*ones(12,1)];
Q_supply = 2500*ones(24,1);
%% Degree of interpolating polynomial
d = 3;
% Get collocation points
tau_root = [0 collocation_points(d, 'radau')]; %can be 'legendre'
% Coefficients of the collocation equation
C = zeros(d+1,d+1);
% Coefficients of the continuity equation
D = zeros(d+1, 1);
% Coefficients of the quadrature function
B = zeros(d+1, 1);
%% Construct polynomial basis
for j=1:d+1
% Construct Lagrange polynomials to get the polynomial basis
% at the collocation point
    coeff = 1;
    for r=1:d+1
```

```
        if r ~= j
            coeff = conv(coeff, [1, -tau_root(r)]);
            coeff = coeff / (tau_root(j)-tau_root(r));
        end
    end
% Evaluate the polynomial at the final time to get the
% coefficients of the continuity equation
    D(j) = polyval(coeff, 1.0);

% Evaluate the time derivative of the polynomial at all collocation
% points to get the coefficients of the continuity equation
    pder = polyder(coeff);
    for r=1:d+1
        C(j,r) = polyval(pder, tau_root(r));
    end
% Evaluate the integral of the polynomial to get the coefficients
% of the quadrature function
    pint = polyint(coeff);

    B(j) = polyval(pint, 1.0);
 end
%% Time horizon
T = 24*60*60;
ndiff = 2;                %Number of differential states
nalg = 2;                 %Number of algebraic states
nu = 3;                   %Number of controls
nx = nalg + ndiff;        %Total number of states
zub = Inf*ones(nalg,1);
zlb = -Inf*ones(nalg,1);
%% SS-opt
run RTO.m
x_0 = x_init;
%% Declare model variables
x1 = SX.sym('x1');      % T_L1
x2 = SX.sym('x2');      % T_R2
x = [x1; x2];
x3 = SX.sym('x3');      % a
x4 = SX.sym('x4');      % b
z = [x3; x4];
u1 = SX.sym('u1');      %q_R2
u2 = SX.sym('u2');      %Q_Market
u3 = SX.sym('u3');      %Q_dump
u = [u1; u2; u3];
%% Model equations
xdot = [(1/V_hex)*(q_L1*(T1-x1) - (h_dot)*(0.5*(x3 + x4))^(1/n));

        (1/V_hex)*(u1*(T2-x2) + (h_dot)*(0.5*(x3 + x4))^(1/n));

        T1 - x2 - x3^(1/n);

        x1 - T2 - x4^(1/n)];
%% Objective term
L = Pm*u2;
%% Continuous time dynamics
f = Function('f', {x, z, u}, {xdot, L});
% Control discretization
N = 24; % number of control intervals
```

```
M = 24; % number of MPC loops
h = T/N;
period = N;
%% Prepare output variables
x_opt = zeros(ndiff,M);
u_opt = zeros(nu,M);
i_infeasible = zeros(M,1);
solver_return = {};
%% Predicted Demand
Q_demand = [Q_demand; Q_demand];
Q_supply = [Q_supply; Q_supply];
%% MPC loop
for i=1:M
    % Start with an empty NLP
    w={};
    w0 = [];
    lbw = [];
    ubw = [];
    J = 0;
    g={};
    lbg = [];
    ubg = [];

    % "Lift" initial conditions
    % Differential states
    Xk = MX.sym('X0', ndiff);
    w = {w{:}, Xk};
    lbw = [lbw; x_init];
    ubw = [ubw; x_init];
    w0 = [w0; x_init];

    % Algebraic states
    Zk = MX.sym('Z0', nalg);
    w = {w{:}, Zk};
    lbw = [lbw; z_init];
    ubw = [ubw; z_init];
    w0 = [w0; z_init];
    % Formulate the NLP
    for k=0:N-1
    % New NLP variable for the control
        Uk = MX.sym(['U_' num2str(k)], nu);
        w = {w{:}, Uk};
        lbw = [lbw; ulb];
        ubw = [ubw; uub];
        w0 = [w0; u_init];

        % State at collocation points
        Xkj = {};

        Zkj = {};
        for j=1:d
            % Differential states
            Xkj{j} = MX.sym(['X_' num2str(k) '_' num2str(j)], ndiff);
            w = {w{:}, Xkj{j}};
            lbw = [lbw; xlb];
            ubw = [ubw; xub];
            w0 = [w0; x_init];
```

```
            % Algebraic states
            Zkj{j} = MX.sym(['Z_' num2str(k) '_' num2str(j)], nalg);
            w = {w{:}, Zkj{j}};
            lbw = [lbw; zlb];
            ubw = [ubw; zub];
            w0 = [w0; z_init];
        end
        % Loop over collocation points
        Xk_end = D(1)*Xk;

        for j=1:d
        % Expression for the state derivative at the collocation point
        xp = C(1,j+1)*Xk;
        zp = zeros(nalg,1);
            for r=1:d
                xp = xp + C(r+1,j+1)*Xkj{r};
            end

        % Append collocation equations
        [fj, qj] = f(Xkj{j}, Zkj{j}, Uk);
        g = {g{:}, h*fj - [xp; zp]};
        lbg = [lbg; zeros(nx,1)];
        ubg = [ubg; zeros(nx,1)];

        % Add contribution to the end state
        Xk_end = Xk_end + D(j+1)*Xkj{j};
        % Add contribution to quadrature function
        J = J + B(j+1)*qj*h;
        end
        % New NLP variable for state at end of interval
        % Differential states
        Xk = MX.sym(['X_' num2str(k+1)], ndiff);
        w = {w{:}, Xk};
        lbw = [lbw; xlb];
        ubw = [ubw; xub];
        w0 = [w0; x_init];

        % Add inequality constraint
        g = {g{:}, Uk(2) + Uk(1)*rho*cp*(Xk(2)-T2)};
        lbg = [lbg; Q_demand((i-1)+k+1)];
        ubg = [ubg; Q_demand((i-1)+k+1)];

        g = {g{:}, Uk(3) + q_L1*rho*cp*(T1-Xk(1))};
        lbg = [lbg; Q_supply(i)];
        ubg = [ubg; Q_supply(i)];

        % Add equality constraint
        g = {g{:}, Xk_end - Xk};
        lbg = [lbg; zeros(ndiff,1)];
        ubg = [ubg; zeros(ndiff,1)];
    end

% Create an NLP solver
opts = struct;
opts.ipopt.max_iter = maxiter;%5000;
opts.ipopt.print_level = 5; %0,3
opts.print_time = 1; %0,1
```

```
opts.ipopt.tol = tol;
opts.ipopt.acceptable_tol = 100*tol; % optimality convergence tolerance

prob = struct('f', J, 'x', vertcat(w{:}), 'g', vertcat(g{:}));
solver = nlpsol('solver', 'ipopt', prob, opts);
% Solve the NLP
sol = solver('x0', w0, 'lbx', lbw, 'ubx', ubw,'lbg', lbg, 'ubg', ubg);
i_infeasible(i) = solver.stats.success;
solver_return{i} = solver.stats.return_status;
w_opt = full(sol.x);
if solver.stats.success == 0
    error('Error: Optimal Solution Not Found')
end
% Simulator
x0 = x_init;
tspan = [0 3600];
p = [q_L1; w_opt(nx+1:nx+nu-2); T1; T2; V_hex; U_hex; A_hex; rho;...
cp; n];
options = odeset('RelTol',1e-5,'Stats','off','OutputFcn',@odeplot);
[t,x] = ode15s(@(t,x) twoPlantModelDirect(t,x,p),tspan,x0,options);
u_opt(:,i) = w_opt(nx+1:nx+nu);
u_init = w_opt(nx+1:nx+nu);
x_m = x(end,:);
x_opt(:,i) = x_m(1:ndiff);
x_init = transpose(x_m(1:ndiff));

x_init
u_init
i
end
Cost = Pm*u_opt(2,:);
Cost = [Cost, NaN];
% Store data
nostorageNMPC = struct('Demand',Q_demand,'OptimalStates',...
x_opt,'OptimalInputs',u_opt,'Measurement',...
x_opt,'Cost',Cost);

%% Plot results
T = T/3600;
tgrid = linspace(0, T, N+1);
clf;
x_opt = [x_0, x_opt];
set(0,'DefaultTextFontName','Times',...
'DefaultTextFontSize',15,...
'DefaultAxesFontName','Times',...
'DefaultAxesFontSize',15,...
'DefaultLineLineWidth',1.5,...
'DefaultLineMarkerSize',7.75,...
'DefaultStairLineWidth',1.5);
set(findall(gcf,'Type','text'),'FontSize',15,'Interpreter','latex');
set(gcf,'color','white');
figure(1)
subplot(311)
plot(tgrid, x_opt(1,:), '-r')
hold on
plot(tgrid, x_opt(2,:), '-b')
% xlabel('time [hr]')
```

```
ylabel('T [$\circ$C]','Interpreter','latex')
legend('$T_{1,o}$','$T_{2,o}$','Interpreter','latex')
hold off
xlim([0 24])
ylim([40 100])
grid on

subplot(312)
stairs(tgrid, [u_opt(1,:)*1e3, nan], '-b')
hold off
% xlabel('time [hr]')
ylabel('$q_{2}$ [$\ell$/s]', 'Interpreter','latex')
xlim([0 24])
ylim([0 10])
grid on

subplot(313)
stairs(tgrid, [u_opt(3,:), nan], '-g')
hold on
stairs(tgrid, [u_opt(2,:), nan], '-r')
hold off
xlabel('time [hr]','Interpreter','latex')
ylabel('Power [kW]','Interpreter','latex')
legend('$Q_{dump}$','$Q_{M}$','Interpreter','latex')
xlim([0 24])
ylim([0 1200])
grid on
```

Case three: Standard NMPC on simple TES with direct solar heating

Listing 8.9: Simple TES system model

```
function [xdot, alg, par, F_integrator, f, fk, ss_opt, dx1dot] = TESmdl(par)
import casadi.*
%% Declare model variables
% Differential states
T_L1 = MX.sym('T_L1');
T_R1 = MX.sym('T_R1');
T_L2 = MX.sym('T_L2');
T_R2 = MX.sym('T_R2');
T_tank = MX.sym('T_tank');
x = [T_L1; T_R1; T_L2; T_R2; T_tank];
% Algebraic states
z1 = MX.sym('z1');
z2 = MX.sym('z2');
z3 = MX.sym('z3');
z4 = MX.sym('z4');
z = [z1; z2; z3; z4];
% Inputs
q_R2 = MX.sym('q_R2');
Q_market = MX.sym('Q_market');
Q_dump = MX.sym('Q_dump');
Q_tank = MX.sym('Q_tank');                    % Q_solar = Q_tank
u = [q_R2; Q_market; Q_dump; Q_tank];
% Disturbances
T1 = MX.sym('T1');
```

```
T2 = MX.sym('T2');
T_s = MX.sym('T_s');
Q_supply = MX.sym('Q_supply');
Q_demand = MX.sym('Q_demand');
d = vertcat(T1, T2, T_s, Q_supply, Q_demand);
%% Variable counting
par.nx = length(x);
par.nz = length(z);
par.nu = length(u);
par.nd = length(d);
%% Model equations
V_hex = par.V_hex;
V_tank = par.V_tank;
n = par.n;
q_L1 = par.q_L1;
q_R1 = par.q_R1;
q_L2 = par.q_L2;
h_dot = par.h_dot;
h_t_dot = par.h_t_dot;
rho = par.rho;
cp = par.cp;
T1 = par.T1;
T2 = par.T2;

xdot = [(1/V_hex)*(q_L1*(T1-T_L1) - h_dot*(0.5*(z1 + z2))^(1/n));
        (1/V_hex)*(q_R1*(T_tank-T_R1) + h_dot*(0.5*(z1 + z2))^(1/n));
        (1/V_hex)*(q_L2*(T_tank-T_L2) - h_dot*(0.5*(z3 + z4))^(1/n));
        (1/V_hex)*(q_R2*(T2-T_R2) + h_dot*(0.5*(z3 + z4))^(1/n));
        (1/V_tank)*(q_R1*(T_R1-T_tank) + q_L2*(T_L2-T_tank) + ...
        (Q_tank/(rho*cp)-h_t_dot*(T_tank-T_s)))];

alg = [T1-T_R1-z1^(1/n);
       T_L1-T_tank-z2^(1/n);
       T_L2-T2-z3^(1/n);
       T_tank-T_R2-z4^(1/n)];

%% Objective term
L = par.Pm*Q_market + par.Pt*Q_tank;
%% Continuous time dynamics
f = Function('f', {x, z, u, d}, {xdot, alg, L}, {'x','z','u','d'},...
    {'xdot','alg','qj'});

dae = struct('x', x, 'z', z, 'p', vertcat(u, d), 'ode', xdot, 'alg',...
    alg, 'quad', L);

opts = struct('t0', 0, 'tf', par.h);
% use collocation instead of IDAS (due to a linesearch backtracking error)
F_integrator = integrator('F', 'collocation', dae, opts);

% Conversion to discrete model
dx1 = [];

for i = 1:par.nx
    dx1 = [dx1; x(i) + par.h*xdot(i)];
end

fk = Function('fk', {x, u, d}, {dx1});
```

```
dx1_x = jacobian(dx1, x);
dx1_u = jacobian(dx1, u);

dx1dot = Function('dx1dot', {x, u, d}, {dx1_x, dx1_u});

%% Solve steady state optimization problem
w = {};
w0 = [];
lbw = [];
ubw = [];
lbg = [];
ubg = [];
g = {};
w = [w(:)', {x}, {z}, {u}, {d}];   % Disturbances are constant!!!
lbw = [lbw; par.xlb; par.zlb; par.ulb; par.Q_tanklb; par.dist];
ubw = [ubw; par.xub; par.zub; par.uub; par.Q_tankub; par.dist];
w0 = [w0; par.x_init; par.z_init; par.u_init; par.dist];

J = L;

g = [g(:)', {xdot}, {alg}];
lbg = [lbg; zeros(size(xdot,1),1); zeros(size(alg,1),1)];
ubg = [ubg; zeros(size(xdot,1),1); zeros(size(alg,1),1)];

g = [g(:)', {u(2) + u(1)*par.rho*par.cp*(x(4)-d(2))-d(4)}];
lbg = [lbg; 0];
ubg = [ubg; 0];

g = [g(:)', {u(3) + par.q_L1*par.rho*par.cp*(d(1)-x(1))-d(5)}];
lbg = [lbg; 0];
ubg = [ubg; 0];

opts = struct;
opts.ipopt.max_iter = par.maxiter;
opts.ipopt.print_level = 0;
opts.print_time = 0;
opts.ipopt.tol = par.tol;
opts.ipopt.acceptable_tol = 100*par.tol;

ss_nlp = struct('x', vertcat(w{:}), 'f', J, 'g', vertcat(g{:}));
ss_solver = nlpsol('ss_solver', 'ipopt', ss_nlp, opts);

% Solve the NLP
sol_ss = ss_solver('x0', w0, 'lbx', lbw, 'ubx', ubw, 'lbg', lbg, 'ubg', ubg);
ss_opt = full(sol_ss.x);

end
```

Listing 8.10: Simple TES system parameters

```
function par = TES_Parameters()
%% Parameters (in l)
par.V_hex = 500;            % Heat exchanger volume (shell or tube side)
par.V_tank = 1e+6;          % Storage tank volume
par.U_hex = 0.5;            % Overall heat transfer coefficient [kW/m^2K]
```

```
par.A_hex = 300;            % Heat transfer area [m^2]
par.rho = 1;                % Density [kg/m^3]
par.cp = 4.186;             % Specific heat capacity of fluid [kJ/kgK]
par.h_s = 5e-4;             % Heat loss coefficient [kW/m^2K]
par.n = 1/3;                % Approximation index
par.A_tank = 100;           % Tank heat loss area [m^2]
par.Pm = 1e-3;
par.Pt = 5e-6;
par.Pu = 50e-6;
par.Px = 5e-6;
par.Q_demand_0 = 2000;
par.tol = 1e-6;
par.maxiter=2e4;
par.std = 2.31;
%% Simulation parameters
par.T = 24*3600;
par.h = 3600;
par.nIter = par.T/par.h;
par.N = 24;
par.d = 3;
%% Disturbances (Boundary conditions)
par.T1 = 95;                % Plant 1 temperature (source)[C]
par.T2 = 20;                % Plant 2 temperature (sink)[C]
par.T_s = 15;               % Ambient temperature [C]
%% Constant inputs
par.q_L1 = 50;
par.q_R1 = 50;
par.q_L2 = 50;
%% Bounds
% States UB
par.T_L1ub = 200;
par.T_R1ub = 200;
par.T_L2ub = 200;
par.T_R2ub = 200;
par.T_tankub = 200;
% Inputs UB
par.q_L1ub = 50;
par.q_R1ub = 50;
par.q_L2ub = 50;
par.q_R2ub = 50;
par.Q_tankub = 3000;
par.Qmub = +Inf;                % Q_market
par.Qdub = +Inf;                % Q_dump
% States LB
par.T_L1lb = 0;
par.T_R1lb = 0;
par.T_L2lb = 0;
par.T_R2lb = 0;
par.T_tanklb = 30;
% Inputs LB
par.q_L1lb = 0;
par.q_R1lb = 0;
par.q_L2lb = 0;
par.q_R2lb = 0;
par.Q_tanklb = 0;
par.Qmlb = 0;
par.Qdlb = 0;
```

```
% Bounds concatenation
par.xub = vertcat(par.T_L1ub, par.T_R1ub, par.T_L2ub, par.T_R2ub,...
    par.T_tankub);
par.xlb = vertcat(par.T_L1lb, par.T_R1lb, par.T_L2lb, par.T_R2lb,...
    par.T_tanklb);
par.uub = vertcat(par.q_R2ub, par.Qmub, par.Qdub);
par.ulb = vertcat(par.q_R2lb, par.Qmlb, par.Qdlb);
par.zub = +Inf*ones(4,1);
par.zlb = -Inf*ones(4,1);

%% Initial values
par.x_init = [80; 60; 40; 30; 55];
par.x_0 = par.x_init;
par.u_init = [0; 0; 0 ; 0];
par.u_n1 = par.u_init;
par.z_init = 3*ones(4,1);
par.h_dot = (par.U_hex*par.A_hex)/(par.rho*par.cp);
par.h_t_dot = (par.h_s*par.A_tank)/(par.rho*par.cp);

end
```

Listing 8.11: Simple TES system NLP problem

```
function [solver, par] = TES_createNLP(f, par, mpcloop, options)

import casadi.*
%% Construct polynomial basis
% Degree of interpolating polynomial
d = 3;
% Get collocation points
tau_root = [0 collocation_points(d, 'radau')]; %can be 'legendre'
% Coefficients of the collocation equation
C = zeros(d+1,d+1);
% Coefficients of the continuity equation
D = zeros(d+1, 1);
% Coefficients of the quadrature function
B = zeros(d+1, 1);
% Construct polynomial basis
for j=1:d+1
    % Construct Lagrange polynomials to get the polynomial basis
    % at the collocation point
    coeff = 1;
    for r=1:d+1
        if r ~= j
            coeff = conv(coeff, [1, -tau_root(r)]);
            coeff = coeff / (tau_root(j)-tau_root(r));
        end
    end
    % Evaluate the polynomial at the final time to get the
    % coefficients of the continuity equation
    D(j) = polyval(coeff, 1.0);

    % Evaluate the time derivative of the polynomial at all collocation
    % points to get the coefficients of the continuity equation
    pder = polyder(coeff);
    for r=1:d+1
```

```
        C(j,r) = polyval(pder, tau_root(r));
    end

    % Evaluate the integral of the polynomial to get the coefficients
    % of the quadrature function
    pint = polyint(coeff);
    B(j) = polyval(pint, 1.0);
end
%% Start with an empty NLP
w = {};
w0 = [];
lbw = [];
ubw = [];
J = 0;
g = {};
lbg = [];
ubg = [];

% States at stage k
Xk = MX.sym('X0', par.nx);
w = [w(:)', {Xk}];
lbw = [lbw; par.xlb];
ubw = [ubw; par.xub];
w0 = [w0; par.x_init];

% "Lift" initial conditions by setting initial value as a parameter
p = MX.sym('p', par.nx);

g = [g(:)', {Xk - p}];
lbg = [lbg; zeros(par.nx,1)];
ubg = [ubg; zeros(par.nx,1)];

% Formulate the NLP
for k = 0:par.N-1
    % New NLP variable for the control
    Uk = MX.sym(['U_' num2str(k)], par.nu);
    w = [w(:)', {Uk}];
    lbw = [lbw; par.ulb; 0];
    ubw = [ubw; par.uub; par.Q_solar(mpcloop)];
    w0 = [w0; par.u_init];

    % State at collocation points
    Xkj = cell(1,d);
    Zkj = cell(1,d);

    for j=1:d
        % Differential states
        Xkj{j} = MX.sym(['X_' num2str(k) '_' num2str(j)], par.nx);
        w = [w(:)', Xkj(j)];
        lbw = [lbw; par.xlb];
        ubw = [ubw; par.xub];
        w0 = [w0; par.x_init];

        % Algebraic states
        Zkj{j} = MX.sym(['Z_' num2str(k) '_' num2str(j)], par.nz);
        w = [w(:)', Zkj(j)];
        lbw = [lbw; par.zlb];
```

```
        ubw = [ubw; par.zub];
        w0 = [w0; par.z_init];
    end

    % Loop over collocation points
    Xk_end = D(1)*Xk;

    for j=1:d
        % Expression for the state derivative at the collocation point
        xp = C(1,j+1)*Xk;
        zp = par.tol*ones(par.nz,1);
        for r=1:d
            xp = xp + C(r+1,j+1)*Xkj{r};
        end

        % Append collocation equations
        [fxj, fzj, qj] = f(Xkj{j}, Zkj{j}, Uk, par.dist);
        g = [g(:)', {par.h*[fxj; fzj] - [xp; zp]}];
        lbg = [lbg; zeros(par.nx + par.nz,1)];
        ubg = [ubg; zeros(par.nx + par.nz,1)];

        % Add contribution to the end state
        Xk_end = Xk_end + D(j+1)*Xkj{j};

        % Add contribution to quadrature function
        J = J + B(j+1)*qj*par.h;
    end

    % Add regularization terms to economic cost objective
    switch options.reg_terms
        case 'on'
            Phi_reg = (Xk -par.xreg)'*par.Q_reg*(Xk -par.xreg);
        case 'off'
            Phi_reg = 0;
    end

    if k>0
        switch options.control_penalty
            case 'on'
                Phi_control = (Uk - Uk_prev)'*par.R_delta*(Uk - Uk_prev);
            case 'off'
                Phi_control = 0;
        end
    else
        Phi_control = 0;
    end

    J = J + Phi_reg + Phi_control;
    Uk_prev = Uk;

    % Add inequality constraint
    g = [g(:)', {Uk(2) + Uk(1)*par.rho*par.cp*(Xk(4)-par.dist(2))}];
    lbg = [lbg; par.Q_demand((mpcloop-1)+k+1)];
    ubg = [ubg; par.Q_demand((mpcloop-1)+k+1)];

    g = [g(:)', {Uk(3) + par.q_L1*par.rho*par.cp*(par.dist(1)-Xk(1))}];
    lbg = [lbg; par.Q_supply(mpcloop)];
```

```
    ubg = [ubg; par.Q_supply(mpcloop)];

    % New NLP variable for state at end of interval
    Xk = MX.sym(['X_' num2str(k+1)], par.nx);
    w = [w(:)', {Xk}];
    lbw = [lbw; par.xlb];
    ubw = [ubw; par.xub];
    w0 = [w0; par.x_init];

    % Shooting gap constraint (continuity for differential states)
    g = [g(:)', {Xk_end-Xk}];
    lbg = [lbg; zeros(par.nx,1)];
    ubg = [ubg; zeros(par.nx,1)];
end

% Add contribution of terminal constraint to objective
if k == par.N-1
    switch options.termin_con
        case 'on'
            Vn = (Xk - par.x_final)'*par.P*(Xk - par.x_final);

            % Add terminal constraints
            g = [g(:)', {(Xk - par.x_final)'*(Xk - par.x_final)}];
            lbg = [lbg; 0];
            ubg = [ubg; par.Cf];
        case 'off'
            Vn = 0;
    end
    J = J + Vn;
end

par.w0 = w0;
par.lbw = lbw;
par.ubw = ubw;
par.lbg = lbg;
par.ubg = ubg;

% Create an NLP solver
opts = struct;
opts.ipopt.max_iter = par.maxiter;
opts.ipopt.print_level = 0;
opts.print_time = 0;
opts.ipopt.tol = par.tol;
opts.ipopt.acceptable_tol = 100*par.tol; % optimality convergence tolerance

prob = struct('f', J, 'x', vertcat(w{:}), 'p', p, 'g', vertcat(g{:}));
solver = nlpsol('solver', 'ipopt', prob, opts);

par.prob = prob;
par.d = d;
end
```

Listing 8.12: Simple TES system main simulator

```
clc;
clear;
```

```
close all;
addpath('C:\Users\dingn\Desktop\Thesis\casadi-windows-matlabR2016a-v3.5.1')
import casadi.*
%% Call TES parameters
par = TES_Parameters();
%% Surplus supply
Q_supply = 2500*ones(24,1);
%% Load California demand
filename = 'CarliforniaDailyPowerDemand.xlsx';
sheet = 1;
xlRange = 'D2:D25';
Q_demand = xlsread(filename,xlRange);
%% Load direct solar supply
filename_1 = 'SolarSupply.xlsx';
sheet_1 = 1;
xlRange_1 = 'E2:E25';
par.Q_solar = xlsread(filename_1,xlRange_1);
%% Predicted (Expected) Demand
par.Q_demand = [Q_demand; Q_demand];
par.Q_supply = [Q_supply; Q_supply];
%% Disturbances
par.dist = [par.T1;
    par.T2;
    par.T_s;
    par.Q_supply(1);
    par.Q_supply(1);];
%% Call TES model (steady state optimization)
[xdot, alg, par, F_integrator, f, fk, ss_opt, dx1dot] = TESmdl(par);

par.x_init = ss_opt(1:par.nx);
par.z_init = ss_opt(par.nx+1:par.nx+par.nz);
par.u_init = ss_opt(par.nx+par.nz+1:par.nx+par.nz+par.nu);

%% Variable counting
par.NXD = par.N*par.nx*(par.d+1);     %TN of x
par.NXA = par.N*par.nz*par.d;         %TN of z
par.NU = par.N*par.nu;                %TN of u
par.NXF = par.nx;                     %TN of end point variables(diff. only)
par.NV = par.NXD + par.NXA + par.NU + par.NXF; %TN of NLP variables
%% Prepare output variables
x_actual = zeros(par.nIter+1, par.nx);
x_actual(1,:) = par.x_init;
u_opt = NaN(par.nIter+1, par.nu);
i_infeasible = zeros(par.nIter, 1);
solver_return = cell(par.nIter, 1);
w_stored = zeros(par.NV, 1);

options = struct('reg_terms', 'off', 'control_penalty', 'off', ...
    'termin_con', 'off');
%% MPC loop
for mpcloop = 1:par.nIter

    [solver, par] = TES_createNLP(f, par, mpcloop, options);

    % Solve the NLP
    if mpcloop ~= 1
        par.w0 = w_stored;
```

```
    end

    sol = solver('x0', par.w0, 'p', par.x_init, 'lbx', par.lbw, 'ubx', ...
        par.ubw,'lbg', par.lbg, 'ubg', par.ubg);

    if solver.stats.success == 0
        error('Error: Optimal Solution Not Found')
    else
        disp('Success: Optimal Solution found.')
    end

    i_infeasible(mpcloop) = solver.stats.success;
    solver_return{mpcloop} = solver.stats.return_status;
    w_opt = full(sol.x);

    % Store the open loop solution
    w_stored = [w_opt((par.nx+par.nu)+par.d*(par.nx+par.nz)+1:end);...
        w_opt(end+1-((par.nx+par.nu)+par.d*(par.nx+par.nz)):end)];

    % Simulator
    x_in = par.x_init;  %x0
    z_in = par.z_init;  %z0
    u_in = [w_opt(par.nx+1:par.nx+par.nu); par.dist(1:3); ...
        par.Q_supply(mpcloop); par.Q_demand(mpcloop)];

    fprintf('------- Sample %d ----- elapsedTime = %2.0f h----- \n', ...
        mpcloop, mpcloop)

    % ---------- Plant Simulator (offline) ---------------------------
    xpred = F_integrator('x0', x_in,'p', u_in,'z0', z_in);
    par.xf = full(xpred.xf);                              % without noise
    par.zf = full(xpred.zf);
    % ----------------------------------------------------------------

    par.x_init = par.xf;
    par.z_init = par.zf;

    x_actual(mpcloop+1,:) = par.x_init;
    u_opt(mpcloop,:) = u_in(1:par.nu);

fprintf('Expected Demand = %4.2f \t Actual Demand = %4.2f \n', ...
        par.Q_demand(mpcloop), par.Q_demand(mpcloop))
fprintf('Optimal Inputs = %2.4f \t %4.2f \t %4.2f \n', u_in(1:par.nu))
fprintf('Predicted State = %3.2f \t %3.2f \t %3.2f \t %3.2f \t %3.2f\n',...
w_opt(par.nu+(par.d+1)*par.nx+par.d*par.nz+1:par.nu+(par.d+1)*par.nx...
        +par.d*par.nz+par.nx))
fprintf('Actual State =  %3.2f \t %3.2f \t %3.2f \t %3.2f \t %3.2f \n', ...
        par.xf)

end

tgrid = linspace(0, par.T/3600, mpcloop + 1);
%% Plot the results
Plot_TES(par, x_actual, u_opt, tgrid)
```

Case four: Multistage NMPC on simple TES with plant-model mismatch

Listing 8.13: Scenario-based multistage NMPC

```
clear;
clc;
close all;
addpath('C:\Users\dingn\Desktop\Thesis\casadi-windows-matlabR2016a-v3.5.1')
import casadi.*
%% Model parameters
run Parameters_scendae.m
%% Demand profile
Q_demand = [1500*ones(12,1); 3500*ones(12,1)];
%% Actual temperatures
T1_actual = 99;
T2_actual = 18;
%% Uncertain parameter(s) (uncertainty space)
par1 = [90; 95; 100];    %T1
par2 = [15; 20; 25];     %T2
[scens, scen_count] = scenpara(par1,par2);
%% Collocation
% Degree of interpolating polynomial
d = 3;
% Get collocation points
tau_root = [0 collocation_points(d, 'legendre')];
% Coefficients of the collocation equation
C = zeros(d+1,d+1);
% Coefficients of the continuity equation
D = zeros(d+1, 1);
% Coefficients of the quadrature function
B = zeros(d+1, 1);
% Construct polynomial basis
for j=1:d+1
% Construct Lagrange polynomials to get the polynomial basis at
% the collocation point
coeff = 1;
   for r=1:d+1
     if r ~= j
       coeff = conv(coeff, [1, -tau_root(r)]); %convolution
       coeff = coeff / (tau_root(j)-tau_root(r));
     end
   end
  % Evaluate the polynomial at the final time to get the
  % coefficients of the continuity equation
  D(j) = polyval(coeff, 1.0);

  % Evaluate the time derivative of the polynomial at all collocation
  % points to get the coefficients of the continuity equation
  pder = polyder(coeff);
  for r=1:d+1
    C(j,r) = polyval(pder, tau_root(r));
  end
  % Evaluate the integral of the polynomial to get the coefficients
  % of the quadrature function
  pint = polyint(coeff);
  B(j) = polyval(pint, 1.0);
```

```
end

%% Time horizon
T = 24*60*60;          %Prediction horizon time
ndiff = 5;             %number of differential states
nalg = 4;              %number of algebraic states
nu = 4;                %number of controls
nx = nalg + ndiff;     %total number of states
zub = Inf*ones(nalg,1);
zlb = -Inf*ones(nalg,1);
%% Run ssopt
run RTO.m
x_0 = x_init;
%% Declare model variables
x1 = SX.sym('x1');
x2 = SX.sym('x2');
x3 = SX.sym('x3');
x4 = SX.sym('x4');
x5 = SX.sym('x5');
x = [x1; x2; x3; x4; x5];
x6 = SX.sym('x6');
x7 = SX.sym('x7');
x8 = SX.sym('x8');
x9 = SX.sym('x9');
z = [x6; x7; x8; x9];
u1 = SX.sym('u1');                % q_L2
u2 = SX.sym('u2');                % q_R2
u3 = SX.sym('u3');                % Q_tank
u4 = SX.sym('u4');                % Q_demand
u = [u1; u2; u3; u4];
p1 = SX.sym('p1');
p2 = SX.sym('p2');
p = [p1; p2];
%% Model equations
xdot = [(1/V_hex)*(q_L1*(p1-x1) - h_dot*(0.5*(x6 + x7))^(1/n));

        (1/V_hex)*(q_R1*(x5-x2) + h_dot*(0.5*(x6 + x7))^(1/n));

        (1/V_hex)*(u1*(x5-x3) - h_dot*(0.5*(x8 + x9))^(1/n));

        (1/V_hex)*(u2*(p2-x4) + h_dot*(0.5*(x8 + x9))^(1/n));

        (1/V_tank)*(q_R1*(x2-x5)+u1*(x3-x5)+(u3/(rho*cp)-h_t_dot*(x5-T_s)));

         p1-x2-x6^(1/n);

         x1-x5-x7^(1/n);

         x3-p2-x8^(1/n);

         x5-x4-x9^(1/n)];

%% Objective term
L = Pm*u4 + Pt*u3 + Pu*(u1^2+ u2^2);
%% Continuous time dynamics
f = Function('f', {x, z, u, p}, {xdot, L});
%% Control discretization
```

```
N = 24;                % Number of control intervals, 24
M = 24;                % Mpc loops
h = T/N;
Nr = 1;                % Robust horizon of multi-stage NMPC
levels = scen_count;
S = levels^Nr;         % Total number of scenarios
period = N;
%% Predicted demand
Q_demand = [Q_demand; Q_demand];
%% Variable counting
NXD = N*ndiff*(d+1);          %Total Number of differential state variables
NXA = N*nalg*d;               %Total Number of algebraic state variables
NU = N*nu;                    %Total Number of input variables
NXF = ndiff;              %Total Number of end point variables (diff. only)
NV = NXD + NXA + NU + NXF;    %Total number of NLP variables

%% Prepare output variables
x_opt = zeros(M,ndiff);
u_opt = NaN(M+1,nu);
i_infeasible = zeros(M,1);
solver_return = cell(1,M);
w_stored = zeros(NV,1);
%% MPC loop
for ii=1:M
   if ii == 1
    % Start with an empty NLP
    w={};
    w0 = [];
    lbw = [];
    ubw = [];
    J = 0;
    g={};
    lbg = [];
    ubg = [];

    % "Lift" initial conditions
    Xkl = MX.sym('X0', ndiff);
    w = [w(:)', {Xkl}];
    lbw = [lbw; x_init];
    ubw = [ubw; x_init];
    w0 = [w0; x_init];
    % Formulate the NLP
    % For each scenario in multi-stage NMPC
    for l=1:S   %S=9
    % New NLP variable for the control
      for k=0:N-1
        Ukl = MX.sym(['U_' num2str(k) '_' num2str(l)], nu);
        w = [w(:)', {Ukl}];
        lbw = [lbw; ulb];
        ubw = [ubw; uub];
        w0 = [w0; u_init];

        % State at collocation points
        Xklj = cell(1,d);
        Zklj = cell(1,d);

        for j=1:d
```

```
            %Differential states
            Xklj{j} = MX.sym(['X_' num2str(k) '_' num2str(l) ...
            '_' num2str(j)], ndiff);
            w = [w(:)', Xklj(j)];
            lbw = [lbw; xlb];
            ubw = [ubw; xub];
            w0 = [w0; x_init];
            % Algebraic states
            Zklj{j} = MX.sym(['Z_' num2str(k) '_' num2str(l) ...
            '_' num2str(j)], nalg);
            w = [w(:)', Zklj(j)];
            lbw = [lbw; zlb];
            ubw = [ubw; zub];
            w0 = [w0; z_init];
        end
        % Loop over collocation points
        Xkl_end = D(1)*Xkl;
        for j=1:d
            % Expression for the state derivative at the
            % collocation point
            xp = C(1,j+1)*Xkl;
            zp = zeros(nalg,1);
            for r=1:d
                xp = xp + C(r+1,j+1)*Xklj{r};
            end

            % Append collocation equations
            [fj, qj] = f(Xklj{j}, Zklj{j}, Ukl, scens{l}');
            g = [g(:)', {h*fj - [xp; zp]}];
            lbg = [lbg; zeros(nx,1)];
            ubg = [ubg; zeros(nx,1)];

            % Add contribution to the end state

            Xkl_end = Xkl_end + D(j+1)*Xklj{j};
            % Add contribution to quadrature function
            J = J + B(j+1)*qj*h;
        end
        % New NLP variable for state at end of interval
        % Differential states
        Xkl = MX.sym(['X_' num2str(k+1) '_' num2str(l)], ndiff);
        w = [w(:)', {Xkl}];
        lbw = [lbw; xlb];
        ubw = [ubw; xub];
        w0 = [w0; x_init];
        % Add inequality constraint
        g = [g(:)', {Ukl(4) + Ukl(2)*rho*cp*(Xkl(4)-scens{l}(2))}];
        lbg = [lbg; Q_demand((ii-1)+k+1)];
        ubg = [ubg; Q_demand((ii-1)+k+1)];

        % Add equality constraint
        g = [g(:)', {Xkl_end - Xkl}];
        lbg = [lbg; zeros(ndiff,1)];
        ubg = [ubg; zeros(ndiff,1)];
    end
  end
else
```

```
% Start with an empty NLP
w={};
w0 = w_stored;
lbw = [];
ubw = [];
J = 0;
g={};
lbg = [];
ubg = [];
% Apply the first control from the previous solution
w0(1:ndiff+nu) = [x_init; u_init];

% "Lift" initial conditions
Xkl = MX.sym('X0', ndiff);
w = [w(:)', {Xkl}];
lbw = [lbw; w0(1:ndiff)];
ubw = [ubw; w0(1:ndiff)];

% Formulate the NLP
% For each scenario
for l=1:S
% New NLP variable for the control
    for k=0:N-1
        Ukl = MX.sym(['U_' num2str(k) '_' num2str(l)], nu);
        w = [w(:)', {Ukl}];
        lbw = [lbw; ulb];
        ubw = [ubw; uub];

        % State at collocation points
        Xklj = cell(1,d);
        Zklj = cell(1,d);

        for j=1:d
            %Differential states
            Xklj{j} = MX.sym(['X_' num2str(k) '_' num2str(l)...
                '_' num2str(j)], ndiff);
            w = [w(:)', Xklj(j)];
            lbw = [lbw; xlb];
            ubw = [ubw; xub];

            % Algebraic states
            Zklj{j} = MX.sym(['Z_' num2str(k) '_' num2str(l) ...
            '_' num2str(j)], nalg);
            w = [w(:)', Zklj(j)];
            lbw = [lbw; zlb];
            ubw = [ubw; zub];
        end

        % Loop over collocation points
        Xkl_end = D(1)*Xkl;
        for j=1:d
            % Expression for the state derivative at the
            % collocation point
            xp = C(1,j+1)*Xkl;
            zp = zeros(nalg,1);
            for r=1:d
                xp = xp + C(r+1,j+1)*Xklj{r};
```

```
                end

                % Append collocation equations
                [fj, qj] = f(Xklj{j}, Zklj{j}, Ukl, scens{l}');
                g = [g(:)', {h*fj - [xp; zp]}];
                lbg = [lbg; zeros(nx,1)];
                ubg = [ubg; zeros(nx,1)];

                % Add contribution to the end state
                Xkl_end = Xkl_end + D(j+1)*Xklj{j};
                % Add contribution to quadrature function
                J = J + B(j+1)*qj*h;
            end
                % New NLP variable for state at end of interval
                % Differential states
                Xkl = MX.sym(['X_' num2str(k+1) '_' num2str(l)], ndiff);
                w = [w(:)', {Xkl}];
                lbw = [lbw; xlb];
                ubw = [ubw; xub];

                % Add inequality constraint
                g = [g(:)', {Ukl(4) + Ukl(2)*rho*cp*(Xkl(4)-scens{l}(2))}];
                lbg = [lbg; Q_demand((ii-1)+k+1)];
                ubg = [ubg; Q_demand((ii-1)+k+1)];
                % Add equality constraint
                g = [g(:)', {Xkl_end - Xkl}];
                lbg = [lbg; zeros(ndiff,1)];
                ubg = [ubg; zeros(ndiff,1)];
        end
    end
    end
    % Non-anticipativity constraints
    U = w(2:2*d+2:end);  % +2 is the start and end point for the state!!!
    for n=1:S-1
        g = [g(:)', {U{N*(n-1)+1} - U{1+N*n}}];
        lbg = [lbg; -tol*zeros(nu,1)];
        ubg = [ubg; tol*zeros(nu,1)];
    end

%% Create an NLP solver
opts = struct;
opts.ipopt.max_iter = maxiter; %5000;
opts.print_time = 1; %0,1
opts.ipopt.tol = tol;
opts.ipopt.acceptable_tol =100*tol; % optimality convergence tolerance
prob = struct('f', J, 'x', vertcat(w{:}), 'g', vertcat(g{:}));
solver = nlpsol('solver', 'ipopt', prob, opts);

%% Solve the NLP
sol = solver('x0', w0, 'lbx', lbw, 'ubx', ubw,'lbg', lbg, 'ubg', ubg);
if solver.stats.success == 0
   error('Error: Optimal Solution Not Found')
end
i_infeasible(ii) = solver.stats.success;
solver_return{ii} = solver.stats.return_status;
w_opt = full(sol.x);
%% Store the open loop solution
```

```
w_stored = [w_opt((ndiff+nu)+d*nx+1:end); ...
w_opt(end+1-((ndiff+nu)+d*nx):end)];
z_init = w_stored((d-1)*ndiff+nu+1:ndiff+nu+nx);

%% Simulator (What actually happening in the plant!!!)
x0 = x_init;
u_init = w_opt(ndiff+1:ndiff+nu);
tsample = T/M;
tspan = [0 tsample];

par = [q_L1; q_R1; u_init(1:nu-1); T1_actual; T2_actual; ...
V_hex; V_tank; U_hex; A_hex; rho;...
cp; h_s; A_tank; T_s; n];
options = odeset('RelTol',1e-5,'Stats','off','OutputFcn',@odeplot);
[t,x] = ode15s(@(t,x) twoPlantModelChen(t,x,par), tspan, x0, options);

u_opt(ii,:) = w_opt(ndiff+1:ndiff+nu);
x_m = x(end,1:ndiff);
x_opt(ii,:) = x_m;
x_init = transpose(x_m);
x_init
u_init
ii
end

T = T/3600;
tgrid = linspace(0, T, N+1);
clf;

x_opt = [x_0'; x_opt];

%% Plot optimal controls
set(0,'DefaultTextFontName','Times',...
'DefaultTextFontSize',15,...
'DefaultAxesFontName','Times',...
'DefaultAxesFontSize',15,...
'DefaultLineLineWidth',1.5,...
'DefaultLineMarkerSize',7.75,...
'DefaultStairLineWidth',1.5);
set(findall(gcf,'Type','text'),'FontSize',15,'Interpreter','latex');
set(gcf,'color','white');

figure(1)
subplot(311)
plot(tgrid, x_opt(:,5), 'k-')
ylabel('$T_{tank}$ [$^{\circ}$C]','Interpreter','latex')
hold off
grid on
xlim([0 24])
ylim([50 70])

subplot(312)
stairs(tgrid, u_opt(:,1)*1e3, 'r-')
hold on
stairs(tgrid, u_opt(:,2)*1e3, 'b-')
hold off
legend('$q_{L2}$','$q_{R2}$','Interpreter','latex')
```

```
ylabel('$q_{i2}$ [$\ell$/s]','Interpreter','latex')
grid on
xlim([0 24])
ylim([0 55])

subplot(313)
stairs(tgrid, u_opt(:,4), 'r-')
hold off
xlabel('time [hr]','Interpreter','latex')
ylabel('Power [kW]','Interpreter','latex')
legend('$Q_{market}$','Interpreter','latex')
grid on
xlim([0 24])
ylim([0 500])
```

Listing 8.14: Scenario selection

```
function [scens, scen_count] = scenpara(a1,a2)
%Box method for scenario selection and return the
%scenarios
%scenpara: creates combinations for uncertain
%parameter levels in scenarioMPC
scen_count = length(a1)*length(a2);        %length(a1)=3
scens = {};                        %scen_count is the area of scenario square
  for i=1:length(a1)
       for j=1:length(a2)
           scens = [scens(:); {[a1(i), a2(j)]}];
       end
  end

end
```

Case four: Standard NMPC on simple TES with plant-model mismatch with slack variables

Listing 8.15: Standard NMPC with slack variables

```
clear;
clc;
close all;
addpath('C:\Users\dingn\Desktop\Thesis\casadi-windows-matlabR2016a-v3.5.1')
import casadi.*
%% Model parameters
run Parameters_TES.m
%% Energy demand profile
Q_demand = [1500*ones(12,1); 3500*ones(12,1)];
%% Actual T1 and T2 for considering plant-model mismatch
T1_actual = 99;
T2_actual = 18;
%% Degree of interpolating polynomial
d = 3;      % Gauss-Radua
%% Get collocation points
tau_root = [0 collocation_points(d, 'radau')];
%% Coefficients of the collocation equation
```

```
C = zeros(d+1,d+1);
% Coefficients of the continuity equation
D = zeros(d+1, 1);
% Coefficients of the quadrature function
B = zeros(d+1, 1);

%% Construct polynomial basis
for j=1:d+1
% Construct Lagrange polynomials to get the polynomial basis
% at the collocation point
 coeff = 1;
  for r=1:d+1
     if r ~= j
       coeff = conv(coeff, [1, -tau_root(r)]);
       coeff = coeff / (tau_root(j)-tau_root(r));
     end
  end
% Evaluate the polynomial at the final time to get the
% coefficients of the continuity equation
D(j) = polyval(coeff, 1.0);
% Evaluate the time derivative of the polynomial at all collocation
% points to get the coefficients of the continuity equation
pder = polyder(coeff);
  for r=1:d+1
     C(j,r) = polyval(pder, tau_root(r));
  end
% Evaluate the integral of the polynomial to get the coefficients
% of the quadrature function
pint = polyint(coeff);
B(j) = polyval(pint, 1.0);

end
%% Time horizon
T = 24*60*60;               % Prediction horizon is 24h
ndiff = 5;                  % Number of differential states, x
nalg = 4;                   % Number of algebraic states, z
nu = 4;                     % Number of control inputs, u
nx = nalg + ndiff;          % Total number of states
zub = Inf*ones(nalg,1);     % Algebraic states, z
zlb = -Inf*ones(nalg,1);

run RTO111.m
x_0 = x_init;
%% Declare model variables
% Diffential states
x1 = SX.sym('x1');
x2 = SX.sym('x2');
x3 = SX.sym('x3');
x4 = SX.sym('x4');
x5 = SX.sym('x5');
x = [x1; x2; x3; x4; x5];
% Algebraic states
x6 = SX.sym('x6');
x7 = SX.sym('x7');
x8 = SX.sym('x8');
x9 = SX.sym('x9');
z = [x6; x7; x8; x9];
```

```
% Control inputs
u1 = SX.sym('u1');                  % q_L2
u2 = SX.sym('u2');                  % q_R2
u3 = SX.sym('u3');                  % Q_tank
u4 = SX.sym('u4');                  % Q_market
u = [u1; u2; u3; u4];
%% Slack variable
s = SX.sym('s', ndiff);
s_init = ones(ndiff,1);
%% Model equations (DEAs)

xdot = [(1/V_hex)*(q_L1*(T1-x1) - h_dot*(0.5*(x6 + x7))^(1/n));

        (1/V_hex)*(q_R1*(x5-x2) + h_dot*(0.5*(x6 + x7))^(1/n));

        (1/V_hex)*(u1*(x5-x3) - h_dot*(0.5*(x8 + x9))^(1/n));

        (1/V_hex)*(u2*(T2-x4) + h_dot*(0.5*(x8 + x9))^(1/n));

        (1/V_tank)*(q_R1*(x2-x5)+u1*(x3-x5)+(u3/(rho*cp)-h_t_dot*(x5-T_s)));

         T1-x2-x6^(1/n);

         x1-x5-x7^(1/n);

         x3-T2-x8^(1/n);

         x5-x4-x9^(1/n)];

%% Objective term
L = Pm*u4 + Pt*u3 + Pu*(u1^2+u2^2) + rho_s*sum(s);
%% Continuous time dynamics
f = Function('f', {x, z, u, s}, {xdot, L});
%% Control discretization
N = 24;              % Number of control intervals, 24 hours
M = 24;              % Number of MPC loops, reoptimizing times
h = T/N;
period = N;
%% Variable counting
NXD = N*ndiff*(d+1);         %Total Number of differential state variables
NXA = N*nalg*d;              %Total Number of algebraic state variables
NU = N*nu;                   %Total Number of input variables
NXF = ndiff;                 %Total Number of end point variables(diffonly)
NV = NXD + NXA + NU + NXF;   %Total number of NLP variables
%% Prepare output variables
x_opt = zeros(ndiff,M);
u_opt = zeros(nu,M);
i_infeasible = zeros(M,1);
solver_return = {};
w_stored = zeros(NV,1);
%% Predicted Demand
Q_demand = [Q_demand; Q_demand];
%% MPC loop
for i=1:M
   if i==1
     % Start with an empty NLP
     w={};
```

```
    w0 = [];
    lbw = [];
    ubw = [];
    J = 0;
    g={};
    lbg = [];
    ubg = [];

    % "Lift" initial conditions
    % Differential states
    Xk = MX.sym('X0', ndiff);
    w = [w(:)', {Xk}];
    lbw = [lbw; x_init];
    ubw = [ubw; x_init];
    w0 = [w0; x_init];

    % Formulate the NLP
    for k=0:N-1
      % New NLP variable for the control
      Uk = MX.sym(['U_' num2str(k)], nu);
      w = [w(:)', {Uk}];
      lbw = [lbw; ulb];
      ubw = [ubw; uub];
      w0 = [w0; u_init];

      Sk = MX.sym(['S_' num2str(k)], ndiff);
      w = [w(:)', {Sk}];
      lbw = [lbw; zeros(ndiff,1)];
      ubw = [ubw; inf(ndiff,1)]; % No upper bound!!!
%       ubw = [ubw; 1*ones(ndiff,1)];
      w0 = [w0; s_init];

      % State at collocation points
      Xkj = cell(1,d);
      Zkj = cell(1,d);
      for j=1:d
        % Differential states
        Xkj{j} = MX.sym(['X_' num2str(k) '_' num2str(j)],ndiff);
        w = [w(:)', Xkj(j)];
        lbw = [lbw; xlb];
        ubw = [ubw; xub];
        w0 = [w0; x_init];

        % Algebraic states
        Zkj{j} = MX.sym(['Z_' num2str(k) '_' num2str(j)],nalg);
        w = [w(:)', Zkj(j)];
        lbw = [lbw; zlb];
        ubw = [ubw; zub];
        w0 = [w0; z_init];
      end
      % Loop over collocation points
      Xk_end = D(1)*Xk;

      for j=1:d
         % Expression for the state derivative at the collocation point
         xp = C(1,j+1)*Xk;
         zp = tol*ones(nalg,1);
```

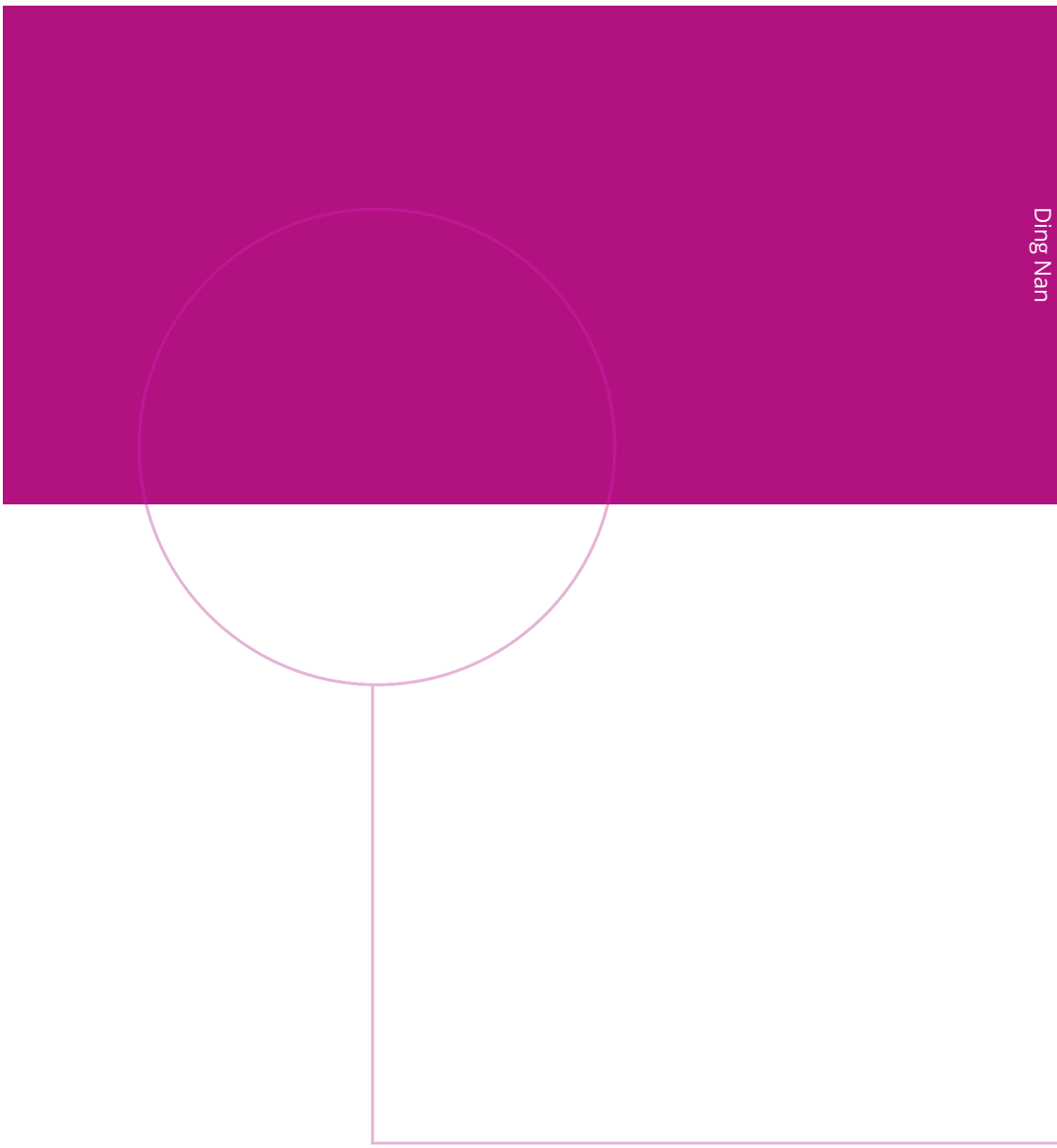
```
        for r=1:d
           xp = xp + C(r+1,j+1)*Xkj{r};
        end

        % Append collocation equations
        [fj, qj] = f(Xkj{j}, Zkj{j}, Uk, Sk);
        g = [g(:)', {h*fj - [xp;zp]}];
        lbg = [lbg; zeros(nx,1)];
        ubg = [ubg; zeros(nx,1)];

        % Add contribution to the end state
        Xk_end = Xk_end + D(j+1)*Xkj{j};
        % Add contribution to quadrature function
        J = J + B(j+1)*qj*h;
      end
      % New NLP variable for state at end of interval
      % Differential states
      Xk = MX.sym(['X_' num2str(k+1)], ndiff);
      w = [w(:)', {Xk}];
      lbw = [lbw; -Inf*ones(ndiff,1)];
      ubw = [ubw; +Inf*ones(ndiff,1)];
      w0 = [w0; x_init];

      % Add inequality constraint
      g = [g(:)', {Uk(4) + Uk(2)*rho*cp*(Xk(4)-T2)}];
      lbg = [lbg; Q_demand((i-1)+k+1)];
      ubg = [ubg; Q_demand((i-1)+k+1)];

      % Add equality constraint
      g = [g(:)', {Xk_end-Xk}];
      lbg = [lbg; zeros(ndiff,1)];
      ubg = [ubg; zeros(ndiff,1)];

      g = [g(:)', {Xk-Sk}, {Xk+Sk}]; % Soft constraint
      lbg = [lbg; -Inf.*ones(ndiff,1); xlb.*ones(ndiff,1)];
      ubg = [ubg; xub.*ones(ndiff,1); +Inf.*ones(ndiff,1)];

    end

  else
    % Start with an empty NLP
    w={};
    w0 = w_stored;  %!!!
    lbw = [];
    ubw = [];
    J = 0;
    g={};
    lbg = [];
    ubg = [];
    % Apply the first control from the previous solution
    w0(1:ndiff+nu) = [x_init; u_init];

    % "Lift" initial conditions
    % Differential states
    Xk = MX.sym('X0', ndiff);
    w = [w(:)', {Xk}];
    lbw = [lbw; w0(1:ndiff)];
```

```
    ubw = [ubw; w0(1:ndiff)];

   % Formulate the NLP
    for k=0:N-1
       % New NLP variable for the control
       Uk = MX.sym(['U_' num2str(k)],nu);
       w = [w(:)', {Uk}];
       lbw = [lbw; ulb];
       ubw = [ubw; uub];
       Sk = MX.sym(['S_' num2str(k)], ndiff);
       w = [w(:)', {Sk}];
       lbw = [lbw; zeros(ndiff,1)];
       ubw = [ubw; inf(ndiff,1)];

       % State at collocation points
       Xkj = {};
       Zkj = {};
       for j=1:d
           % Differential states
           Xkj{j} = MX.sym(['X_' num2str(k) '_' num2str(j)],ndiff);
           w = [w(:)', Xkj(j)];
           lbw = [lbw; xlb];
           ubw = [ubw; xub];

           % Algebraic states
           Zkj{j} = MX.sym(['Z_' num2str(k) '_' num2str(j)],nalg);
           w = [w(:)', Zkj(j)];
           lbw = [lbw; zlb];
           ubw = [ubw; zub];
       end

       % Loop over collocation points
       Xk_end = D(1)*Xk;
       for j=1:d
           % Expression for the state derivative at the collocation point
           xp = C(1,j+1)*Xk;
           zp = tol*ones(nalg,1);
           for r=1:d
              xp = xp + C(r+1,j+1)*Xkj{r};
           end

           % Append collocation equations
           [fj, qj] = f(Xkj{j}, Zkj{j}, Uk, Sk);
           g = [g(:)', {h*fj - [xp;zp]}];
           lbg = [lbg; zeros(nx,1)];
           ubg = [ubg; zeros(nx,1)];

           % Add contribution to the end state
           Xk_end = Xk_end + D(j+1)*Xkj{j};
           % Add contribution to quadrature function
           J = J + B(j+1)*qj*h;
       end
       % New NLP variable for state at end of interval
       % Only for differential states
       Xk = MX.sym(['X_' num2str(k+1)], ndiff);
       w = [w(:)', {Xk}];
%        lbw = [lbw; xlb];
```

```
%          ubw = [ubw; xub];
      lbw = [lbw; -Inf*ones(ndiff,1)];
      ubw = [ubw; +Inf*ones(ndiff,1)];

         % Add inequality constraint
         g = [g(:)', {Uk(4) + Uk(2)*rho*cp*(Xk(4)-T2)}];
         lbg = [lbg; Q_demand((i-1)+k+1)];
         ubg = [ubg; Q_demand((i-1)+k+1)];

         % Add equality constraint (only differential states)
         g = [g(:)', {Xk_end-Xk}];
         lbg = [lbg; zeros(ndiff,1)];
         ubg = [ubg; zeros(ndiff,1)];

        g = [g(:)', {Xk-Sk}, {Xk+Sk}];
        lbg = [lbg; -Inf.*ones(ndiff,1); xlb.*ones(ndiff,1)];
        ubg = [ubg; xub.*ones(ndiff,1); +Inf.*ones(ndiff,1)];
      end
end
%% Create an NLP solver
opts = struct;
opts.ipopt.max_iter = maxiter; %5000;
opts.ipopt.print_level = 3; %0,3
opts.print_time = 1; %0,1
opts.ipopt.tol = tol;
opts.ipopt.acceptable_tol =100*tol; % optimality convergence tolerance
% opts.ipopt.jacobian_approximation = 'finite-difference-values';
opts.ipopt.hessian_approximation = 'limited-memory';
% % opts.ipopt.limited_memory_update_type = 'bfgs';
% % % opts.ipopt.limited_memory_max_skipping= '1';
% % % opts.ipopt.limited_memory_special_for_resto = 'no';

prob = struct('f', J, 'x', vertcat(w{:}), 'g', vertcat(g{:}) );
solver = nlpsol('solver', 'ipopt', prob, opts);
%% Solve the NLP
sol = solver('x0', w0, 'lbx', lbw, 'ubx', ubw,'lbg', lbg, 'ubg', ubg);

if solver.stats.success == 0
    error('Error: Optimal Solution Not Found')
end

i_infeasible(i) = solver.stats.success;
solver_return{i} = solver.stats.return_status;
w_opt = full(sol.x);

%% Store the open loop solution
w_stored = [w_opt((ndiff+nu)+d*nx+1:end); ...
w_opt(end+1-((ndiff+nu)+d*nx):end)];

%% Simulator
x0 = x_init;
tspan = [0 3600];

p = [q_L1; q_R1; w_opt(ndiff+1:ndiff+nu-1); ...
T1_actual; T2_actual; V_hex; V_tank; U_hex; A_hex; rho;...
cp; h_s; A_tank; T_s; n];
```

```
options = odeset('RelTol', 1e-8,'Stats','off','OutputFcn', @odeplot);
[t,x] = ode15s(@(t,x) twoPlantModelChen(t,x,p), tspan, x0, options);
u_opt(:,i) = w_opt(ndiff+1:ndiff+nu);
u_init = w_opt(ndiff+1:ndiff+nu);
x_m = x(end,:);
x_opt(:,i) = x_m;
x_init = x_m';
x_init(5)
u_init
i
end
%% Adding additional regularisation term!!!
Cost = Pm*u_opt(2,:) + Pu*(u_opt(1,:).^2+u_opt(3,:).^2) + rho_s*sum(s);
Cost = [Cost, NaN];

%% Store data
storageNMPC = struct('Demand',Q_demand,'OptimalStates',...
x_opt,'OptimalInputs',u_opt,'Measurement',...
x_opt,'Cost',Cost);
%% Plot results
T = T/3600;
tgrid = linspace(0, T, N+1);
clf;

x_opt = [x_0, x_opt];

%% Plot optimal controls
set(0,'DefaultTextFontName','Times',...
'DefaultTextFontSize',15,...
'DefaultAxesFontName','Times',...
'DefaultAxesFontSize',15,...
'DefaultLineLineWidth',1.5,...
'DefaultLineMarkerSize',7.75,...
'DefaultStairLineWidth',1.5);
set(findall(gcf,'Type','text'),'FontSize',15,'Interpreter','latex');
set(gcf,'color','white');
figure(1)

subplot(311)
plot(tgrid, x_opt(5,:), 'k-')
ylabel('$T_{tank}$ [$^{\circ}$C]','Interpreter','latex')
hold off
grid on
xlim([0 24])
ylim([55 76])

subplot(312)
stairs(tgrid, [u_opt(1,:)*1e3, nan], 'r-')
hold on
stairs(tgrid, [u_opt(2,:)*1e3, nan], 'b-')
hold off
ylabel('$q_{i2}$ [$\ell$/s]','Interpreter','latex')
legend('$q_{L2}$','$q_{R2}$','Interpreter','latex')
grid on
xlim([0 24])
ylim([0 52])
```

```
subplot(313)
stairs(tgrid, [u_opt(4,:), nan], 'r-')
xlabel('time [hr]','Interpreter','latex')
ylabel('Power [kW]','Interpreter','latex')
legend('$Q_{market}$','Interpreter','latex')
grid on
xlim([0 24])
ylim([0 1100])
```



Ding Nan

NTNU

Norwegian University of
Science and Technology