NTNU – Trondheim
Norwegian University of
Science and Technology

Optimal control of thermal energy storage under supply and demand uncertainty

Zawadi Ntengua Mdoe

Submission date:	June 18, 2019
Project Supervisor:	Johannes Jäschke, Associate Professor IKP
Co-Supervisor:	Mandar Thombre, PhD Candidate IKP

Norwegian University of Science and Technology

Department of Chemical Engineering

I dedicated this work to my fianceé, Naomi.
This is for you Mom and Dad, for sending me to the library,
And in the memory of my beloved sister, Rehema. May her dear soul rest in eternal peace.

Summary

Thermal energy storage is becoming increasingly important in many sectors including the industrial sector. This is due to increasing environmental awareness, and energy saving is a major action to minimise environmental footprint. When optimal control of thermal energy storage is established, it ensures automated profitable operation of such systems. The aim of this thesis was to improve a mathematical model representing a thermal storage system and to continue the previous work done in the specialisation project (Mdoe, 2018). The goal is to investigate the performance of a multi-stage NMPC on a thermal energy storage under uncertainty or plant-model mismatch. The general scenario used for numerical case study is a simple thermal energy storage system that consists of at least one supplier, one consumer, a storage unit, a direct and cheap heat source and an expensive emergency heat source from the external market. To obtain a basis for comparison, a similar numerical case study was performed on a similar two plant system but without a storage unit. A modified mathematical model was obtained and has shown better robustness on handling numerical issues by avoiding singularity problems when calculating derivatives for the optimisation problem solver. It also assured convergence from any feasible starting point. After that, case studies on the system were done by performing simulations on specific operation scenarios.

To begin with, the importance of the thermal storage was illustrated by comparing the operation of a two plant system with storage and without storage. Both cases were controlled by an economic standard model-predictive controller without plant-model mismatch that is, assumming perfect prediction. Inclusion of a thermal storage exhibited lesser need to purchase energy from the external market than when there was no storage. The buffering effect of the thermal storage was clearly seen when there was a step increase in the expected demand. A trade-off exists between purchasing extra energy to fulfil consumer requirements and using the limited storage efficiently. These optimal decisions are calculated and implemented at each hour by the standard NMPC on the thermal energy storage.

Moreover, the thermal energy storage was integrated with an intermittent supply such as solar power. This is a co-generation scenario for storage with one energy resource being limited. The standard NMPC controlled the system to utilise the use of the cheaper solar heating and store it as much as possible to prepare for peak demand periods.

Lastly, optimal control of the thermal energy grid under uncertainty was investigated assuming mismatch in supply and demand stream temperatures. A multi-stage NMPC was implemented and their results compared with the performance of a standard NMPC. The multi-stage NMPC showed poor economic performance but without constraint violations. The constraints satisfaction in industrial operations is paramount because its lack of that results to unsafe operation and loss of product quality which translates to huge losses. Therefore, the multi-stage NMPC was found to be a better control strategy in the presence of large supply and demand uncertainties. The optimal control problem in all cases was discretised into a non-linear program using direct collocation approach. The nonlinear program was defined using CasADi framework and solved by using IPOPT solver in MATLAB environment.

Preface

This report is written for the fulfillment of the requirements of the course *TKP4900 Chemical Process Technology, Master's Thesis* carried out at the Norwegian University of Science and Technology (NTNU). The work presented here was carried out during the Spring semester of 2019 at the Department of Chemical Engineering.

I would like to thank my supervisor Associate Professor Johannes Jäschke for the guidance and mentoring he has gladly offered. He has always worked with me closely and has offered great inspiration throughout the duration of the thesis work. My sincere gratitude also goes to my co-supervisor in this thesis, PhD candidate Mandar Thombre who has been giving me close assistance throughout the semester. Our common enthusiasm on the field of optimal control of thermal energy storage had brought up helpful discussions in the topic and through them great insights in the course of implementing the thesis objectives. This work has given me an opportunity to obtain hands on programming experience with MATLAB and familiarity with IPOPT optimisation tool and CasADi framework. Lastly, I also thank the Process Systems Engineering research group for their readiness to offer help, advice and tips on particular challenges I faced. They created a special environment for me to facilitate the completion of my work in the 20 week period.

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).

[illegible]

Zawadi Ntengua Mdoe
Trondheim
June 18, 2019

Table of Contents

List of Tables

List of Figures

Abbreviations

AM	=	Arithmetic Mean
CM	=	Chen Mean Approximation
CAISO	=	California Independent system Operator
DAE	=	Differential Algebraic Equations
ES	=	Energy Storage
GM	=	Arithmetic Mean
HEX	=	Heat Exchanger
HTF	=	Heat Transfer Fluid
IPOPT	=	Interior Point Optimiser
KKT	=	Kahrush-Kuhn-Tucker
LMTD	=	Log Mean Temperature Difference
MPC	=	Model Predictive Control
NLP	=	Nonlinear program
NMPC	=	Nonlinear Model Predictive Control
ODE	=	Ordinary Differential Equations
PCM	=	Phase Change Material
TES	=	Thermal Energy Storage
UM	=	Underwood Mean Approximation

Chapter 1

Introduction

The idea of sustainability in process industries has become prominent due to growth of environmental awareness. There are increasing concerns on climate change and environmental emissions linked to industrial activities. Consequently, minimisation of resource utilisation in human activities is paramount towards achieving sustainability. Scientific innovation is focused on developing technology that relieves the burden on the environment. One of the most impactful human activities to the environment is industrial manufacturing. Industries require huge amounts of raw materials that are derived from the environment. The growth of human world population has increased the amount significantly and questions are being asked about the capability of the world to sustain itself in the future. Caution is required when utilising resources for industrial production that are of essence to the world. Humanity cannot ultimately stop production, but the goal is optimal exploitation of the environmental resources while satisfying customer needs. Industrial processes have been enhanced for better efficiency, and process technology is aimed at producing more from as little resources as possible.

Energy is the most coveted resource in all industrial processes. Energy is a valuable resource that can be generated for use from various sources. They can be categorised as renewable or non-renewable resources depending on their ability to regenerate and replenish. Non-renewable resources are easily available, cheaper to harness but to a great cost on the environment.

Moreover, non-renewable energy resources have a great contribution to carbon emissions to the atmosphere. The energy crisis in the 1980s due to skyrocketed oil prices led to search for alternative energy resources, and renewable energy technology arose. Renewables are environmentally friendly but have unpredictable availability and are expensive to produce. Regardless of energy source, produced energy must be meticulously consumed . This makes energy management an important technology that plays a major role in sustainability. Good energy management results into lesser environmental impact and saves a significant amount of operational costs.

Energy storage technology is a promising field to be applied for energy management. This approach facilitates both supply-side and demand-side energy management, and removes the gap present between power demand and the quality of power supplied and reliability for both short and long-term basis. Conventional energy grids without storage would meet peak demands by purchasing extra energy from emergency generators which is expensive. Inclusion of energy storage increases the flexibility of energy planning and management. It also allows for contribution of other energy production sources notably renewables, whose availability can be shifted by storage and redistribution of the stored energy at peak demand times. Human activity sectors that have highest energy demand include transport, industry and residential sectors. The industry and residential sectors are special because they require unceasing energy supply. Industries require continuous supply of energy while in production and residential areas need energy for domestic hot water heating, air conditioning and lighting almost at all times of the day. All these sectors desire economic operation with less environmental footprint and therefore optimal energy storage technology is demanded. (Kalaiselvam and Parameshwaran, 2014c)

The development of renewable energy technologies and waste heat recovery in chemical processes have further promoted the idea of energy storage. These renewable sources are intermittent and uncertain in terms of their availability and magnitude. Therefore, energy storage is a promising utility to be applied in tandem with renewables because of resource reallocation ability offered. The supply-demand mismatch calls for smart energy distribution technologies that are optimally combined with energy storage.

Produced energy can be stored in various forms. The most common form of energy used is electrical energy. The storage methods for electrical energy include electrical batteries (chemical storage) and capacitors. Electrical batteries are practical for storing power just enough to operate small to medium electrical appliances and not industrial scale plants. Other forms of energy storage are underdeveloped and less practical. The storage of large quantities of energy that can be used to supply hundreds of megawatts of electrical power output in an 8 hour period is only possible using thermal energy storage, pumped hydroelectric energy storage, or compressed air storage. Pumped HEP storage requires pumping to high heights and compressed air storage require large volume for highly pressurised air to occupy which are unrealistic choices. Thermal energy storage turns out to be the best technology candidate out there that is available without limitation on location or quantity of energy stored for a short time, up to 12 hours. (Li and Chan, 2017)

In brief, due to a growing energy demand, that is expected to escalate further due to world population growth, the challenges that energy sector face that force energy scientists and engineers to rethink and obtain energy efficient technologies or measures include: depletion of primary energy resources, complex energy extraction process from primary energy sources, atmospheric emissions such as green-house gases and associated carbon gases, climate change, depletion of ozone layer and global warming, and steep increase in price of energy fuels. (Kalaiselvam and Parameshwaran, 2014a)

In order to mitigate the current energy challenges highlighted above and the prevailing energy issues, the following actions need to be taken to improve energy efficiency: developing smart energy strategies and management techniques that minimise energy demand and associated carbon footprints, implementing energy saving technologies that will ensure reliable energy supply, and optimising the operation of energy distribution systems for efficient energy performance at all times regardless of drastic and unexpected disturbances in the environmental conditions or supply and demand patterns, promote energy distribution frameworks that are flexible to integrate renewable energy sources when needed, invent new energy policies that target on laying plans for expected (obvious) future energy demand or consumption. (Kalaiselvam and Parameshwaran, 2014a)

Thermal energy storage combined with good process control strategy promises to achieve optimal economic operation. Model-predictive control (MPC) has the ability of including and tracking an economic objective to a control problem. The controller constantly re-optimises and manipulates the process inputs according to the economic objective that is subject to predefined operational constraints. It is a centralised control strategy that can handle nonlinearities in the process dynamics. This is an exciting research area to achieve smart and automated energy distribution with minimum effort. The MPC controller can perform online process optimisation and is therefore convenient for optimal disturbance rejection.

1.1 Motivation

As discussed in the previous section, energy storage is useful in energy distribution grids because they hold and release energy between the integrated units. These integrated units are mainly the energy sources and energy sink units. When the energy sources have excess energy, and there is a smaller demand requirement in the sinks, the storage holds it. The stored energy can be released to match future power demand peaks that are typically much higher than the instant supply rate. Energy storage systems usually have multiple sources which vary in availability, prices and quality. Therefore the operation of such a grid requires optimal control. Optimal control of energy grids create smart grids which ensure that the system operates at the optimal point or near-optimal point.

Thermal energy storage is as important and are applied in industrial clusters, hot water systems in buildings, air conditioning etc. An economic optimal operation strategy that ensures cost minimisation and energy demand satisfaction at all times is needed in these systems to save resources. Thermal energy smart grids have the goal of meeting heat demands in the system and hence its operation strategy could be based on a predictive model that is used to predict the best actions. The advantages of an optimal thermal energy storage system include:

1. Higher peak capacity: This means that the storage is always heated up in advance to prepare for peak demands and meet short-term fluctuations in demand. Therefore, the system can satisfy peak

demands better without huge reliance on emergency power generators that consume expensive primary energy sources.

2. Exploit energy market prices: The storage heats up when the market prices are lowest and opts not to when most expensive. This is a price-based strategy for storage.

3. Reject power supply surges: The energy storage improves reliability of energy sources even when there is a fluctuation in the supply or transitions. The storage gives the system inert ability to reject large supply disturbances and uncertainties.

A general problem with energy distribution and management is that power supply and demand are stochastic quantities and can never be precisely anticipated. Optimal control can be coupled with supply and demand forecasting to handle these supply surges and demand fluctuations in a better way. The supply and demand historical data can be analysed and used to learn about their behaviour, which in turn produces satisfactory models that can forecast supply and demand fluctuations. Attaining an accurate forecast is impossible but having a rough expectation of what will happen in the future results in rational planning and optimal control operation.

Uncertainties in supply and demand properties such as magnitude of power and the timing of peaks are a major characteristic in energy distribution systems. This work aims at contribution towards achieving a smart controlled thermal energy grid system that is robust in presence of large uncertainties. Robust model-predictive control approaches come to play here and there has been quite significant studies performed for example by Lucia et al. (2013). However, particular studies of robust control on thermal energy grids with storage are yet to be explored.

1.2 Objectives

The goal of this thesis is to develop a robust model-predictive control strategy on a thermal energy storage system that rejects large disturbances in supply and demand. These disturbances are a result of plant-model mismatch and uncertainty in forecasted and random quantities, especially energy supply and demand. The project builds up on work that has been done in a specialisation project as part of the masters program requirement at NTNU (Mdoe, 2018). From that work a numerical case study of open loop optimisation of a simple two plant energy storage system which consists of a thermal supply, a thermal sink, sensible heat storage tank and an external expensive heating source was done. The work produced a working mathematical model which was used to obtain an optimal open loop control strategy for the system. Following the previous work, this project focuses on the following objectives:

1. Improving suitability of the thermal energy storage system mathematical model for numerical optimisation and control purposes. The model has to be formulated in such a way that numerical issues are avoided when performing calculations.

2. Illustrating the role of thermal energy storage in thermal energy supply grids by comparing its performance with a system that has no storage.

3. Implementing standard model-predictive control on a thermal storage system, first without uncertainties or plant-model mismatch.

4. Developing a robust model-based control approach on a thermal storage to handle supply and demand uncertainties and discuss its performance. The performance is compared with that of a non-robust control strategy.

1.3 Thesis structure

The report has a total of six chapters. Apart from the introduction chapter the next five chapters are briefly described as follows:

Preliminaries
First, a concise background theory on thermal energy storage, modeling of thermal storage systems, dynamic optimisation and model-predictive control, energy supply and demand modeling is covered.

Implementation methods
The next section describes the default approach and tools used in this work

in modeling and performing simulations for the systems and cases of interest.

Modeling of thermal energy grids
Then the system description, assumptions and steps taken to derive the governing equations are listed for two plant system with thermal storage and for two plant direct coupling with no storage cases.

NMPC control on thermal energy grids
This is followed by implementation of model-predictive control on thermal energy grids section where the importance of thermal storage operated by model-predictive control is illustrated. In this chapter, a scenario with an intermittent supply source in presence of thermal storage is also shown. Moreover, a robust model-predictive control (multi-stage NMPC) is implemented when there is uncertainty in supply and demand stream temperatures.

Discussion and conclusion
This chapter is followed by a general discussion on the results and issues encountered in this work. In addition to that, recommendations on future exploration in this topic are given. Finally, a conclusion is drawn regarding the questions that have been asked at the beginning of the research.

Chapter 2

Preliminaries

This chapter introduces the basic theory and concepts applied in this thesis. To begin with, a foundation about energy storage and thermal energy storage systems is presented. This is followed by an explanation of heat transfer modeling in a heat exchanger unit for model derivation and simulation purposes. There is also a concise description on dynamic optimisation and application of direct discretisation method called direct collocation in order to formulate a nonlinear program. Finally, theoretical concepts and robust control approaches that can be applied to optimisation problems under significant uncertainty are highlighted.

2.1 Energy storage systems

Energy storage (ES) systems are playing a major role in achieving energy resource shifting to meet energy demand. Their development has impacted the modern technology significantly. The varying and intermittent available resources such as renewables including solar and wind power can be harnessed when in abundance and the energy stored to cater demand when at peak and/or in periods of low energy availability. Energy storages are environmentally friendly because they ensure minimum resource utilisation by avoiding energy wastage. One of the applications of such energy storages are heating and cooling in industrial processes.

According to Dincer and Rosen (2002), the benefits of energy storage are: economic operation due to diminished energy costs and consumption, improved operational flexibility and reliability, smaller equipment sizes, improved efficiency of process equipment, minimization of fossil fuel used and pollutant emissions.

The 21st century industrial and process technology requires energy abundance and reliability. In addition, it is highly preferred to minimise operating costs which are highly influenced by the energy consumed. The primary energy resources used are non-renewables which are mostly fossil fuels that primarily release energy through combustion as heat. Even though electricity is the conventional energy form, it originates from electrical power generators that are driven by fuel combustion hot gases or indirectly heated steam. The fuel combustion process to release heat for power production is inefficient and hence a large chunk of lost energy is bound to the waste streams.

On the contrary, renewable energy sources are influenced by the environment, hence their availability is not easily predictable. They may also be highly available when the demand is lowest and vice versa, leading to supply-demand mismatch. This unsteady supply of energy resources is characteristic of most renewables. For example, solar power is abundant in some hours of the day and unavailable a night. Therefore, the commitment for attaining efficient, minimal waste and reliable energy supply for industrial processes has called forth the application of energy storage. The scope of energy storage is not only limited to industries but is used in domestic, small-scale power grids. For example, in centralised cooling and heating systems for households and buildings or hot water supply and storage in a household cluster. Moreover, a steep rise in fuel prices has made energy a valuable process utility that has to be managed carefully, and energy storage is important for energy resource management.

2.1.1 Energy demand

Energy demand is the amount of energy that an energy sink requires and it is seldom constant. It has stochastic nature but can show some time dependent trends. In order to decide on how much energy to generate, or in the case of storage, how much to store, the practice is to design energy systems that can adapt to demand even at peak moments. When a combination of energy produced and energy stored fails to satisfy the instant demand, a system must take emergency measures where an external energy source caters the deficit. This subsidiary energy source can be termed as external market energy. The peak demands are usually met by operating extra gas turbines or oil generators that increase the operating costs significantly. This is because fossil fuels are scarce and expensive. Energy storage is another option to satisfy peak energy demand which is also flexible enough to attach cheaper renewable energy sources or waste energy streams. An energy storage strategy can be applied as follows, depending on the field of application:

- Utility: a cheaper utility such as electricity at base load or hot waste streams for example flue gases can "charge" the storage during low demand periods. The stored energy is released in peak demand times, to minimise cost of purchasing energy from the market.

- Industry: in industry, some process units have high temperature outlet streams with excess heat generated that can be used to preheat the storage for later use in peak demand periods. It can also be a part of an industrial cluster which consists of high energy plants such as metallurgy processes and low energy plants that are biochemical processes, greenhouses or even district heating.

- Co-generation: when several energy sources and generators are able to contribute for consumer supply then energy reliability is ensured with minimal costs. Energy storage assist co-generation for example, combining of thermal energy from hot streams and solar energy sources can be done via a thermal storage unit in order to meet energy demand.

Understanding the demand pattern of the energy storage system is the key to execute better decisions for optimal operation. Therefore, energy demand forecasting is a vital tool that is applied in energy systems to ensure that there is always just enough energy available to satisfy the demand side at all instances. However, the actual demand will never be equal to the forecast due to its stochastic nature.

2.1.2 Categories of energy storage methods

Energy storages are made up of substances that have the capacity to hold energy in some form. They are usually classified depending on the form of energy in which the excess energy is converted to before being stored. The classes of energy storage are mechanical, chemical, biological, magnetic and thermal energy storage. Thermal energy storage is to be discussed in this report. Other classes of energy storage are out of the scope of this work.

2.1.3 Thermal energy storage systems

Thermal energy storage (TES) systems are capable of holding heat for later use depending on factors such as temperature, time, place, power demand and price. As expected from an energy storage, thermal energy storages are used to nullify the discrepancy between thermal energy supply and demand when a thermal energy grid is in operation. The working principle of a TES is analogous to an electric battery with a synchronous cycle of charging and discharging processes.

Dincer and Rosen (2002) highlight that a TES needs a high thermal capacity medium. They also mention that it is desired to have a good heat transfer between the storage material and the heat transfer fluid (HTF) or the supply and demand streams. Moreover, according to Cabeza et al. (2015), the important design criteria for a TES are operation strategy, the maximum load needed, the nominal temperature and enthalpy drop, and the integration to the whole application system. A thermal storage system commonly consists of energy collection, storage and distribution via a heat transfer fluid. A TES can be designed with a HTF as the thermal storage medium or with another material packed in a bed.

An ideal thermal storage is the one that heat or cold is released without temperature degradation (Li et al., 2011). To avoid loss by temperature degradation, the HTF must be released at the same temperature at which it was stored. Therefore, the most efficient thermal storage when the HTF is the storage material is by using two storage tanks. The hot fluid carrying thermal energy is stored in a hot tank. Then from there it is pumped through the heat sink where it “discharges” and the temperature drops before being stored in a cold tank. When “charging” is required, the fluid

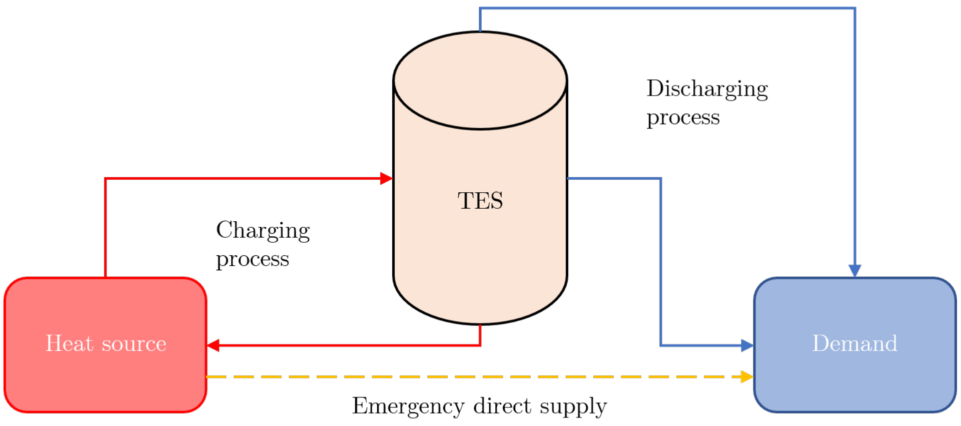

Figure 2.1: Schematic for a thermal energy storage system

from the cold tank is pumped through the source and its temperature rises before being stored in the hot tank. If the tanks are highly insulated, then storage efficiency approaches 100%. Another thermal storage system has the HTF and thermal storage material different. The two-tank thermal energy storage system has ideal energy storage efficiency but keeps one of the tank space empty causing space wastage therefore not cost-effective.

The concept of thermal energy storage system as applied in industrial clusters is illustrated in figure 2.2. The energy sources are high-energy plants and processes. These include cement plants, metallurgy and high energy chemical plants that have hot streams in their process which require cooling. The energy sinks are processes which require heating such as greenhouses, biochemical plants and drying plants. The supply and demand is uneven, and to avoid lack of energy utility in the sinks, the system can purchase heat from an external supplier. On the other hand, when the heat stored is in excess, the heat can be dumped to a variable energy sink such as district heating.

Types of thermal energy storage

TES systems can be classified into sensible heat storage, latent heat storage or thermo-chemical heat storages.

1. **Sensible heat storage**: Sensible heat storage also known as heat

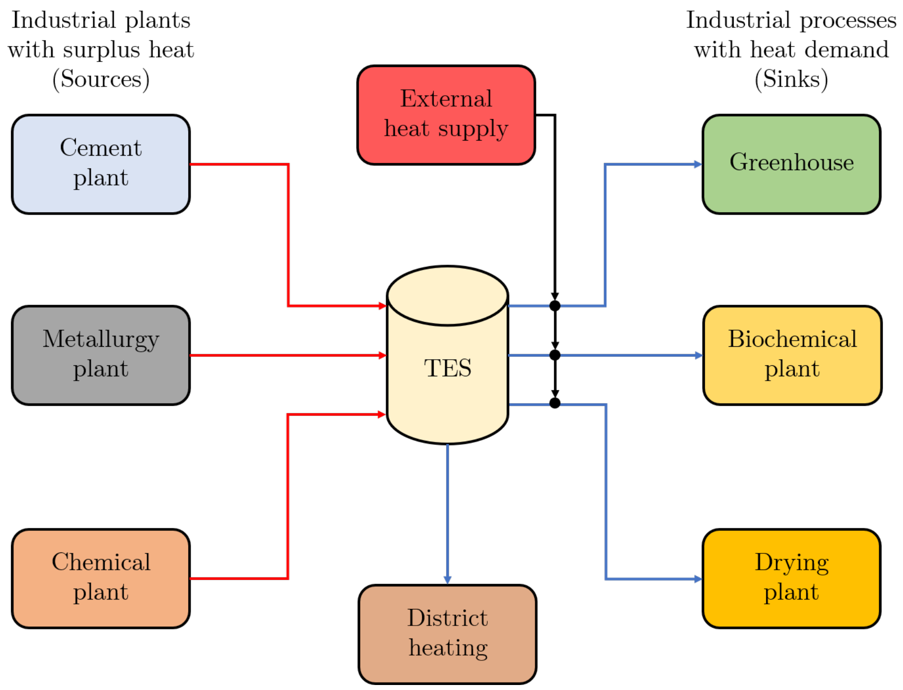

Figure 2.2: Conceptual illustration of energy or heat exchange in industrial cluster with heat exchange and thermal storage. The red lines indicate heat flow from sources to storage and blue lines is heat flow from storage to sinks

capacity storage, is done by initially supplying heat to the storage material to raise the temperature of the storage material. The temperature rise is due to rise in internal energy, meaning that the initial supplied energy is stored in form of sensible heat. This form of storage ensures that supply and demand requirements are met by storing and discharging thermal energy from the heat storage without the storage material changing in phase or chemical composition. The amount of thermal energy stored in a sensible heat storage at constant pressure is given by equation 2.1.

$$E = \int_{T_{\text{ref}}}^{T} \left(\frac{\partial H}{\partial T} \right)_{p,n} \mathrm{d}T \tag{2.1}$$

where, E = thermal energy stored, H = enthalpy, T = absolute temperature, T_{ref} = reference temperature. The equation can be written in form of specific heat capacity at constant pressure, c_p in equation 2.2a

$$E = m \int_{T_{\text{ref}}}^{T} c_p(T)\, \mathrm{d}T \tag{2.2a}$$

where,

$$c_p(T) = \left(\frac{\partial h}{\partial T} \right)_{p,n} \tag{2.2b}$$

and m = mass of stored material and, h = specific enthalpy

Some storage materials have a small variation in their specific heat capacity values within an operating temperature range. In such cases a valid assumption can be made that the specific heat capacity is independent of temperature then it is approximately constant. The constant value used is the mean specific heat. The energy stored can be written simply as equation 2.3.

$$E = mc_p(T - T_{\text{ref}}) \tag{2.3}$$

The sensible energy storage materials are either solids or liquids. For solids and liquids, the specific heat capacities at constant volume and constant pressure are always equal[1] ($c_p = c_v$). Therefore from the equation 2.2a, the amount of thermal storage stored in a material due to raising its temperature can be determined. (Kalaiselvam and Parameshwaran, 2014b)

Liquid storage is used for applications that involve low-temperature to medium-temprature storage. Water is the most common liquid storage material used in practice due to its cheapness, availability and high energy density. Currently, most solar thermal energy storage uses the mentioned qualities of water to achieve the required sensible heat storage. Kalaiselvam and Parameshwaran (2014d) describes in detail on sensible thermal energy storages, more information about them can be found there.

2. **Latent heat storage**: have storage media that stores thermal energy in form of their latent heat during a constant temperature process like phase change. Solid-liquid phase change is the most common method used. Liquid-gas phase change has the highest latent heat of phase change but the huge volume change of the storage material is a problem and hence this method is rarely applied (Alva et al., 2018). The thermal energy stored by latent heat can be expressed as in equation 2.4.

$$E = mL \tag{2.4}$$

where, m = the mass of stored material and, L = latent heat of phase change.

The storage material that undergoes phase change in latent thermal energy storage are known as phase change materials (PCM). There are very few PCMs that have been commercialised. The society working on such storage materials face a huge challenge due to observed problems like phase separation, corrosion, indelible stability, sub-cooling, and low heat conductivity. (Cabeza et al., 2015)

[1]Incompressible materials have the specific heat at constant volume equal that at constant pressure

3. **Thermochemical heat storage**: Thermochemical energy storage is a result of a high energy chemical reaction that stores energy. The reaction products must be stored and the heat generated during the reaction must be available when backward reaction occurs. Therefore, this storage method involves only reversible reactions. In the charging process, injected heat is used to drive an endothermic chemical reaction. The chemical products are later used to restore thermal energy by performing the reverse exothermic reaction. (Orosz and Dickes, 2017)

2.1.4 Thermal storage using water

Sensible thermal storage desires a material that can absorb a large amount of heat. Therefore a material with inherently high heat storage density per volume is preferred. This implies that the storage material must have both high specific heat value and high density. A summary of storage materials with their specific heats and heat content per volume is shown in table 2.1. Water has a very high heat storage density both per weight and per volume compared to other candidate storage materials. Moreover, water is an available material which is relatively cheaper, less reactive and easy to handle. These properties are valid for water in the temperature range between $0°\text{C}$ and $100°\text{C}$. For energy systems that have processes operating at temperatures between $0 - 100°\text{C}$, water is suitable for their heat storage.

The storage tanks involving water are constructed with watertight materials and highly insulating materials. This insulation is to minimise heat losses of the stores. It is a normal occurrence in the hot water storage to exhibit temperature stratification. The rate of heat loss from the hot water storage to the surroundings should be as small as possible to improve storage performance. However, one should expect a higher loss rate when the storage is heated up to elevated temperatures.When charging and discharging the storage, the rate of charging or discharging must be as high as possible in order to ensure that heat is withdrawn at larger temperature differences as possible thus greater efficiency. (Furbo, 2015)

The material properties of water that affect the thermal and flow quantities of interest in storage include density, viscosity, thermal conductivity and specific heat. At an operation range between $0 - 100°\text{C}$, the density,

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

Material	**Specific heat capacity** (kJ/(kgK))	**Heat content per volume** (MJ/(m^3K))
Water	4.2	4.2
Oil	2.0	1.7
Ice	2.0	1.8
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
Aluminium	0.9	2.5
Gold	0.1	2.5

thermal conductivity and specific heat vary non-linearly with temperature. The variation in specific heat capacity within that range is not very large and average property value can be used without creating a huge discrepancy.

Water density decreases with increasing temperature. Variation in density with temperature causes the hot water (less dense) to rise up the storage and the cold water (more dense) moves downwards in the storage. This causes a strong thermal stratification in the hot water store.

Different configurations of hot water storage tanks are shown in figure 2.3. The immersed coils exchanger has the heating coils placed at the bottom where it is coldest for maximum heat transfer. This configuration can be expensive because it may require specialized designs in most cases to improve the storage efficiency. It is also less flexible if there is a need to integrate several different sources of heat simultaneously. The external heat exchanger design configuration is more appealing due to its simplicity coimpared to the former. Moreover the design is more energy efficient, cost effective and reliable because of the pronounced thermal stratification that it creates in the storage. Due to better practicality of using heat exchangers to collect and distribute energy to and from the

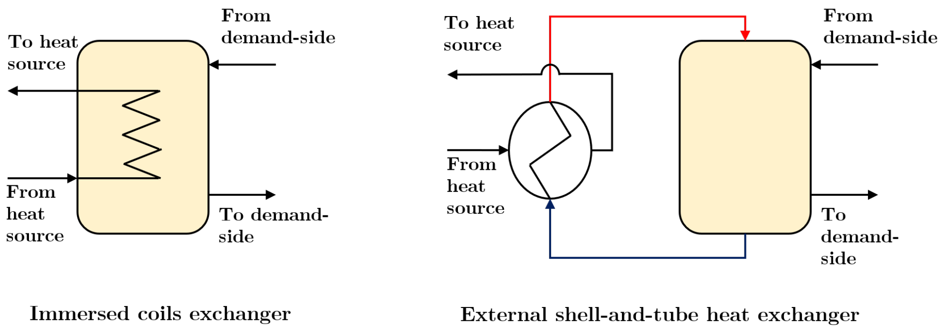

Figure 2.3: Different configurations of storage tanks using water as a storage medium

storage, it is wise to understand the theory about heat exchangers and heat exchanger modeling.

2.2 Heat transfer modeling in heat exchangers

A heat exchanger is a process unit that allows transfer of thermal energy between two fluid streams without having the fluids mix together or come into direct contact. A heat exchanger ensure that there is heat transfer from a hot fluid to a cold one without any mass transfer between them. Therefore the equipment has two sections which separate hold and cold fluid streams by a solid conductor. The arrangement of the two sides of a heat exchanger is designed to create maximum possible heat transfer area and residence time between the two streams.

Heat exchangers are used to extract heat from hot environments and release the heat to cold environments. Therefore one can anticipate their application in thermal energy grids. The heat transfer fluid gathers heat from the source via a heat exchanger and consequently distributes heat from the storage to the sinks using the same equipment. There are various ways of classifying heat exhangers but they can easily be classified in terms of flow arrangement or type of construction. In this thesis, there is minimum focus of the heat exchanger construction design aspect, hence only flow arrangements will be discussed. Based on flow arrangement, the simplest heat exchangers with double-pipes according to Bergman et al.

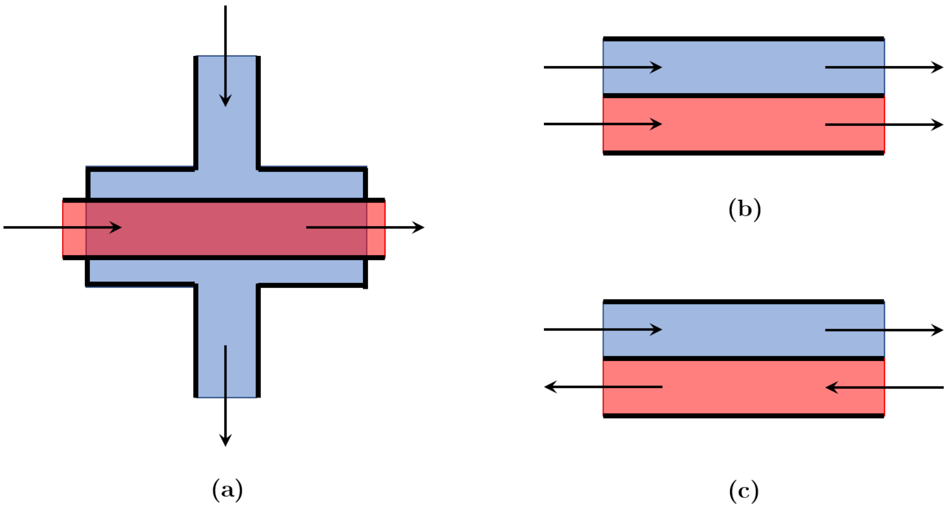

Figure 2.4: Heat exchanger types according to flow configuration (a) cross flow (b) parallel flow (c) counter flow

(2011), can be classified as follows:

- *Cross flow heat exchangers*: the flow of cold and hot streams are arranged in a way that one stream flows perpendicular to the other.
- *Parallel flow heat exchangers*: the cold and hot fluid enter the heat exchanger in the same end and therefore flow in the same direction.
- *Counter-current flow heat exhangers*: the cold and hot fluids enter at different ends and have opposite directed flows.

A common design in practice is a *shell-and-tube heat exchanger* that has a single shell and tube passes inside the shell. Inside the shell are baffles attached in order to maximise convective heat transfer by increasing turbulence. In a shell and tube construction, the flow configuration can not be clearly identified as any of the three discussed above.

2.2.1 Overall heat transfer coefficient

The heat transfer between the two streams is proportional to the differences in absolute temperature of the two streams at that point. The constant of proportionality is the total thermal resistance to heat transfer between the two fluids. This parameter is the overall heat transfer coefficient which is a function of the convective and conductive fluid properties, and conductive solid material properties. The determination of this value is important to analyse heat exchanger performance but is often the most uncertain parameter.

2.2.2 Log mean temperature difference

To model the performance of a heat exchanger one can relate the inlet and outlet temperatures of exchanging streams, overall heat transfer coefficient and total heat transfer surface area. This is shown in equation 2.5.

$$q = UA\Delta T_m \tag{2.5}$$

where U = overall heat transfer coefficient, A = heat transfer area and ΔT_m = appropriate *mean* temperature difference.

The appropriate mean temperature difference can be found by performing energy balances across an infinitesimal transfer area and integrating w.r.t the distributed stream temperatures throughout all the positions (length), assuming parallel flow or counter-current flow. After some derivation steps shown in the books by Bergman et al. (2011) and Lienhard IV and Lienhard V (2018), the appropriate average temperature difference is a *log mean temperature difference*, ΔT_{LM}. Therefore 2.5 can be written as 2.6

$$q = UA\Delta T_{LM} \tag{2.6}$$

where,

$$\Delta T_{LM} = \frac{\Delta T_2 - \Delta T_1}{\ln\left(\Delta T_2/\Delta T_1\right)} = \frac{\Delta T_1 - \Delta T_2}{\ln\left(\Delta T_1/\Delta T_2\right)} \tag{2.7}$$

where positions 1 and 2 are the inlet or outlet positions of the double-pipe heat exchanger.

It may be noted that, for the same inlet and outlet temperatures, the LMTD for counter-current flow is always greater than that for parallel flow. Hence, for the same value of U, the heat transfer area A required to achieve the same heat transfer rate q is smaller for counter-current flow than for parallel flow arrangements. It is also possible for the outlet stream temperature of the cold stream to be greater than the outlet temperature of the hot stream in a counter-current flow setting but never in parallel flow. (Bergman et al., 2011)

2.2.3 Approximation to the LMTD

The logarithmic mean temperature difference is an accurate expression for the mean temperature difference between the hot and cold streams provided that the overall heat transfer coefficient is not a function of the position. However, it has been globally accepted that the logarithmic mean causes inconvenience to chemical engineering programmers. Zavala-Río et al. (2005), Paterson (1984) and Chen (1987), communicated the difficulties associated with performing heat exchanger calculations with the log mean function. It is common practice in iterative equation solving schemes that equality of stream temperatures is assumed as a starting value. This will result into an indeterminate form of the log mean. Moreover, at that limit, the derivatives of LMTD (which are needed in Newton iterative methods) are undefined.

Paterson (1984) derived a new expression that overcomes the aforementioned difficulties, to replace the log mean. The expression is a good enough approximation to LMTD that can be used in calculations for practical purposes. He obtained the *new mean* as a weighted arithmetic mean (linear combination) of *geometric mean* and *arithmetic mean* temperature differences shown in eq. (2.8).

$$\Delta T_{NM} \equiv \frac{2}{3}\Delta T_{GM} + \frac{1}{3}\Delta T_{AM} \simeq \Delta T_{LM} \tag{2.8}$$

where

$$\Delta T_{AM} \equiv \frac{\Delta T_1 + \Delta T_2}{2} \tag{2.9a}$$

$$\Delta T_{GM} \equiv \sqrt{\Delta T_1 \Delta T_2} \tag{2.9b}$$

The arithmetic mean is considered a useful approximation for mean temperature difference in economic analysis but the new mean has refined it for more practical uses.

Underwood (1933, 1970) and then Chen (1987) both derived an approximation that is a weighted geometric mean of the arithmetic mean and geometric mean. Their new means are a polynomial of the temperature differences at the ends of the exchanger ΔT_1 and ΔT_2.

$$\Delta T_{UM}^{1/3} = \frac{1}{2}(\Delta T_1^{1/3} + \Delta T_2^{1/3}) \tag{2.10}$$

$$\Delta T_{CM}^{0.3275} = \frac{1}{2}(\Delta T_1^{0.3275} + \Delta T_2^{0.3275}) \tag{2.11}$$

Generally,

$$\Delta T_m = \left[\frac{1}{2}(\Delta T_1^n + \Delta T_2^n)\right]^{1/n} \tag{2.12}$$

such that the n values will determine the type of approximation used. This has been summarised in table 2.2.

Table 2.2: Values of n and their respective approximations

n	Approximation (ΔT_m)
1	Arithmetic mean (ΔT_{AM})
$\frac{1}{3}$	Underwood's mean (ΔT_{UM})
0.3275	Chen's mean (ΔT_{CM})

2.3 Dynamic optimisation

An optimisation problem includes three main parts which are an *objective function, decision variables* and *constraints*. The objective function is a scalar function which describes the quantity to be minimised or maximised. Decision variables can either be real numbers, integers or binary numbers. Usually decision variables are vectors of real numbers. The constraints are divided into equality and inequality constraints. All these elements together define an optimisation problem. When the objective function or constraints are nonlinear functions then the optimisation problem becomes a *nonlinear program* (NLP). (Foss and Heirung, 2013)

$$\min_{z \in \mathbb{R}^n} f(z) \tag{2.13a}$$

subject to

$$c_i(z) = 0, \quad i \in \mathcal{E} \tag{2.13b}$$
$$c_i(z) \geq 0, \quad i \in \mathcal{I} \tag{2.13c}$$

where, z is the decision variable, $\mathcal{E}$ is the equality constraint index set and $\mathcal{I}$ is the inequality constraints index set.

Usually optimization problems are stated as minimisation problems. To obtain an optimal point, either a local or global minimum must satisfy the *Karush-Kuhn-Tucker (KKT) conditions* which are conditions for optimality. This has been discussed properly in Nocedal and Wright (2006).

2.3.1 Discretisation of a dynamic system

Dynamic systems have decision variables that evolve with time. They are described mathematically by a set of Ordinary Differential Equations (ODEs) or Differential Algebraic Equations (DAEs). Therefore, dynamic optimisation involves computation of the optimal decision and state variables at each point in time. This implies that the problem extends to an infinite dimension. The problem becomes difficult to solve using standard methods. In order to solve the problem, it must be discretised to obtain a finite approximation. The discretisation methods used are either direct or indirect. Most common dynamic optimisation solvers employ direct discretisation methods. Direct methods can either be seqeuntial or simultaneous. If discretisation is performed only on the control inputs, then it is a called a *sequential approach*; while if both the control inputs and states are discretised then it is known as a *simulataneous approach*.

Sequential approach described by Sargent and Sullivan (1978) considers the discretised control inputs to be piece-wise constant within the discretisation time intervals which is usually equal to the sampling time interval. The approach requires an initial guess for the control inputs that are fed to the optimiser. Then the system is simulated using an integrator (ODE or DAE solver) depending on the control inputs given. The objective function, constraints and derivatives are then calculated together with optimality conditions check. The optimal solution is found when these optimality conditions are satisfied. Otherwise, new control input guesses are provided by the NLP solver. the challenge of this approach is the number of initial value problems that the solver has to compute per scenario.

Simultaneous approaches as described by Biegler (2010) have a dichotomy property where bounds are imposed on state variables to avoid system instability within the prediction horizon. This is a merit when compared to the simple to implement sequential approach for integration of unstable systems. The only disadvantage is that the optimisation problem becomes much bigger. Consequently, in the simultaneous approach both the control inputs and the states must be discretised. The two main simultaneous approaches are: *full discretisation* and *multiple shooting*.

Multiple shooting method involves state discretisation and are added as optimisation variables to the overall optimisation problem. The system is integrated in finite elements which usually coincide with the sampling time. To make sure there is state continuity at the finite elemnt boundaries, a continuity constraint is enforced at each finite element and added to the optimisation problem. Continuity means that the end point of a particular finite element must be equal to the starting point of the next finite element. More explanation of multiple-shooting method has been given by Bock and Plitt (1984).

Another type of simultaneous approach is the full discretisation method which uses orthogonal collocation on finite elements. The difference of this approach from multiple-shooting method is the approximation of the state dynamics inside the finite elements to avoid the necessity of using an ODE solver at every stage in the prediction horizon. The approach is a direct method that employs orthogonal collocation and thus alternatively known as *direct collocation method*.

2.3.2 Direct collocation method

This discretisation method has regularly been applied in state trajectory optimisation and parameter optimisation problems or a combination of the two. Every control interval is split into finite elements where the state trajectory is parameterised using Lagrange polynomials. This method improves the ability of an optimisation problem solver to find a solution when a problem is further constrained, even if it increases the dimensionality. In addition, direct collocation method better relates the states to the augmented objective function. (Diehl and Gross, 2017)

Direct collocation method is considered an implicit variation of Runge-Kutta method (Diehl and Gross, 2017). Runge-Kutta allows calculation of states at each discrete time element by forward integration. In direct collocation, the states are expressed implicitly as the function of states and their derivatives. The method employs polynomials to interpolate state variables inside the discrete time element (Hargraves and Paris, 1987).

Polynomial Interpolation

In the temporal domain $\{t_{k,0}, \ldots, t_{k,K}\} \in [t_k, t_{k+1}]$ it is possible to express the state trajectory within a discrete time interval as a function of time points within the interval known as collocation points. The functions used are called Lagrange polynomials ($P_{k,j}(t)$, at time step k and collocation point j) whose order K depends on the number of points taken inside the interval.

$$P_{k,j}(t) = \prod_{j=0,\, j\neq i}^{K} \frac{t - t_{k,j}}{t_{k,i} - t_{k,j}} \in \mathbb{R} \tag{2.14}$$

Lagrange polynomials in an interval $[t_k, t_{k+1}]$, have the property (eq. (2.15)), which also implies that they are orthorgonal.

$$P_{k,i}(t_{k,I}) = \begin{cases} 1 & \text{if} \quad I = i \\ 0 & \text{if} \quad I \neq i \end{cases} \tag{2.15}$$

Figure 2.5 shows all the Lagrange polynomials of the order $K = 4$ in the interval $[t_k,\ t_{k+1}]$.

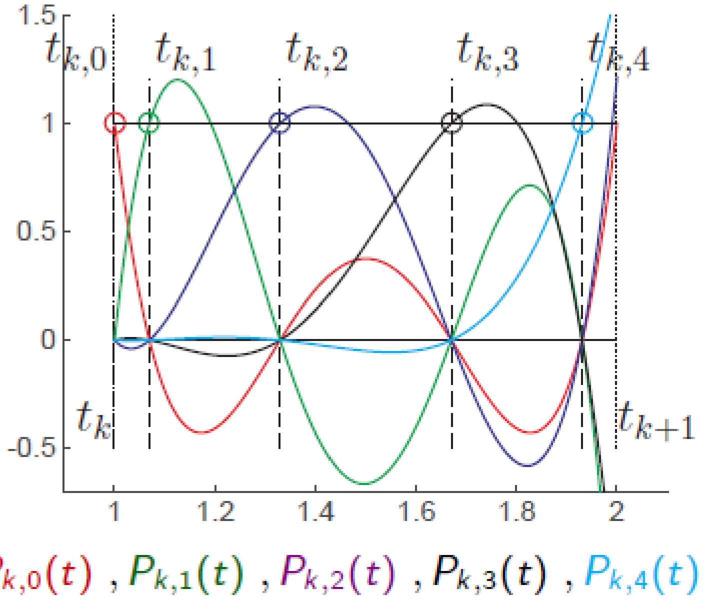

Figure 2.5: Possible 4th order Lagrange polynomials in a time interval Gross (2016)

In order to approximate the state trajectory the linear combination of all these polynomials is used as shown in eq. (2.16).

$$\mathrm{s}(\theta_k, t) = \sum_{i=0}^{K} \theta_{k,i} P_{k,i}(t) \tag{2.16}$$

where

$$\mathrm{s}(\theta_k, t_{k,j}) = \theta_{k,j} \tag{2.17}$$

The main idea of direct collocation method is using eq. (2.16) to fit the state trajectory by selecting parameters $\theta_{k,i}$ to approximate the system dynamics $\mathbf{F}(\dot{\mathbf{x}}, \mathbf{x}, \mathbf{u}) = \mathbf{0}$. In that case we obtain $K + 1$ degrees of freedom per state. Due to the property of Lagrangian basis, the value of the states is equal to the sample taken at the collocation points. Thus, the constraints that should be satisfied in direct collocation are given by eq. (2.18), and the states $\mathbf{x}_k$ and inputs $\mathbf{u}_k$ are the degrees of freedom for the optimisation problem.

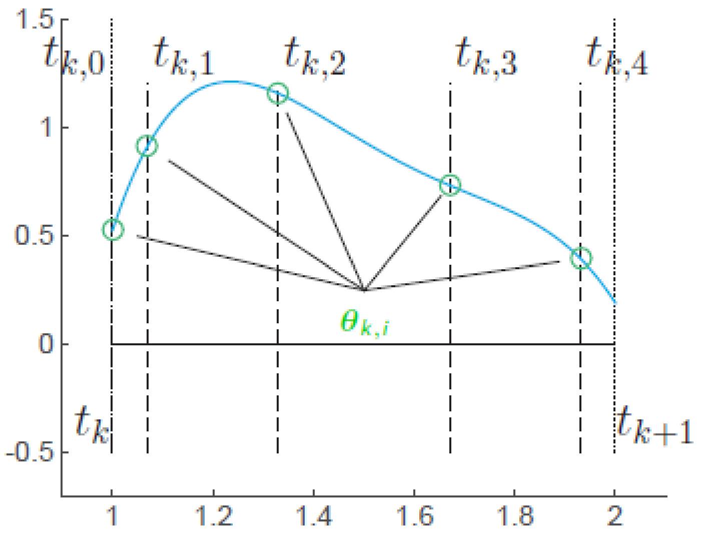

Figure 2.6: Interpolated state trajectory as a function of parameters $\theta_{k,i}$ Gross (2016)

$$\mathbf{s}(\theta_k, t_k) = \theta_{k,0} = \mathbf{x}_k \tag{2.18a}$$

$$\frac{\partial}{\partial t}\mathbf{s}(\theta_k, t_{k,j}) = \mathbf{F}(\mathbf{s}(\theta_k, t_{k,j}, \mathbf{u}_k)), \quad j = 1, \ldots, K \tag{2.18b}$$

Equation 2.18a are the *continuity constraints*. Equation 2.18b are the *dynamic constraints* that can be further modified to obtain eq. (2.19).

$$\sum_{i=0}^{K} \theta_{k,i}\dot{P}_{k,i}(t_{k,j}) = \mathbf{F}(\theta_{k,j}, \mathbf{u}_k) \tag{2.19}$$

The best parameters $\theta_{k,j}$ can be solved for using Newton method from the constraints at each collocation point. The resulting state trajectory is shown in fig. 2.6, with the values at the collocation points equal to their corresponding parameters $\theta_{k,j}$.

Selection of Time Grid

As discussed previously, there are many possibilities of selecting collocation points within the interpolation interval. However, there are proven set of collocation points which deliver an exact integration solution for any polynomial of order $< 2K$ (Legendre) and $< 2K - 1$ (Radau) [2]. (Gross, 2016)

Error and Stability

Collocation methods are A-stable: They can handle stiff model equations. This implies that even larger time steps can be used to predict steady state and slow dynamics correctly in the presence of very fast dynamics. (Biegler, 2010)

Radau collocation is L-stable: In addition to A-stability, Radau collocation handles eigenvalues at $-\infty$. (Gross, 2016)

Order of integration error: It depends on K, integration error is $O(h^{2K})$ for Legendre and $O(h^{2K-1})$ for Radau. Runge-Kutta schemes have an order of $O(h^4)$. Moreover, the error only occurs to the end-state (t_k) of the integrator but not the intermediate points. (Biegler, 2010; Gede, 2011)

2.3.3 Implementation of Direct Collocation to solve an NLP

Solving an NLP is done by passing the constraints and the objective function through a nonlinear program solver. Direct collocation approximates the dynamics within the finite elements, and enforces continuity at the interval boundaries which are then added as constraint equations for the decision variables (states and inputs) at all collocation points and end-points. The optimal solution is found by solving for all the decision variables such that an objective function is minimum. The vector dimension of decision variables $\mathbf{w}$ is equal to $N(n_x(K + 1) + n_u)$ where

[2]Gauss-Legendre and Radau roots define the position of collocation points for any order K Lagrange polynomial. The collocation points are presented in table 10.1 by Biegler (2010)

n_x is the number of state variables, n_u is the number of input variables and N is the number of discrete time elements. The NLP is transformed into a form as shown in equation 2.20.

$$\min_{\mathbf{w}} \quad \Phi(\mathbf{w}) \tag{2.20a}$$

$$\text{subject to} \quad \mathbf{g}(\mathbf{w}) = \begin{bmatrix} \theta_{0,0} - \overline{\mathbf{x}}_0 \\ s(\theta_0, t_1) - \theta_{1,0} \\ \mathbf{F}(\theta_{0,i}, \mathbf{u}_0) - \sum_{j=0}^{K} \theta_{0,j} \dot{P}_{0,j}(t_{0,i}) \\ \vdots \\ s(\theta_k, t_{k+1}) - \theta_{k+1,0} \\ \mathbf{F}(\theta_{k,i}, \mathbf{u}_k) - \sum_{j=0}^{K} \theta_{k,j} \dot{P}_{k,j}(t_{k,i}) \\ \vdots \end{bmatrix} = \mathbf{0} \tag{2.20b}$$

where $\mathbf{w} = \{\theta_{0,0}, \ldots, \theta_{0,K}, \mathbf{u}_0, \ldots, \theta_{N-1,K}, \mathbf{u}_{N-1}\}$ are the decision variables.

Since, the parameter $\theta_{k,i}$ are solved together with the decision variables $\mathbf{x}_k$ and $\mathbf{u}_k$, then direct collocation is a fully simultaneous approach. The integration and optimisation actions are performed together in the NLP solver.

The input is usually chosen as piecewise-constant within each discrete time element. Picking a different input at different collocation times can also be done but is not common and may cause problems in converging to a solution.

How to treat a differential algebraic system of equations?
A system is said to be a differential algebraic system when it has both differential and algebraic states. Differential states ($\mathbf{x}$) are those variables that are dynamic changing with time. Algebraic states ($\mathbf{z}$) are variables that are not time-differentiated and are dependent on differential states.

DAEs can be written in a fully implicit form as shown in eq. (2.21).

$$\mathbf{F}(\dot{\mathbf{x}}, \mathbf{x}, \mathbf{z}, \mathbf{u}) = \mathbf{0} \tag{2.21}$$

They can also be expressed semi-explicitly as shown in eq. (2.22).

$$\dot{\mathbf{x}} = \mathbf{F}(\mathbf{x}, \mathbf{z}, \mathbf{u}) \tag{2.22a}$$

$$\mathbf{0} = \mathbf{G}(\mathbf{x}, \mathbf{z}, \mathbf{u}) \tag{2.22b}$$

It may be more convenient in complex systems to express dynamics constraints as a set of DAEs instead of ODEs. Therefore when solving for optimal decision variables governed by the system, one has to define an NLP that is DAE-constrained. Using direct collocation discretisation, the algebraic states are treated differently from the differential ones. This is because algebraic states do not have to satisfy continuity. The following must be considered when listing constraints on algebraic variables:

1. The continuity equality constraints are not included for the algebraic states at the end-points.
2. The algebraic states appear only in the dynamic constraints. Hence, the degree of freedom per algebraic state per finite element is K and not $K + 1$ as in differential states.

The general form NLP for a DAE-constrained optimal control problem through direct collocation discretisation method is represented by eq. (2.23).

$$\min_{\mathbf{w}} \quad \Phi(\mathbf{w}) \tag{2.23a}$$

$$\text{subject to} \quad \mathbf{g}(\mathbf{w}) = \begin{bmatrix} \theta_{0,0} - \overline{\mathbf{x}}_0 \\ \theta_{0,K} - \theta_{1,0} \\ \mathbf{F}(\frac{\partial}{\partial t}\mathbf{x}(\theta_k, t_{k,0}), \theta_{k,0}, \mathbf{z}_{k,0}, \mathbf{u}_k) \\ \vdots \\ \theta_{k,K} - \theta_{k+1,0} \\ \mathbf{F}(\frac{\partial}{\partial t}\mathbf{x}(\theta_k, t_{k,K}), \theta_{k,i}, \mathbf{z}_{k,K}, \mathbf{u}_k) \\ \vdots \end{bmatrix} = 0 \tag{2.23b}$$

where the decision variables are:
$\mathbf{w} = \{\ldots, \theta_{k,0}, \theta_{k,1}, \mathbf{z}_{k,1}, \ldots, \theta_{k,K}, \mathbf{z}_{k,K}, \mathbf{u}_k, \ldots\} \quad \forall k = \{0, \ldots, N-1\}$

What about other operational constraints on the system apart from the model equations?
A controlled process is usually subjected to operational constraints that it has to satisfy. These constraints are based upon the control objective which is usually economic-driven or safety-driven. For example, the desired operation of the system requires its states to be bounded due to material limits. Moreover, to avoid input saturation, the inputs must be bounded to avoid the NLP providing solutions outside the saturation limits that cannot be implemented. These state and inputs bounds are inequality constraints that must be explicitly defined at each collocation point for the states and at each finite element for the inputs. There might also be hard equality constraints on the decision variables. For example the system has to reach a desired terminal state or a desired quantity must be accurately satisfied at each finite element in order to avoid losses. It is advised to enforce other equality constraints on the finite elements and not on each collocation point.

2.4 Model predictive control

Model Predictive Control (MPC) involves application of optimisation theory in control. It uses the system model to describe and predict the future process states and optimises the current actions to achieve the control objective. This control strategy allows inclusion of feedback measurements to update the optimisation problem when the system is driven away from the model prediction (which is the case for chemical processes). This sampling action at discrete time intervals updates and recalculates the optimal control problem at each sampling time step allowing for better control compared to open loop optimisation since mathematical models are not exact representations of a plant process. The idea of MPC was first proposed by Richalet et al. (1978) and then Cutler and Ramaker (1980).

2.4.1 The MPC Algorithm

The MPC strategy is described by Mayne Mayne et al. (2000) as:

> Model predictive control is a form of control in which the current control action is obtained by solving, at each sampling instant, a finite horizon open loop optimal control problem, using the current state of the plant as the initial state; the optimization yields an optimal control sequence and the first control in this sequence is applied to the plant.

$$\min_{(\mathbf{x},\mathbf{u})} \quad \phi(\mathbf{x},\mathbf{u}) \tag{2.24a}$$

subject to: Equality constraints (ODEs or DAEs):

$$\mathbf{F}(\dot{\mathbf{x}},\mathbf{x},\mathbf{u}) = \mathbf{0} \tag{2.24b}$$

and, Inequality constraints (Variable bounds):

$$\mathbf{x}^{\text{low}} \leq \mathbf{x} \leq \mathbf{x}^{\text{high}} \tag{2.24c}$$

$$\mathbf{u}^{\text{low}} \leq \mathbf{u} \leq \mathbf{u}^{\text{high}} \tag{2.24d}$$

The basic MPC algorithm is summarised in table 2.3. A prediction horizon is the length of time in to the future which the calculations predict. It must be noted that once an optimal solution for the first step is computed, the prediction horizon for the next optimisation problem will move one step forward. This is a moving horizon approach and it can be seen in figure 2.7. Therefore, for a system under control the same optimal problem eq. (2.24) is solved over and over again at each time step to obtain a sequence of optimal control inputs. (Foss and Heirung, 2013) These control inputs are applied within one control step (finite element).

When the ODEs and DAEs that are constraints to the MPC problem are nonlinear functions, the problem is known as nonlinear model predictive control (NMPC).

2.4.2 Robust NMPC

Nonlinear model predictive control (NMPC) handles highly coupled systems with multiple inputs and outputs compared to classical linear

Table 2.3: Basic MPC algorithm (Foss and Heirung, 2013)

```
Algorithm: State feedback MPC procedure

for   $t = 0, 1, 2, \ldots$ 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 initial
          condition
          Apply the first control move $u_t$ from the solution
          above.
end for
```

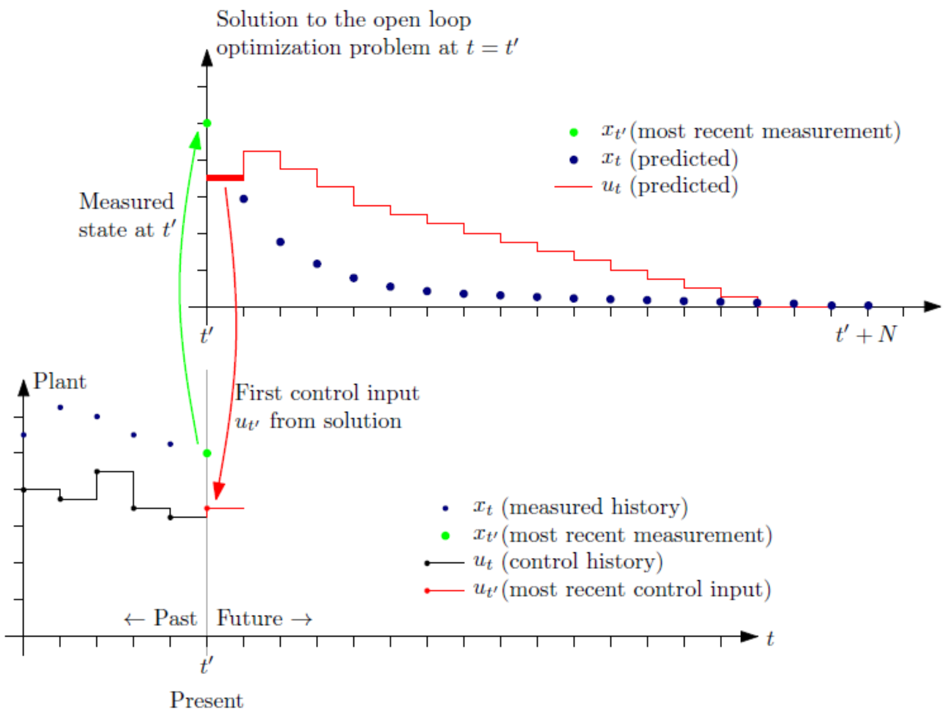

Figure 2.7: Illustration of MPC principle (Foss and Heirung, 2013)

control strategies. It is ideal for plants with tight constraints. Regardless of its advantages it is still not a commonly sought for approach to control industrial processes because model based approaches introduce uncertainty to the optimal control problem. The stability and performance of the NMPC controller is not guaranteed under uncertainty. It is always the case that a model has some degree of inaccuracy known as plant-model mismatch.

Standard MPC has issues dealing with uncertainty because the optimisation technique does not consider uncertainty. Over the years, developments of the optimisation formulations to handle huge uncertainty have been done and as a result optimisation techniques such as stochastic programming (Birge, 1997; Shapiro et al., 2009), dynamic programming (Bellman, 1957) and robust optimisation (Soyster, 1973) were discovered. Robust NMPC approach aspires to overcome the effect of model inaccuracy and errors in the standard NMPC while ensuring stability, no constraint violation, and recursive feasibility for all possible values in the uncertainty space. The standard MPC has an innate ability to reject large disturbances but is only valid when the strong assumptions mentioned by Grimm et al. (2004) hold. The assumptions are invalid for nonlinear constrained systems thus the need for robustification. The general formulation of robust optimisation is such that all parameters are uncertain but are known to be a part of a given uncertainty set. The optimal solution is found by minimising the cost for the worst case uncertainty realisation. This is also known as a min-max optimisation form.

Consequently, min-max MPC was conceived by describing the optimisation problem with an objective function of the worst case scenario minimisation and the constraints for all minimum and maximum uncertainty value cases. This ensured constraint satisfaction for the optimal control sequence solved based on the objective of minimizing the worst possible loss. When optimising a system with uncertain variables along the prediction horizon, we obtain prompt decisions that have to be made, and a set of possible future decisions which depend on where the system is driven. Therefore, first-stage decisions need to be made at the current time and depending on the uncertainty realized in the next step the optimiser will take a corresponding second-stage action. This phenomenon is known as *recourse*, which reduces the conservativeness of the current decisions. The min-max approach does not allow for recourse when new information from the plant measurements is already available. Scokaert and Mayne (1998) showed that this may lead to highly conservative solutions and infeasibilities.

A feedback min-max NMPC was designed by Lee and Yu (1997) and then Mayne (2001) which includes a sum of the costs for each control policies and not control input sequences in min-max without feedback. The approach is a closed-loop min-max NMPC which makes it adaptable and feasible due to possibility of taking recourse actions. However, this robustification technique generates a problem with infinite dimensions which deems the problem extremely difficult to solve.

Tube-based MPC approach, which was applied to linear systems has attained recent interest in nonlinear cases to replace min-max approaches. Tube based MPC is based on the solution of a nominal control problem and inclusion of an ancillary controller that ensures that the evolution of the real uncertain system stays in a tube whose cross-section is a positive invariant set, centered around the nominal trajectory. This approach can be difficult to adapt for nonlinear cases because it is a challenge to entirely define the possible set for the nominal problem, which can also be conservative.

Lucia et al. (2013) presented another approach called the multi-stage NMPC. The multi-stage approach is described as a promising framework for solving robust NMPC problems by including both standard and min-max NMPC. The main idea of this approach is to model the growth of uncertainty with each discrete time step as a scenario tree. Since measurement feedback will only be available in the future, it allows recourse which in turn minimizes the conservativeness of the resulting decisions.

2.4.3 Multi-stage NMPC

Multi-stage NMPC is a robustification scheme that combines standard and min-max NMPC approaches. The principal assumption that is taken to build up a multi-stage NMPC is that the uncertainty can be perfectly modeled by a scenario tree. The evolution of uncertainty takes up discrete values at each discrete time step. If the assumption is not valid, that we have a continuous uncertainty space (which is the case for real systems), the approach calculates the approximate optimal feedback policy that is near-optimal. However, much care is needed to pick the discrete values from the continuous uncertainty space.

The scenario tree that is used in multi-stage NMPC shown in fig. 2.8 describes the uncertainty evolution. The tree grows progressively towards the future by branching at the nodes. Every node represents an unknown uncertain event that influences the system apart from the applied control input. The tree indicates the relationship between the future control actions and their previous uncertainty realisations extracted from measurement information. This gives the ability for future control inputs to act as recourse variables to cancel out the effect of future uncertainties. In fact, multi-stage NMPC approach is a closed loop robust NMPC with a lower degree of conservativeness compared to others.

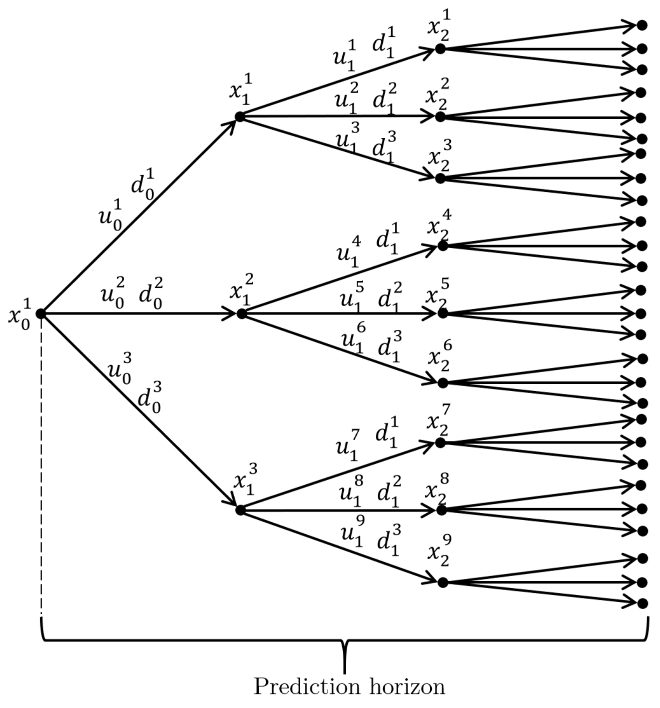

Figure 2.8: Scenario tree representation of uncertainty evolution for multi-stage NMPC

The tree growth represents the fact that if uncertainty at the future sampling time is unknown, then it will remain unknown for the next sampling time when the horizon moves forward. The branches in the tree are the uncertain parameter realisations and constraint violations can definitely occur for the values that are not represented in the tree. However, the worst case parameters are used as the bound values to create an uncertainty parameter interval and the branches are the combinations of these extreme parameter values. Therefore, it is the designer's task to chose a proper scenario tree for the desired system. Consequently if we have the combination, then the scenario tree will always grow exponentially with the number of realisations and prediction horizon length. This creates a rapid increase in problem size with prediction length, and is a major drawback of this approach.

When making current decisions of the control input, it is impossible to know the future outcome. The controller is unable to anticipate what will happen in the next time step and therefore it is required to enforce non-anticipativity constraints which ensure all the control inputs originating from the same node are always equal.

The scenario tree as in NLP formulation assumes not only discrete time steps but also discrete uncertainty realisations of the nonlinear system described. The realisations depends on the set of the disturbances (uncertain parameters) and thus the scenario tree model function $f : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_d} \to \mathbb{R}^{n_x}$ that maps current state to future state is written as in eq. (2.25) [3].

$$x_{k+1}^{l} = f(x_k^{p(l)}, u_k^l, d_k^{r(l)}), \tag{2.25}$$

The superscript $^{p(l)}$ identifies the parent node of the branched future state x_{k+1}^l. The parent node depends on the tree path taken that is a function of the tree position $p(l,k)$. Representation of stage index k is dropped to simplify the expressions. The superscript $^{r(l)}$ is the realisation of uncertainty at sampe time k and is also dependent on the scenario tree position $r(l,k)$. n_d is the size of the uncertainty vector.

[3]Note: index l is used to represent a scenario since index i indicates position in variable vectors, index k time discretisation, and j indicates collocation points for the states.

From fig. 2.8, $x_2^6 = f(x_1^2, u_1^6, d_1^3)$. It is convenient to have equal number of branches from all nodes hence a constant dimension for vector $d_k^{r(l)} \in \{d_k^1, d_k^2, \ldots, d_k^s\}$ at time k and s uncertainty realisations.

Referencing in the scenario tree can be further simplified by using I to denote the set of all node positions (l, k). A scenario is any continuous path through a set of nodes that begins from the root node x_0 to any leaf (end) node $x_{N_P}^l$. A set of all the nodes and their variables (states and control inputs) through which a one scenario takes path is denoted by S_i. Therefore, an ith scenario can be generically represented as:

$$S_i = \{x_{N_P}^i, x_{N_P-1}^{p(i)}, x_{N_P-2}^{p(p(i))}, \ldots, x_0^1, u_{N_P-1}^i, u_{N_P-2}^{p(i)}, x_{N_P-3}^{p(p(i))}, \ldots, u_0^i\}, \quad \forall i = 1, \ldots, N, \tag{2.26}$$

Here, N is the total number of scenarios, which in the case of equal branching nodes is the same as the leaf node count. N_P is the length of prediction horizon. A subset of S_i that contains all the states in the ith scenario is denoted as:

$$X_i = \{x_{N_P}^i, x_{N_P-1}^{p(i)}, x_{N_P-2}^{p(p(i))}, \ldots, x_0^1, \}, \qquad \forall i = 1, \ldots, N, \tag{2.27}$$

In the same way, the sequence of control inputs for the ith scenario is written as:

$$U_i = \{u_{N_P-1}^i, u_{N_P-2}^{p(i)}, u_{N_P-3}^{p(p(i))}, \ldots, u_0^i\}, \qquad \forall i = 1, \ldots, N, \tag{2.28}$$

After understanding and identifying the scenarios and set of variables in the multi-stage optimisation formulation the optimisation problem resulting from this setting is written as in eq. (2.29).

$$\min_{x_k^l, u_k^l \forall (l,k) \in I} \quad \sum_{i=1}^{N} \omega_i J_i(X_i, U_i) \tag{2.29a}$$

subject to:

$$x^l_{k+1} = f(x^{p(l)}_k, u^l_k, d^{r(l)}_k), \qquad \forall (l, k+1) \in I, \tag{2.29b}$$

$$g(x^l_{k+1}, u^l_k) \leq 0, \qquad \forall (l, k+1) \in I, \tag{2.29c}$$

$$u^l_k = u^m_k \text{ if } x^{p(l)}_k = x^{p(m)}_k \qquad \forall (l, k), (m, k) \in I, \tag{2.29d}$$

where, $g : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \rightarrow \mathbb{R}^{n_g}$ (eq. (2.29c))is a general inequality constraint function on the states and inputs at each scenario tree node. n_g is the number of constraints at each node. The scenario tree cost function in eq. (2.29a) is a weighted sum of the cost J_i of each scenario S_i assigned weight ω_i where $J_i : \mathbb{R}^{n_x \times N_P + 1} \times \mathbb{R}^{n_u \times N_P} \rightarrow \mathbb{R}$ is a scalar cost.

Non-anticipativity constraints (eq. (2.29d)) are listed in such a way that the current decision vectors u^l_k having the same parent node $x^{p(l)}_k$ have to be equal. The problem is reduced to a standard NMPC when $N = 1$. The weights ω_i are decided based on stochastic data or parameter estimation when available. It is normal to chose identical weights in absence of such information. A drawback of an evolving scenario tree is how fast the problem grows with increase in uncertainty realisations considered and the prediction horizon. The number of scenarios is given by s^{N_P} ,when there are s uncertainty realisations branching from each parent node. Moreover, the total number of nodes is a sum of an exponential sequence involving N_P and s. A large number of nodes implies larger number of decision variables which deems the problem difficult to compute. This challenge also known as the *curse of dimensionality* can be overcame by making the following important assumption as outlined by Lucia et al. (2013):

> A simple strategy to deal with tree growth with prediction horizon is to assume that the uncertainty remains constant after a certain point in time (called the robust horizon).

This assumption can be justified by the receeding horizon feature of NMPC, which makes accurate modeling of future uncertainties unnecessary. The control inputs will be recomputed at the next time step anyway. Figure 2.9 illustrates such simplification. The simplification does not affect the performance of the multi-stage NMPC as shown by Lucia et al. (2013).

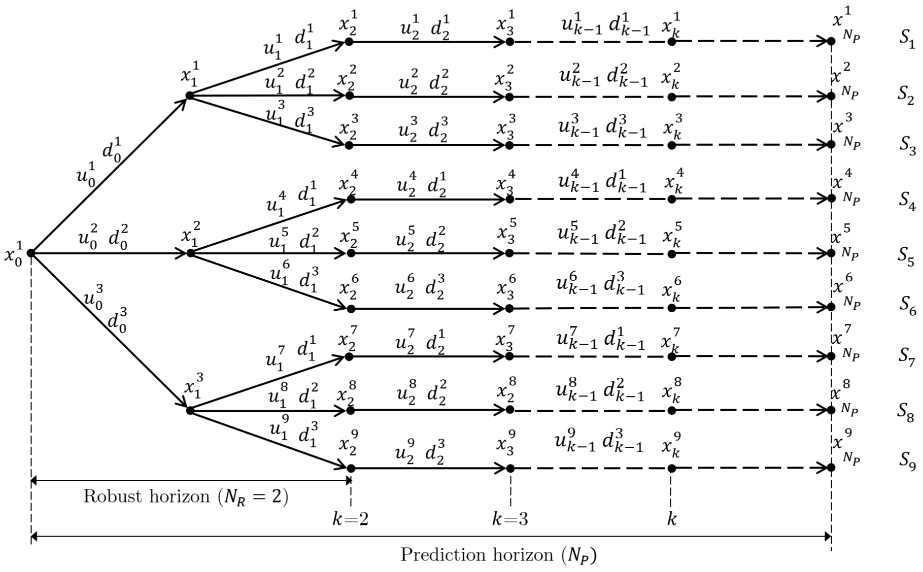

Figure 2.9: Scenario tree representation of the uncertainty evolution for multi-stage NMPC with robust horizon

Chapter 3

Implemetation methods

This chapter explains the default tools and methods that are employed to perform simulation case studies in this thesis. The chapter presents the general approach used in modeling and simulation of the controlled systems. All the descriptions here are assumed to be applied to obtain the presented results unless specified otherwise.

3.1 Modeling

The process that has to be controlled is described. In this work, case studies have been done on two systems. The first system is a direct thermal energy transfer from a thermal source to a demand stream without any thermal storage. The other system of interest in this work is a two plant thermal energy storage system.

Modeling of the two-plant thermal energy storage has been done in the previous specialisation project (Mdoe, 2018) but some modifications on the model have been added in this work. Due to the modifications, the modeling of the system has been presented again in this report. Direct thermal heat transfer model is written down as simple heat exchanger model using similar equations from the two-plant energy storage system. The models used are based on physical laws subject to some assumptions that are clearly explained in the modeling section.

3.2 Simulations

After obtaining the model for the controlled systems, which is a set of DAES, simulations for interesting control scenarios were done. Unless stated otherwise, all the simulations were performed in a MATLAB (2018) environment installed on an Intel Core i7 CPU at 2.7 GHz running Windows 10 Pro with 8 GB RAM.

In the MATLAB environment, CasADi was used as a tool for improving computational performance and simpler code implementation. CasADi is an open-source framework which allows for easy and efficient implementation of optimal control (Andersson, 2013). The advantage is that an NLP can be represented in a high-level symbolic form but in the background it is represented as an expression graph with the help of included ODE or DAE integrators. The representation is easy for any available NLP solver to solve the NLP, or even a custom one. CasADi is an automatic differentiator and so it calculates and provides the first-order and second-order derivative information for the NLP solver. Therefore, CasADi helps user avoid errors in derivative calculations, which is often very tedious for nonlinear models.

The NLP solver used in this work is IPOPT (Interior Point OPTimiser) developed by Wächter and Biegler (2006). The solver uses the automatic derivative information provided by CasADi. In this thesis, a direct collocation approach was implemented for all cases using Radau collocation points with 3rd order interpolation polynomials.

To illustrate plant-model mismatch, a real plant had to be implemented that produces real measurement data. The simulation of real plant was done by writing a dynamic model function in MATLAB that was integrated for a length of sample time with optimal inputs provided by the NLP solver (optimiser) using `ode15s` function. `ode15s` is a stiff ODE solver and it is also capable of solving DAEs when a Mass matrix is specified and passed as an argument in the `odeset` structure. The Mass matrix is a square diagonal matrix whose dimension matches the dynamic variables.

The diagonal index corresponding to differential variables are set to ones while those for algebraic variables are set to be zero.

The standard NMPC was performed for both cases: first perfect knowledge of the plant behaviour is assumed, then when there is plant-model mismatch. In both cases, the general implementation follows what has been explained above.

For multi-stage NMPC, an overview of all the possible uncertain parameters was done. The most important parameters whose uncertainty has greater effect on the operation were identified. The uncertainty space of each parameter was established. In this case a mean value was assumed and depending on the level of confidence of the mean, an assuming normal Gaussian distributions for the parameters, high and low uncertainty realisations were obtained. then according to the box design the extreme realisations were set as the uncertainty levels in which the scenario tree evolves. In the simulations involving Gauss distributions about a mean, a 95% confidence level was assumed. The multi-stage NMPC were implemented with a robust horizon of 1 ($N_R = 1$).

Chapter 4

Modeling of thermal energy grids

In this chapter, governing equations for thermal energy grids are derived. Simple case studies with two plants where one is a heat source and another a heat sink are considered. The case studies are done on the following simple systems of interest:

1. A two plant system with thermal storage and direct storage heating, and
2. A simple two plant system of direct thermal transfer without storage.

The model equations are derived from principles of mass and energy conservation following some assumptions. The result is a mathematical model which is necessary for the numerical case studies performed in this thesis.

4.1 System description and assumptions

Mdoe (2018) derived a model that describes a systems which is a thermal energy grid that consists of one supply from a hot stream, a variable direct storage heating for example flue gases and solar heating that directly heats a storage tank, a sink plant or demand stream with demand satisfaction constraints that can be met by purchasing energy from an external source in the market. The following assumptions were taken in order to derive the system model:

1. The sources and sinks are considered as reservoirs such that flow streams originating from them have their intensive properties. This implies that the temperature of the source and sink streams are only dependent of the plants.

2. All heat exchangers are modeled as two lumps (hot and cold sides) with heat transfer flow between them.

3. The heat exchanger sides are assumed to have uniform temperature, and equal to the temperature of their outlet streams. This assumption is valid because the heat exchanger exit temperatures have faster dynamics compared to the storage tank temprature dynamics. It is also possible to make a steady state approximation for the heat exchangers since the modeling is focused on optimal control which is at a slower time scale.

4. The model does not account for heat losses from the heat exchanger or flow pipelines to the surroundings. It also assumes that the overall heat transfer coefficient U_{hex} is constant.

5. All heat exchangers in the system are identical, that is, they have the same dimensions and parameters.

6. The storage has no temperature stratification. It is uniform throughout the volume and the storage holdup is controlled and always constant at V_{tank}.

7. The exit temperature from the tank is equal to the storage temperature.

8. The fluid streams have constant specific heat capacity c_p, and have the physical properties of water.

9. The conventional energy flow is from the supply side to the demand.

The following two sections present how the mathematical model is obtained with the aforementioned assumptions for our systems of interest.

4.2 Modeling of a two plant energy storage system

A pictorial representation of the system that is a framework for model derivation is given by a topology in figure 4.1.

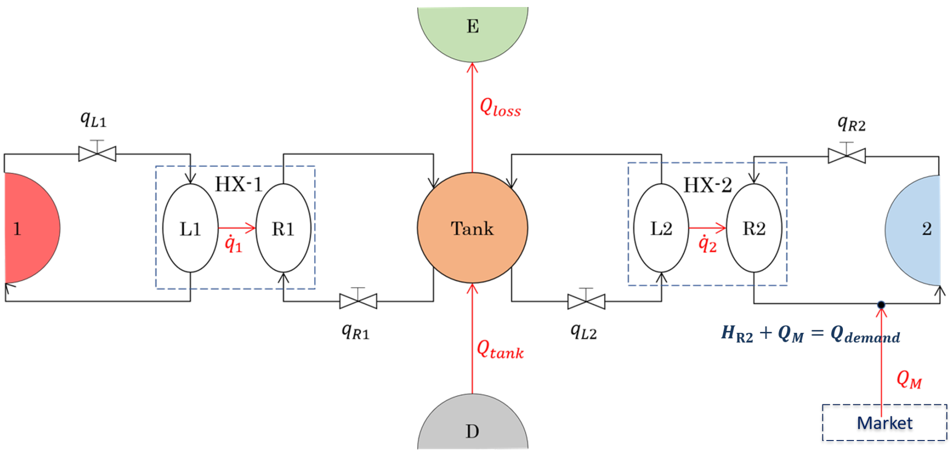

Figure 4.1: Topology of a simple thermal energy grid including thermal storage

Energy and Mass Balances

Following assumption 1, the sources and sinks have temperatures T_1 and T_2 respectively that are not affected by the system dynamics. They will be considered constant or given.

Source:

$$\frac{\mathrm{d}H_1}{\mathrm{d}t} = 0, \quad T_1 = \text{constant (given)} \tag{4.1}$$

Sink:

$$\frac{\mathrm{d}H_2}{\mathrm{d}t} = 0, \quad T_2 = \text{constant (given)} \tag{4.2}$$

Lumps:

1. **Heat exchanger 1:** Starting with the mass balances across the heat exchanger gives equation 4.3.

$$\frac{\mathrm{d}(\rho V_{\text{hex}})}{\mathrm{d}t} = \rho q_{1|L1} - \rho q_{L1|1} \tag{4.3a}$$

$$\frac{\mathrm{d}V_{\text{hex}}}{\mathrm{d}t} = q_{1|L1} - q_{L1|1} \tag{4.3b}$$

While in operation, the heat exchanger is filled with fluid so it is always constant i.e. $\frac{\mathrm{d}V_{\text{hex}}}{\mathrm{d}t} = 0$. Therefore, the inlet and outlet flows of the heat exchanger are equal. For simplicity, the following notations in eq. (4.4) will be used from now on in this report instead.

$$q_{1|L1} = q_{L1|1} = q_{L1} \tag{4.4a}$$

$$q_{\text{tank}|R1} = q_{R1|\text{tank}} = q_{R1} \tag{4.4b}$$

$$q_{\text{tank}|L2} = q_{L2|\text{tank}} = q_{L2} \tag{4.4c}$$

$$q_{2|R2} = q_{R2|2} = q_{R2} \tag{4.4d}$$

The outlet temperatures of the left and right side of the heat exchanger 1 (HX-1) are denoted as T_{L1} and T_{R1} respectively. The temperature of all the storage tank's outlet streams are equal and denoted as T_{tank}. Energy is conserved across each side of the heat exchanger. Starting with the left side L1 the energy conservation equation is eq. (4.5a).

$$\frac{\mathrm{d}H_{L1}}{\mathrm{d}t} = \sum H_{\text{in}} - \sum H_{\text{out}} + Q_{\text{net}} - W_s \tag{4.5a}$$

Adiabatic conditions are assumed and there is no shaft work done by the system i.e. Q_{loss} = 0 and W_s = 0, then

$$\frac{\mathrm{d}(\rho c_p V_{\text{hex}} T_{L1})}{\mathrm{d}t} = \rho c_p q_{L1} T_1 - \rho c_p q_{L1} T_{L1} - Q_{L1|R1} \tag{4.5b}$$

$$\rho c_p V_{\text{hex}} \frac{\mathrm{d}T_{L1}}{\mathrm{d}t} = \rho c_p q_{L1}(T_1 - T_{L1}) - Q_{L1|R1} \tag{4.5c}$$

(4.5d)

Where $Q_{L1|R1}$ stands for energy transfer rate from lump L1 to R1. Hence, equation for temperature dynamics of stream q_{L1} given by equation 4.7.

$$\frac{\mathrm{d}T_{L1}}{\mathrm{d}t} = \frac{1}{V_{\text{hex}}}\left\{ q_{L1}(T_1 - T_{L1}) - \frac{Q_{L1|R1}}{\rho c_p} \right\} \tag{4.6}$$

The right side, R1 dynamics can be found with the same steps done in the left side L1 to give equations 4.7.

$$\frac{\mathrm{d}T_{R1}}{\mathrm{d}t} = \frac{1}{V_{\text{hex}}}\left\{ q_{R1}(T_{\text{tank}} - T_{R1}) + \frac{Q_{L1|R1}}{\rho c_p} \right\} \tag{4.7}$$

2. **Heat exchanger 2:** The design parameters of every heat exchanger is the same so the energy balances for the second heat exchanger will result to similar equations with different variables shown in equation 4.8 and 4.9. The outlet temperatures of the left and right sides of the heat exchanger 2 (HX-2) are denoted as T_{L2} and T_{R2} respectively.

$$\frac{\mathrm{d}T_{L2}}{\mathrm{d}t} = \frac{1}{V_{\text{hex}}}\left\{ q_{L2}(T_{\text{tank}} - T_{L2}) - \frac{Q_{L2|R2}}{\rho c_p} \right\} \tag{4.8}$$

$$\frac{\mathrm{d}T_{R2}}{\mathrm{d}t} = \frac{1}{V_{\text{hex}}}\left\{ q_{R2}(T_2 - T_{R2}) + \frac{Q_{L2|R2}}{\rho c_p} \right\} \tag{4.9}$$

3. **Heat transferred from hot to cold side:** Heat transfer model between cold and hot streams in heat exchangers is given by 2.5.

$$Q = U_{\text{hex}} A_{\text{hex}} \Delta T_m \tag{4.10}$$

where, ΔT_m is the mean temperature difference between the cold and hot streams. The appropriate value for this is the LMTD, (ΔT_{LM}) but a polynomial approximation is suited for practical purposes instead. These approximations are by Chen (1987) ΔT_{CM}, Underwood (1970) ΔT_{UM}. The ΔT_m can be simply expressed as the differences between the inlet temperatures of hot and cold sides or as an arithmetic mean ΔT_{AM}. The simple approximations are suitable for cases when the ratio of flow rates of either streams is close to 1. Otherwise when one of the flows in very large compared to another then the approximation becomes very poor. In the specialisation project work (Mdoe, 2018) performed on this same system, simulation results indicated that the Underwood and Chen mean approximations produced the best results. In this thesis the Underwood mean approximation has been considered.

$$Q_{L1|R1} = U_{\text{hex}} A_{\text{hex}} \Delta T_{m,1} \quad \text{and} \quad Q_{L2|R2} = U_{\text{hex}} A_{\text{hex}} \Delta T_{m,2}$$

where,

$$\Delta T_{m,1}^{n} = 0.5\left[(T_1 - T_{R1})^n + (T_{L1} - T_{\text{tank}})^n\right], \quad n = \frac{1}{3}$$

$$\Delta T_{m,2}^{n} = 0.5\left[(T_{L2} - T_2)^n + (T_{\text{tank}} - T_{R2})^n\right], \quad n = \frac{1}{3}$$

Using the simpler approximation the mean temperature differences can be estimated as equations 4.11.

$$\Delta T_{m,1} = T_1 - T_{R1} \tag{4.11a}$$

$$\Delta T_{m,2} = T_{\text{tank}} - T_{R2} \tag{4.11b}$$

4. **Storage tank:** Beginning with mass balances we obtain equation 4.12.

$$\frac{\mathrm{d}(\rho V_{\text{tank}})}{\mathrm{d}t} = \rho q_{R2} + \rho q_{L2} - \rho q_{R2} - \rho q_{R2} = 0 \tag{4.12}$$

Energy balance across the tank following the general energy balance equation 4.5a provided the enthalpies and the net heat flow in equations 4.13 leads up to equation 4.14.

$$\sum H_{\text{in}} = \rho c_p (q_{R1} T_{R1} + q_{L2} T_{L2}) \tag{4.13a}$$

$$\sum H_{\text{out}} = \rho c_p (q_{R1} T_{\text{tank}} + q_{L2} T_{\text{tank}}) \tag{4.13b}$$

$$Q_{\text{net}} = Q_{D|\text{tank}} - Q_{\text{tank}|E} \tag{4.13c}$$

$$\frac{\mathrm{d}H_{\text{tank}}}{\mathrm{d}t} = \rho c_p(q_{R1}(T_{R1} - T_{\text{tank}}) + q_{L2}(T_{L2} - T_{\text{tank}})) + Q_{D|\text{tank}} - Q_{\text{tank}|E} - 0 \tag{4.14}$$

Let $Q_{D|\text{tank}} = Q_{\text{tank}}$ and $Q_{\text{tank}|E} = Q_{\text{loss}}$

$$\frac{\mathrm{d}T_{\text{tank}}}{\mathrm{d}t} = \frac{1}{V_{\text{tank}}}\left\{q_{R1}(T_{R1} - T_{\text{tank}}) + q_{L2}(T_{L2} - T_{\text{tank}}) + \frac{Q - Q_{\text{loss}}}{\rho c_p}\right\} \tag{4.15}$$

Hence storage tank temperature dynamics.

Basing on the listed assumptions, the system model was derived and the final model is a set of ordinary differential equations listed in eq. (4.16).

$$\frac{\mathrm{d}T_{L1}}{\mathrm{d}t} = \frac{1}{V_{\text{hex}}}\left\{q_{L1}(T_1 - T_{L1}) - \frac{U_{\text{hex}}A_{\text{hex}}}{\rho c_p}\Delta T_{m,1}\right\} \tag{4.16a}$$

$$\frac{\mathrm{d}T_{R1}}{\mathrm{d}t} = \frac{1}{V_{\text{hex}}}\left\{q_{R1}(T_{\text{tank}} - T_{R1}) + \frac{U_{\text{hex}}A_{\text{hex}}}{\rho c_p}\Delta T_{m,1}\right\} \tag{4.16b}$$

$$\frac{\mathrm{d}T_{L2}}{\mathrm{d}t} = \frac{1}{V_{\text{hex}}}\left\{q_{L2}(T_{\text{tank}} - T_{L2}) - \frac{U_{\text{hex}}A_{\text{hex}}}{\rho c_p}\Delta T_{m,2}\right\} \tag{4.16c}$$

$$\frac{\mathrm{d}T_{R2}}{\mathrm{d}t} = \frac{1}{V_{\text{hex}}}\left\{q_{R2}(T_2 - T_{R2}) + \frac{U_{\text{hex}}A_{\text{hex}}}{\rho c_p}\Delta T_{m,2}\right\} \tag{4.16d}$$

$$\frac{\mathrm{d}T_{\text{tank}}}{\mathrm{d}t} = \frac{1}{V_{\text{tank}}}\left\{q_{R1}(T_{R1} - T_{\text{tank}}) + q_{L2}(T_{L2} - T_{\text{tank}}) + \frac{1}{\rho c_p}(Q - Q_{\text{loss}})\right\} \tag{4.16e}$$

where,

$$Q_{\text{loss}} = (UA)_{\text{tank}}(T_{\text{tank}} - T_{\text{surr}}) \tag{4.16f}$$

$$\Delta T_{m,1}^n = 0.5\big[(T_1 - T_{R1})^n + (T_{L1} - T_{\text{tank}})^n\big], \quad n = \frac{1}{3} \tag{4.16g}$$

$$\Delta T_{m,2}^n = 0.5\big[(T_{L2} - T_2)^n + (T_{\text{tank}} - T_{R2})^n\big], \quad n = \frac{1}{3} \tag{4.16h}$$

There are five differential states in the model. The other variables can be classified as disturbances or inputs depending on whether they can be manipulated within the system boundaries or not.

- Differential states ($\mathbf{x}$): there are 5 states variables in the system.

$$\mathbf{x} = \begin{bmatrix} T_{L1} & T_{R1} & T_{L2} & T_{R2} & T_{\text{tank}} \end{bmatrix}^\top \tag{4.17}$$

- Inputs ($\mathbf{u}$): there are 5 input variables in the system. These variables could be possibly manipulated.

$$\mathbf{u} = \begin{bmatrix} q_{L1} & q_{R1} & q_{L2} & q_{R2} & Q_{\text{tank}} \end{bmatrix}^\top \tag{4.18}$$

- Disturbances ($\mathbf{d}$): there are at least 3 disturbances in the system. These variables are uncertainties that can not be manipulated within the system boundaries.

$$\mathbf{d} = \begin{bmatrix} T_1 & T_2 & T_{\text{surr}} \end{bmatrix}^\top \tag{4.19}$$

- Parameters: the remaining variables are parameters as long as they are constant with respect to time. They depend on the design of the system, material properties of the storage fluid and heat exchanger.

Therefore, the system's model can be simply written as: $\dot{\mathbf{x}} = \mathbf{F}(\mathbf{x}, \mathbf{u}, \mathbf{d})$

In the Underwood approximation for heat exchanger modeling, where $n = \frac{1}{3}$, it is best to express the model as a set of differential and algebraic equations to avoid having terms with root expressions (fraction indexes). This is because the derivatives of roots are rational, generating singularity problems when gradients are calculated by CasADi at points with state values approaching zero.

Therefore, new states are assigned to slightly adapt the model to deal with the aforementioned numerical issues. The resulting model will be of a higher dimension with extra algebraic states. For the simple two plant model case, 4 new algebraic states are defined which are new variable vector $\mathbf{z}$.

$$\mathbf{z} = \begin{bmatrix} a & b & c & d \end{bmatrix}^\top \tag{4.20}$$

where:

$$a = (T_1 - T_{R1})^n \tag{4.21a}$$
$$b = (T_{L1} - T_{\text{tank}})^n \tag{4.21b}$$
$$c = (T_{L2} - T_2)^n \tag{4.21c}$$
$$d = (T_{\text{tank}} - T_{R2})^n \tag{4.21d}$$

which can be written in an implicit form as eq. (4.22).

$$a^{\frac{1}{n}} - (T_1 - T_{R1}) = 0 \tag{4.22a}$$
$$b^{\frac{1}{n}} - (T_{L1} - T_{\text{tank}}) = 0 \tag{4.22b}$$
$$c^{\frac{1}{n}} - (T_{L2} - T_2) = 0 \tag{4.22c}$$
$$d^{\frac{1}{n}} - (T_{\text{tank}} - T_{R2}) = 0 \tag{4.22d}$$

The set of algebraic equations can be written in general form as $\mathbf{G}(\mathbf{x}, \mathbf{z}, \mathbf{u}, \mathbf{d}) = \mathbf{0}$. The model becomes a differential algebraic system of equations (DAE) when combined with the rest of the differential states. The algebraic equations eq. (4.22) are added to the first five differential equations in eq. (4.16). The equations eq. (4.16g) and eq. (4.16h) are written as eq. (4.23a) and eq. (4.23b) respectively.

$$\Delta T_{m,1}^n = 0.5(a + b), \quad n = \frac{1}{3} \tag{4.23a}$$
$$\Delta T_{m,2}^n = 0.5(c + d), \quad n = \frac{1}{3} \tag{4.23b}$$

The DAE model is expressed in a general semi-explicit form expression as eq. (4.24)

$$\dot{\mathbf{x}} = \mathbf{F}(\mathbf{x}, \mathbf{z}, \mathbf{u}, \mathbf{d}) \tag{4.24a}$$
$$\mathbf{0} = \mathbf{G}(\mathbf{x}, \mathbf{z}, \mathbf{u}, \mathbf{d}) \tag{4.24b}$$

Expressing the model as a set of DAEs with 9 states (5 differential, 4 algebraic) instead of an ODE with only 5 differential states makes the optimal control problem and consequently the NLP larger and slightly slower to solve compared to the previous model but has the following advantages:

1. DAE formulation increases the chances of the solver, IPOPT, converging to an optimal solution of the NLP by avoiding errors when the differentiation algorithm, CasADi, fails to compute the Jacobians for the NLP due to singularity.

2. The NLP can be solved from any possible initial state of the system given to the solver.

We need a model for a system that has no storage. This model is to be used as a base case (control) when showcasing the performance of a thermal energy storage.

4.3 Modeling thermal supply without storage

The pictorial representation of the system without storage is shown in fig. 4.2. The rate of heat transfer from the supply side is limited by the heat transfer across the heat exchanger. Otherwise, the rest of the heat is dumped out by a cooling process, for example. Applying the same principles and assumptions taken in the derivation of the simple two plant system with storage we obtain the following equations 4.25.

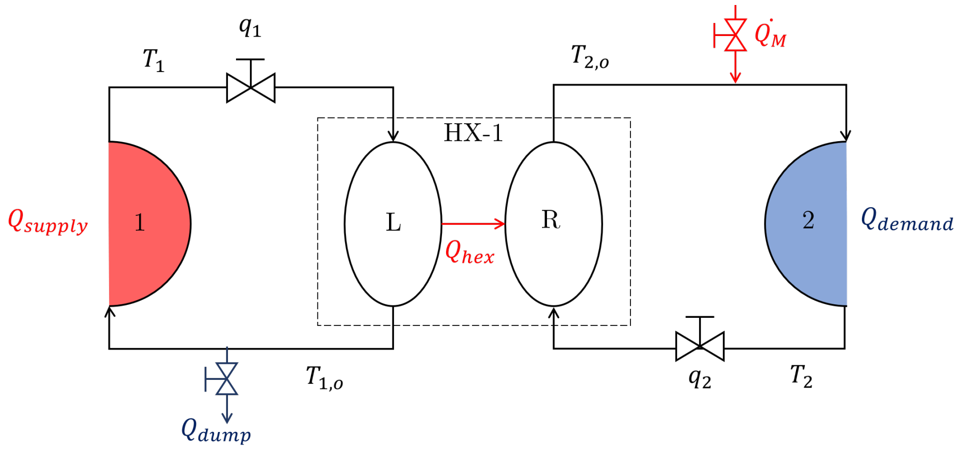

Figure 4.2: Topology of a simple thermal energy grid without thermal storage (direct heat supply)

$$\frac{\mathrm{d}T_{1,o}}{\mathrm{d}t} = \frac{1}{V_{\text{hex}}}\left\{q_1(T_1 - T_{1,o}) - \frac{U_{\text{hex}}A_{\text{hex}}}{\rho c_p}\Delta T_m\right\} \tag{4.25a}$$

$$\frac{\mathrm{d}T_{2,o}}{\mathrm{d}t} = \frac{1}{V_{\text{hex}}}\left\{q_2(T_2 - T_{2,o}) + \frac{U_{\text{hex}}A_{\text{hex}}}{\rho c_p}\Delta T_m\right\} \tag{4.25b}$$

$$0 = a^{\frac{1}{n}} - (T_1 - T_{2,o}) \tag{4.25c}$$

$$0 = b^{\frac{1}{n}} - (T_{1,o} - T_2) \tag{4.25d}$$

where,

$$\Delta T_m^n = 0.5(a+b), \quad n = \frac{1}{3} \tag{4.25e}$$

$$\tag{4.25f}$$

The model variables in this no storage system can be classified as follows:

- Differential states ($\mathbf{x}$): there are 2 state variables in the system.

$$\mathbf{x} = \begin{bmatrix} T_{1,o} & T_{z,o} \end{bmatrix}^\top \tag{4.26}$$

- Inputs ($\mathbf{u}$): there are 2 input variables in the system. These variables could be possibly manipulated.

$$\mathbf{u} = \begin{bmatrix} q_1 & q_2 & Q_{\text{M}} & Q_{\text{dump}} \end{bmatrix}^\top \tag{4.27}$$

As a part of modeling a thermal energy supply grid, it is important to understand how to model energy demand. The demand must be anticipated and forecasted. These forecasted demand profiles are set as operational constraints that a control strategy must abide to. Depending on the nature of the demand-side, there are common demand patterns that are discussed in the next section.

4.4 Energy demand modeling

To perform energy storage control simulations there must be a model for prediction of energy demand. This is usually a forecast that depends on the historical data of the demand-side and other factors such as season of the year, time of the day. The process of obtaining such models is out of the scope of this report. However, we shall discuss some interesting demand trends. Since thermal energy storages are used for intermittent storage to cater for diurnal supply-demand mismatch we are interested on daily demand curves (within 24 hours).

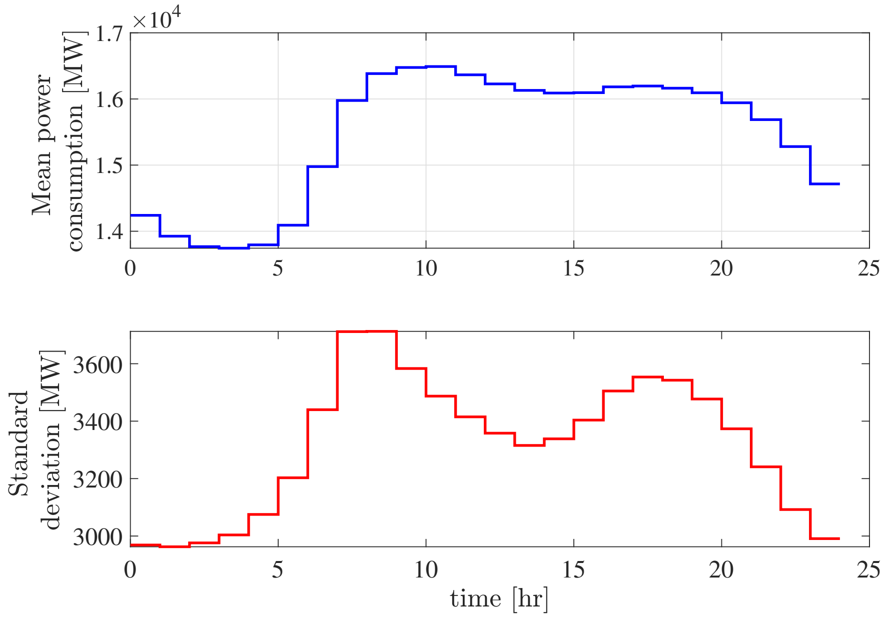

Figure 4.3: Average diurnal electricity demand in Norway for the year 2018 (NordPool, 2018)

Figure 4.3 shows the mean hourly electrical power demand in a single day in Norway as recorded by NordPool (2018). The feature of this demand curve is a sharp rise to a peak at 9-10 AM in the morning. This is when most households are awake and are using a lot of electrical devices for cooking, heating water. As the day goes on the demand gradually decreases and past 8 PM it sharply decreases because people are going to sleep. The lowest demands are at 4 AM the morning.

The same figure 4.3 also shows the standard deviation (uncertainty) associated with the mean hourly demands. It is vivid that there is some correlation between the magnitude of demand with uncertainty. The uncertainty is highest at the peak demand hours and lowest at the low demand hours. This gives us an idea that when modeling uncertainty in demand one must consider the change in uncertainty levels at each hour depending on the magnitude of the mean hourly demand. The variance of the demand also depends on the season of operation. There is the greatest variance in the winter season and variance is lowest in summer period.

Another typical daily energy demand curve is the "duck curve" demand profile. California Independent system Operator (CAISO)'s duck curve is shown in figure 4.4. The duck curve is interesting since it shows the challenge of intermittent renewable resources such as solar and wind integration to energy supply grids. In the case of California state in the United States, there is abundant sunlight but only available in the day and not at night. This challenge supports the role of energy storage for successful integration.

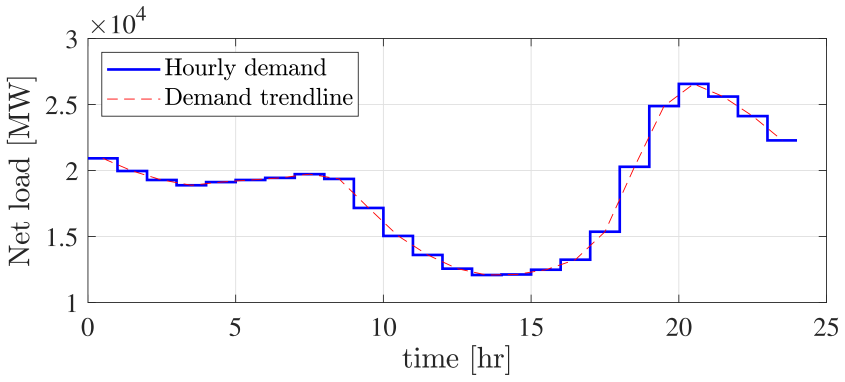

Figure 4.4: Estimated net load daily trend for the year 2020 from CAISO showing the "duck curve" (Burnett, 2016)

The demand data was scaled down to generate the same trend and a hypothetical supply curve from a solar supply was considered to show the mismatch of supply and demand peaks. See fig. 4.5. This scaled down data has been used as a case study in section 5.4.

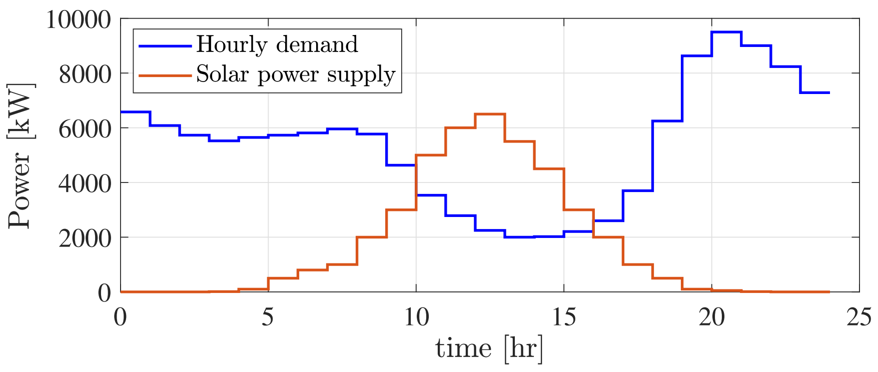

Figure 4.5: Scaled demand values of the California duck curve with a typical solar supply curve to illustrate the intermittent supply mismatch challenge

Chapter 5

NMPC control on thermal energy grids

This chapter demonstrates the implementation of an MPC controller on a simple thermal energy supply grid. First, we present a numerical case study that compares operation of the direct thermal supply system (see section 4.3) against a two plant energy grid with a thermal storage as described in the section 4.2. In both mentioned cases, there is zero direct tank heating (Q_{tank}), for better comparison. This is followed with another case study to demonstrate integration of direct storage heating using a renewable energy resource of intermittent solar power. Both of the cases already mentioned are standard NMPC control with state feedback and no uncertainty. Lastly, a plant-model mismatch is introduced in the supply and demand temperatures. This final case study is used to compare the performance of standard NMPC with multi-stage NMPC considering an operational constraint on the tank temperature.

5.1 Implementation description

After obtaining the model equations for the systems, they should be represented in the computer code in MATLAB. The code includes the following sections:

1. Parameter specification: The nominal system parameters must be specified before running a simulation for the system. The parameters used in the model are approximately realistic and are listed in table 5.1

Table 5.1: Model parameters and values used

Parameter	Description	Value	Units
U_{hex}	Overall heat transfer coefficient for heat exchanger	0.5	$\mathrm{kW/m^2K}$
A_{hex}	Heat transfer area for heat exchanger	300	$\mathrm{m^2}$
U_{tank}	Overall heat loss coefficient for tank	0.5	$\mathrm{W/m^2K}$
A_{tank}	Heat loss area for storage tank	100	$\mathrm{m^2}$
V_{tank}	Volume of the storage	10^3	$\mathrm{m^3}$
c_p	Specific heat capacity of storage material	4.186	$\mathrm{kJ/kgK}$
ρ	Density of storage material	1000	$\mathrm{kg/m^3}$
P_{M}	Cost per unit external energy	10^{-3}	-
P_{T}	Cost per unit direct tank heating	5×10^{-6}	-
T_1	Supply stream temperature	95	$°\mathrm{C}$
T_2	Demand stream temperature	20	$°\mathrm{C}$
T_{surr}	Ambient temperature	15	$°\mathrm{C}$

2. Optimiser: This is the part of the code that represents the NMPC controller. It includes the approximate model (expected model) and the formulation of the NLP. The direct collocation discretisation method was implemented using Radau 3rd order polynomials. As mentioned in section 3.2, IPOPT was the optimisation solver used to compute the optimal solution.

3. Plant simulator and observer: This part is used for sampling the state of the real plant. The system of DAEs are solved by an integrator for discretised time steps to obtain the approximate system dynamics. In this thesis, `ode15s` integrator in MATLAB was used to simulate the differential equations for the system representation. This

simulation was used to represent the “real plant”. The MATLAB function allows passing the system model and the required time span it should simulate in order to obtain the state values at every sampling time.

A standard MPC with state feedback is investigated here. The model derived in sections 4.3 and 4.2 were used in the calculations to represent their respective systems. To begin with the simulation results for a thermal energy supply grid without storage are presented in the next section.

5.2 Standard NMPC without thermal storage

To illustrate the importance of a thermal storage controlled by a standard NMPC controller we use a simple supply-demand mismatch scenario. Figure 5.1 shows the demand and supply profiles over 24 hours. The expected daily supply is always constant at 2500 kW, while the expected demand is 1500 kW for the first 12 hours and then immediately rises to 3500 $\mathrm{kW}$for the remainder of the day.

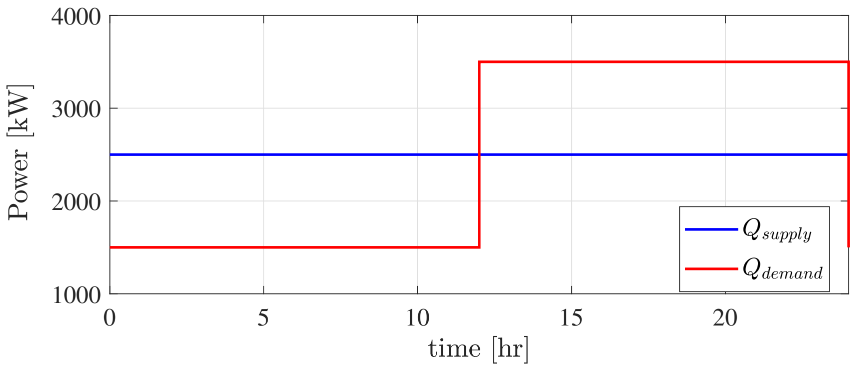

Figure 5.1: Simple supply-demand mismatch scenario in a thermal energy system

To begin with, let us investigate the operation of energy supply to a demand stream without storage as illustrated in fig. 4.2. A standard NMPC controller manipulates the flow of the demand-side stream q_2 into the heat exchanger, the purchased energy from the external heating source Q_{M} and the amount of heat dumped in the supply side Q_{dump}. The supply-side flow

q_1 is fixed at 50 ℓ/s. Therefore, in this case the manipulated variables $\mathbf{u} = \begin{bmatrix} q_2 & Q_\mathrm{M} & Q_\mathrm{dump} \end{bmatrix}^\top$. The control objective is an economic function to minimise the cost of external heating. The objective function is written in eq. (5.1).

$$\min_{(\mathbf{x},\mathbf{u})} \quad \phi(\mathbf{x},\mathbf{u}) \tag{5.1a}$$

where

$$\phi(\mathbf{x},\mathbf{u}) = P_\mathrm{M} Q_\mathrm{M} \tag{5.1b}$$

The optimisation objective eq. (5.1) is constrained by the system dynamics, demand satisfaction constraint and supply constraints. The constraints are categorised as follows:

1. Equality constraints: These are hard constraints that the system has to strictly satisfy for optimal and stable operation.
   (a) The decision variables are related by the system dynamic model and they can not violate it. If these constraints are violated, then the system will go unstable. The model of the system is an equality constraint in this optimal problem.

   $$\mathbf{F}(\dot{\mathbf{x}}, \mathbf{x}, \mathbf{z}, \mathbf{u}, \mathbf{d}) = \mathbf{0} \tag{5.2}$$

   (b) Consumer demand satisfaction: The scenario expects that the demand-side requires a specified amount of energy at a specific time. If the demand profile is constant then the demand is independent of time. However, it is almost always the case that there is a varying demand in the sink. It is of interest to see how a system without thermal storage handles this variations from the normal base load. In this scenario, a step increase in demand was considered assuming that the supply was always higher before the step time. The thermal power demand can be satisfied by the enthalpy gain of demand stream across the heat exchanger and power purchased from external resource. Hence eq. (5.3).

   $$Q_\mathrm{demand}(t_k) = Q_{\mathrm{M},k} + \rho c_p q_{2,k}(T_{2,o,k} - T_2) \tag{5.3}$$

where $Q_{\mathrm{M},k}$ is the energy purchased from external market at the k^{th} hour.

(c) Supply constraint: The thermal energy supply rate and the supply stream temperature at which the energy is available are both specified. In practice, these supply streams are hot process streams that require cooling to a specific temperature. If the energy system cannot provide enough cooling, the stream must be cooled further. Therefore a supply constraint is added to ensure that the system does not draw more energy that what is available. For the cases when the supply is higher than demand, energy can be dumped at a rate Q_{dump}. When the thermal energy supply rate is 2500 kW, it implies a return temperature of 83.05 °C.

$$Q_{\text{supply}}(t_k) = Q_{\text{dump},k} + \rho c_p q_{2,k}(T_1 - T_{1,o,k}) \tag{5.4}$$

where $Q_{\text{dump},k}$ is the rate of heat dumped in the supply side at the k^{th} hour.

2. Inequality constraints: These are relaxed conditions that the system must satisfy. They include:

(a) State bounds: The system is assumed to have a storage fluid with properties of water. Therefore, the storage fluid does not exceed a temperature of 100 °C and does not go below zero. The tank temperature is not allowed to go below 30 °C. Hence eq. (5.5).

$$\begin{bmatrix} 0 \\ 0 \end{bmatrix} \leq \begin{bmatrix} T_{1,o} \\ T_{2,o} \end{bmatrix} \leq \begin{bmatrix} 100 \\ 100 \end{bmatrix} \tag{5.5}$$

(b) Input bounds: The manipulated variables have saturation limits to which they cannot exceed. It is undesirable in practice to have an optimal solution outside the MV saturation limit range. In this scenario the volumetric flows can be adjusted between 0 and 50 ℓ/s. The power flows Q_M and Q_{dump} must be non-negative. Hence eq. (5.6).

$$\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \leq \begin{bmatrix} q_2 \\ Q_{\mathrm{M}} \\ Q_{\mathrm{dump}} \end{bmatrix} \leq \begin{bmatrix} 50 \\ +\infty \\ +\infty \end{bmatrix} \tag{5.6}$$

The standard NMPC above was implemented to the above scenario without thermal storage using the methods described earlier in section 3.2. The simulation assuming a perfect model for 24 hour operation yielded results that are plotted in figure 5.2.

Figure 5.2 shows that without storage the system dumps all the extra heat supplied in periods of low demand. The supply-demand difference is 1000 kW, and that is the thermal energy dumped in the low-demand phase. In the high demand period, the system purchases extra heating from the external source. The amount of extra heating purchased is equal to the difference between the peak demand and supply, that is 1000 kW. The flow rate of the demand stream increases from 4.78 to 8.12 ℓ/s at the demand step time. The controller cannot increase the flow to the maximum because it reaches equilibrium. Moreover, it is impossible to transfer thermal energy through the heat exchanger at a higher rate than the rate at which it is supplied.

The temperature difference between $T_{1,o}$ and $T_{2,o}$ is increased when the peak demand rises. This is also because of the same reason that the NMPC manipulates the heat exchanger to transmit all the available thermal power now that the demand has exceeded the supply. The rest of the energy required to meet the demanded thermal power is purchased from the external market. As it is evident in figure 5.2, past the 12th hour mark there is zero dumped heat, and the external energy purchased at a rate of 1000 kW.

The performance of the economic standard NMPC on a thermal energy supply system without storage has been presented in this section. The next section presents the case where a thermal storage is part of the thermal supply grid. The performance of the latter system is discussed and compared in the next section.

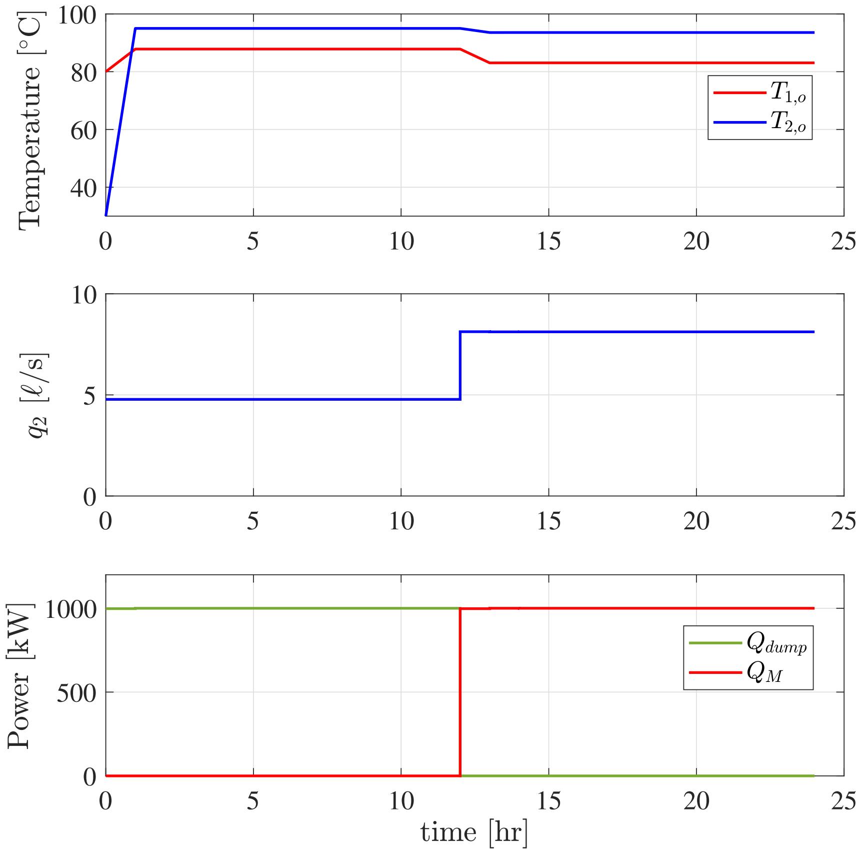

Figure 5.2: Simple scenario of power supply and demand mismatch for an energy system that has no thermal storage

5.3 Standard NMPC with thermal storage

Now we consider a thermal storage included in the thermal energy supply system (see fig. 4.1). The thermal supply is not connected directly to the demand side. The storage tank is expected to be heated up first by the supply stream, and then the storage “discharges” to the demand stream to satisfy demand requirements.

Again a standard NMPC operates the system under the similar supply-demand scenario in fig. 5.1. In this case we want to compare the behaviour of a thermal power supply with and without storage therefore the direct tank heating, Q_{tank} is set to 0 kW. The supply-side and storage outlet flows (q_{L1}, q_{R1} and q_{L2}) are fixed at 50 ℓ/s and only the demand-side flow q_{R2} is manipulated. The control objective is the same as before in eq. (5.1). The constraints here are the similar to those in section 5.2 except for the state bounds (see eq. (5.7)) and the equality constraints, where the set of DAEs corresponding to the thermal storage system (eq. (4.23)) were used. The 24 h simulation was done assuming no plant-model mismatch and the results were plotted as shown in figure 5.3.

$$\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 30 \end{bmatrix} \leq \begin{bmatrix} T_{L1} \\ T_{R1} \\ T_{L2} \\ T_{R2} \\ T_{\text{tank}} \end{bmatrix} \leq \begin{bmatrix} 100 \\ 100 \\ 100 \\ 100 \\ 100 \end{bmatrix} \tag{5.7}$$

Figure 5.3 shows that the tank temperature rises at the beginning when the demand is lower than the supply. Instead of the system dumping the extra heat, the tank provides capacity for storage of surplus thermal energy. However, as the temperature of the storage rises, the storage’s ability to withdraw heat from the supply stream decreases. This is due to the decrease in the mean temperature difference across the supply-side heat exchanger. The temperature difference can be thought of as the force that drives thermal current from the source. Therefore, as the storage temperature rises the rate of “charging” decreases.

At the 12th hour mark, the storage starts to cool down at a decreasing rate. Similar to charging process, the rate of storage thermal discharge is higher

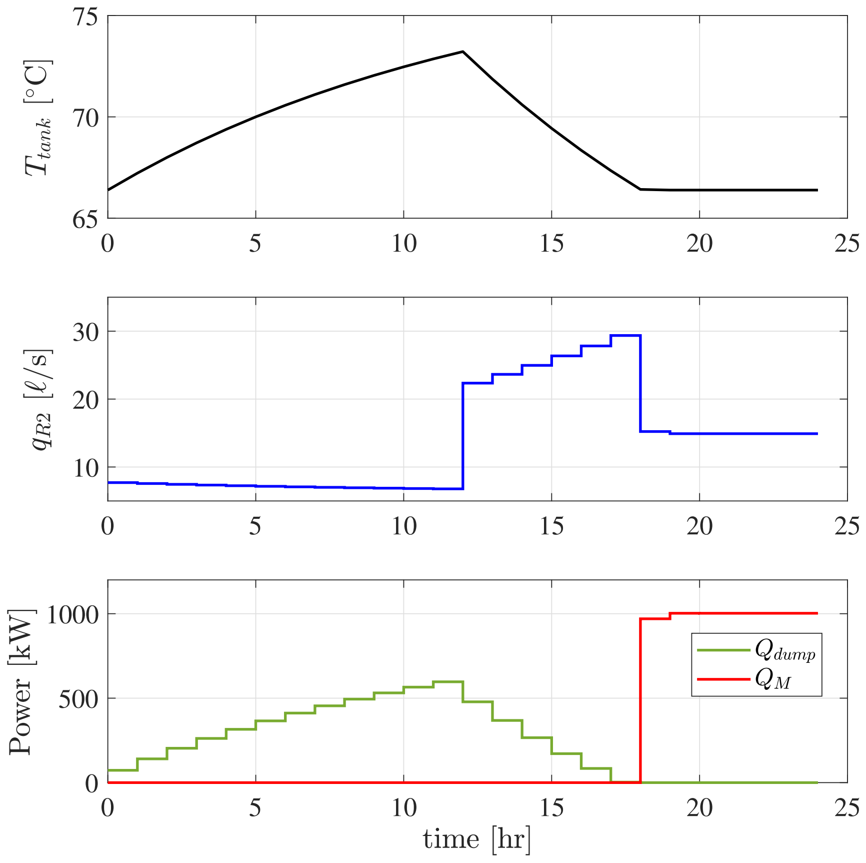

Figure 5.3: Simple scenario of power supply and demand mismatch for an energy system with thermal storage tank of volume 10^6 ℓ

than when the storage is hotter. However, the flow rate of the cold stream q_{R2} is manipulated by the NMPC to ensure thermal power demand is satisfied by the stored heat first as long as possible before purchasing from the market. This is why at the demand step time the controller increases the flow q_{R2} sharply to match the sudden increased demand. It is followed by a gradual step wise increase as a consequence of the storage gradual cooling until the storage is completely discharged at $t = 18\mathrm{h}$. Complete thermal discharge of the storage is when the storage temperature cools down to an equilibrium temperature. After that, the controller will increase purchased power Q_{M} to meet the demand requirements.

The amount of market energy purchased is lower compared to the no storage case because the system will purchase only when the thermal storage is fully discharged. In this case, the system purchases energy only for the remaining 6 hours. Moreover, the rate of rejected heat is highest when the storage is hottest, and is zero when the storage begins to "charge" or has completely "discharged" and is at its equilibrium temperature. The magnitudes of Q_{dump} are lower than that observed in the no storage case because most of extra energy supplied is stored in the tank and reallocated. The period in which the storage dicharges depends on the storage size and the thermal energy demand rate.

To observe the effect of storage size on the system, simulations were performed with different storage sizes. Smaller storage will heat up faster to higher temperatures than larger storage. When the demand is higher than supply and the tank starts cooling, a smaller storage exhibits faster cooling rates than a larger storage. Therefore, the discharge times for the storage depends on the storage size. Even though a larger storage does not heat up as high temperature as the small storage, it has bigger storage material mass and therefore a larger thermal capacity. This implies that less heat is dumped in the low demand phase and the storage can supply energy to the supply at much longer periods before purchasing power from external sources.

The trend of discharge time with storage material volume can be seen using a scatter plot shown in fig. 5.5. The larger storage size the longer the discharge time. However the plot shows that the effect on discharge time is less pronounced when the size is large and for infinite large sizes an asymptote discharge time is reached. This shows that there is a limit for

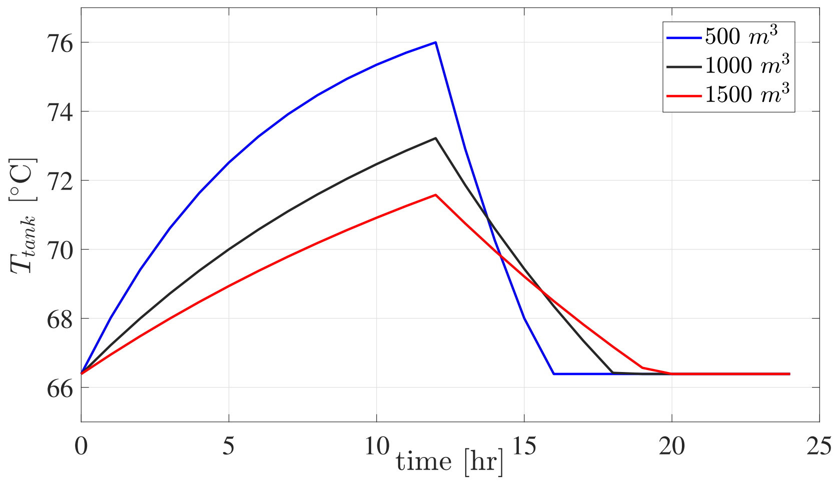

Figure 5.4: Comparison of storage temperatures with different storage sizes showing the effect of storage size

the effect of storage size on thermal discharge time. In this example case, the net thermal charging rate equals net discharge rate at 1000 kW. Due to that we expected to see for an infinitely large storage that the thermal discharge time is at least equal the charging time of 12 h. That is not the case because the quality of thermal energy stored will depend on the temperature at which it is stored. When thermal energy is stored at a lower temperature than it was originally produced, it losses its ability to transfer known as exergy losses. When the design optimisation is performed, it is expected that the tank costs are significant with larger storage volumes. This will lead to a solution for an optimal storage size.

Therefore, these simulation results indicate that it is possible to satisfy peak demand requirements and significantly save purchase costs from external sources by a good choice of thermal storage size and good control structure such as the standard NMPC. A thermal storage controlled by an optimal controller such as a standard economic NMPC will always be cheaper to operate than direct coupling of the supply and demand streams.

After assuming zero direct storage heating in the previous case studies, it is also interesting to see the behaviour of a thermal energy grid with storage and two integrated sources. The additional thermal energy source is intermittent, for example solar power. The following section presents

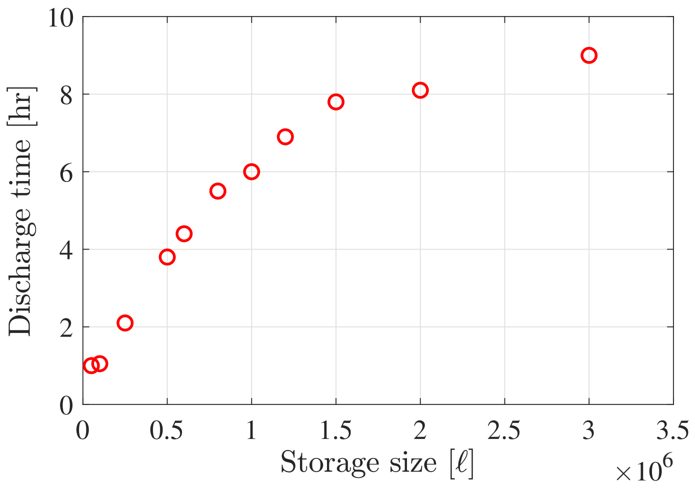

Figure 5.5: Scatter plot of storage discharge time (t_d) after a 12 h charging duration against storage size

such a case study and discussion of the simulation results.

5.4 Standard NMPC on thermal storage with direct solar heating

Consider the scaled California "duck curve" demand data shown in fig. 4.5 with an intermittent direct storage heating supply that comes from solar power. The solar power is only available at night and the availability is variable depending on the hour of the day. It is most available at noon and least available in the morning and afternoon hours of the day.

This case was implemented using a standard NMPC that controls not only q_{R2}, Q_{dump} and Q_M but also Q_{tank}. The Q_{tank} comes from solar heating thus can be denoted as Q_{solar} interchangeably. The standard NMPC implements the predicted hourly solar power levels as upper bounds for Q_{tank}. The input bounds are represented in eq. (5.9). The state bounds used here are equal to those in eq. (5.7). The objective function has an added term from price of direct storage heating ($P_{\text{T}}Q_{\text{tank}}$) in this case (see eq. (5.8)).

$$\phi(\mathbf{x}, \mathbf{u}) = P_{\mathrm{M}} Q_{\mathrm{M}} + P_{\mathrm{T}} Q_{\mathrm{tank}} \tag{5.8a}$$

where

$$\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \leq \begin{bmatrix} q_{R2} \\ Q_{\mathrm{tank}} \\ Q_{\mathrm{M}} \\ Q_{\mathrm{dump}} \end{bmatrix} \leq \begin{bmatrix} 50 \\ Q_{\mathrm{solar},k} \\ +\infty \\ +\infty \end{bmatrix} \tag{5.9}$$

and $Q_{\mathrm{solar},k}$ is the amount of solar heating available at the k^{th} hour of the day.

The simulation was performed using the default implementation described in section 3.2 of this thesis. The results for the simulation are shown in figure 5.6.

The storage remains at a constant temperature for the first 9 hours because the net available supply is less than the demand. Therefore, its starts purchasing energy from the external market at the beginning to cater the demand gap. When the minimum demand hours are approaching the solar power availability increases. The storage temperature starts to rise rapidly to store the relatively cheaper solar power for future high demand ($P_{\mathrm{T}} << P_{\mathrm{M}}$). The controller anticipates a future peak demand that is at 2100 hrs. This is because the MPC recomputes the optimal control problem at every time step with a prediction of the future supply and demand values in the coming 24 hours. The storage charges to the maximum at 1700 hrs before it starts cooling due to increase in energy demand.

The following section introduces discrepancies between the model in the optimiser and the actual plant (simulator). The case study compares the performance of standard NMPC and a multi-stage NMPC whose theory has been discussed in section 2.4.3.

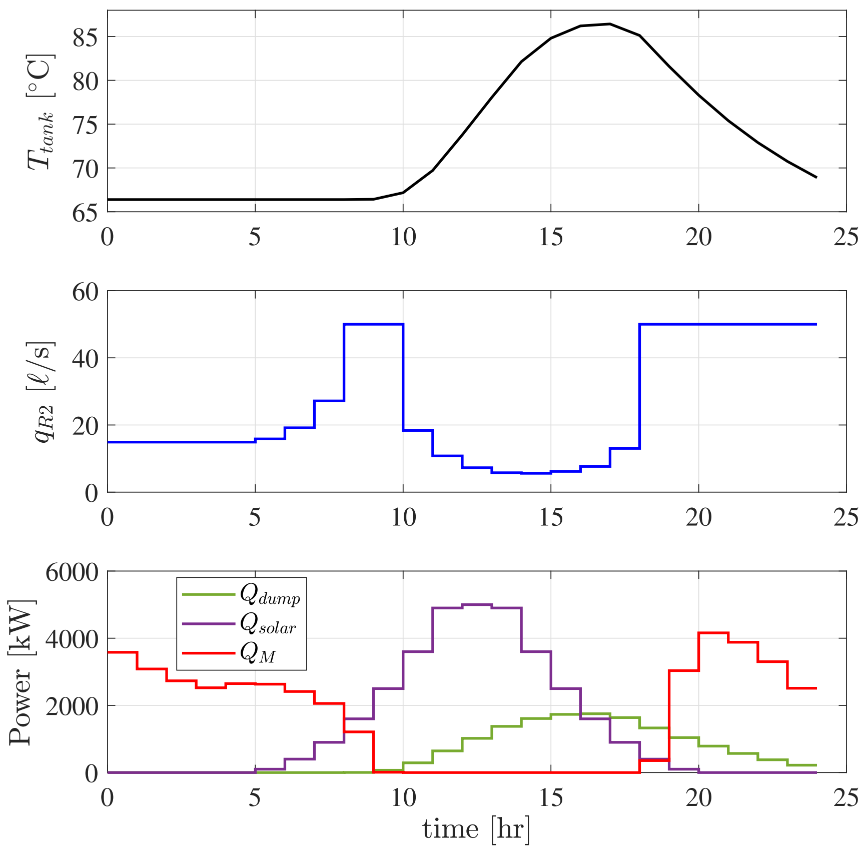

Figure 5.6: Integration of intermittent solar power source with the heat source stream via thermal storage

5.5 Multi-stage NMPC on two-plant thermal storage system

The previous numerical case studies assumed perfect knowledge of the system dynamics. This is not possible in practice. There are always discrepancies between the mathematical model and the actual plant. The plant model mismatch will result to slightly different operation. There are also imperfections in measurements and the supply and demand profiles are always different from the expected values. These unanticipated variations in the parameters are known as uncertainties.

For the energy storage system, there are uncertainties that are associated with:

1. Actual supply and demand values: The NMPC calculates over a prediction horizon into the future, thus requiring a forecast of these values. However, in actual case they are different from expected values.
2. Disturbances in the temperature of supply and demand flow streams (T_1 and T_2), surrounding temperature (T_{surr}), volumetric flow of supply stream (q_{L1}).
3. Approximations in the mathematical model.
4. Changes in design parameters such as overall heat transfer coefficient (U_{hex}) due to fouling etc.

These uncertainties can result to:

1. Failure to exactly match power demand requirements, especially when the expected demand is underestimated. Extra thermal power supply to the demand side when the expected demand is overestimated results to unnecessary losses. However, the underestimated expected demand is a more critical issue.
2. Violation of model equality constraints that can result to poor and unsafe operation, and even process instability.

5.5.1 Implementation of multi-stage NMPC on thermal storage

The multi-stage NMPC was designed to reject huge disturbances in the supply temperature (T_1) and demand temperature (T_2). The NLP optimiser, which is basically our controller is programmed to expect a supply temperature $T_1 = 95°\text{C}$ and a demand stream temperature $T_2 = 20°\text{C}$. However, this is not the case for actual values in the plant. The actual values for T_1 and T_2 are 99°C and 18 °C. Physically this can be interpreted that there is a potential to supply heat at a higher rate than expected, and the potential of the sink plant to extract heat from the storage is much greater than expected. Simulations for a standard NMPC with plant-model mismatch were performed and the results are shown in fig. 5.8. The objective function in this case include a regularisation term for the flow inputs as shown in eq. (5.10). This term reduces wild changes in the manipulated variables and the obtained solution has smoother input transitions. (Biegler, 2010)

$$\phi(\mathbf{x}, \mathbf{u}) = p_{\text{M}} Q_{\text{M}} + P_{\text{u}}(q_{L2}^2 + q_{R2}^2) \tag{5.10a}$$

where, $P_u = 5 \times 10^{-5}$

Moreover, in the same case a multi-stage NMPC was implemented with three levels for both uncertainty variables T_1 and T_2. The assumed uncertainty space for the T_1 and T_2 values was a deviation of 5°C either side of the expected mean temperatures. Therefore the uncertainty space interval considered for the two parameters are listed in eq. (5.11).

$$T_{1,actual} = [90,\ 100] \tag{5.11a}$$

$$T_{2,actual} = [15,\ 25] \tag{5.11b}$$

Figure 5.7 shows the possible uncertainty realisations taken were all the combinations of the extreme levels and the mean values for each parameter. This is a conventional BOX method for scenario selection. The scenario NMPC was implemented with as robust horizon (N_r) of 1.

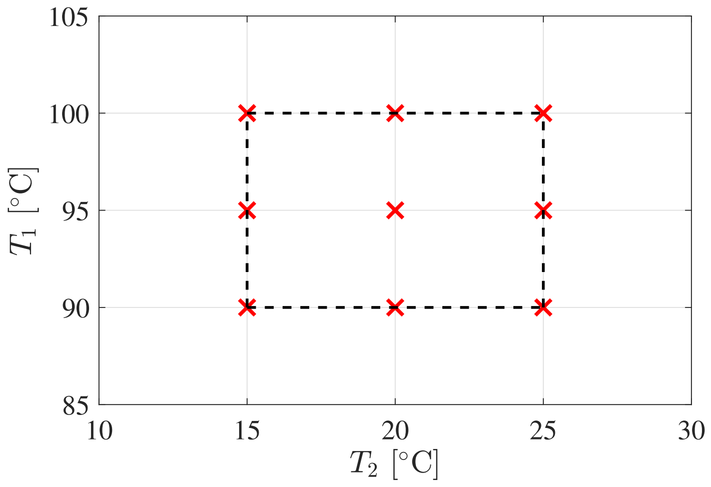

Figure 5.7: The BOX method for selection of uncertainty realisations in the scenario tree

A custom function in MATLAB was created to automatically return the scenarios for two pairs of parameters. The function is called `scenpara()` and can be found in the Appendix. The implementation of the non-anticipativity constraints in the MATLAB code was done by use of a `for` loop to list all the equality constraints for the equal inputs originating from the root node. More details on the MATLAB code implementation of multi-stage NMPC can be seen in the code included in the Appendix.

The figure 5.8 shows results for simulation with plant-model mismatch as explained before. The figure compares both standard NMPC and multi-stage NMPC control on the exact case scenario over one day of operation. It can be seen that economically the multi-stage NMPC is worse than standard NMPC because it purchases more energy from the market. The standard NMPC does not purchase external energy at all times because the actual supply is always higher than expected and therefore the stored heat is capable of satisfying demand in the peak demand phase without need of extra purchase. However, when a constraint is set for tank temperature, the standard NMPC violates it while multi-stage NMPC does not. This shows that multi-stage NMPC is much more conservative but the price must be paid by spending more to avoid

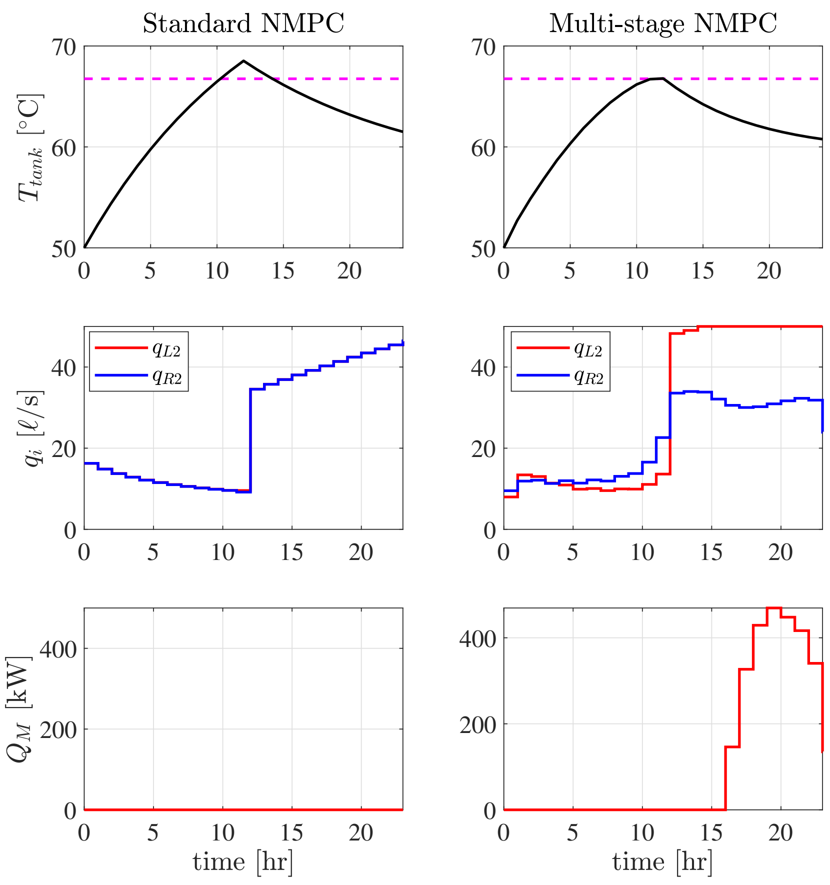

Figure 5.8: Plots for both standard NMPC and multi-stage NMPC with plant-model mismatch $T_{1,actual} = 99°\text{C}$, $T_{2,actual} = 18°\text{C}$, and $T_{tank} \leq 66.7°\text{C}$

the constraint violation. Therefore, due to the critical nature of the constraint violations in plant operation, multi-stage NMPC is better than standard NMPC. This might be for example, high tank temperature should not be allowed to avoid vapour pressure build up in the tank for safer operation. In another plant process, this might be a quality constraint and its violation implies loss of production and product quality. Otherwise, if there is no operational constraint in the system, then standard NMPC definitely shows the best economic performance and should be encouraged.

In addition to that, when there are larger disturbances present, and there are tight constraints imposed, the standard NMPC could easily run into infeasibility issues. As for the case of this project, if the tank temperature bound is chosen much lower than 66.7 $^{\circ}\mathrm{C}$, there is no guarantee that there would be recursive convergence to a feasible solution by the solver. A way around this is to express the bounds as soft constraints and penalise the magnitude of soft constraint violation in the objective function. This constraint formulation has not been implemented in this thesis.

5.5.2 Issues with implementation of multi-stage NMPC

In these two simulations the flows in the heat exchanger 2 were both manipulated. This is because if q_{L2} was fixed as in previous cases and we only allow q_{R2} to change then the solver fails to converge to a solution. The difficulty is due to the supply side from the tank being constrained, the only way to satisfy the demand constraint is by manipulating q_{R2}, but since multi-stage NMPC has non-anticipativity constraints, this flow is fixed for all scenarios with a specific demand and market energy. This defines a required temperature T_{R2} for each scenario which might not be feasible with the storage dynamic constraints. Therefore, the issue with the definition of demand satisfaction constraint has resulted into the implementation of both flows as manipulated inputs. This implies that the solution in the multi-stage NMPC is not a unique solution. Hence, further work can be done in this area to reformulate the demand satisfaction constraint in a better way and avoid these mentioned numerical problems.

Chapter 6

Discussion

Included in this chapter is a general discussion about the results of this work focusing on the objectives set at the beginning of the research period. Some numerical issues that occurred while conducting this work and how they were resolved have been presented. Suggestions on future work and possible improvements of results have been highlighted.

6.1 TES versus no storage

The inclusion of a thermal storage in a thermal supply grid is cost saving compared to direct thermal supply. When the system is controlled by an economic MPC, it ensures minimum purchase of energy as possible from external sources. Inclusion of thermal storage gives the energy supply grid capacity of storing as much thermal energy as possible when supply is greater than demand and utilise the stored energy in peak demand periods. The quality of thermal energy stored for an non-stratified storage is lower because it will always be stored at a lower temperature than it was previously available. This lack of temperature stratification lowers the storage efficiency but the effect of storage in the energy system is still evident.

6.2 Operation of TES using NMPC

Thermal energy storage is advantageous for energy supply and demand management when operated by an economic optimal control policy using a standard NMPC. They exhibit cheaper operation and energy savings when there is a characteristic supply and demand mismatch in daily operation. The standard NMPC with an economic objective function decides on current inputs that will lead to profitable operation. The optimal operation is storing as much as possible and releases the stored heat efficiently when the thermal power demand is higher than thermal power supplied. The storage size affect the duration of thermal discharge, storage temperature and amount of thermal energy that can be stored.

When another supply source is available, it can be integrated into the system via a thermal storage flexibly. An intermittent source such as a variable solar heating supply which is cheaper than external heating is prioritised by the standard NMPC control when available to heat up the storage and prepare for peak demand periods.

Uncertainty in actual plant operation is better handled by the robust multi-stage NMPC for thermal energy storage compared to standard NMPC at the expense of higher operating costs. This is a merit because operational constraint satisfaction in chemical processes is paramount. Occurrence of constraint violations can be translated to economic losses due to unsafe operation and substandard product quality. Therefore, profitable operation is not the main concern, and instead constraint violations that correspond to safe operation and product quality standards must always be prioritised. In this case of thermal storage, there could be storage temperature bound to avoid vapour pressure build-up in the tank. Multi-stage NMPC will ensure operation without violating the constraint, but standard NMPC results to constraint violation and may fail to converge to a feasible solution. This makes multi-stage NMPC a better controller than standard NMPC under uncertainty in supply and demand.

6.3 Numerical Issues

The numerical case studies that have been done in this thesis brought forth some numerical problems which were tackled using some mathematical tricks. These numerical issues include:

- Singularity: To avoid singularity and computation of non-real jacobians by CasADi, it is important for the NLP to be formulated with additional algebraic states. These states were introduced such that polynomial terms including differential states with fractional exponents were substituted from the model. This formulation also ensures the convergence from any feasible starting points.
- Computational speed in NMPC: To improve computational speed, initial guesses provided to IPOPT must be near to the optimal solution. To ensure fast convergence, the solution from the previous open loop problem is stored and used as an initial guess for the next NMPC reoptimisation step. Apart from fast convergence, this also improved the chances converging to a feasible point.
- Feasible demand side temperatures (T_{R2}): To avoid failure of convergence to a feasible solution, the numerical case study must be formulated wisely. The demand side heat exchanger will always transmit thermal energy and the rate of thermal power transferred is dependent on the ratio of heat capacities of the cold and hot side. The return temperature of the cold side stream (T_{R2}) has a limit and that may cause an enthalpy gain larger than the expected demand. Therefore, for an equality demand satisfaction constraint, the expected demand (Q_{demand}) must always be larger than this limit in enthalpy gain.

6.4 Unresolved Issues

The implementation of multi-stage NMPC with control of only one flow in the demand side remains unresolved. The issue might be because of how the demand satisfaction constraint function is expressed. When the value of return stream temperature T_{R2} determined by the dynamics of the thermal storage system is high enough to make its change in enthalpy higher than Q_{demand}, then the equality constraint can never be satisfied since Q_M is non-negative. A suggestion to solve this problem could be by using an inequality constraint with a slack variable instead of an equality constraint for the demand satisfaction constraint.

The slack variable will physically represent dumped heat since the demand stream has already satisfied the demand requirements, but the storage dynamics make it impossible to match it exactly. The formulation of the inequality constraint could be written as equation 6.1.

$$Q_{\text{M},k} + \rho c_p q_{R2,k}(T_{R2,k} - T_2) - Q_{\text{slack},k} \geq Q_{\text{demand}}(t_k) \tag{6.1}$$

where Q_{slack} should be always positive, and the value may be penalised in the objective function.

It was also found impossible for IPOPT to converge to an optimal solution when considering uncertainties in Q_{demand} and Q_{supply} instead of the temperatures. This is because the plant model does not include the demand and supply rate as parameters in the model. Therefore, a better formulation of the plant model is to required to include the aforementioned parameters.

6.5 Further work

More improvements are possible in the multi-stage NMPC implementation. These improvements include:

- Reformulation of the model to explicitly include thermal power supply and demand in the system dynamics, both in the optimiser and the simulator. This reformulation should also be included in the constraints.
- Implementation of soft constraints and slack variable tricks seems extremely important in order to tackle infeasibility problems in both standard NMPC and multi-stage NMPC cases.
- It could also be interesting to select other uncertain parameters such as overall heat transfer coefficient to design the robust NMPC.

In general, future work on optimal control of thermal energy systems could be on determination of algorithms to predict thermal power supply and demand with little uncertainty. It might be a far-fetched idea to completely eliminate the effect of uncertainty on optimal control but by application of learning algorithms on historical data better forecasts can be obtained and improve the control of a smart thermal energy grid.

Chapter 7

Conclusion

The drawn conclusions regarding the thesis objectives are presented together in this chapter after the general discussions.

The mathematical model of a simple two-plant thermal energy storage system from previous work has been successfully reformulated from an ODE to a set of DAEs which is now suitable for numerical computations. This formulation generates non-singular jacobians for values of decision variables that are close to zero and hence a more robust NMPC code. The reformulation increases the dimension of the optimisation problem but it is important anyway to ensure convergence.

It has been found that thermal storage is important in thermal supply grids and when controlled by an economic standard NMPC, it operates at minimal costs. The direct thermal supply without thermal storage results to dumping of excess supplied heat at periods of abundant thermal supply. Moreover, the system without storage purchases the exact amount of extra demanded heat from the market at peak demand periods.

Standard NMPC on a thermal storage with an economic objective and assuming perfect prediction behaves correctly. With a varying supply and demand, the controller manipulates the system to heat up storage when the supply is highest and to cool it down to release the stored thermal energy when the demand is highest. The use of an additional cheaper but intermittent solar supply has been showcased and the controller works perfectly by preferring to store the cheaper source when it is available. It prepares the system to use the stored heat at peak periods instead of using emergency sources.

When there is a plant-model mismatch due to supply and demand uncertainty, the implementation of multi-stage NMPC shows better performance than standard NMPC on the thermal storage. Multi-stage NMPC incurs more costs than standard NMPC but it obeys operation constraints. Constraint satisfaction is argued as a more important criteria that cost savings. This is because processes have to operate within their limits to ensure safety and product quality. Otherwise, constraint violations happening in standard NMPC control causes huge losses and also results to infeasibilities when constraints are not softened.

Bibliography

Alva, G., Lin, Y., Fang, G., 2018. An overview of thermal energy storage systems. Energy 144, 341 – 378.
URL http://www.sciencedirect.com/science/article/pii/S036054421732056X

Andersson, J., October 2013. A General-Purpose Software Framework for Dynamic Optimization. PhD thesis, Arenberg Doctoral School, KU Leuven, Department of Electrical Engineering (ESAT/SCD) and Optimization in Engineering Center, Kasteelpark Arenberg 10, 3001-Heverlee, Belgium.

Bellman, R., 1957. Dynamic programming. Princeton University Press, Princeton, N. J.

Bergman, T. L., Incropera, F. P., DeWitt, D. P., Lavine, A. S., 2011. Fundamentals of heat and mass transfer. John Wiley & Sons.

Biegler, L. T., 2010. Nonlinear programming: concepts, algorithms, and applications to chemical processes. Vol. 10. Siam.

Birge, J. R., 1997. State-of-the-art-surveystochastic programming: Computation and applications. INFORMS journal on computing 9 (2), 111–133.

Bock, H. G., Plitt, K.-J., 1984. A multiple shooting algorithm for direct solution of optimal control problems. IFAC Proceedings Volumes 17 (2), 1603–1608.

Burnett, M., 2016. Energy storage and the california ”duck curve”. http://large.stanford.edu/courses/2015/ph240/burnett2/.

Cabeza, L., Martorell, I., Mir, L., Fernndez, A., Barreneche, C., 2015. 1 - introduction to thermal energy storage (tes) systems. In: Cabeza, L. F. (Ed.), Advances in Thermal Energy Storage Systems. Woodhead Publishing Series in Energy. Woodhead Publishing, pp. 1 – 28. URL http://www.sciencedirect.com/science/article/pii/B9781782420880500018

Chen, J., 1987. Comments on improvements on a replacement for the logarithmic mean. Chemical Engineering Science 42 (10), 2488–2489.

Cutler, C. R., Ramaker, B. L., 1980. Dynamic matrix control?? a computer control algorithm. In: joint automatic control conference. No. 17. p. 72.

Diehl, M., Gross, S., 2017. Numerical optimal control. https://www.syscop.de/files/2017ss/NOC/script/book-NOCSE.pdf.

Dincer, I., Rosen, M., 2002. Thermal energy storage: systems and applications. John Wiley & Sons.

Foss, B., Heirung, T. A. N., 2013. Merging optimization and control. Lecture Notes.

Furbo, S., 2015. Using water for heat storage in thermal energy storage (tes) systems. In: Advances in Thermal Energy Storage Systems. Elsevier, pp. 31–47.

Gede, G., 2011. The direct collocation method for optimal control.

Grimm, G., Messina, M. J., Tuna, S. E., Teel, A. R., 2004. Examples when nonlinear model predictive control is nonrobust. Automatica 40 (10), 1729–1738.

Gross, S., 2016. Numerical optimal control lecture 6: Direct collocation. NTNU PhD Course.

Hargraves, C. R., Paris, S. W., 1987. Direct trajectory optimization using nonlinear programming and collocation. Journal of Guidance, Control, and Dynamics 10 (4), 338–342.

Kalaiselvam, S., Parameshwaran, R., 2014a. Chapter 1 - energy and energy management. In: Kalaiselvam, S., Parameshwaran, R. (Eds.), Thermal Energy Storage Technologies for Sustainability. Academic Press, Boston, pp. 1 – 19.
URL http://www.sciencedirect.com/science/article/pii/B9780124172913000013

Kalaiselvam, S., Parameshwaran, R., 2014b. Chapter 10 - thermal energy storage systems design. In: Kalaiselvam, S., Parameshwaran, R. (Eds.), Thermal Energy Storage Technologies for Sustainability. Academic Press, Boston, pp. 237 – 245.
URL http://www.sciencedirect.com/science/article/pii/B9780124172913000104

Kalaiselvam, S., Parameshwaran, R., 2014c. Chapter 2 - energy storage. In: Kalaiselvam, S., Parameshwaran, R. (Eds.), Thermal Energy Storage Technologies for Sustainability. Academic Press, Boston, pp. 21 – 56.
URL http://www.sciencedirect.com/science/article/pii/B9780124172913000025

Kalaiselvam, S., Parameshwaran, R., 2014d. Chapter 4 - sensible thermal energy storage. In: Kalaiselvam, S., Parameshwaran, R. (Eds.), Thermal Energy Storage Technologies for Sustainability. Academic Press, Boston, pp. 65 – 81.
URL http://www.sciencedirect.com/science/article/pii/B9780124172913000049

Lee, J. H., Yu, Z., 1997. Worst-case formulations of model predictive control for systems with bounded parameters. Automatica 33 (5), 763–781.

Li, P.-W., Chan, C. L., 2017. Chapter 1 - introduction. In: Li, P.-W., Chan, C. L. (Eds.), Thermal Energy Storage Analyses and Designs. Academic Press, pp. 1 – 6.
URL http://www.sciencedirect.com/science/article/pii/B9780128053447000018

Li, P. W., Van Lew, J., Karaki, W., Chan, C. L., Stephens, J., OBrien, J. E., 2011. Transient heat transfer and energy transport in packed bed thermal storage systems. In: Developments in heat transfer. IntechOpen.

Lienhard IV, J., Lienhard V, J., 2018. A Heat Transfer Textbook, 4th Edition. Phlogiston Press, Cambridge, MA, version 2.12.
URL http://ahtt.mit.edu

Lucia, S., Finkler, T., Engell, S., 2013. Multi-stage nonlinear model predictive control applied to a semi-batch polymerization reactor under uncertainty. Journal of Process Control 23 (9), 1306–1319.

MATLAB, 2018. version 9.5.0.944444 (R2018b). The MathWorks Inc., Natick, Massachusetts.

Mayne, D. Q., 2001. Control of constrained dynamic systems. European Journal of Control 7 (2-3), 87–99.

Mayne, D. Q., Rawlings, J. B., Rao, C. V., Scokaert, P. O., 2000. Constrained model predictive control: Stability and optimality. Automatica 36 (6), 789–814.

Mdoe, Z. N., December 2018. Dynamic optimisation and control of thermal energy storage systems.

Nocedal, J., Wright, S. J., 2006. Numerical optimization 2nd.

NordPool, 2018. Historical market data. https://www.nordpoolgroup.com/historical-market-data/.

Orosz, M., Dickes, R., 2017. 16 - solar thermal powered organic rankine cycles. In: Macchi, E., Astolfi, M. (Eds.), Organic Rankine Cycle (ORC) Power Systems. Woodhead Publishing, pp. 569 – 612.
URL http://www.sciencedirect.com/science/article/pii/B9780081005101000168

Paterson, W., 1984. A replacement for the logarithmic mean. Chemical Engineering Science 39 (11), 1635–1636.

Richalet, J., Rault, A., Testud, J., Papon, J., 1978. Model predictive heuristic control. Automatica (Journal of IFAC) 14 (5), 413–428.

Sargent, R., Sullivan, G., 1978. The development of an efficient optimal control package. In: Optimization Techniques. Springer, pp. 158–168.

Scokaert, P. O., Mayne, D., 1998. Min-max feedback model predictive control for constrained linear systems. IEEE Transactions on Automatic control 43 (8), 1136–1142.

Shapiro, A., Dentcheva, D., Ruszczyński, A., 2009. Lectures on stochastic programming: modeling and theory. SIAM.

Soyster, A. L., 1973. Convex programming with set-inclusive constraints and applications to inexact linear programming. Operations research 21 (5), 1154–1157.

Underwood, A. J. V., 1933. Ind. Chemist., 167–170.

Underwood, A. J. V., 1970. Chemical Engineering, 192.

Wächter, A., Biegler, L. T., 2006. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. Mathematical programming 106 (1), 25–57.

Zavala-Río, A., Femat, R., Santiesteban-Cos, R., 2005. An analytical study of the logarithmic mean temperature difference. Revista Mexicana de Ingeniería Química 4 (3).

Appendix

System Models

This section presents the set of mathematical model equations used in the optimal control problems for numerical case study.

DAE model for two plant system

The system described in has a mathematical model in semi-implicit DAE form shown in eq. (7.1l).

$$\frac{\mathrm{d}T_{L1}}{\mathrm{d}t} = \frac{1}{V_{\text{hex}}}\left\{q_{L1}(T_1 - T_{L1}) - \frac{U_{\text{hex}}A_{\text{hex}}}{\rho c_p}\Delta T_{m,1}\right\} \tag{7.1a}$$

$$\frac{\mathrm{d}T_{R1}}{\mathrm{d}t} = \frac{1}{V_{\text{hex}}}\left\{q_{R1}(T_{\text{tank}} - T_{R1}) + \frac{U_{\text{hex}}A_{\text{hex}}}{\rho c_p}\Delta T_{m,1}\right\} \tag{7.1b}$$

$$\frac{\mathrm{d}T_{L2}}{\mathrm{d}t} = \frac{1}{V_{\text{hex}}}\left\{q_{L2}(T_{\text{tank}} - T_{L2}) - \frac{U_{\text{hex}}A_{\text{hex}}}{\rho c_p}\Delta T_{m,2}\right\} \tag{7.1c}$$

$$\frac{\mathrm{d}T_{R2}}{\mathrm{d}t} = \frac{1}{V_{\text{hex}}}\left\{q_{R2}(T_2 - T_{R2}) + \frac{U_{\text{hex}}A_{\text{hex}}}{\rho c_p}\Delta T_{m,2}\right\} \tag{7.1d}$$

$$\frac{\mathrm{d}T_{\text{tank}}}{\mathrm{d}t} = \frac{1}{V_{\text{tank}}}\left\{q_{R1}(T_{R1} - T_{\text{tank}}) + q_{L2}(T_{L2} - T_{\text{tank}}) + \frac{1}{\rho c_p}(Q - Q_{\text{loss}})\right\} \tag{7.1e}$$

$$0 = a^{\frac{1}{n}} - (T_1 - T_{R1}) \tag{7.1f}$$

$$0 = b^{\frac{1}{n}} - (T_{L1} - T_{\text{tank}}) \tag{7.1g}$$

$$0 = c^{\frac{1}{n}} - (T_{L2} - T_2) \tag{7.1h}$$

$$0 = d^{\frac{1}{n}} - (T_{\text{tank}} - T_{R2}) \tag{7.1i}$$

where,

$$Q_{\text{loss}} = (UA)_{\text{tank}}(T_{\text{tank}} - T_{\text{surr}}) \tag{7.1j}$$

$$\Delta T_{m,1}^{n} = 0.5(a+b), \quad n = \frac{1}{3} \tag{7.1k}$$

$$\Delta T_{m,2}^{n} = 0.5(c+d), \quad n = \frac{1}{3} \tag{7.1l}$$

Source codes

Here the MATLAB source codes used for different numerical cases in this thesis are presented.

Direct supply with no storage dynamics

ODE function

```
function dxdt = twoPlantModelDirect(~,x,p)
% Model for direct heat exchange system between two plants
% a heat supplier and consumer external heating.
%=========================================================================
% Author: Zawadi Mdoe
% Date: May 2019
%=========================================================================
%% Description of the states:
T_L1 = x(1);
T_R1 = x(2);

%========================================================================
%% Assignment of inputs and disturbances
% Input variables u's
q_L1 = p(1);
q_R1 = p(2);

% Distubances d's
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
```

Two plant dynamics

ODE function

```
function dxdt = twoPlantModelChen(~,x,p)
% Model for an energy storage system with two plants
% a heat supplier and consumer, a storage tank and external heating.
%--------------------------------------------------------------------
% Description of the states:
%--------------------------------------------------------------------
T_L1 = x(1);
T_R1 = x(2);
T_L2 = x(3);
T_R2 = x(4);
T_tank = x(5);
%--------------------------------------------------------------------
% Reassignment of inputs and disturbances
% Manipulated or fixed input variables u's
q_L1 = p(1);
q_R1 = p(2);
q_L2 = p(3);
q_R2 = p(4);
Q_tank = p(5);

%% Distubances
T1 = p(6);
T2 = p(7);

%% Design and physical parameters
```

```
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 =  [(1/V_hex)*(q_L1*(T1-T_L1) - (U_hex*A_hex/(rho*cp))*...
        (0.5*((abs(T1-T_R1))^n + (abs(T_L1-T_tank))^n))^(1/n));

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

         (1/V_hex)*(q_L2*(T_tank-T_L2) - (U_hex*A_hex/(rho*cp))*...
         (0.5*((abs(T_L2-T2))^n + (abs(T_tank-T_R2))^n))^(1/n));

         (1/V_hex)*(q_R2*(T2-T_R2) + (U_hex*A_hex/(rho*cp))*...
         (0.5*((abs(T_L2-T2))^n + (abs(T_tank-T_R2))^n))^(1/n));

         (1/V_tank)*(q_R1*(T_R1-T_tank) + q_L2*(T_L2-T_tank) ...
        + (Q_tank-h_s*A_tank*(T_tank-T_s))/(rho*cp))];

end
```

Standard NMPC code

No Storage

```
% An implementation of direct collocation to open loop
% dynamic optimisation of a two plant
% direct supply without storage using CasADi
%% VARIABLE DEMAND!!!

% Zawadi Mdoe, 2019
% ==========================================================================
clear;
clc;
close all;
```

```
addpath('C:\Users\DELL\Desktop\Matlab\casadi-windows-matlabR2016a-v3.4.5')
import casadi.*

run Parameters_NoStorage.m

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);

run SSOptNS.m

% Declare model variables
x1 = SX.sym('x1');
x2 = SX.sym('x2');
x3 = SX.sym('x3');
x4 = SX.sym('x4');
x = [x1; x2];
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;                  %no tank heating

% 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,:)+ Pu*u_opt(3,:);
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('Temperature [$\circ$C]','Interpreter','latex')
legend('$T_{1,o}$','$T_{2,o}$','Interpreter','latex')
hold off
grid on

subplot(312)
stairs(tgrid, [u_opt(1,:), nan], '-b')
hold off
% xlabel('time [hr]')
ylabel('$q_{2}$ [$\ell$/s]', 'Interpreter','latex')
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')
ylim([0 1200])
grid on
```

With Storage

```
% An implementation of direct collocation to open loop
% dynamic optimisation of a two plant
% Energy Storage System using CasADi
%% VARIABLE DEMAND!!!

% Zawadi Mdoe, 2019
% ===============================================================================
clear;
clc;
close all;

addpath('C:\Users\DELL\Desktop\Matlab\casadi-windows-matlabR2016a-v3.4.5')
import casadi.*

run Parameters_TES.m


Q_demand = [1500*ones(12,1); 3500*ones(12,1)];
Q_supply = 2500*ones(24,1);

% Q_demand = CalDemand;
% 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 = 5;                    %number of differential states
nalg = 4;                    %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);

run SSOpt.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');
x6 = SX.sym('x6');
x7 = SX.sym('x7');
x8 = SX.sym('x8');
x9 = SX.sym('x9');
x = [x1; x2; x3; x4; x5];
z = [x6; x7; x8; x9];
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*(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;

% 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;
% Q_demand = zeros(N+M,1);
% 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(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 = [];

        % "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];
            % 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; %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

    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

Cost = Pm*u_opt(2,:) + Pu*u_opt(3,:);
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

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

subplot(313)
```

```
stairs(tgrid, [u_opt(3,:), nan], 'm-')
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')
grid on
```

Multi-stage NMPC

Supply and demand temperature uncertainty

```
% An implementation of direct collocation to open loop
% dynamic optimisation of a two plant energy Storage System using CasADi
% VARIABLE DEMAND!!! with plant model mismatch an implementation of
% multi-stage MPC to handle uncertainties.

% Author: Zawadi Mdoe
% Date: March 2019
% ==========================================================================

clear;
clc;
close all;

addpath('C:\Users\DELL\Desktop\Matlab\casadi-windows-matlabR2016a-v3.4.5')
import casadi.*
run Parameters_scendae3.m
load CaliforniaDemand3.mat

Q_demand = [1500*ones(12,1); 3500*ones(12,1)];

T1_actual = 99;
T2_actual = 18;

% Uncertain parameter(s)
par1 = [90; 95; 100];    %Source temperature
par2 = [15; 20; 25];     %Sink temperature

[scens, scen_count] = scenpara(par1,par2);

% 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);

% 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');
u2 = SX.sym('u2');
u3 = SX.sym('u3');
u4 = SX.sym('u4');
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
M = 24;          %mpc loops
h = T/N;
Nr = 1;
levels = scen_count;
S = levels^Nr;  % number of scenarios
period = N;
% Q_demand = [CalDemand; CalDemand];
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
        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];
                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);

    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
    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];
```

Scenario generation (scenpara())

```
function [scens, scen_count] = scenpara(a1,a2)
%scenpara: creates combinations for uncertain
%parameter levels in scenarioMPC
```

```
%
% Author: Zawadi Mdoe 2019
%===========================================================
    scen_count = length(a1)*length(a2);

    scens = {};

    for i=1:length(a1)
        for j=1:length(a2)
            scens = [scens(:); {[a1(i), a2(j)]}];
        end
    end
end
```