

FMH606 Master's Thesis 2022

Master of Science, Industrial IT and Automation

Stochastic MPC for Optimal Operation of Hydropower Plant

Nabin K C

239255

Course: FMH606 Master's Thesis, 2022

Title: Stochastic MPC for Optimal Operation of Hydropower Plant

Number of pages: 83

Keywords: MPC, stochastic MPC, prediction horizon, weighted sum approach, MOO, non-linear MPC

Student: Nabin K C

Supervisor: Roshan Sharma

External partner: Skagerak Energy AS

Summary:

Different categories of industries, such as manufacturing plants, power production plants etcetera, are equipped with a control mechanism to keep their process-variables of interest within the desired limit. Among different prevalent advanced control systems, Model Predictive Controller (MPC) has gained significant interest because of its ability to act based on the uncertainties that may occur in the future.

Skagerak Energi has predicted ensemble of inflow uncertainties that may occur in Lake Toke. The spreading of these ensembles increases as the prediction days march forward. The main goal of this thesis is to design a stochastic MPC (SMPC) that can handle these inflow uncertainties, while maintaining the optimal operation of the hydropower plant by maximizing the reservoir's water level within the limits.

This thesis starts with an understanding of possible methods for SMPC formulation, such as multistage nonlinear MPC (NMPC), multi-objective optimization (MOO), and Min-Max MPC. Such stochastic formulations are computationally demanding, therefore efficient numerical methods for solving these formulations are often sought out. In this thesis, the direct multiple shooting method is explored as a numerical scheme. The thesis proceeds further with comparison of two different types of objective functions based on the results from deterministic MPC. The objective function which meets the requirement of optimal operation of the hydropower plant is then selected to design the SMPC.

SMPC based on the weighted sum MOO is designed. When full spreading of the inflow ensembles was used, the results obtained for the gate opening signals are changing irregularly along with the level constraint violation. However, the results were improved when the inflow ensembles spreading was limited to 100 m³/s. From the author's experience with the problem, two important factors are outlined for improving the results. First, the spreading of the inflow ensembles should be handled for the outliers, and second, a suitable choice of weighting parameters should be taken into consideration.

Preface

This thesis has been performed for the fulfillment of the master's degree program in Industrial IT and Automation at the University of South-Eastern Norway (USN), Porsgrunn. The purpose of this work is to develop a stochastic MPC (Model Predictive Control) that facilitates the optimal operation of the hydropower plant by maximizing the water levels and handling the uncertainties that may occur in Lake Toke with optimal flood gate opening.

Over the past few months, the support and guidance of my supervisor, Associate Professor Roshan Sharma, and co-supervisor Changhun Jeong played a crucial role to carry out this thesis in the proper direction. Their invaluable support through hours-long meaningful discussion, techniques to tackle the hurdles, and motivation was the key asset that drive me to learn and perform for this thesis. Therefore, I would like to express my special gratitude to them.

Furthermore, I would like to thank Skagerak Energy and USN for the allocation of this topic as one of the master's thesis. It was a great learning experience in the field of MPC and I am sure that the ideas I gained will help me in the future as well.

Finally, I would like to acknowledge the support of my family, friends, and those who directly or indirectly provided me the right thrust over these periods.

Porsgrunn, May 2022

Nabin K C

Contents

Preface	i
Contents	ii
List of Figures	iv
List of Tables	vii
Nomenclature	viii
1 Introduction	1
1.1 Background.....	1
1.2 Objectives.....	1
1.3 Previous Work.....	1
1.4 Requirements.....	2
1.5 Thesis structure.....	2
2 System Overview	3
2.1 Lake Toke and operational regulations	3
2.2 Process Model	5
2.3 Time period selection for simulation.....	7
2.4 Open-loop simulation.....	8
2.5 Inflow data visualization	8
3 Theory	11
3.1 Some methods for SMPC formulation	11
3.1.1 Multi-stage scenario-based NMPC.....	11
3.1.2 MPC based on multi-objective optimization.....	13
3.1.3 Min-Max MPC	14
3.2 Numerical Methods for the optimal control problem.....	15
3.3 CasADi – A framework for algorithmic differentiation and numerical optimization	17
4 Deterministic MPC	18
4.1 Type I OCP formulation based on reference region tracking.....	19
4.2 Type II OCP formulation based on levels maximization	26
4.3 Comparison of best results	31
4.3.1 Results comparison based on Case 1.....	31
4.3.2 Results comparison based on Case 2.....	32
4.4 Discussion on deterministic MPC	33
4.4.1 Discussion on weighting parameters.....	33
4.4.2 Discussion on OCP formulation.....	33
5 Stochastic MPC	34
5.1 WSA for SMPC	34
5.2 Performance testing of SMPC at low turbine intake.....	36
5.3 Performance testing of SMPC at maximum turbine intake.....	42
5.4 Discussion on stochastic MPC	45
5.4.1 Trajectories of inflow ensembles.....	45
5.4.2 Weight adjustment.....	45

5.4.3 Computational load	46
6 Conclusion	47
6.1 Conclusion.....	47
6.2 Future Work.....	48
References	49
Appendices	51

List of Figures

Figure 2-1: Map Showing Lake Toke, Merkebekk and Dalfoss point [3].....	3
Figure 2-2: Constraints on water levels throughout the year [3]	4
Figure 2-3: Functional block diagram of lake Toke [3].....	5
Figure 2-4: Schematic of flood gates [3]	6
Figure 2-5: Schematic of Lake Toke [3].....	6
Figure 2-6: Indication of simulation time period.....	8
Figure 2-7: Open-loop simulation result: with higher inflow in the reservoir, levels increase faster, and therefore more gate opening is required to meet constraints.....	9
Figure 2-8: Ensemble forecast recorded on April 22, 2020.....	9
Figure 2-9: Each day inflow variations starting from April 15, 2020 - May 15, 2020.....	10
Figure 2-10: Each day inflow variations starting from April 15, 2021 - May 15, 2021.....	10
Figure 3-1: Stochastic Programming in multistage [1].....	11
Figure 3-2: Example of scenario diagram for multistage NMPC [8]	12
Figure 3-3: Methods to solve MOO problems [12]	14
Figure 3-4: Numerical methods for dynamic optimization problem [20].....	15
Figure 3-5: Illustration of single shooting: discretization of control inputs only [25].....	16
Figure 4-1: Operational inflow selection from the 50 different possible inflows each day.....	18
Figure 4-2: Type I OCP, initially water at the lower region of the reservoir - [57.75;57.70], turbine operating in full capacity- $V_t = 36 \text{ m}^3/\text{s}$, flood coefficient = 1, Group 1 tuning parameters.....	21
Figure 4-3: Type I OCP, initially water level lower region of the reservoir- [57.75;57.70], turbine operating in full capacity – $V_t = 36 \text{ m}^3/\text{s}$, flood coefficient = 3, results comparison using all groups of parameters	22
Figure 4-4: Type I OCP, initially water at the lower region of the reservoir - [57.75;57.70], turbine operating in full capacity- $V_t = 36 \text{ m}^3/\text{s}$, flood coefficient = 3, Group 1 tuning parameters.....	23
Figure 4-5: Type I, Case I OCP timing diagram; the results obtained within average loop run time of 0.2s with Group 1 tuning parameters.	23
Figure 4-6: Type I OCP, initially water level at the higher region of the reservoir- [59.55;59.50], turbine operating in full capacity – $V_t = 36 \text{ m}^3/\text{s}$, flood coefficient = 3, results from comparison using all groups of parameters.....	24
Figure 4-7: Type I OCP, initially water at the lower region of the reservoir - [59.55;59.50], turbine operating in full capacity- $V_t = 36 \text{ m}^3/\text{s}$, flood coefficient = 3, Group 3 tuning parameters.....	25
Figure 4-8: Type I OCP, Case II timing diagram; the results obtained within an average loop run time of 0.13s with Group 3 tuning parameters.	25

Figure 4-9: Type II OCP, initially water at level lower region of the reservoir- [57.75;57.70], turbine operating in full capacity – $V_t = 36 \text{ m}^3/\text{s}$, flood coefficient = 3, results comparison using all groups of parameters	27
Figure 4-10: Type II OCP, initially water at the lower region of the reservoir - [57.75;57.70], turbine operating in full capacity- $V_t = 36 \text{ m}^3/\text{s}$, flood coefficient = 3, Group 3 tuning parameters	28
Figure 4-11: Type II OCP, Case I timing diagram; the results obtained within average loop run time of 0.20s with Group 3 tuning parameters.	28
Figure 4-12:Type II OCP, initially water level at higher region of the reservoir- [59.55;59.50], turbine operating in full capacity – $V_t = 36 \text{ m}^3/\text{s}$, flood coefficient = 3, results comparison using all groups of parameters	29
Figure 4-13:Type II OCP, initially water at the upper region of the reservoir - [59.55;59.50], turbine operating in full capacity- $V_t = 36 \text{ m}^3/\text{s}$, flood coefficient = 3, Group 3 tuning parameters	30
Figure 4-14: Type II OCP, Case II timing diagram; the results obtained within average loop run time of 4.18s with Group 3 tuning parameters.	30
Figure 4-15: Type I vs Type II OCP formulation, comparison of level change and control signal using best performing weighting factor, initial level in the lower region	31
Figure 4-16:Type I vs Type II OCP formulation, comparison of level change and control signal using best performing weighting factor, initial level in the upper region	32
Figure 5-1: A framework showing the connection between the MPC and process.....	34
Figure 5-2: Flowchart showing implementation of SMPC using WSA method	35
Figure 5-3: Levels and gate openings change using 10 same ensembles in MOO formulation	36
Figure 5-4: Changes in levels with 10 ensembles data taken from year 2020, level constraints violation around 400 hours	37
Figure 5-5: Changes in one control signal with 10 ensembles data taken from year 2020	37
Figure 5-6: Example of chopped data to limit the maximum inflow to $100 \text{ m}^3/\text{s}$,	38
Figure 5-7: Weightings are set differently as time marches forward in the prediction horizon	38
Figure 5-8: Level variations due to 50 ensembles with maximum inflow limit to $100 \text{ m}^3/\text{s}$, ..	39
Figure 5-9: A close look on levels variation from 340 hours to 470 hours	39
Figure 5-10: Changes in gate opening signal with 50 ensembles, data taken from the year 2021,.....	40
Figure 5-11: Changes in gate opening signal with 50 ensembles, Figure 5-10 view	40
Figure 5-12: Average loop run time considering 50 ensembles and low turbine inflow.....	41
Figure 5-13: Level variations and gate openings change with prediction horizon reduced to 6 days, but ensembles without limiting the inflow	42
Figure 5-14: Level variations, flood coefficient 1, maximum inflow from year 2021 limited to $100 \text{ m}^3/\text{s}$	43

Figure 5-15: Gate openings variations, maximum inflow from the year 2021 limited to 100 m³/s43

Figure 5-16: Gate opening variations, Figure 5-15 viewed from 350 hours to 650 hours.44

Figure 5-17: Average loop run time considering 50 ensembles and high turbine inflow.....44

Figure 5-18: Ensembles numbered 1,2 and 3 are different than majority of the ensembles, data recorded on 20 April 202045

List of Tables

Table 2-1: Constraints on water level throughout the year[3]	4
Table 2-2: Parameters used in Lake Toke model equations [3].....	7
Table 4-1: Parameters used in Type I OCP formulation.....	20
Table 4-2: Groups of tuning parameters to analyze the performance of the controller	20

Nomenclature

Here is the list of variables that are frequently used in this thesis.

Quantity	Unit	Description
A	$[m^2]$	Cross sectional area of the reservoir
h_g	$[m]$	Gate opening signal, this is the control signal evaluated by MPC
h_1	$[m]$	Absolute water level in Merkebekk
h_2	$[m]$	Absolute water level in Dalsfoss
\dot{V}_o	$[m^3/s]$	The total volumetric water outflow from Dalsfoss station
\dot{V}_t	$[m^3/s]$	Volumetric inflow to the turbines
\dot{V}_i	$[m^3/s]$	Inflow into the Lake Toke
\dot{V}_g	$[m^3/s]$	Volumetric outflow through the flood gates
w_R	—	Weight given to level while formulating objective function
$w_{\Delta u}$	-	Weight given to change in gate opening while formulating objective function
w_u	-	Weight given to gate opening while formulating objective function
x_M	$[m]$	Relative water level at Merkebekk from sea level
x_D	$[m]$	Relative water level at Dalsfoss from sea level

Here is the list of abbreviations that are often used throughout the report.

Abbreviations	Full form
CPU	Central Processing Unit
MPC	Model Predictive Control
MOO	Multi Objective Optimization

NLP	Non-Linear Programming
NMPC	Non-Linear Model Predictive Control
OCP	Optimal Control Problem
WSA	Weighted Sum Approach

1 Introduction

1.1 Background

Although the application of MPC was originally designed to be applicable in the field of chemical processes [1], MPC now has been of immense interest in a wide range of fields including hydropower plants. The ability of MPC to handle uncertainties that may occur in the system and realization of constraints in the process variables are the major features researchers in the field of hydropower are attracted to and motivated to implement MPC in this sector. Furthermore, an optimal centralized control system can be achieved using MPC for the smooth operation of hydropower plants [2].

The operation of hydropower production is directly associated with the water available in the reservoir, and it is always desirable to maintain the maximum water level in the reservoir. However, the unforeseen circumstances due to weather change result in the uneven water quantity in the reservoir. One situation may be unanticipated flooding in the reservoir, resulting in water overflow, while another could be a water deficit in the reservoir. In both scenarios, the proper operation of the water handling mechanism, known as flood gates, becomes crucial for water resource management.

State-of-art water resource management applications for the hydropower plant include components like hydrological modeling and optimization techniques for decision-making in reservoir operation [2]. Different possible inflows in the reservoirs are predicted over a certain period using hydrological modeling, and these inflows cause the behavioral change of the plant as they incorporate disturbance in its operation. To handle such dynamical behavior change, techniques based on optimization can suggest the proper operation of flood gates such that the optimal operation is ensured. A deterministic or stochastic MPC can be the solution to address this scenario. These controllers can take all the ensembles disturbance, optimize the system, and suggest the control strategy for the reservoir operation addressing the constraints imposed on the system.

1.2 Objectives

The main objective of this thesis is to design a SMPC that can handle inflow uncertainties in Lake Toke and determine the optimal flood gate openings to maximize water levels in the reservoirs. Because choosing the right objective function is critical on achieving this goal, the first step is to find a suitable objective function. Because the inflow uncertainties utilized in simulations are distributed over the next 13 days, SMPC considers 13 days of prediction horizon.

1.3 Previous Work

The model of Lake Toke was developed back in 2014, and is based on mass balance equations as given in [3]. The differential and algebraic equations representing Lake Toke were simulated to see the behavior of the model. Furthermore, a deterministic MPC algorithm was proposed based on the reference region tracking cost function. The inflow to the system was assumed to be constant over the whole prediction horizon when formulating MPC which does not reflect the reality. Later in 2019, the author in [4] designed a deterministic MPC with the same cost

function, furthermore extended to stochastic analysis. However, the inflow ensembles used for the simulation were fictitious. Although the simulation results look sensible for both deterministic and stochastic design, the data used to design MPC was never the real data therefore may not represent the actual behavior of the MPC controller. Moreover, the stochastic MPC was designed in python platform with control signal grouping technique to reduce the computational load. In [5], the authors looked for the changes in the cost function and designed deterministic MPC accordingly.

In this thesis, the two cost functions proposed in [5] are revisited and compared to observe the best-performing cost function based on deterministic MPC results. Finally, the cost function suitable for this thesis is selected to design stochastic MPC. Although ensemble forecasts used to simulate stochastic MPC are real data, the inflows are limited to certain value in some of the simulations. The reason behind limiting the inflow will be discussed later in Chapter 5.

1.4 Requirements

MPC design requires a computer platform with an optimization tool, such as MATLAB or Python. The optimization tool might be embedded within the programming environment, or it could be a compatible framework containing optimization techniques. This thesis uses IPOPT (Interior Point Optimizer) from CasADi framework. Furthermore, high-performance CPUs are recommended for dealing with the computing demands of stochastic simulation. The results in this thesis, however, were achieved with an 8-gigabyte CPU.

1.5 Thesis structure

The thesis is divided into six chapters, the first of which is the introduction. The second chapter discusses the system under investigation, the regulations that have been put on it, and quick visualization of the inflow uncertainty. Chapter 3 covers stochastic MPC formulation, numerical approaches for optimal control problems, and an introduction to CasADi. With a brief explanation, Chapter 4 summarizes the findings obtained using two alternative objective functions. The results of stochastic MPC are summarized and discussed in Chapter 5. Finally, a conclusion is given in Chapter 6.

Furthermore, the contextual equations, figures, chapters, and sections are cross-linked such that clicking on them leads to the connected part, avoiding information duplication across the report.

2 System Overview

This chapter describes the system under study with its governing models, different parameters, regulations, and uncertainties that may occur in the process.

2.1 Lake Toke and operational regulations

Kragerø Waterways located in Telemark Norway has been used by the Skagerak Energi for hydropower operation. Altogether five hydropower plants are being operated in this waterway starting at Dalsfoss hydropower station. This station uses Lake Toke as the reservoir for its operation which has a catchment area of over 1200 square kilometers [3].

Over a year, Lake Toke faces drastic changes in the water level; due to the ice melting and rain in the months of April-May, it faces heavy inflow causing floods in Lake Toke. The two flood gates operating in Dalsfoss serve to bypass the heavy inflow. However, it is important to maintain the levels at Merkebekk and Dalsfoss, shown in Figure 2-1, to a certain limit. The regulations on water levels are imposed by the Norwegian government authority, Norwegian Water Resources and Energy Directorate (NVE), and these levels constraints change over the years. The water level, mainly at Merkebekk, should not cross the upper regulated and lower regulated value named as x_{LRV} and x_{HRV} shown by the red and blue lines respectively as shown in Figure 2-2.

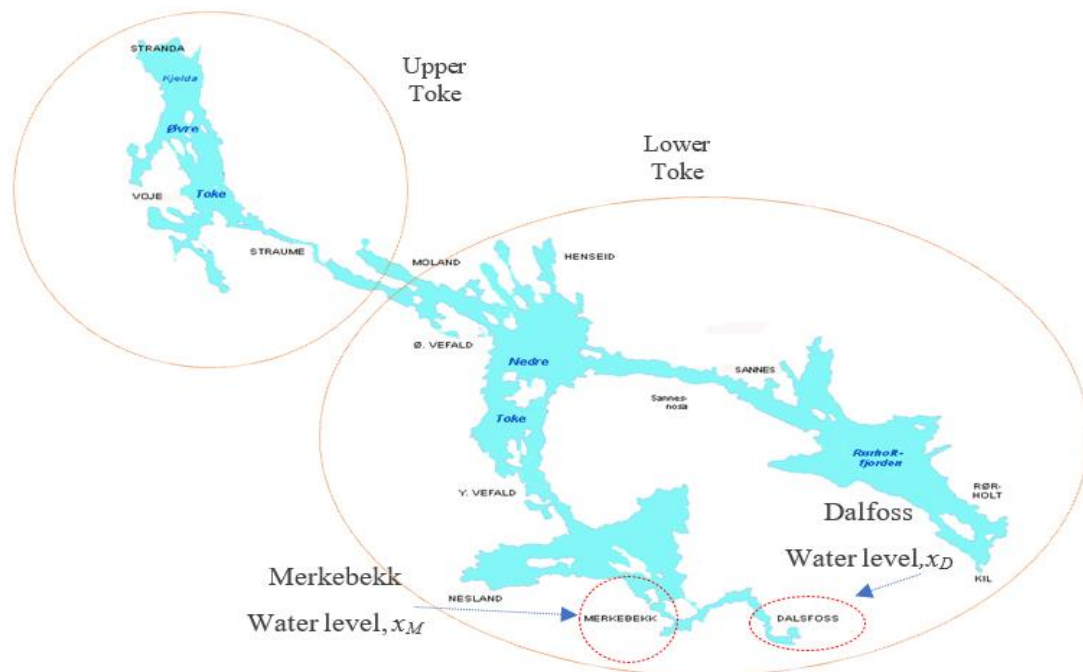


Figure 2-1: Map Showing Lake Toke, Merkebekk and Dalsfoss point [3]

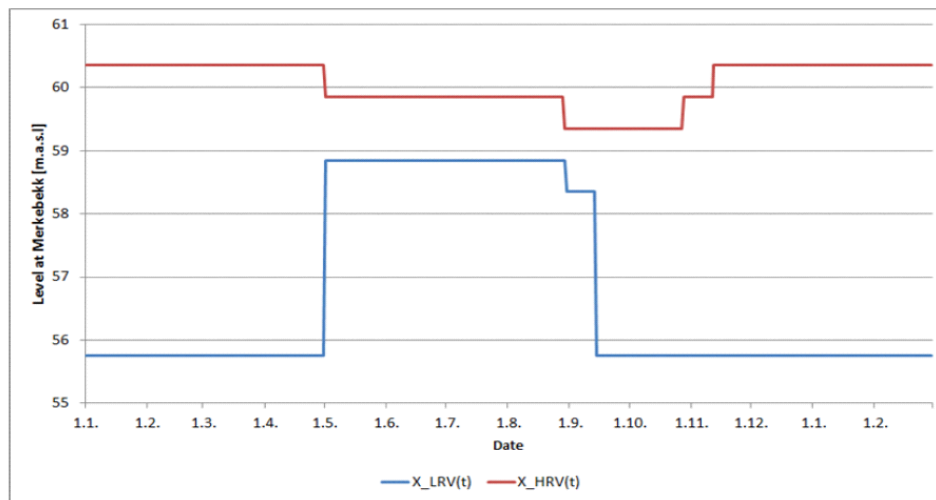


Figure 2-2: Constraints on water levels throughout the year [3]

Table 2-1: Constraints on water level throughout the year[3]

Date range	Lowest Regulated value x_{LRV} [m]	Highest regulated value x_{HRV} [m]
Jan 1 – April 30	55.75	60.65
May 1 – Aug 30	58.85	59.85
Sept 1 – Sept 14	58.35	59.35
Sept 15 – Oct 27	55.75	59.35
Oct 28 – Nov 11	55.75	59.85
Nov 12 – Dec 31	55.75	60.35

Furthermore, the downstream outflow from the Dalfoss station operation should not be abruptly changed since this might endanger humans and animals. The minimum outflow should be more than $4 \text{ m}^3/\text{s}$, although it is expected that a flow rate greater than $10 \text{ m}^3/\text{s}$ is good since it supports the functioning of other downstream hydroelectric plants. This need does not appear to be an issue while hydropower is in operation, as Dalsfoss station has three hydro turbines with a total capacity of $36 \text{ m}^3/\text{s}$ [3].

2.2 Process Model

Figure 2-3 shows the functional block diagram of Lake Toke with its input, outputs, states and disturbance variables. The control signal, h_g , is applied for the flood gates opening and closing such that the levels at Merkebekk x_M and Dalsfoss x_D , are maintained to the desired limit. The disturbance acting on the system is the volumetric flow, \dot{V}_i , into Lake Toke; these volumetric inflows are computed by the Skagerak Energi using a hydrological model based on the measurements and historic data. The water inflow to the turbine, V_t , also acts as a disturbance to the system in reality and varies depending upon the power production forecast, however, for simplicity V_t will be assumed constant in this thesis.

The outflow from the flood gates, \dot{V}_g , and through the turbines constitute the total outflow, \dot{V}_o , from the Dalsfoss station as given in Equation 2-1.

$$\dot{V}_o = \dot{V}_g + \dot{V}_t \quad 2-1$$

As there are two flood gates, therefore the total flow through both gates is given by Equation 2-2.

$$\dot{V}_g = \dot{V}_{g,1} + \dot{V}_{g,2} \quad 2-2$$

The individual flow through the gates is evaluated using Equation 2-3, and the schematic of the flood gate is shown in Figure 2-4.

$$\dot{V}_{g,j} = C_d * w_j * \min(h_g, h_2) * \sqrt{2 * g * \max(h_2, 0)} \quad 2-3$$

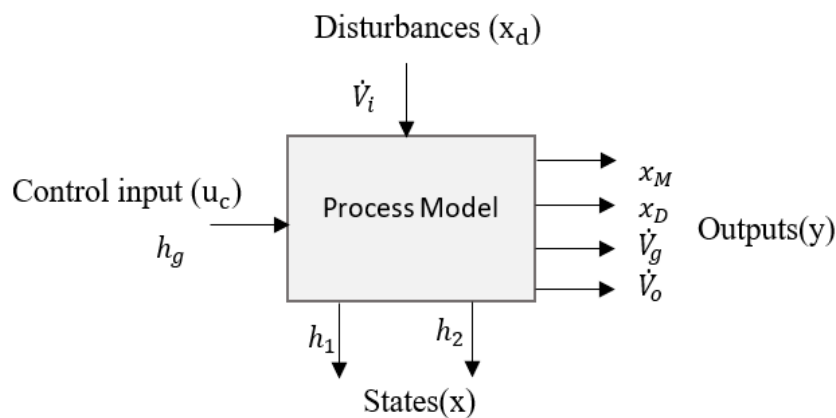


Figure 2-3: Functional block diagram of lake Toke [3]

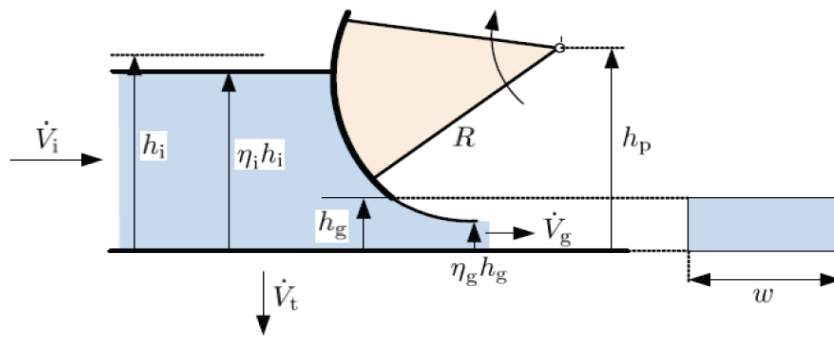


Figure 2-4: Schematic of flood gates [3]

The cross-sectional area of the reservoir is the function of the water level in it which can be seen in Figure 2-5 and given by Equation 2-4.

$$A(h) = \max(28 * 10^6 * 1.1 * 10^{(1/h)}, 10^3) \quad 2-4$$

The system states, h_1 and h_2 , are the absolute levels of Lake Toke at Merkebekk and Dalsfoss respectively, and given by Equation 2-5 and Equation 2-6 respectively.

$$\frac{dh_1}{dt} = \frac{1}{(1 - \alpha)A(h_1)} \left((1 - \beta)\dot{V}_i - \dot{V}_{12} \right) \quad 2-5$$

$$\frac{dh_2}{dt} = \frac{1}{\alpha A(h_2)} (\beta\dot{V}_i + \dot{V}_{12} - \dot{V}_t - \dot{V}_g) \quad 2-6$$

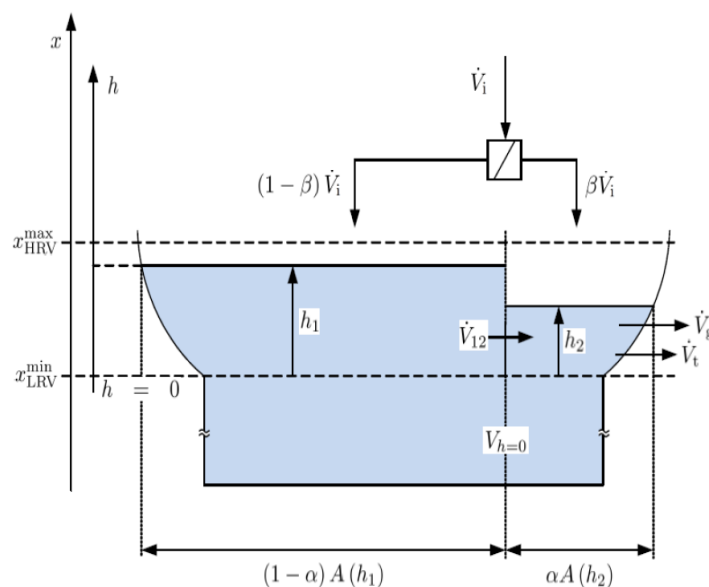


Figure 2-5: Schematic of Lake Toke [3]

The heights of water level relative to sea level at Merkebekk and Dalsfoss are given by Equation 2-7 and Equation 2-8 respectively.

$$x_M = h_1 + x_{LRV}^{min} \quad 2-7$$

$$x_D = h_2 + x_{LRV}^{min} \quad 2-8$$

And the intercompartmental flow in the reservoir is given by Equation 2-9.

$$\dot{V}_{12} = K_{12} \cdot (h_1 - h_2) \cdot \sqrt{abs(h_1 - h_2)} \quad 2-9$$

The parameter values that are used in Equations from 2-1 to 2-9 are provided in Table 2-2.

Table 2-2: Parameters used in Lake Toke model equations [3]

Parameter	Value	Unit	Comment
α	0.05	-	Fraction of surface in compartment 2
β	0.02	-	Fraction of inflow to compartment 2
K_{12}	800		Intercompartment flow coefficient
C_d	0.7	-	Discharge coefficient, Dalsfoss gate
w_1	11.6	<i>m</i>	Width of Dalsfoss gate 1
w_1	11	<i>m</i>	Width of Dalsfoss gate 2
x_{LRV}^{min}	55.75	<i>m</i>	Minimum low regulated level value
x_{LRV}^{max}	60.35	<i>m</i>	Maximum high regulated level value
g	9.81		Acceleration due to gravity

2.3 Time period selection for simulation

In Section 2.1, the need for maximum permissible levels in the reservoir for the entire year was provided. In the simulations reported in this thesis, the level limitations are evaluated for a certain period, from April 15 to May 15, as shown in Figure 2-6. This period is particularly of interest for the hydroelectric operation since substantial inflows into the reservoir are expected. The goal is to see if hydropower can be run in the best possible environment, meaning that water levels are optimal for power generation even when a flood is a possibility.

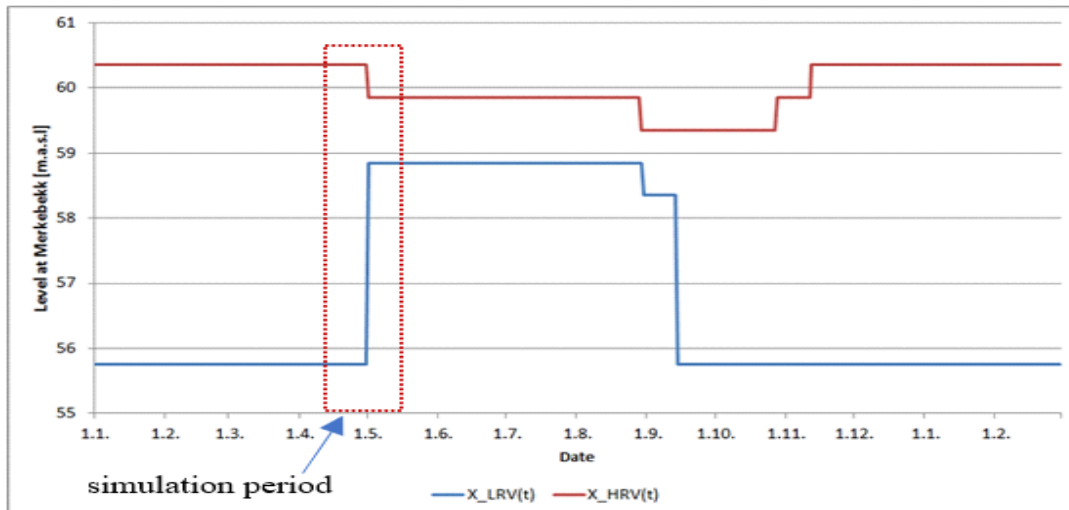


Figure 2-6: Indication of simulation time period

2.4 Open-loop simulation

The model equations presented in Section 2.2 are simulated in MATLAB with a sampling time of 1 hour and a total simulation time of 1 month. For this purpose, the flow through the turbines is set to their maximum capacity, $36 \text{ m}^3/\text{s}$, and the inflow to the reservoir is assumed to have two different scenarios of $200 \text{ m}^3/\text{s}$ and $250 \text{ m}^3/\text{s}$. The simulation results are shown in Figure 2-7.

The results seem logical because when the flood gates are closed, the levels go on increasing in the reservoir. On the other hand, with the opening of flooding gates, the level decreases. Furthermore, when the inflow to the reservoir is $200 \text{ m}^3/\text{s}$, the rise in the levels is slow as compared to the inflow of $250 \text{ m}^3/\text{s}$. When high inflow occurs in the reservoir, flood gates needed to open more to prevent flooding situation which is justified in Figure 2-7.

It is important to note that the gate openings in this simulation are set manually and adjusted as required to realize the flooding control scenario: levels within the given constraints. However, the design of MPC facilitates the same operation with the optimal evaluation of the gate opening/control signal.

2.5 Inflow data visualization

Skagerak Energi records the forecast of ensembles for the next 13 days by updating twice a day. Although the ensembles are updated twice, one-time data are used for the simulation purpose in this thesis. The forecasts consist of 50 different ensembles generated with slight variations in the initial condition of the hydrological model and weather conditions [4]. One example of such an ensemble forecast is shown in Figure 2-8.

From Figure 2-8, it can be observed that the ensemble forecasts tend to scatter from each other when moved ahead in time which signifies that the range of uncertainty in a forecast increases when the number of prediction days becomes greater.

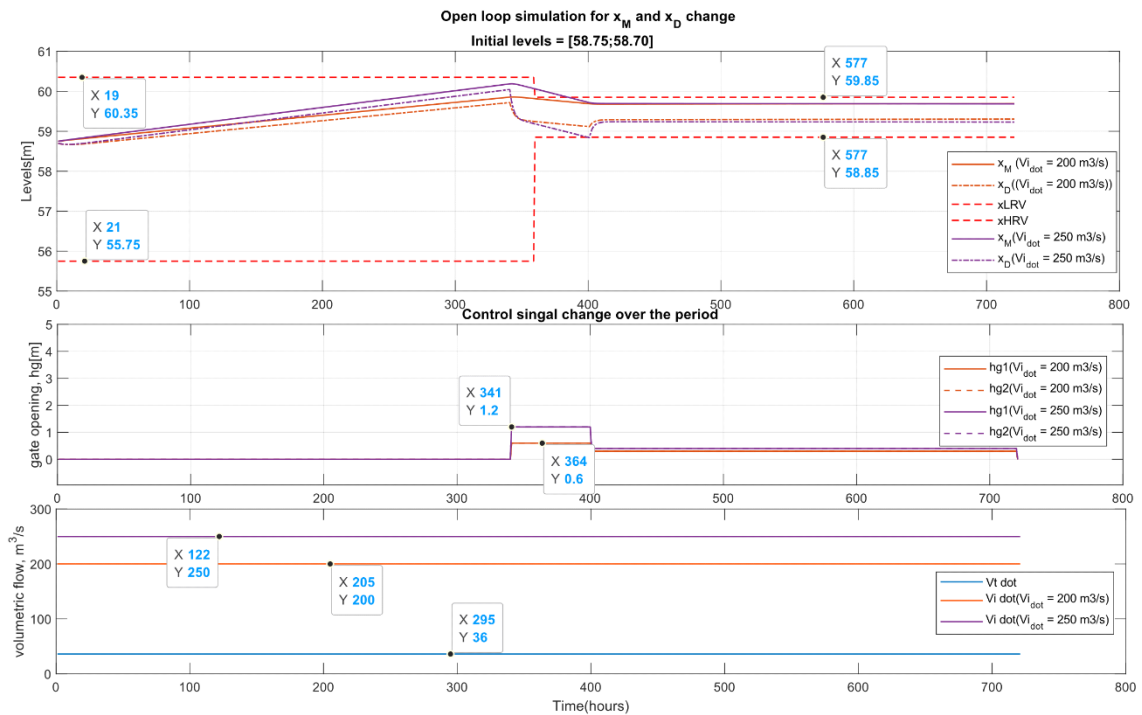


Figure 2-7: Open-loop simulation result: with higher inflow in the reservoir, levels increase faster, and therefore more gate opening is required to meet constraints

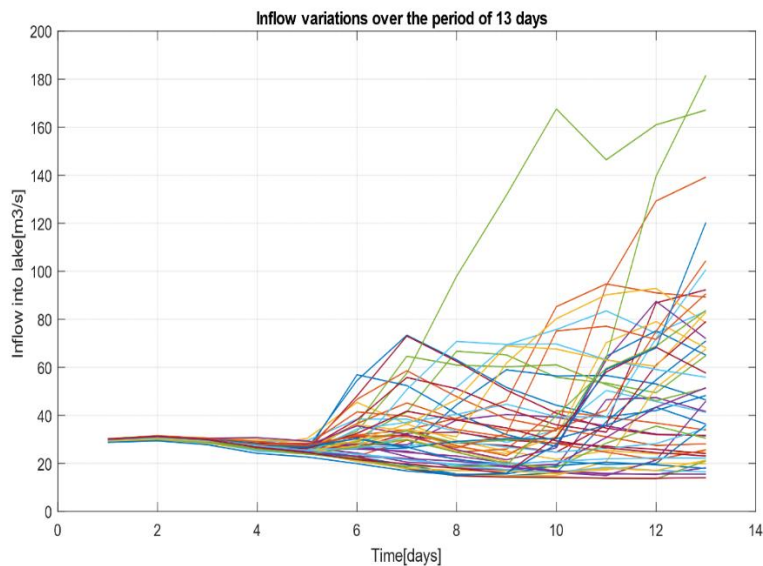


Figure 2-8: Ensemble forecast recorded on April 22, 2020

The forecast ensembles recorded over the whole year of 2020 and 2021 have been provided by Skagerak. The 50 different variations in inflow obtained each day from April 15, 2020, to May 15, 2020, and April 15, 2021, to May 15, 2021 are shown in Figure 2-9 and Figure 2-10 respectively. It is important to note that the data updated each day are repeated 24 times and then plotted. The inflow into the reservoirs has been predicted quite high at the start of May in 2020, however, the inflow is higher around the middle of May in the year 2021.

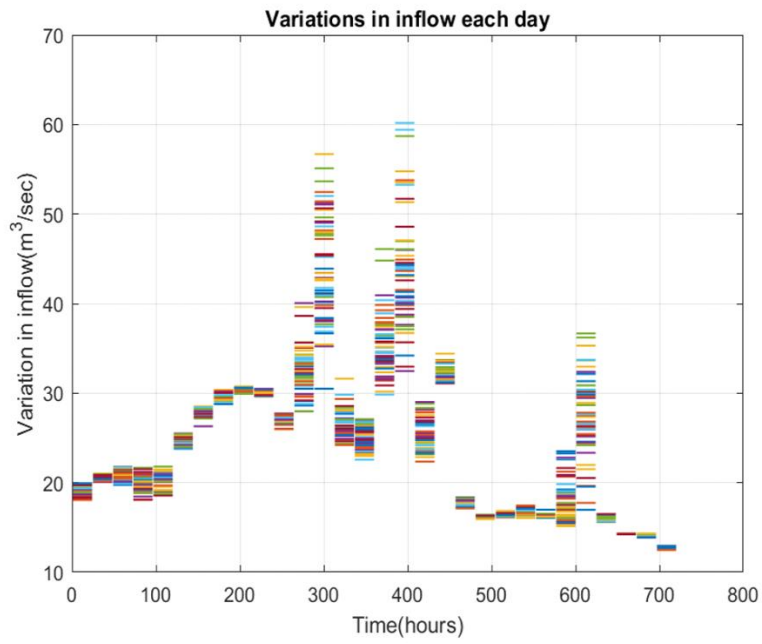


Figure 2-9: Each day inflow variations starting from April 15, 2020 - May 15, 2020

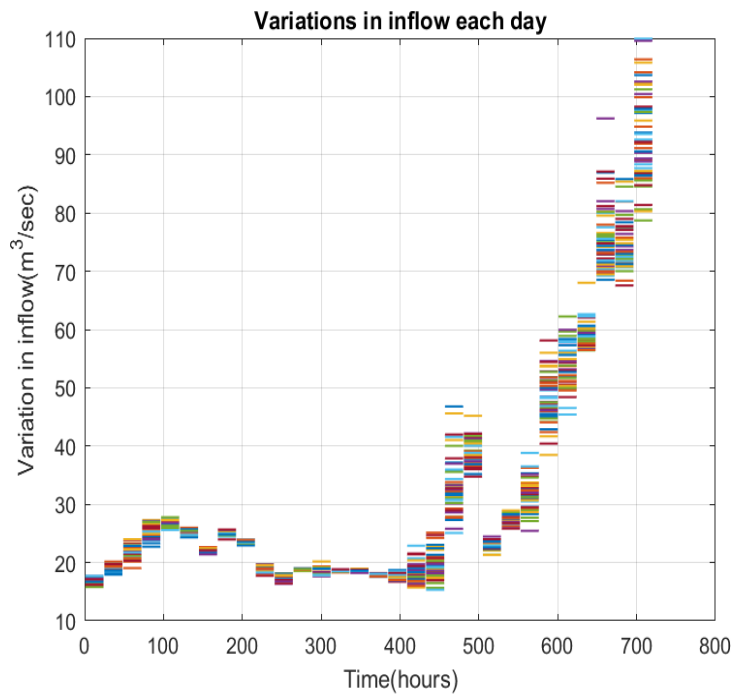


Figure 2-10: Each day inflow variations starting from April 15, 2021 - May 15, 2021

3 Theory

The formulation of stochastic MPC can be carried out using a wide range of methods. However, this section primarily focuses on some methods named scenario-based multistage NMPC, weighted sum method; one of the methods to solve MOO and Min-Max MPC. These formulations are the optimization problem with non-linear functions involved, and therefore can be transformed into Non-Linear Programming Problems (NLP). Some of the methods to transcribe optimization problems into NLP problems are also discussed in this section. Furthermore, a numerical optimization framework – CasADi – will be introduced.

3.1 Some methods for SMPC formulation

3.1.1 Multi-stage scenario-based NMPC

The idea behind the multistage MPC formulation comes from the multistage stochastic programming. The first output variable, y_1 must be selected anyway however the outputs afterward, y_2, y_3, \dots, y_N , are the results of uncertainties that have been acted on the system as shown in Figure 3-1. To be specific, y_3 has the impact of both uncertainties represented by the symbols δ_1 and δ_2 as shown in Figure 3-1. To counteract the changes in the system due to these uncertainties, the optimal moves are evaluated over a certain time, which is also called the prediction horizon [5]. Furthermore, the optimal moves must be evaluated in each time step to encounter the changes of uncertainties in the future which is called closed-loop optimization [6].

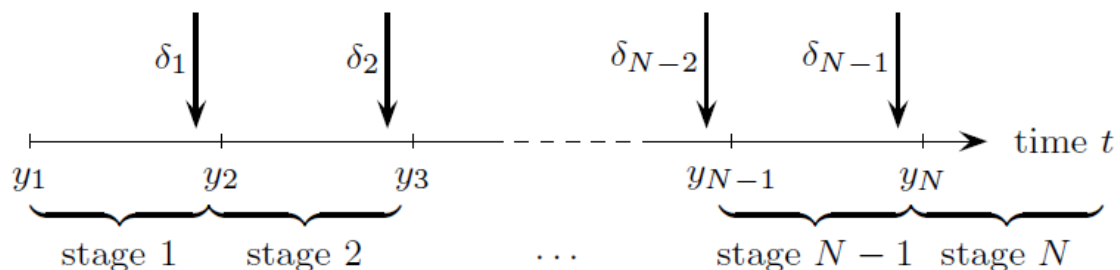


Figure 3-1: Stochastic Programming in multistage [1]

Multistage NMPC is also known as the closed loop optimization method since it explicitly takes into account the new information or changes that happen to the system in the future and recomputes optimal control input to the system. In this approach, the uncertainties are modeled as a moving scenario tree as shown in Figure 3-2. Each branch of the scenario diagram represents variable trajectories of the states or outputs considering the future evolutions in the uncertainties at each point over the prediction horizon. It is important to note that the control moves acting in each branch are set equal (i.e., $u_0^1 = u_0^2 = u_0^3, \dots$) to realize real time decisions to the system. This idea of setting the same control action is called non-anticipativity constraints [7].

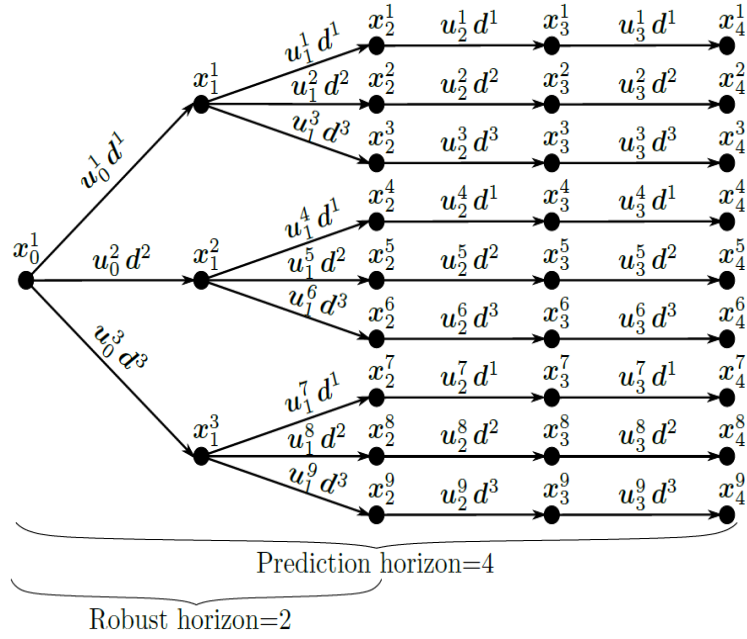


Figure 3-2: Example of scenario diagram for multistage NMPC [8]

In [9], the author emphasizes the importance of considering the robust horizon when the length of the prediction horizon increases. With the increase in the number of the prediction horizon, the branching from each node or uncertainty increases resulting in computational complexity. The idea to deal with this case is to limit the branching after a certain point of time, called a robust horizon as shown in Figure 3-2, such that uncertainties remain constant.

The optimization problem resulting from the scenario-based multistage formulation can be written as:

$$\min_{x_k^j, u_k^j \forall (j,k) \in I} \sum_{i=1}^N w_i J_i(X_i, U_i) \quad 3-1$$

subject to,

$$x_{k+1}^j = f(x_k^{p(j)}, u_k^k, d_k^{r(j)}), \quad \forall (j, k+1) \in I \quad 3-2$$

$$g(x_{k+1}^j, u_k^j) \leq 0 \quad \forall (j, k) \in I \quad 3-3$$

$$u_k^j = u_k^l \text{ if } x_k^{p(j)} = x_k^{p(l)} \quad \forall (j, k), (l, k) \in I \quad 3-4$$

where, in Equation 3-1, X_i and U_i represent the set of states and control inputs that belong to scenario S_i . S_i denotes the i th scenario which is the path from the root node x_0 to one of the leaf nodes and it contains all the states x_k^j and control inputs u_k^j that belong to the i th scenario. N represents the number of scenarios (or leaf nodes).

The cost of each scenario is represented by $J_i(\cdot)$ and given by Equation 3-5

$$J_i = \sum_{k=1}^{N_p} L(x_{k+1}^j, u_k^j) \quad \forall (x_{k+1}^j, u_k^j) \in S_i \quad 3-5$$

where N_p represents the length of the prediction horizon.

Equation 3-2 above represents the state trajectories and x_{k+1}^j at stage $k+1$ and position j is the function of the previous state $x_k^{p(j)}$, vectors of control inputs u_k^j and the corresponding realization r of the uncertainty $d_k^{r(j)}$. The superscript $p(j)$ denotes the index of the previous node in the tree, which is the function of its position j and stage k . The index set of all occurring indices (j, k) is denoted by I . Similarly, Equation 3-3 represents the nonlinear constraints and Equation 3-4 represents an anticaptivity constraint. Details of the symbolic representation can be found in [9].

3.1.2 MPC based on multi-objective optimization

The general MOO can be posed as:

$$\min_x F(x) \quad 3-6$$

$$\text{where, } F(x) = [F_1(x), F_2(x), \dots, F_k(x)]$$

subject to,

$$g_j(x) \leq 0 \quad j \in 1, 2, 3 \dots, m \quad 3-7$$

$$h_p(x) = 0 \quad p \in 1, 2, 3 \dots, n \quad 3-8$$

where k is the number of the objective function, m is the number of inequality constraints, n is the number of the equality constraints and x is the design or optimization variable.

The optimized variable x that minimizes all the objectives simultaneously normally does not exist. In most of the real-life examples, these objectives are contradictory to each other, for example, maximizing quality versus minimizing the cost of the goods. Therefore, there comes a concept of Pareto optimality to describe the solutions for MOO problems. The solution of MOO is Pareto optimal if it is not possible to improve one objective without harming the other objectives [10]. Furthermore, for the given MOO problem, the solution set, also called Pareto optimal set, can be more than one in numbers and all these solution sets are collectively known as Pareto Front. All the possible points in Pareto Front are equally acceptable solutions, however, the selection of the best solution depends upon the decision-makers [11]. There are different methods available to solve such kinds of problems and these methods are briefly listed in Figure 3-3.

Among these available methods, Weighted Sum Approach (WSA) has been used to design stochastic MPC in this thesis which is described in detail in Chapter 5.

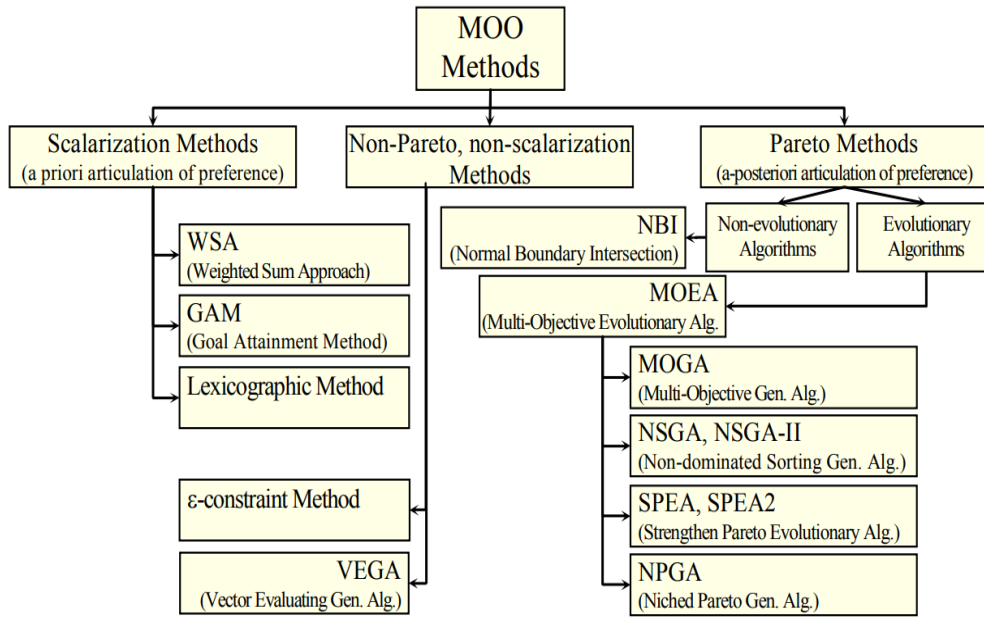


Figure 3-3: Methods to solve MOO problems [12]

3.1.3 Min-Max MPC

The formulation of min-max MPC can be either an open loop, which is also called classical min-max MPC, or can be closed-loop [13]. The classical min-max MPC determines the control action that can guarantee the constraints along the predicted trajectory of the states and minimizes the cost associated with the worst-case uncertainty. It is important to note that these control actions ignore the sliding horizon strategy [14]; the first control action is applied to the process and the future controls are evaluated by solving the optimization problem in each time step addressing the new uncertainties. Such open-loop control actions give poor closed-loop performance when the uncertainties are assumed time-invariant in the formulation [15] and sometimes lead to extremely conservative solutions [13]. On the other hand, in the closed-loop type, control actions are part of the optimization problem in each step. The solution obtained from this method demonstrates a better closed-loop performance index, however, introduces a computation burden.

The closed-loop min-max NMPC can be presented in a form similar to multistage NMPC by simply replacing the summation sign of Equation 3-1 with a max operator as shown in Equation 3-9.

$$\min_{x_k^j, u_k^j \forall (j,k) \in I} \max w_i J_i(X_i, U_i) \quad 3-9$$

The constraints to Equation 3-9 are similar as given in Equation 3-2 to Equation 3-4.

3.2 Numerical Methods for the optimal control problem

The formulation of a dynamic optimization problem or optimal control problem can be done in a variety of ways with the use of different types of objective functions/cost functions and constraints. The constraints in the formulation can be represented by ordinary differential equations (ODEs), differential-algebraic equations (DAEs), or partial differential equations (PDEs). The basic solution policy is to minimize/maximize the objective function by addressing the defined constraints. Over the past decades, there has been significant development in the efficient approaches for solving optimal control problems such as control parametrization [16], collocation method [17], and model-reality difference approach [18],[19]. Basically, the algorithms developed are divided into indirect and direct methods as shown in Figure 3-4.

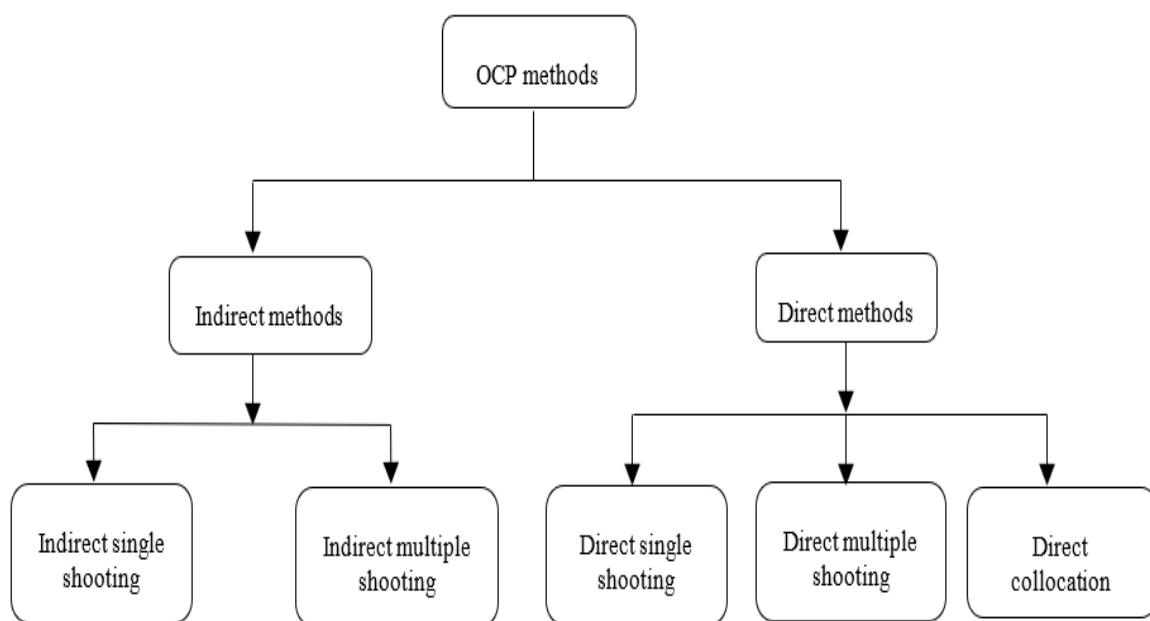


Figure 3-4: Numerical methods for dynamic optimization problem [20]

Indirect methods are based on finding optimal solutions based on the optimality condition criteria instead of minimizing or maximizing the cost criteria. The optimality criteria lead to a two-point or multi-point boundary value issue, depending on the specified optimal control problem (MPBVP), and may give accurate solutions [21]. Nevertheless, these methods suffer significant numerical difficulty and make a computational burden, if the initial guess is not quite good enough [22]. These methods are not implemented in any simulations presented in this thesis.

Direct methods [21], which are based on a discretization to time, address optimum control problems for purely continuous and nonlinear systems. That is, by assessing state and control values only at a fixed number of time samples, the infinite-dimensional issue of identifying optimum state and control trajectories is reduced to a finite-dimensional challenge. In other words, these states and control variables are replaced with piecewise constant parametrization [23]. The implementation of these methods may be easier as compared to the indirect methods; however, the optimality results are usually not guaranteed [22], therefore can only be the

approximate solutions. In this thesis, the results based on the application of direct multiple shooting will be presented.

Direct single shooting method

The direct single shooting method is based on the discretization of control inputs only as the decision variables in the optimization algorithm. The control input $u(t)$ is discretized on the fixed number of time grid N , for example, $0 = t_0 < t_1 < \dots < t_N = T$ as shown in Figure 35, and the states $x(t; q)$, q being control parameters such that $q = (q_0, q_1, \dots, q_{N-1})$, on $[0, T]$ are considered as dependent variables evaluated using numerical integration methods. After control discretization and the numerical solution for states, an optimization algorithm is solved using a finite-dimensional optimization solver, for example, Sequential Quadratic Programming (SQP) by defining cost function and constraints. This method is also called the sequential method because numerical integration and optimization are performed one after the other [24].

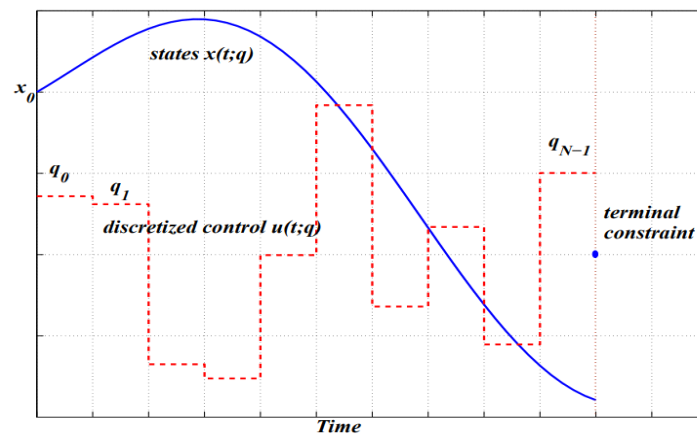


Figure 3-5: Illustration of single shooting: discretization of control inputs only [25]

Direct single shooting method

Direct multiple shooting method based on the discretization of both the control inputs and the states. Each step involved in the transcription of the optimal control problems into NLP can be pointed out as:

- Divide total time, T , intervals into N equal subintervals.

$$t_0 < t_1 < \dots < t_N = T \quad 3-10a$$

- Parameterize the control input and the initial condition of the states in each subinterval.

$$\begin{aligned} u_i(t) & \quad \forall t \in [t_i, t_{i+1}] & 3-10b \\ x_i(t) & \quad \forall t \in [t_i, t_{i+1}] \end{aligned}$$

- Calculate the state trajectories in each subinterval and the values of the states at the end of each subinterval using the parameterized values of the states at the beginning of each subinterval.

$$x_{i+1}(t) = f(x_i(t), u_i(t)) \quad \forall t \in [t_i, t_{i+1}] \quad 3-10c$$

- Define continuity/equality constraints to maintain continuity of the states between the subintervals.

$$h_{i+1} - x_{i+1}(t) = 0 \quad i = 0, 1, \dots, N - 1 \quad 3-11c$$

- In each subinterval, evaluate the objective function and formulate the NLP problem.
- Solve the NLP problem

3.3 CasADi – A framework for algorithmic differentiation and numerical optimization

CasADi [26] is an open-source software framework that is primarily used to tackle optimization issues in a flexible manner. It's developed in self-contained C++, but it may be used with a variety of programming languages, including Python, MATLAB, and Octave. CasADi originated as an algorithmic differentiation (AD) tool with a syntax similar to that of a computer algebra system (CAS), thus the name. While cutting-edge AD is still a big part of CasADi, in recent years the focus has shifted to optimization. It includes a set of methods that make solving numerical optimum control problems easier without sacrificing efficiency.

CasADi is based on a symbolic framework that allows users to build expressions and use them to create functions that are automatically differentiable. These general-purpose expressions are analogous to the Symbolic Toolbox in MATLAB and the SymPy package in Python.

With CasADi, NLPs can be solved using block structure or generic sparsity utilizing sequential quadratic programming (SQP) or interfaces to IPOPT/BONMIN, BlockSQP, WORHP, KNITRO, and SNOPT.

4 Deterministic MPC

The findings of deterministic MPC with two alternative objective function formulations are presented in this section. The purpose of implementing two alternative functions is to determine the best-performing objective function that can fulfill the optimal operating condition while attempting to keep the reservoir level at its maximum level.

The power production plan and inflows into Lake Toke are the disturbances affecting the hydropower plant, as described in Section 2.2. To satisfy the production plan, the equations determining power generation are solved to calculate how much water input through the turbine is necessary. However, if the turbines are permitted to work at full capacity, which is $36\text{m}^3/\text{s}$, the maximum production is expected. Here, the simulation results of deterministic MPC assume the full operation of turbines.

Furthermore, another disturbance, water inflow into the reservoir within the period of 15, April – 15, May is shown in Figure 2-9. From the 50 different possible inflows each day, a single inflow is selected which altogether forms a possible operational/realization inflow acting in the system as shown in Figure 4-1. The inflow scenario represented by the red solid line in Figure 4-1 is the disturbance inflow used for the simulation of deterministic MPC. The realization inflow has a maximum value of around $55\text{ m}^3/\text{s}$ and a minimum of $15\text{ m}^3/\text{s}$.

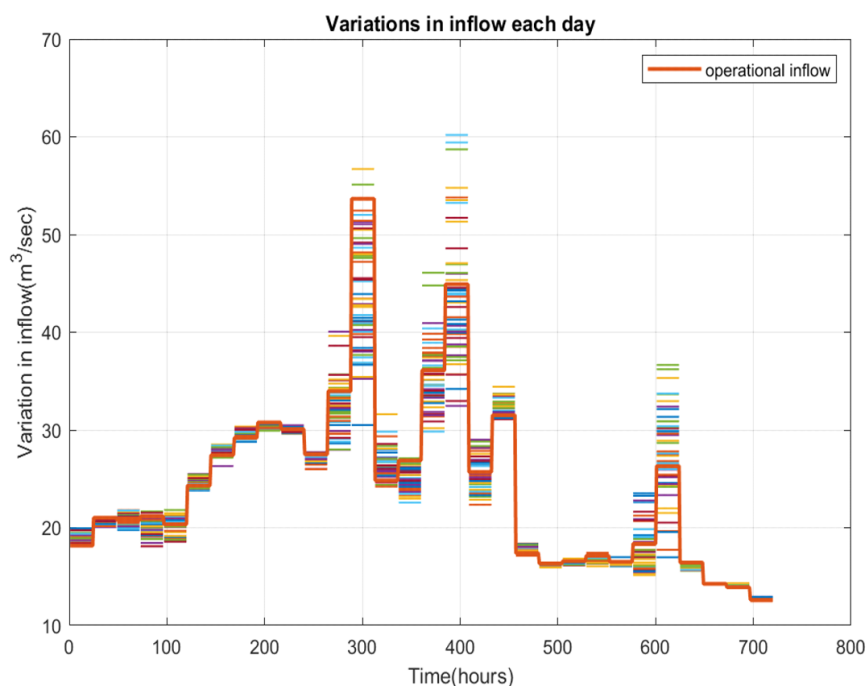


Figure 4-1: Operational inflow selection from the 50 different possible inflows each day.

The selection of a particular inflow for a day is based on the mean methods as given by Equation 4-1 and Equation 4-2. Equation 4-1 evaluates the mean of all the 50 ensembles by taking 13 days of predicted inflow data. Each mean value is denoted by M_1, M_2, \dots, M_{50} as shown in the first column of Equation 4-1. Equation 4-2 calculates mean of these mean values, M_m and then an inflow close to M_m is selected.

$$\dot{V}_i = \begin{pmatrix} M_1 & \leftarrow & \text{Mean}(V_{i,1}^{(1)}) & V_{i,1}^{(2)} & \dots & V_{i,1}^{(13)} \\ M_2 & \leftarrow & \text{Mean}(V_{i,2}^{(1)}) & V_{i,2}^{(2)} & \dots & V_{i,2}^{(13)} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ M_{50} & \leftarrow & \text{Mean}(V_{i,50}^{(1)}) & V_{i,50}^{(2)} & \dots & V_{i,50}^{(13)} \end{pmatrix} \quad 4-1$$

$$\text{Mean of mean, } M_m = \frac{M_1 + M_2 + \dots + M_{50}}{50} \quad 4-2$$

4.1 Type I OCP formulation based on reference region tracking

The traditional reference point tracking objective function is slightly modified to make reference region tracking as given by Equation 4-3 which is valid at time $t \in N$, N is the set of natural numbers.

$$\min_u J(h, u) = \sum_{k=1}^{N_p} w_R R^2 + \Delta u' w_{\Delta u} \Delta u + u' w_u u \quad 4-3$$

$$\Delta u = u_{t+k-1} - u_{t-1} \quad 4-4a$$

This cost function is subject to the dynamic model of the system along with an additional constraint that limits the maximum and the minimum control signal. These constraints are represented by Equation 4-5.

$$h_u(k+1) = f(h_u(k), u(k), d(k)) \quad 4-5$$

$$u \in [0, u^{max}]$$

The term R in Equation 4-3 represents the reference region contribution to the cost function which is evaluated as given by Equation 4-6.

$$R(h_{t+k}) = \min(x_{M,t+k} - r_{t+k}^l, 0) + \max(x_{M,t+k} - r_{t+k}^u, 0) \quad 4-6$$

where,

$$r_k^l = (1 - \chi) x_{LRV,k} + \chi x_{HRV,k} \quad 4-6a$$

$$r_k^u = x_{HRV,k} - \delta_{HRV} \quad 4-6b$$

The terms r_k^l in Equation 4-6a and r_k^u in Equations 4-6b represent the customized lower and upper reference boundaries respectively, although the permissible boundaries are x_{HRV} and x_{LRV} .

It is important to note that the term $R(h_{t+k}) = 0$, if $x_{M,t+k} \in [r_{t+k}^l, r_{t+k}^u]$. A normally used value of χ_R is 0.75 which signifies that the level of Merkebekk lies closer to the upper boundary x_{HRV} . The parameter values used in Equation 4-3 to Equation 4-6 are given in Table 4-1.

Table 4-1: Parameters used in Type I OCP formulation

Parameter	Value	Comment
χ_R	0.75	Location of r^u in $[x_{HRV}, x_{LRV}]$
δ_{HRV}	0.05	Safety margin for upper reference region

The cost function proposed in Equation 4-3 contains three weighting factors which are the tuning parameters of the controller. The performance of the controller is analysed with different groups of weighting factors as given in Table 2-1. Although the variations of these parameters can be done in any way, these groups are selected randomly to get an overview of the controller behaviour.

Table 4-2: Groups of tuning parameters to analyze the performance of the controller

Groups	w_R	$w_{\Delta u}$	w_u
Group 1	10	1	0.1
Group 2	1	100	0.1
Group 3	10	500	0.01
Group 4	1	10	100

The MPC simulation can begin at any moment in terms of reservoir levels. However, the simulations offered only explore two alternative beginning levels, with one set of levels in the lower zone and the other set of levels in the upper region.

Case I: Water levels initialized below the narrow reference boundaries

In this example, there is initially little water in the reservoir, yet the hydroelectric operation continues at full capacity for a month. The inflow in the reservoir is considered as it is, which means that the flood coefficient 1 is used in the given inflow data. In such a scenario, the behaviour of the levels and the control signals are shown in Figure 4-2. The levels of Merkebekk and Dalsfoss are constantly decreasing, and the gate opening is a straight line with a value of zero.

These results show that with limited inflow and low reservoir levels, operating the turbines at maximum capacity is not ideal since the water levels would decrease to the low region. In addition, the controller's behaviour is rational because the gate should not be opened to throw

water in such condition. It signifies that the power production plan should be changed accordingly rather than operating to produce maximum power.

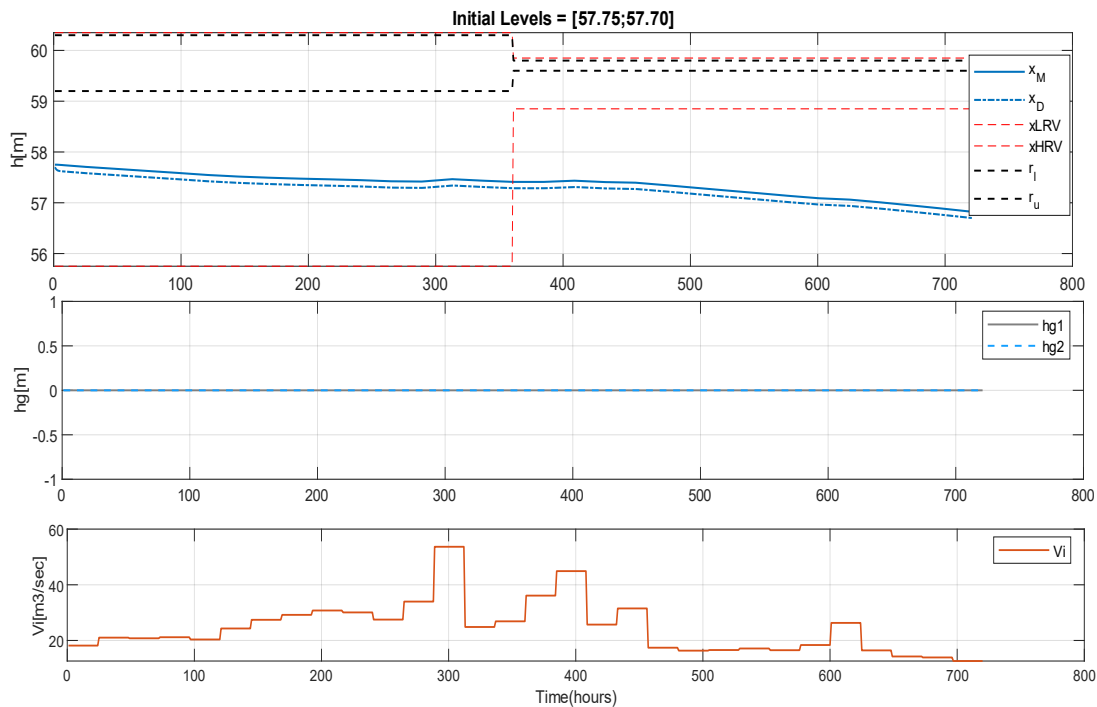


Figure 4-2: Type I OCP, initially water at the lower region of the reservoir - [57.75;57.70], turbine operating in full capacity- $V_i = 36 \text{ m}^3/\text{s}$, flood coefficient = 1, Group 1 tuning parameters

The outcome of considering low inflow situations provides two avenues to pursue to have a better knowledge of MPC performance. Either consider limited turbine input and use flood coefficient 1 or operate the turbine at full capacity and use flood coefficient 3 to account for larger reservoir inflow. Therefore, the simulation results presented in this Chapter consider the later assumption.

Figure 4-3 depicts the results of using the flooding coefficient 3, and all of the tuning parameter groups. Although the pattern of gate opening differs between tuning parameter groups, the effect on level is nearly the same. The water level in each case is increasing gradually until it hits the maximum allowed limit, x_{HRV} . Then, the gate opening signal is activated because the level has reached to bounds.

The flood gates of Dalsfoss are operated manually, therefore if the operator is allowed to choose the best control signal, the selection would be the signal that is less variable or practically constant for most of the time, since this would reduce the need for frequent visits and gate operations. If this is the case, Group 1 tuning appears to perform better than the others. It's worth noting that just the Merkebekk level is depicted because it represents a major part of the reservoir.

Figure 4-3 shows variation in the level of Merkebekk and Dalsfoss, gate opening signals, outflow through each gate, and the total outflow, and the inflow in the reservoir with flooding coefficient 3 respectively from top to bottom row using Group 1 tuning parameters.

Figure 4-5 shows the execution time to run Case I with Group 1 tuning parameters. It shows that the evaluation of the control signal to be used next hour only takes a fraction of seconds, that is, 0.19 seconds.

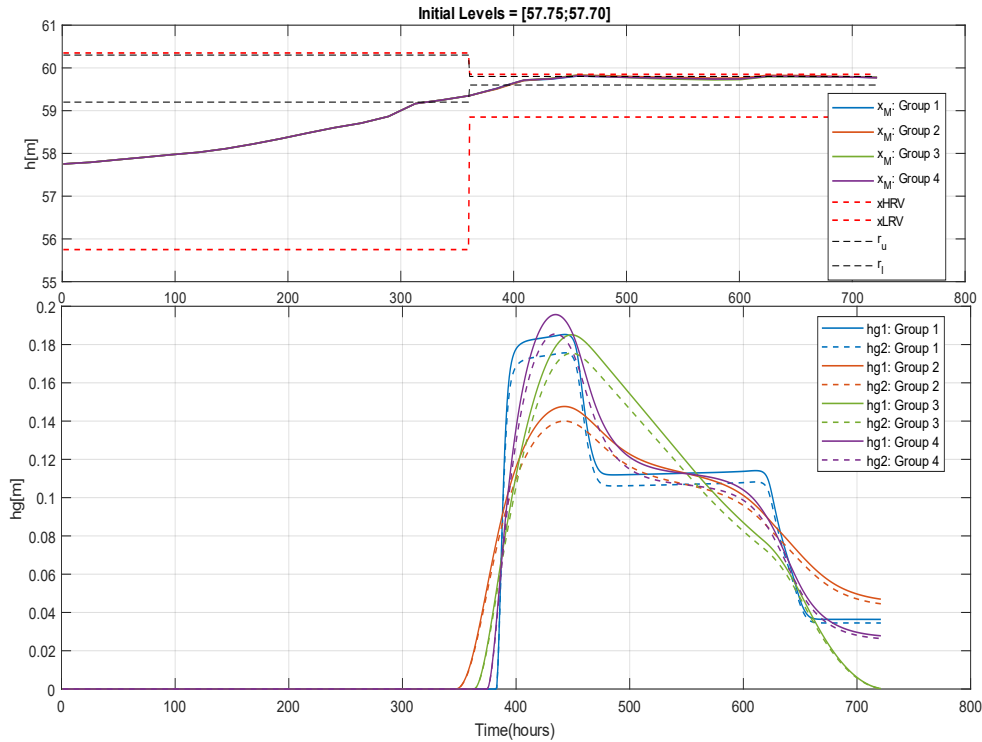


Figure 4-3: Type I OCP, initially water level lower region of the reservoir- [57.75;57.70], turbine operating in full capacity – $V_t = 36 \text{ m}^3/\text{s}$, flood coefficient = 3, results comparison using all groups of parameters

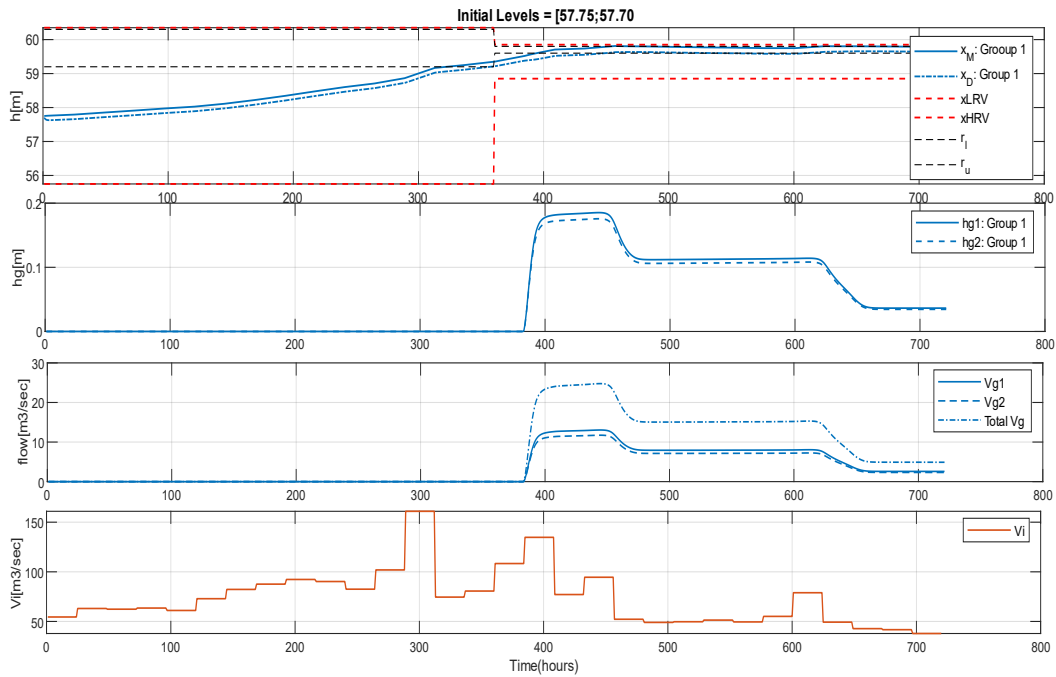


Figure 4-4: Type I OCP, initially water at the lower region of the reservoir - [57.75;57.70], turbine operating in full capacity- $V_t = 36$ m³/s, flood coefficient = 3, Group 1 tuning parameters

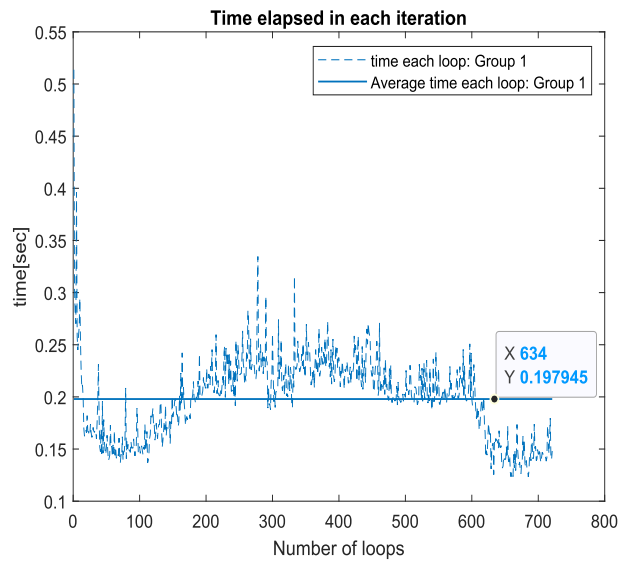


Figure 4-5: Type I, Case I OCP timing diagram; the results obtained within average loop run time of 0.2s with Group 1 tuning parameters.

Case 2: Water levels initialized inside the narrow reference boundaries

This case assumes that the levels are already in the upper region of the reservoir. Figure 4-6 shows the findings obtained after taking this case into account.

It can be noticed that the outcome for group 4 tuning settings differs from the other groups. As the flooding is very high, it is obvious to increase the level. At the same time, MPC should have suggested sufficient gate openings to throw extra flood water, but the MPC could not control the flooding in the reservoir with proper gate openings.

However, the first three groups have nearly identical results in terms of water levels, although the pattern of gate opening is slightly different. It seems difficult to prefer gate opening signal one over the other as the performance is almost similar. If the optimal gate opening from group 3 is chosen, Figure 4-7 shows the variation in the levels, control signals, flow through the gates, and inflow with flood coefficient 3 respectively.

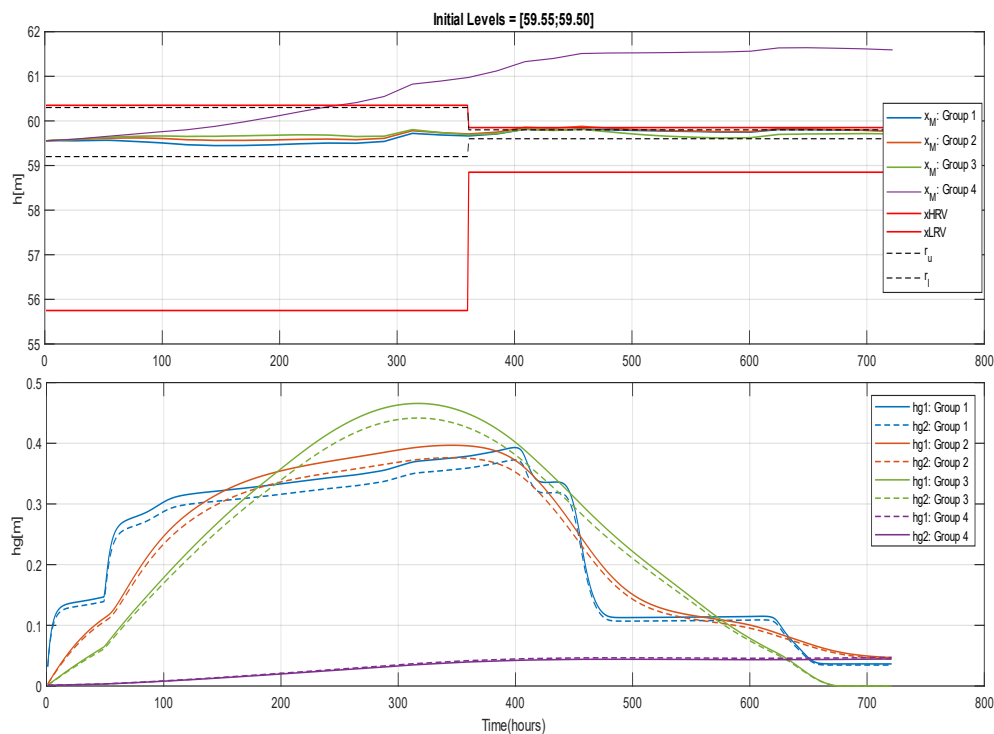


Figure 4-6: Type I OCP, initially water level at the higher region of the reservoir- [59.55;59.50], turbine operating in full capacity – $V_t = 36 \text{ m}^3/\text{s}$, flood coefficient = 3, results from comparison using all groups of parameters

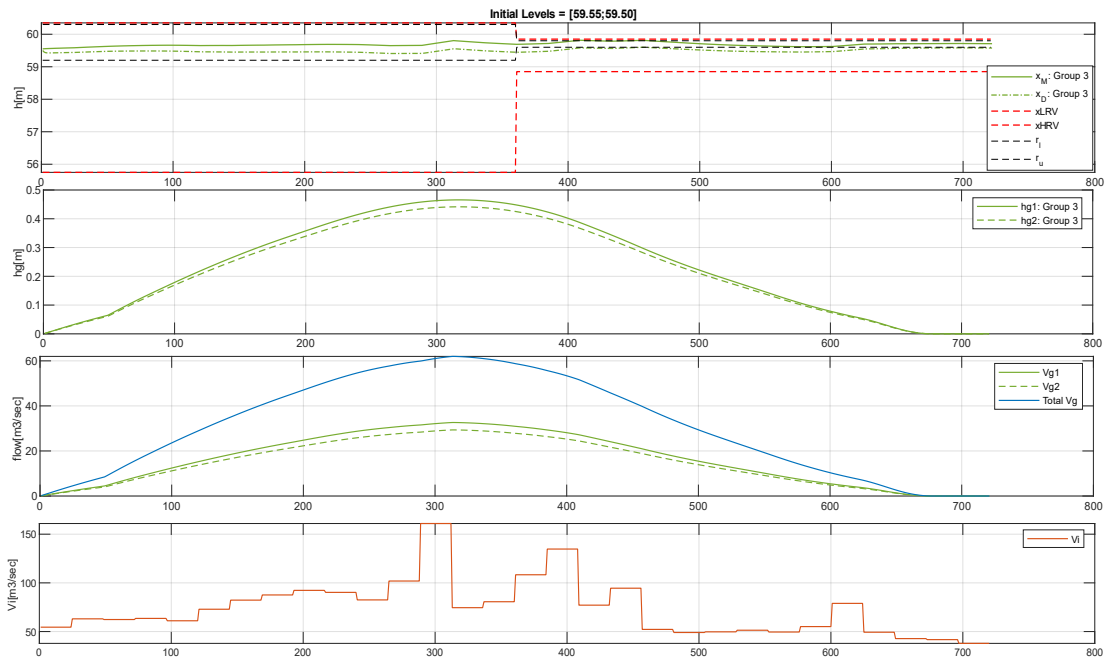


Figure 4-7: Type I OCP, initially water at the lower region of the reservoir - [59.55;59.50], turbine operating in full capacity- $V_t = 36 \text{ m}^3/\text{s}$, flood coefficient = 3, Group 3 tuning parameters

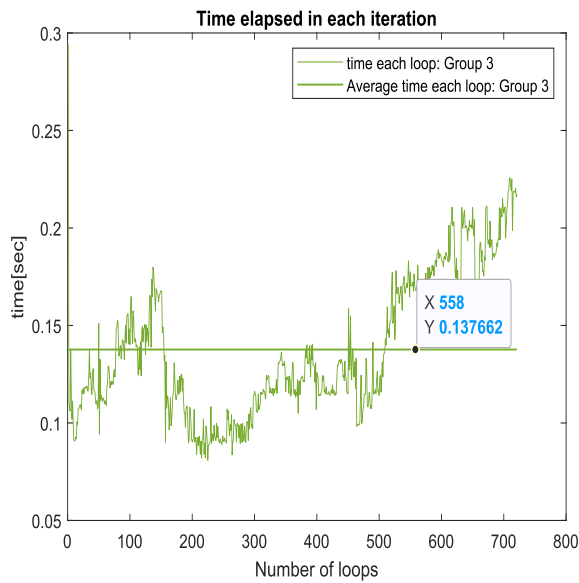


Figure 4-8: Type I OCP, Case II timing diagram; the results obtained within an average loop run time of 0.13s with Group 3 tuning parameters.

4.2 Type II OCP formulation based on levels maximization

The amount of water available in the reservoir plays a crucial role in power production; the abundance of water is the opportunity for the hydropower operation. Therefore, it is desirable to have sufficient water in lake Toke. Therefore, the objective is defined in such a way that the level of Merkebekk is maximized; minus sign has been assigned to it as given in Equation 4-7. Furthermore, as compared to the formulation in Section 4.1, an additional constraint given by Equation 4-9 is added in the formulation to limit the level within x_{LRV} and x_{HRV} .

$$\min_u J(h, u) = \sum_{k=1}^{N_p} -w_R x_M^2 + \Delta u' w_{\Delta u} \Delta u + u' w_u u \quad 4-7$$

subject to,

$$h_u(k+1) = f(h_u(k), u(k), d(k)) \quad 4-8$$

$$x_M, x_D \in [x_{LRV}, x_{HRV}] \quad 4-9$$

$$u \in [0, u^{max}] \quad 4-10$$

Case I: Water levels initialized below the narrow reference boundaries

Although the influence of the weighting groups on level is almost the same, the type of the gate opening signal is distinct as shown in Figure 4-9. To be specific, the control signal with group 1 weighting is noisier as compared to the other control signals. It has unusual peaks when opening and closing the gates as shown at time 425 hours and 450 hours. Furthermore, the gate opening signals tuned with group 2 and group 3 are almost similar. However, the gate openings with group 4 start early but open less. In this case also, there is not a strong point to select among group's 2,3, and 4. However, if group 3 is chosen as the best performing weighting set, its effect on the level and flow through gates can be seen in Figure 4-10.

With this tuning, the average loop run time taken to compute the optimal gate opening is shown in Figure 4-11 which is around 0.20 seconds.

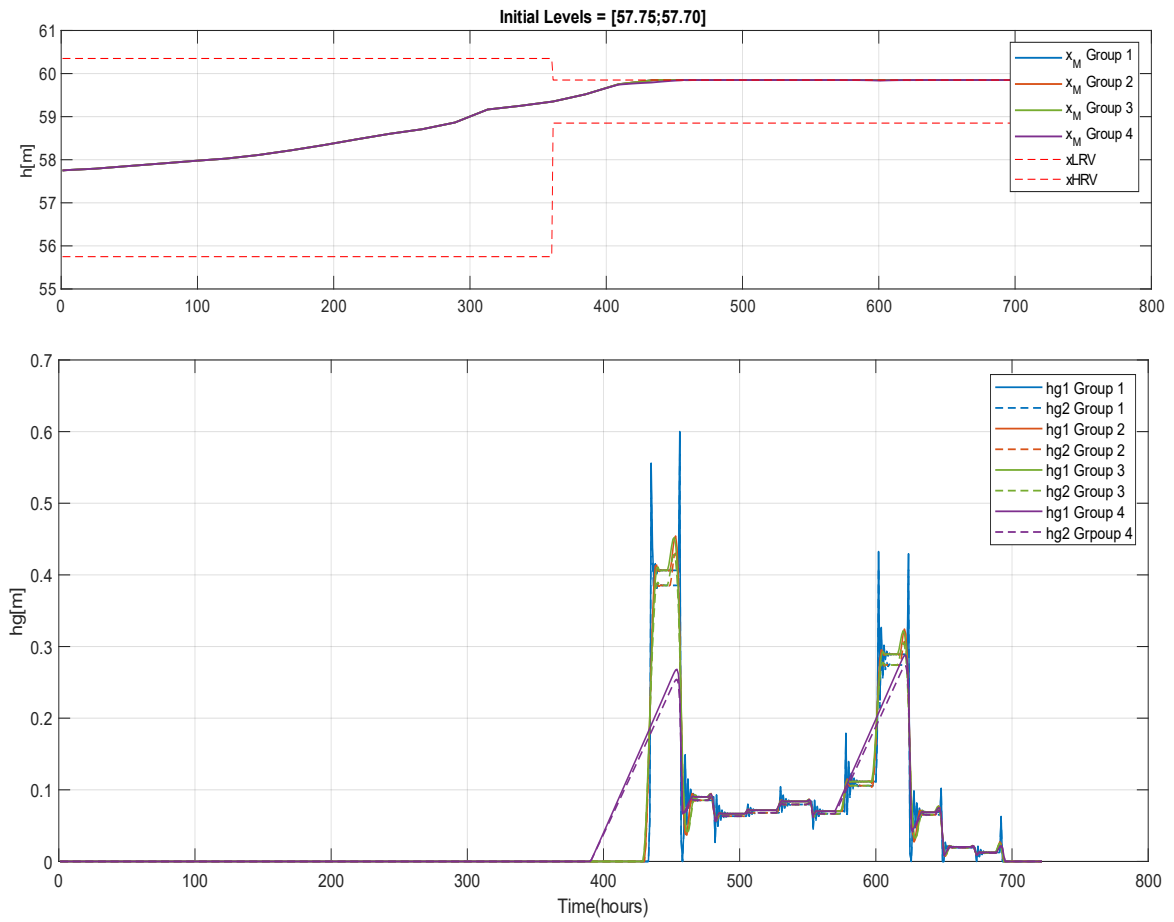


Figure 4-9: Type II OCP, initially water at level lower region of the reservoir- [57.75;57.70], turbine operating in full capacity – $V_t = 36 \text{ m}^3/\text{s}$, flood coefficient = 3, results comparison using all groups of parameters

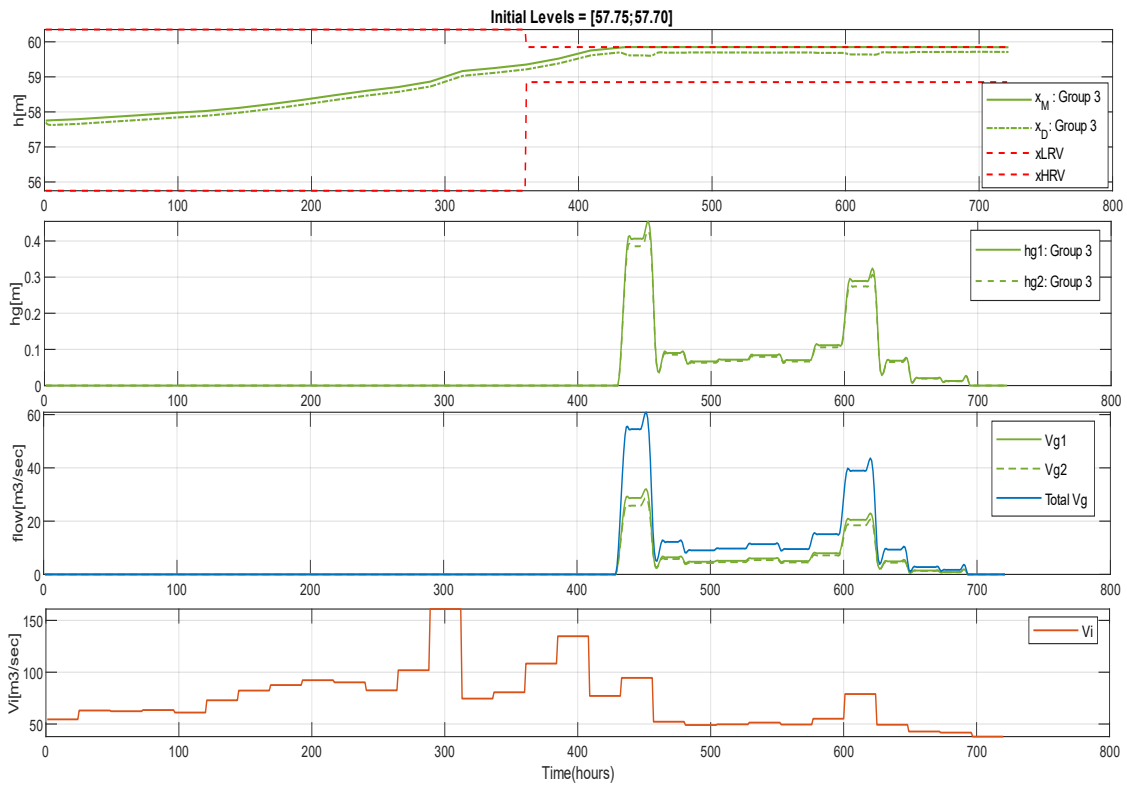


Figure 4-10: Type II OCP, initially water at the lower region of the reservoir - [57.75;57.70], turbine operating in full capacity- $V_t = 36 \text{ m}^3/\text{s}$, flood coefficient = 3, Group 3 tuning parameters

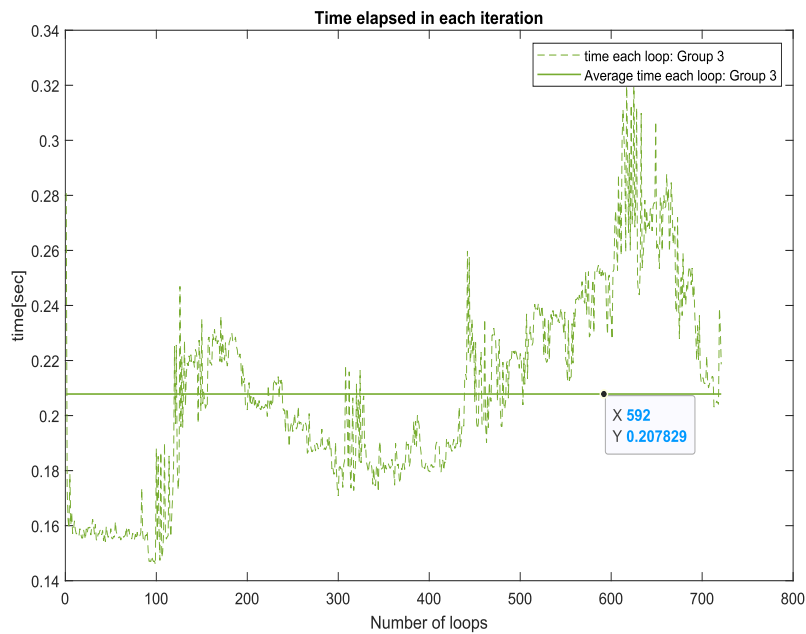


Figure 4-11: Type II OCP, Case I timing diagram; the results obtained within average loop run time of 0.20s with Group 3 tuning parameters.

Case 2: Water levels initialized inside the narrow reference boundaries

The effect of group 4 tuning, in this case, is not desirable which can be seen in Figure 4-12. The gate opening signal starts early and slowly goes on increasing. Furthermore, the performance of this signal in terms of level maximization is not good as compared to the other groups.

Group 1's performance is quite similar to group's 2 and 3 in terms of level maximization, however, the gate opening itself reaches maximum opening with slight distortion. Therefore, it can be deduced that group 2 and group 3 are performing optimally to meet the desired objective. If group 3 is chosen optimally performing, the variations in the levels, gate openings and the flow through the gates are clearly shown in Figure 4-13. Similarly, the time taken to solve the optimization problem in each time step and the average time is shown in Figure 4-14.

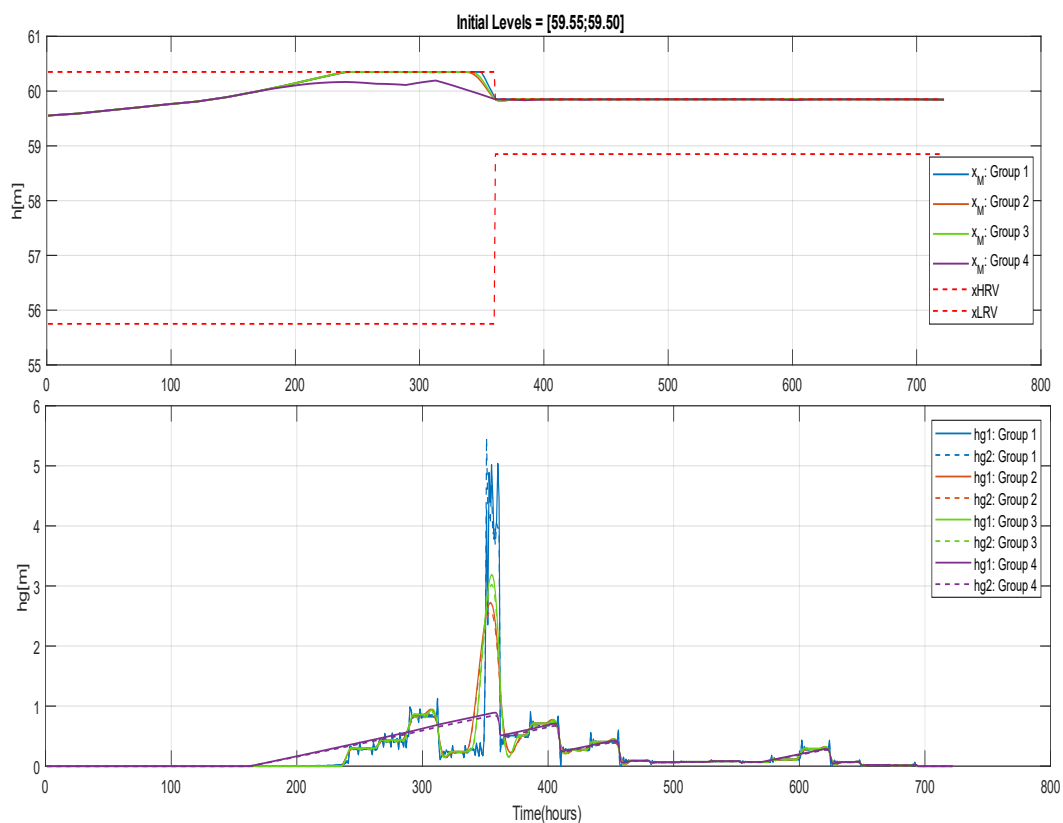


Figure 4-12: Type II OCP, initially water level at higher region of the reservoir - $[59.55; 59.50]$, turbine operating in full capacity - $V_t = 36 \text{ m}^3/\text{s}$, flood coefficient = 3, results comparison using all groups of parameters

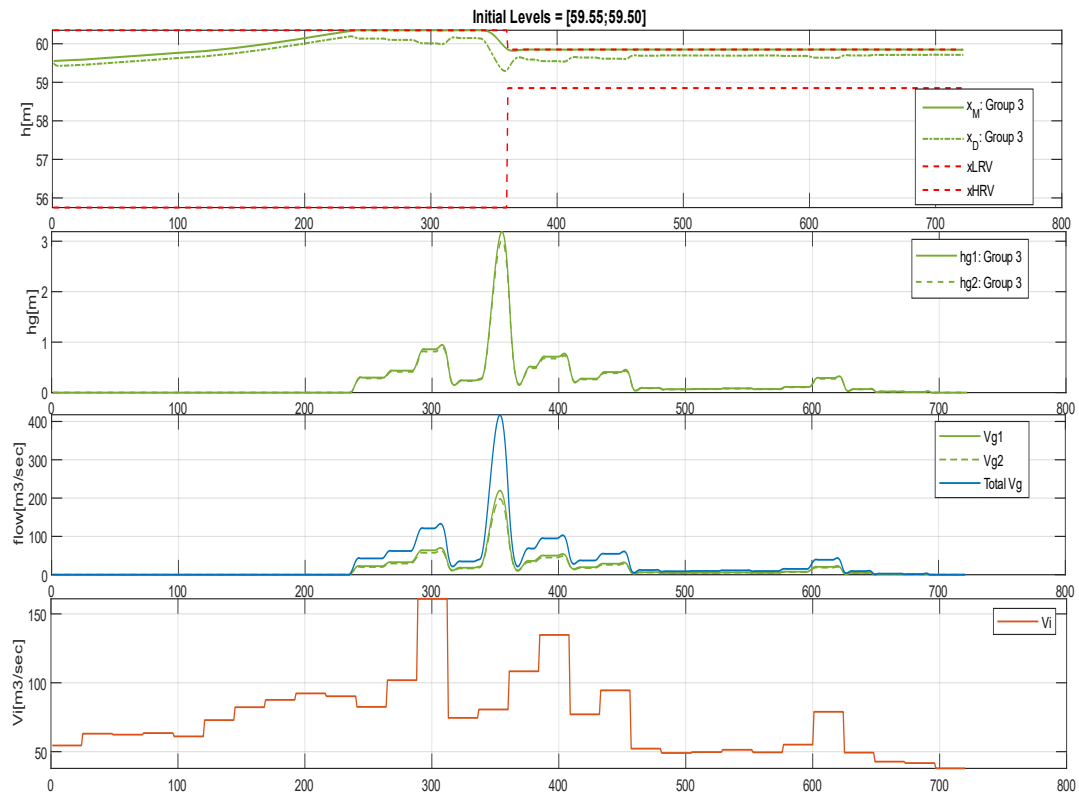


Figure 4-13: Type II OCP, initially water at the upper region of the reservoir - [59.55;59.50], turbine operating in full capacity- $V_t = 36$ m³/s, flood coefficient = 3, Group 3 tuning parameters

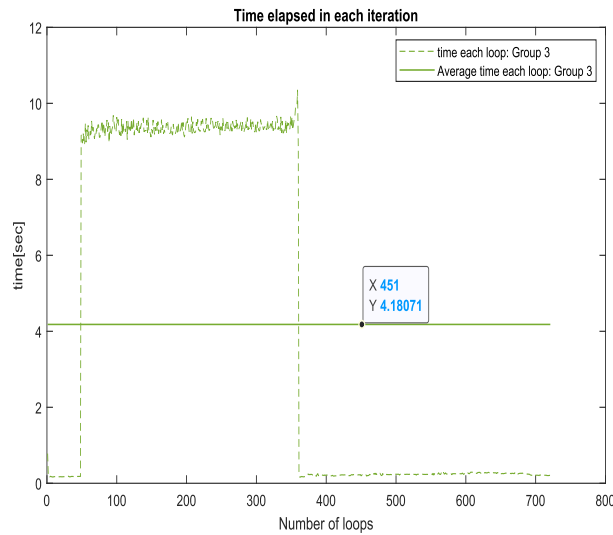


Figure 4-14: Type II OCP, Case II timing diagram; the results obtained within average loop run time of 4.18s with Group 3 tuning parameters.

4.3 Comparison of best results

In Section 4.1 and Section 4.2, two different objective functions were formulated and tuned using groups of parameters considering two separate cases for each objective. Although the tuning parameters can take any value to give better results, here the best results are selected only from the 4 sets of grouping parameters as given in Table 4-2. In each case, the best results will be compared obtained from these two objectives.

4.3.1 Results comparison based on Case 1

When the initial levels are lower in the reservoir, the controller appears to behave similarly in terms of level control except for very small differences from 400 hours. The nature of the gate opening, on the other hand, is entirely different. The gate opening signals obtained from Type I objective begin gate opening early, at time around 380 hours, but maintain it low. The gate opening signals obtained from Type II objective, on the other hand, keep the gate closed for longer, up to around 440 hours, but the opening is greater, as illustrated in Figure 4-15.

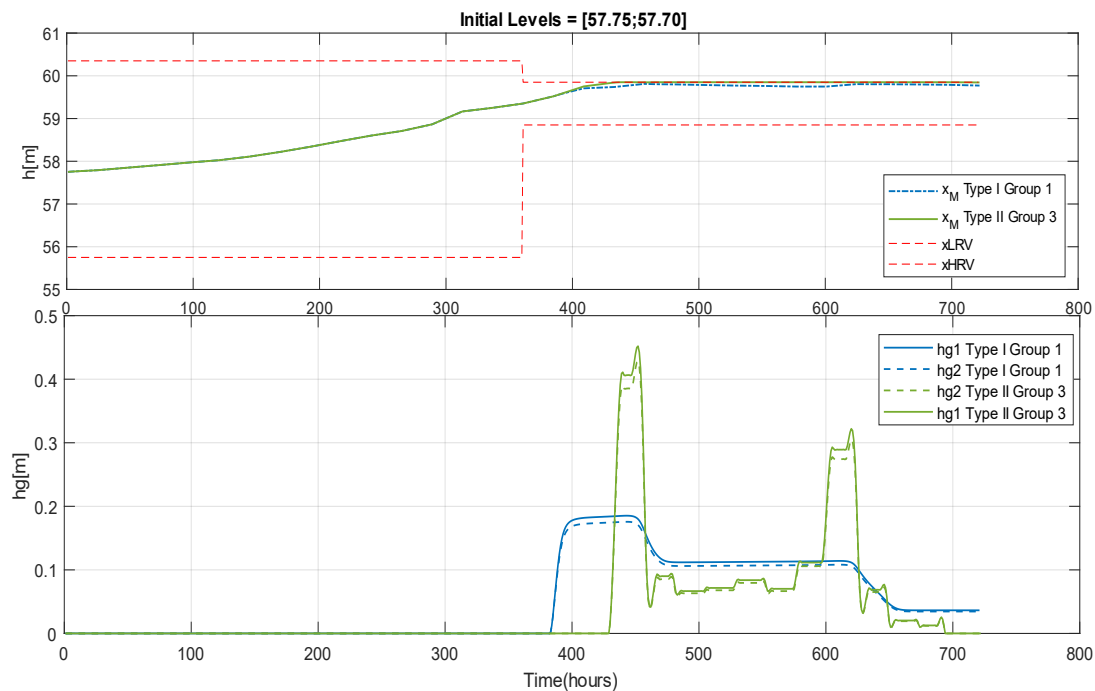


Figure 4-15: Type I vs Type II OCP formulation, comparison of level change and control signal using best performing weighting factor, initial level in the lower region

4.3.2 Results comparison based on Case 2

The effect on results due to two different objective functions is clear when the initial levels are in the upper region of the reservoir. Type II objective clearly gives superior results both in terms of levels maximization and the gate opening signal as shown in Figure 4-16. The level of Merkebekk is maximized up to the maximum allowed level as compared to the result given by Type I. Furthermore, Type I gate opening signal is arduous for the operator to inject into the real system because the gate opening signal should be adjusted every day over the whole month. Type II gate opening signal, on the other hand, is closed for a longer period and has step opening for the majority of the month except for a steep opening at around 350 hours.

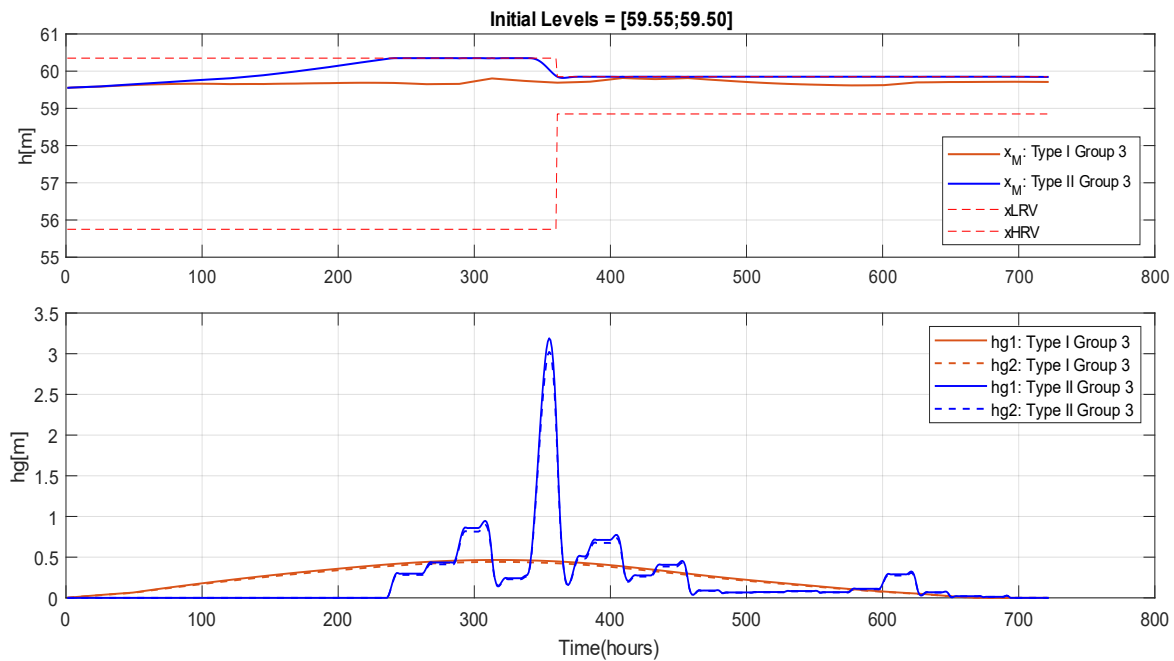


Figure 4-16: Type I vs Type II OCP formulation, comparison of level change and control signal using best performing weighting factor, initial level in the upper region

4.4 Discussion on deterministic MPC

4.4.1 Discussion on weighting parameters

The results obtained using different groups of weighting parameters suggest that the performance of a MPC depends upon how well the weighting parameters are tuned. With Type I OCP formulation, the effect is even more noticeable. To be specific, an improperly tuned controller can even result in overflowing in the reservoir as shown in Figure 4-6. On the other hand, with Type II OCP formulation, because of explicit constraint on the level as given in Equation 4-9, the effect of improperly tuned parameters causing flood is comparably less likely, however, can take more computational time. This signifies the importance of the proper weighting factor that is assigned to the controller. Therefore, it is crucial to monitor the changes in the weighting factors if conditions of operation change.

4.4.2 Discussion on OCP formulation

As already mentioned, the term reference region, R , evaluated in Equation 4-6 in Type I OCP formulation becomes zeros, when the level of Merkebekk is inside r^l and r^u . This signifies that the effect of level itself becomes insignificant when the objective function is evaluated. Furthermore, this formulation introduces fictitious boundaries r^l and r^u introducing complexity, although the permissible boundaries are already defined by x_{HRV} and x_{LRV} .

On the other hand, Type II OCP formulation uses x_{HRV} and x_{LRV} as the upper and lower level boundaries by eliminating r^l and r^u from Type I OCP formulation. Moreover, an additional output constraint, as given in Equation 4-9, is introduced which mostly ensures level constraints except in the worst scenario of flooding. This formation not only simplifies the problem formulation, performs better in terms of holding maximum allowed level in the reservoir, as shown in Figure 4-16, ensuring optimal operation of the hydropower plant.

5 Stochastic MPC

The results obtained from Section 4.3.2 suggested that the objective function defined in Type II deterministic OCP gives better results in terms of maintaining the higher water level in the reservoir. Therefore, the same OCP formulation has been chosen for SMPC and solved using WSA. This section details the WSA implementation for SMPC as well as the findings.

5.1 WSA for SMPC

The WSA-based SMPC tries to satisfy the total objective, given by Equation 5-1, obtained from all the different inflow ensembles. Specifically, the individual ensemble give rise to separate objective function denoted by J_n in Equation 5-1, and subject to the constraints as given in Equations 4-8, 4-9, and 4-10.

$$J_{total} = w_1 * J_1 + w_2 * J_2 + \dots + w_n * J_n \quad 5-1$$

where, w_1, w_2, \dots, w_n are the weights assigned to each ensemble. However, the ensembles here are not specifically prioritized, which means, any ensemble is likely to occur, therefore are given equal weights. The total objective obtained from Equation 5-1 is a scalar value, therefore called the scalarization method, which is passed to MPC along with the constraints, initial states, and the disturbance acting in the system to evaluate the optimal gate opening which ultimately is used to the process as shown in Figure 5-1.

In this thesis, MPC runs with a sampling time of 1 hour, however, new inflow disturbances acting in the system are updated every 24 hours. Therefore, over the period of 24 hours, the ensembles acting on the previous time are adjusted slightly and then utilized in the current time step. The adjustment is such that the stochastic ensembles acting in the previous time step are chopped and the last column data from the inflow matrix, as given in Equation 4-1, is repeated.

The overall implementation of the algorithm to design stochastic MPC using the WSA method is shown in Figure 5-2. The algorithm contains two loops while evaluating the total objective, one is used to run through each ensemble and the other calculates objective and constraints over the whole prediction horizon. The realization inflow acting in the process is assumed to be the same as the ensemble used in deterministic MPC which is shown in Figure 4-1.

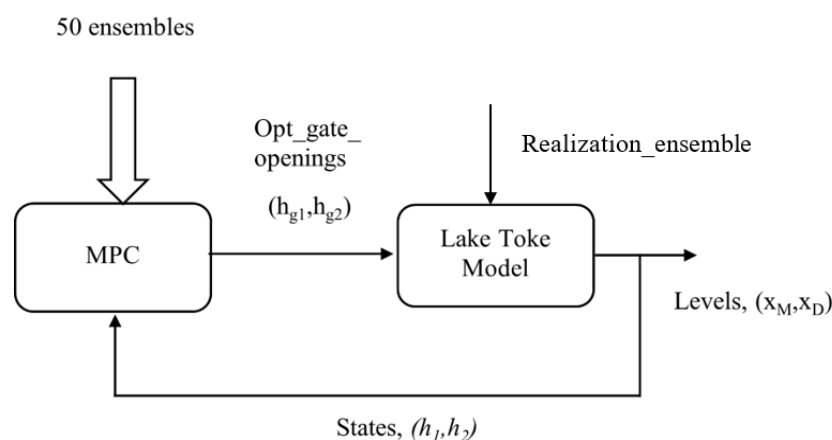


Figure 5-1: A framework showing the connection between the MPC and process

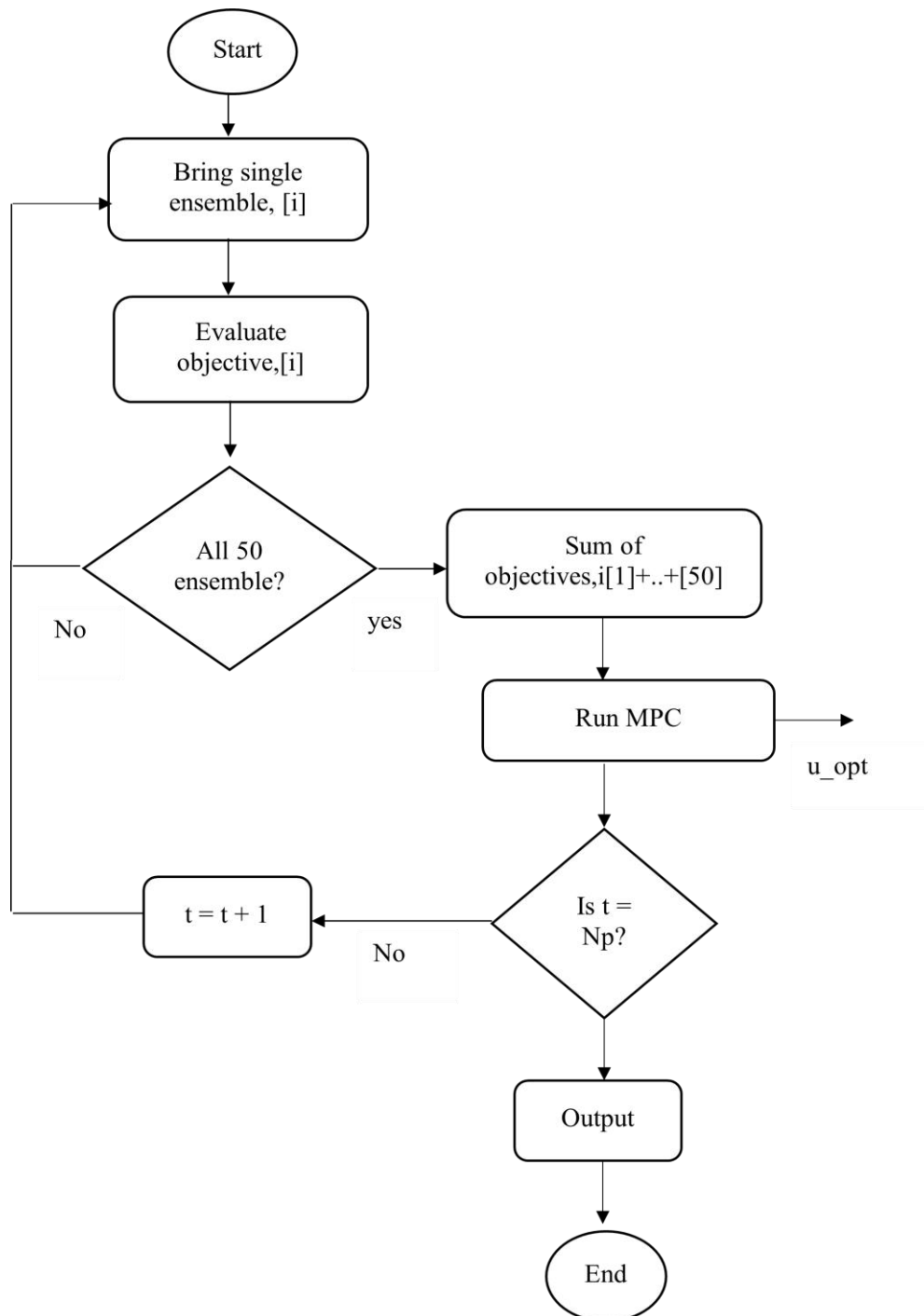


Figure 5-2: Flowchart showing implementation of SMPC using WSA method

5.2 Performance testing of SMPC at low turbine intake

The MOO SMPC's performance has been evaluated using different numbers of ensembles and data sets from two distinct years, 2020 and 2021. The results presented in this section consider low turbine inflow of $10\text{m}^3/\text{s}$ and flood coefficient 1. Each scenario assumed in the simulation is described below with corresponding results.

- a. The first ensemble is selected from 50 different ensembles predicted each day from the year 2020 and repeated to make 10 same ensembles. Although the same ensembles acting on the system do not represent the stochastic nature, this case has been assumed for the integrity test of the MPC algorithm which is developed to handle stochastic ensembles.

Figure 5-3 shows that the levels are maximized until the upper-level constraint, x_{HRV} , is about to hit, then the gate opening signals become active to meet constraint satisfaction. This result testifies the implementation of the WSA algorithm.

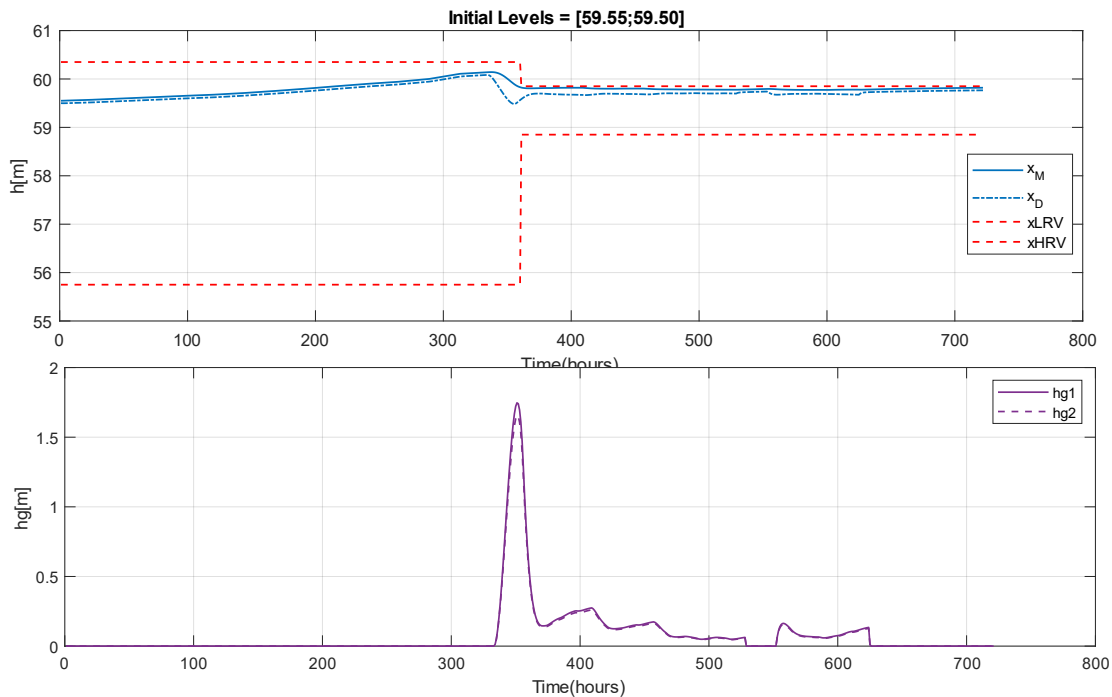


Figure 5-3: Levels and gate openings change using 10 same ensembles in MOO formulation

- b. However, instead of 10 same ensembles, 10 different ensembles from the year 2020 are used in the MPC algorithm. The change in the levels and control signals are shown in Figure 5-4 and Figure 5-5 respectively. With this scenario, the MPC's performance appears anomalous as the gate opening signal changes abruptly within 200 hours, despite the levels not reaching the maximum boundaries. Furthermore, between 350 and 500 hours, the upper-level restriction has been violated.

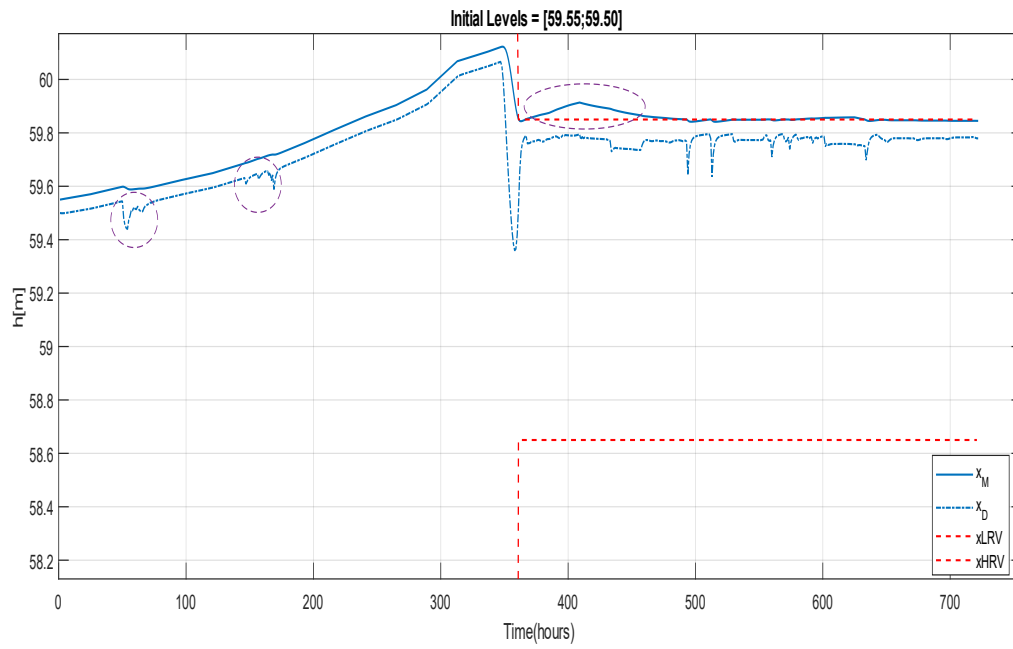


Figure 5-4: Changes in levels with 10 ensembles data taken from year 2020, level constraints violation around 400 hours

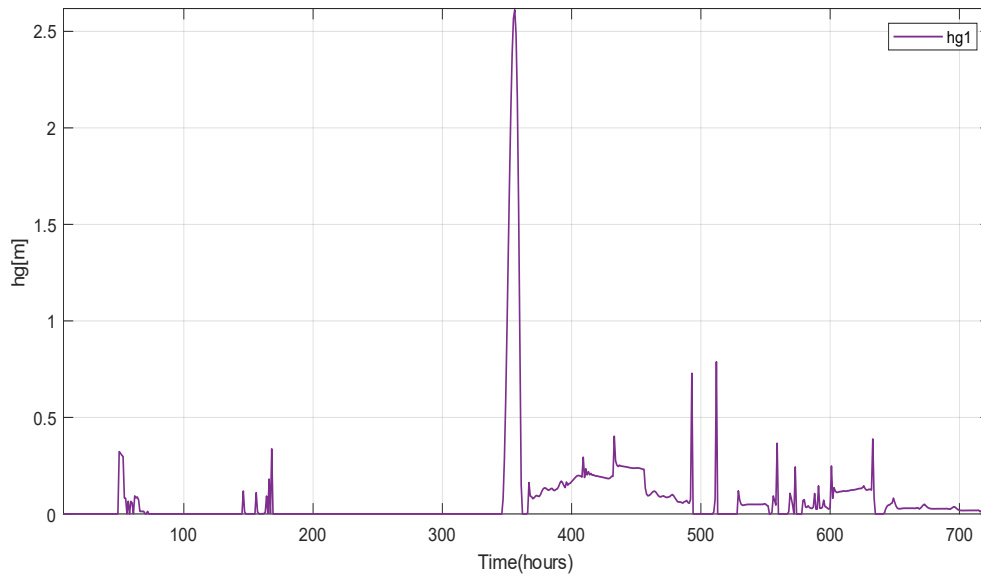


Figure 5-5: Changes in one control signal with 10 ensembles data taken from year 2020

- c. In attempt to observe the performance of the MPC with different settings, the ensembles data is taken from a different year, year 2021, ensembles spread is restricted to the maximum limit of $100 \text{ m}^3/\text{s}$ as shown in Figure 5-6, and the weightings are adjusted differently as prediction horizon marches forward as shown in Figure 5-7.

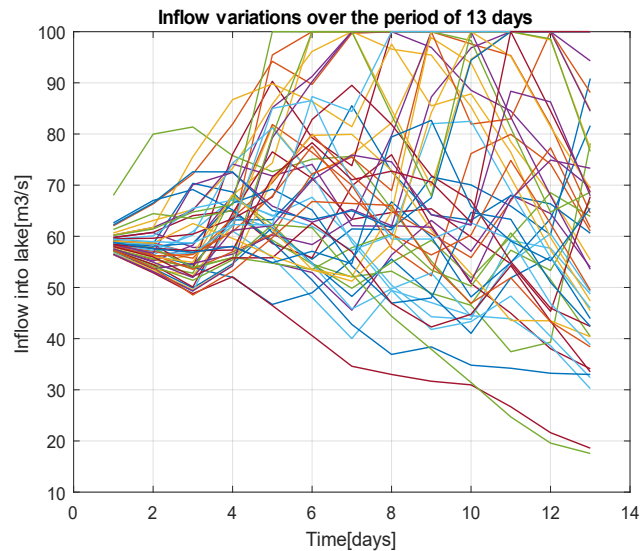


Figure 5-6: Example of chopped data to limit the maximum inflow to $100 \text{ m}^3/\text{s}$, data obtained on May 5, 2021,

```
%Weighting matrices for MPC
w_R = repelem([1,1,1,1,1,1,0.9,0.9,0.8,0.8,0.7,0.7,0.7],24); %Weight given to the level of
Merkebekk
w_delU = repelem([100,100,100,100,100,100,1000,1000,1000,2000,2000,2000,2000],24); %Weight on
change of flood gate opening
w_U = repelem([0.1,0.1,0.1,0.5,0.5,0.5,0.95,0.85,0.75,0.75,0.75,0.75,0.75],24); % Weight on
flood gate opening
```

Figure 5-7: Weightings are set differently as time marches forward in the prediction horizon

All the 50 ensembles are fed to stochastic MPC with the settings mentioned above. The result of levels variations is shown in Figure 5-8. It can be seen that the levels are within the defined level constraints, x_{HRV} and x_{LRV} , with slightly noticeable variation on Dalsfoss level from the period of 370 hours to 437 hours as shown in Figure 5-9. The maximum variation of Dalsfoss is 0.27 m within 67 hours. It signifies that it takes 2.7 days to balance the water dynamics once the constraints are around the narrow region.

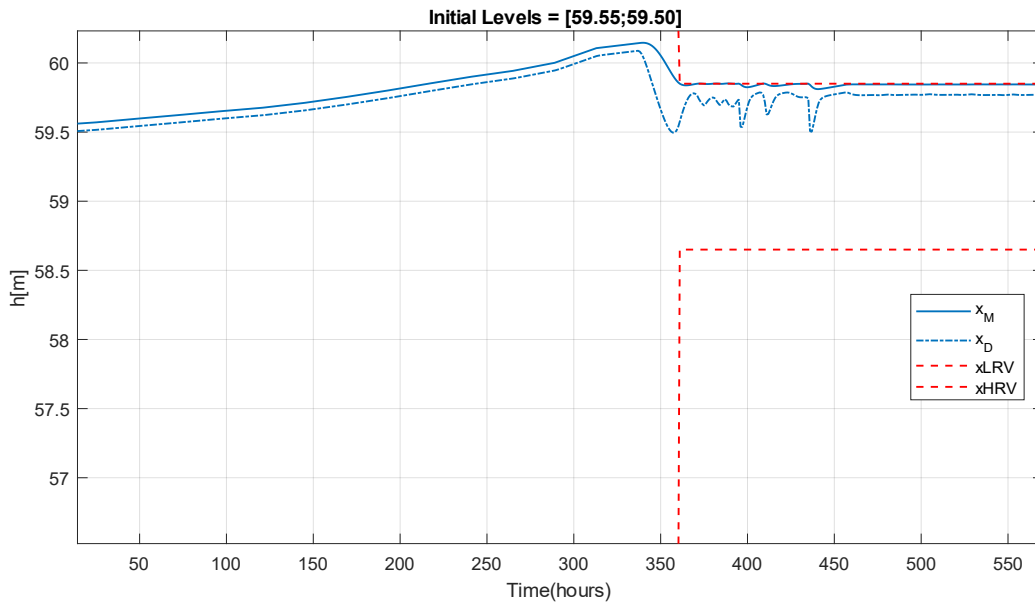


Figure 5-8: Level variations due to 50 ensembles with maximum inflow limit to 100 m³/s, ensemble data taken from year 2021

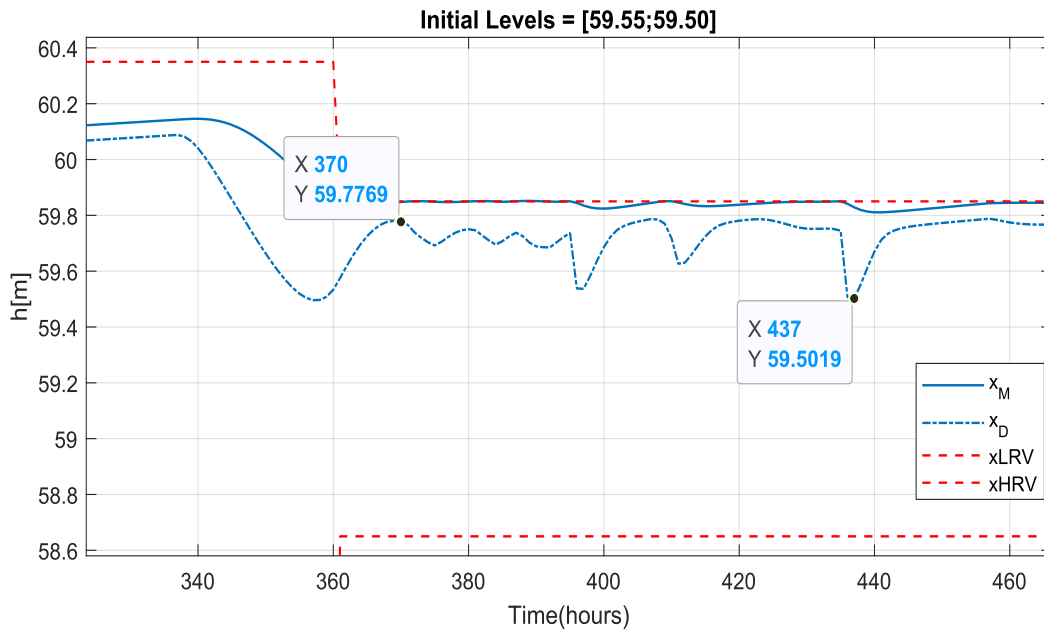


Figure 5-9: A close look on levels variation from 340 hours to 470 hours

The gate openings are smooth most of the time except with some steep openings as shown in Figure 5-10. A closer look at Figure 5-10 is shown in Figure 5-11.

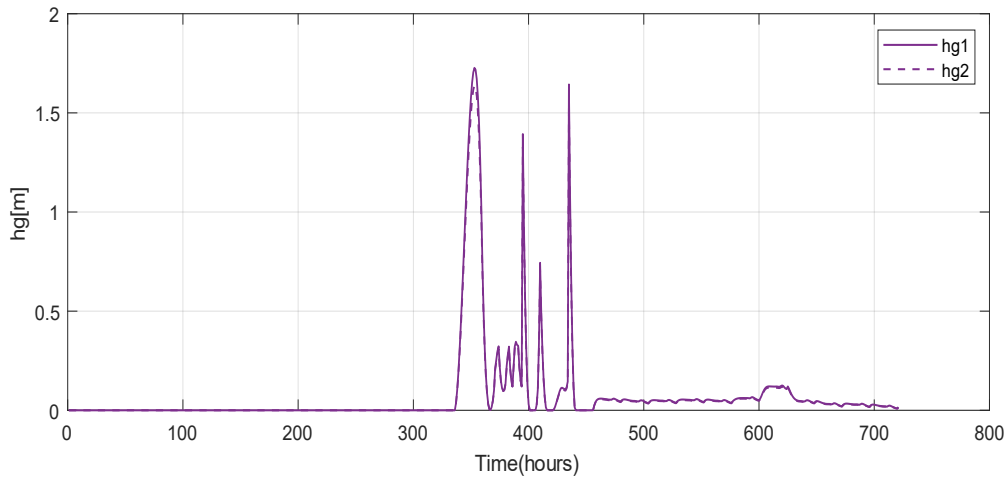


Figure 5-10: Changes in gate opening signal with 50 ensembles, data taken from the year 2021, inflow limited to 100 m³ /s

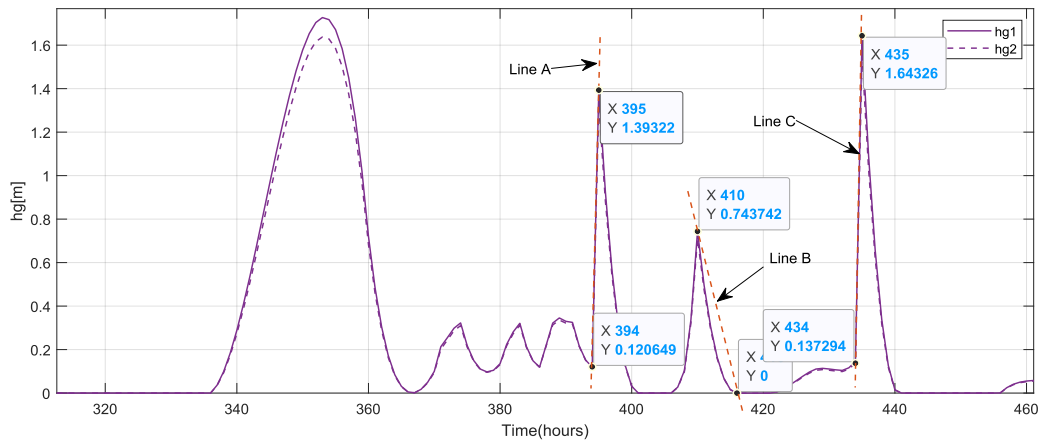


Figure 5-11: Changes in gate opening signal with 50 ensembles, Figure 5-10 view from time 300 hours to 460 hours

The steep gate openings are marked with lines A, B, and C in Figure 5-11, and their slope is given as:

$$\text{Slope of Line A} = \frac{1.39 - 0.12}{395 - 394} = 1.27 \text{ m/hour} \quad 5-2$$

$$\text{Slope of Line B} = \frac{0 - 0.74}{418 - 410} = -0.09 \text{ m/hour} \quad 5-3$$

$$\text{Slope of Line C} = \frac{1.64 - 0.13}{435 - 434} = 1.51 \text{ m/hour} \quad 5-4$$

The steepest openings have occurred at 394 hour and 434 hours with slope given as 1.27 m/hour and 1.51 m/hour respectively. These slopes suggest opening 1.27 m at 394 hours and 1.51 m at 434 hours respectively to meet the level constraints.

The average time taken to compute optimal gate openings hourly is around 24.55 seconds which is shown in Figure 5-12.

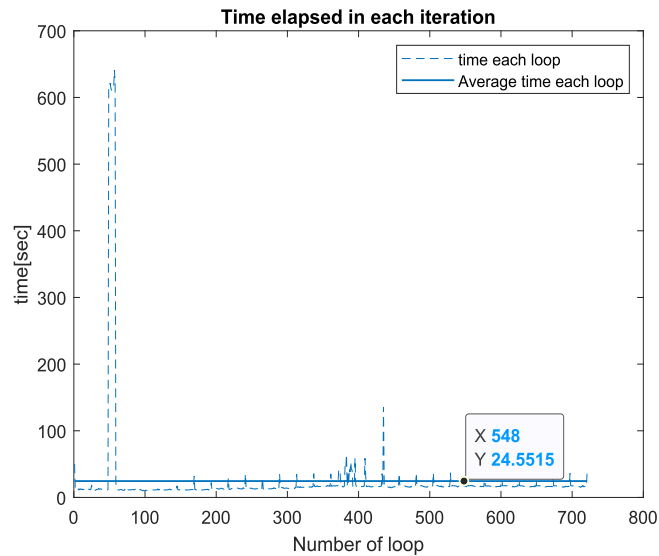


Figure 5-12: Average loop run time considering 50 ensembles and low turbine inflow

- d. The SMPC's performance is further examined by reducing the number of prediction horizon to six days. The prediction horizon is reduced but the ensembles are kept intact without limiting to any value. The corresponding results in terms of level variations and gate opening signal are shown in Figure 5-13. From this figure, it is seen that the levels are maximized up to the upper constraint limit with very less and smooth variations. Furthermore, the gate openings are smooth, and step-change throughout the simulation period. Note that the results plotted in Figure 5-13 are zoomed in to visualize the variations.

However, reducing prediction horizon disregards the possible variabilities of the ensembles in the future. That is, ensembles spread are taken only up to the six days in this case, although the provided ensembles spread are up to 13 days. As a result, while the stochastic MPC performs better, the results may not be substantial for Dalfoss operation.

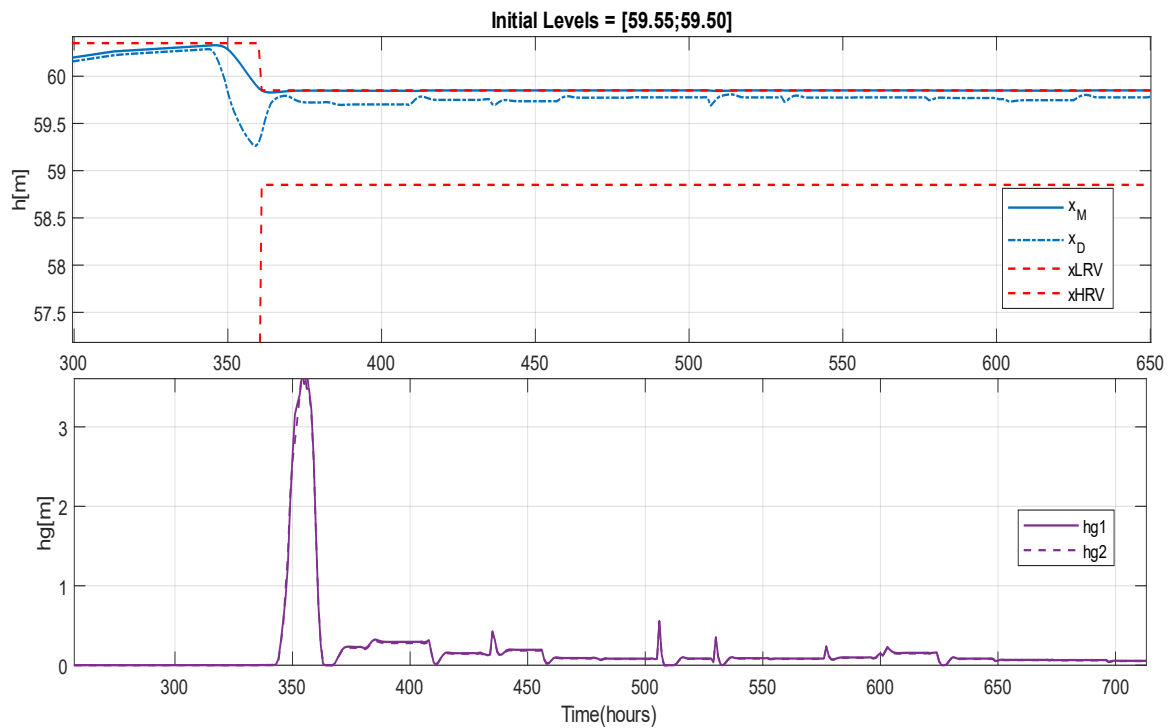


Figure 5-13: Level variations and gate openings change with prediction horizon reduced to 6 days, but ensembles without limiting the inflow

5.3 Performance testing of SMPC at maximum turbine intake

The maximum inflow data limited to $100 \text{ m}^3/\text{s}$ from the year 2021 is used to test the performance of the MPC considering the full operation of the turbine; $36 \text{ m}^3/\text{s}$ inflow to the turbine. From Figure 5-14 it is seen that the levels are within the constraints, therefore MPC can meet the level requirements. Merkebekk level is maximum throughout the simulation period, except from 400 to 500 hours, where it seems to drop by a little margin.

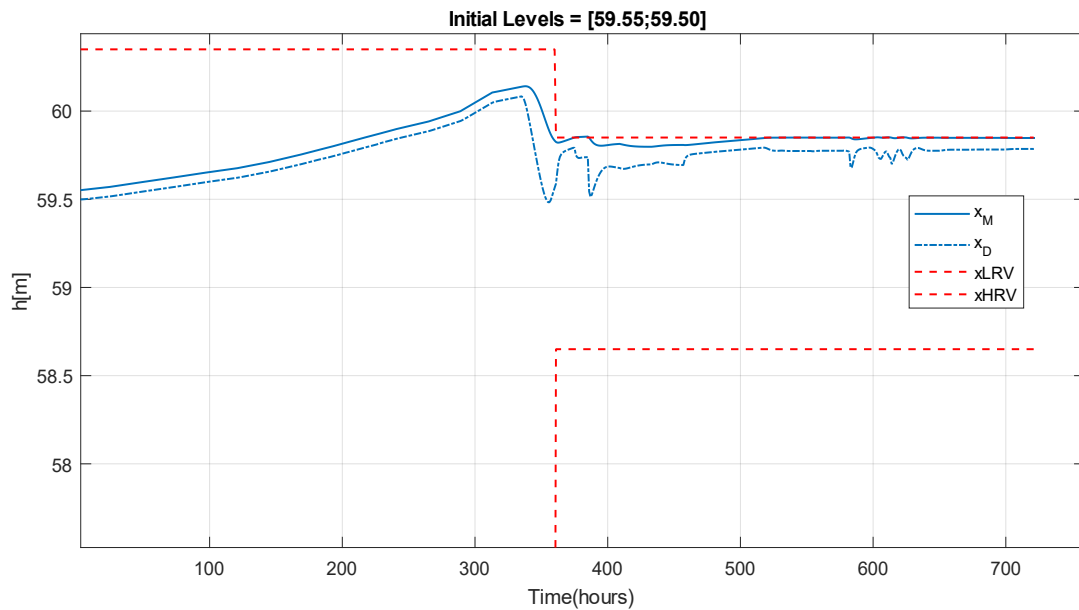


Figure 5-14: Level variations, flood coefficient 1, maximum inflow from year 2021 limited to 100 m³/s

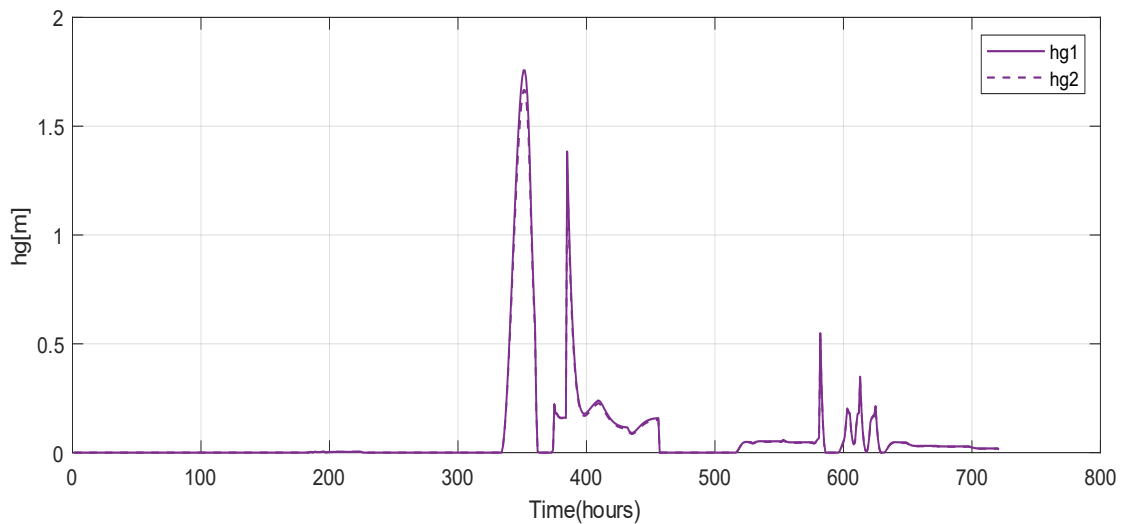


Figure 5-15: Gate openings variations, maximum inflow from the year 2021 limited to 100 m³/s

The corresponding optimal gate openings evaluated are shown in Figure 5-15. When the level constraints become narrow, gate opening signals vary frequently with some steep suggestions. To see the range of variation closely, Figure 5-16 is narrower down from 350 hours to 650 hours as shown in Figure 5-16. It is seen that at a time around 350 hours and 580 hours, optimal gate openings are steep.

The time taken to evaluate the gate opening signals in each time step is around 22 seconds as shown in Figure 5-17.

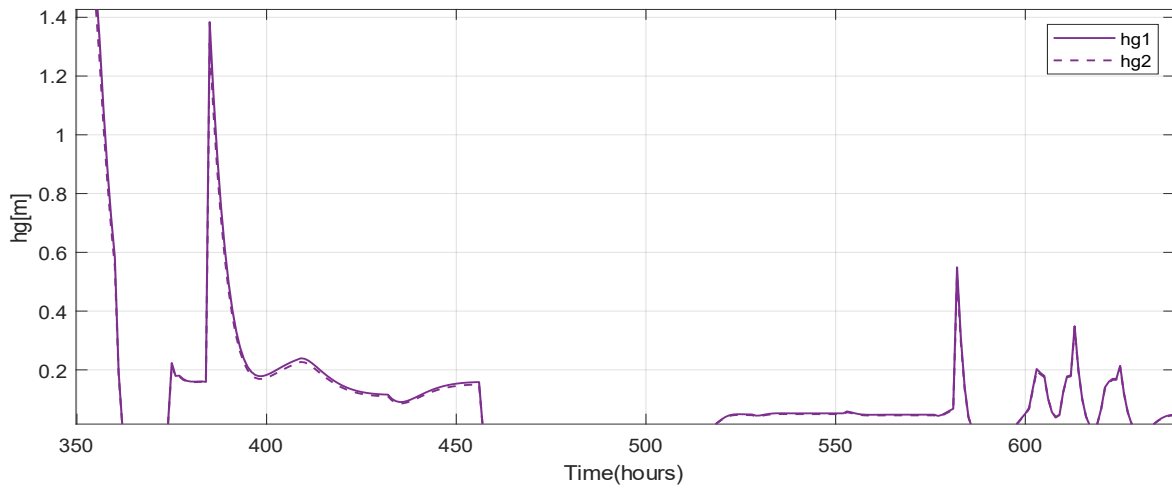


Figure 5-16: Gate opening variations, Figure 5-15 viewed from 350 hours to 650 hours.

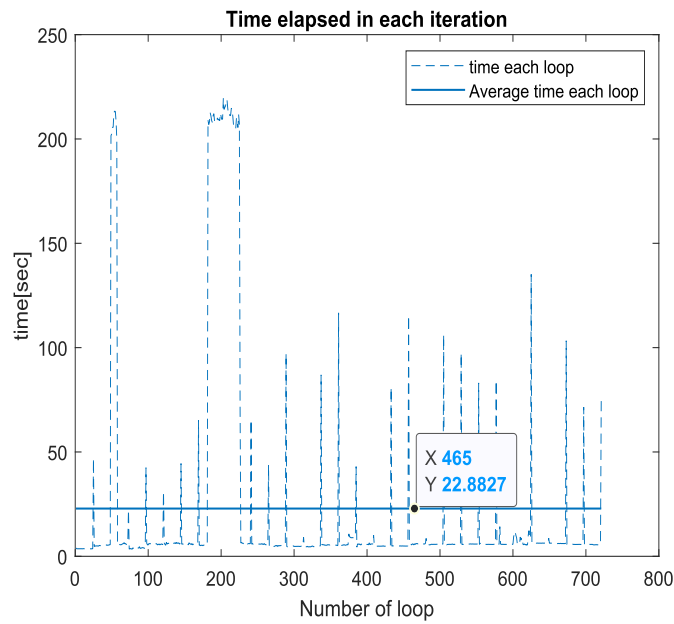


Figure 5-17: Average loop run time considering 50 ensembles and high turbine inflow

5.4 Discussion on stochastic MPC

5.4.1 Trajectories of inflow ensembles

The WSA method evaluates optimal gate openings considering the spread of every ensemble and giving them equal weights. However, among the provided 50 ensembles each day, some of the ensembles have very large spread as compared to most of the ensembles as shown in Figure 5-18. The ensemble numbered 1 has reached up to 150 m³/s, the ensembles numbered 2 and 3 are over 100 m³/s, however, the majority of the ensembles have spread under 82 m³/s. This might be one of the reasons stochastic MPC performed strangely when the original data set was fed; however, when the ensemble spread was limited to 100 m³/s, the performance dramatically improved.

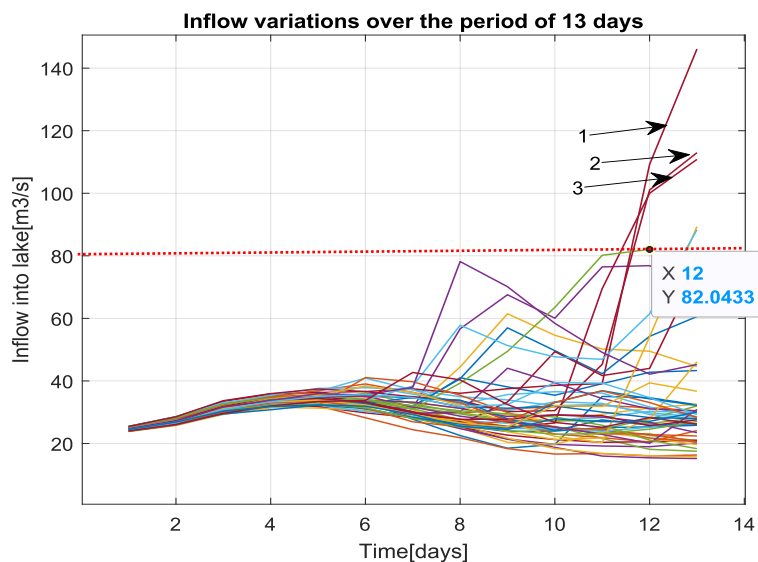


Figure 5-18: Ensembles numbered 1,2 and 3 are different than majority of the ensembles, data recorded on 20 April 2020

5.4.2 Weight adjustment

The same weight given to each weighting parameter over the whole prediction horizon might not be sufficient to tune the controller to get better results. As time moves forward, the stochastic behavior of the ensembles increases, therefore the tuning is needed in such a way to counter this phenomenon. For example, with reference to Figure 5-18, the same weighting might work up to 6 days of prediction length, however, after 6 days the ensembles change drastically, therefore the weights assigned should be different after 6 days to have less effect of these changes while the optimization process runs. One example of such a setting is already shown in Figure 5-7. The performance of stochastic MPC can be enhanced by assigning appropriate weighting factors over the prediction horizon, however, finding the ideal values is challenging.

5.4.3 Computational load

The ensemble spread given to the optimizer, the MPC weighting factors, and most crucially, the numerical approach utilized for MPC all have a significant influence on the computation time required to evaluate the optimal gate openings. The author in [4] used single shooting method to solve stochastic MPC with fictitious sinusoidal ensembles and it took, on average, 20 minutes to solve each iteration. If the same simulation is performed for a whole month, it seems to take a couple of days. In this thesis, multiple shooting has been used with ensembles restricted to a maximum value of $100 \text{ m}^3/\text{s}$, and it took around 23 seconds in each iteration. With this average timing, it takes a couple of hours to complete the simulations. Moreover, when the original ensembles data provided by Skagerak are used for MPC, even with multiple shooting, it takes minutes to solve each iteration, consequently days to simulate for a month. In such a scenario, tuning the controller with proper weighting values and re-running the simulation is extremely time-consuming. This is also one of the major reasons that imposed difficulty in this thesis to find better solutions using stochastic MPC. The availability of a high-performance CPU might have helped to have a detailed analysis of stochastic MPC, however conventional CPU unit used to perform simulation in this thesis could not meet the computational load.

6 Conclusion

6.1 Conclusion

It is crucial to operate the flood gate openings of Dalsfoss power plant considering the possibilities of several inflows into lake Toke. Furthermore, it is desirable to hold water in the lake Toke up to the maximum allowed limit as it ensures the optimal operation of the hydropower plant. The solution to this description can be fulfilled with the design of the stochastic MPC.

The objective function that was used to design SMPC was determined based on the necessity of maintaining the reservoir's maximum level while fulfilling the level constraints. The objective function for SMPC was chosen after the comparison of results obtained from two different OCP formulations: a) Type I formulation with reference region tracking b) Type II formulation with level maximization. The later formulation not only meets the criteria of level maximization, but also simplifies the problem formulation by eliminating the artificial level constraints inherent in the former version.

The MOO based WSA SMPC was formulated, which determines the optimal gate openings based on inflow ensemble uncertainties. The SMPC's performance assessment started using just 10 ensembles with full spreading and low turbine intake. The results obtained did not fulfill the objective of level maximization and constraint satisfaction. That is, the optimal gate opening signals were active even though the levels were not maximized. However, when the ensemble's spreading was limited to 100 m³/s, and the weightings parameters were adjusted differently over the prediction horizon, the performance met the requirements of level maximization and constraints satisfaction. SMPC also performed satisfactorily with these adjustments while considering the optimal operation of the hydropower. The performance of the SMPC was improved when the length of the prediction horizon was reduced, although this solution does not consider the ensemble spreading up to 13 days.

Furthermore, the computational burden that comes with stochastic simulation limited the task of detail analysis. With full ensembles spreading, it takes couple of minutes just to run a single iteration, resulting days to run SMPC with simulation period of 1 month. However, the computational time was reduced from minutes to seconds when ensembles spreading was limited to 100 m³/s.

Therefore, it can be inferred that examining the ensembles before using them, and re-running SMPC with modified weighting parameters can produce better results even with using full ensembles spreading.

6.2 Future Work

Based on the experience of the author, the following future work are worth on implementing.

- Revisit the stochastic MPC framework used in this thesis by minutely studying implementation details, assigning appropriate values to tuning parameters, and inspecting ensemble variations
- Although the multiple shooting technique is considered as an efficient approach to solve the problem, considering the code optimization technique would help to solve the problem faster
- Validate the integrity of the designed stochastic MPC by testing the performance on different ensemble data set with different inflow trajectories. Thereafter, study the robustness and conservativeness of the stochastic MPC.

References

- [1] G. Schildbach, "Scenario-based optimization for multi-stage stochastic decision problems," Doctor of Sciences, Automatic control laboratory, Darmstadt University of Technology, 2014.
- [2] D. Schwanenberg, F. M. Fan, S. Naumann, J. I. Kuwajima, R. A. Montero, and A. Assis dos Reis, "Short-Term Reservoir Optimization for Flood Mitigation under Meteorological and Hydrological Forecast Uncertainty," *Water Resources Management*, vol. 29, no. 5, pp. 1635-1651, 2015, doi: 10.1007/s11269-014-0899-1.
- [3] B. Lie, "KONTRAKT NR INAN-140122 Optimal Control of Dalsfos Flood Gates - control algorithm," 2014.
- [4] I. Menchacatorre, R. Sharma, B. Furenes, and B. Lie, "Flood Management of Lake Toke: MPC Operation under Uncertainty," presented at the Proceedings of The 60th SIMS Conference on Simulation and Modelling SIMS 2019, August 12-16, Västerås, Sweden, 2019.
- [5] B. F. Changhun Jeong, Roshan Sharma, "MPC Operation with Improved Optimal Control Problem at Dalsfoss Power Plant," 2021, doi: 10.3384/ecp21185226.
- [6] D. Q. Mayne, "Model predictive control: Recent developments and future promise," *Automatica*, vol. 50, no. 12, pp. 2967-2986, 2014, doi: 10.1016/j.automatica.2014.10.128.
- [7] S. Lucia and S. Engell, "Multi-stage and Two-stage Robust Nonlinear Model Predictive Control," *IFAC Proceedings Volumes*, vol. 45, no. 17, pp. 181-186, 2012, doi: 10.3182/20120823-5-nl-3013.00015.
- [8] S. Thangavel, R. Paulen, and S. Engell, "Robust Multi-Stage Nonlinear Model Predictive Control Using Sigma Points," *Processes*, vol. 8, no. 7, 2020, doi: 10.3390/pr8070851.
- [9] S. Lucia, "Robust Multi-stage Nonlinear Model Predictive Control," Technischen Universitat Dortmund, Dortmund, Phd thesis 2015.
- [10] S. Peitz and M. Dellnitz, "A Survey of Recent Trends in Multiobjective Optimal Control—Surrogate Models, Feedback Control and Objective Reduction," *Mathematical and Computational Applications*, vol. 23, no. 2, 2018, doi: 10.3390/mca23020030.
- [11] R. Sharma, "Lectures note for course IIA," University of South Eastern Norway, 2020.
- [12] A. Gambier, *MPC and PID control based on Multi-Objective Optimization*. 2008, pp. 4727-4732.
- [13] D. M. Raimondo, D. Limon, M. Lazar, L. Magni, and E. Camacho, "Min-max Model Predictive Control of Nonlinear Systems: A Unifying Overview on Stability," *European Journal of Control*, vol. 15, 12/31 2009, doi: 10.3166/ejc.15.5-21.
- [14] K. Jayamanne, "Optimal operation of processes under uncertainty using robust Model Predictive Control," Master's thesis report, Faculty of Technology, Natural Sciences and Maritime Sciences, University of South Eastern Norway, Porsgrunn, 2021.
- [15] J. H. Lee and Z. Yu, "Worst-case formulations of model predictive control for systems with bounded parameters," *Automatica*, vol. 33, no. 5, pp. 763-781, 1997/05/01/ 1997, doi: [https://doi.org/10.1016/S0005-1098\(96\)00255-5](https://doi.org/10.1016/S0005-1098(96)00255-5).

- [16] C. J. Goh and K. L. Teo, "Control parametrization: A unified approach to optimal control problems with general constraints," *Automatica*, vol. 24, no. 1, pp. 3-18, 1988/01/01/1988, doi: [https://doi.org/10.1016/0005-1098\(88\)90003-9](https://doi.org/10.1016/0005-1098(88)90003-9).
- [17] C. Hargraves and S. Paris, "Direct Trajectory Optimization Using Nonlinear Programming and Collocation," *AIAA J. Guidance*, vol. 10, pp. 338-342, 07/01 1987, doi: 10.2514/3.20223.
- [18] S. L. Kek, M. Aziz, K. Teo, and R. Ahmad, "An iterative algorithm based on model-reality differences for discrete-time nonlinear stochastic optimal control problems," *Numerical Algebra, Control and Optimization (NACO)*, vol. 3, pp. 109-125, 03/01 2013, doi: 10.3934/naco.2013.3.109.
- [19] S.-L. Kek, "Nonlinear programming approach for optimal control problems " presented at the The 2nd international conference on global optimization and its application, Malaysia, 2013.
- [20] J. Andersson, "A General-Purpose Software Framework for Dynamic Optimization (Een algemene softwareomgeving voor dynamische optimalisatie)," 2013.
- [21] B. Passenberg, "Theory and algorithms for indirect methods in optimal control of hybrid systems," Technische Universitat Munchen, PhD thesis 2012.
- [22] O. von Stryk and R. Bulirsch, "Direct and indirect methods for trajectory optimization," *Annals of Operations Research*, vol. 37, no. 1, pp. 357-373, 1992/12/01 1992, doi: 10.1007/BF02071065.
- [23] C. Durazzi and E. Galligani, "Nonlinear Programming Methods for Solving Optimal Control Problems," in *Equilibrium Problems: Nonsmooth Optimization and Variational Inequality Models*, F. Giannessi, A. Maugeri, and P. M. Pardalos Eds. Boston, MA: Springer US, 2001, pp. 71-99.
- [24] M. Diehl, H. J. Ferreau, and N. Haverbeke, "Efficient Numerical Methods for Nonlinear MPC and Moving Horizon Estimation," *Lecture Notes in Control and Information Sciences*, pp. 391-417, 2009.
- [25] T. A. Johansen, "Chapter 1 Introduction to Nonlinear Model Predictive Control and Moving Horizon Estimation," 2011.
- [26] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi: a software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, vol. 11, no. 1, pp. 1-36, 2018, doi: 10.1007/s12532-018-0139-4.

Appendices

Appendix A – Thesis task description

FMH606 Master's Thesis

Title: Stochastic MPC for optimal operation of hydropower plant

USN supervisors: Main supervisor: Roshan Sharma; co-supervisor: Changhun Jeong

External partner: Skagerak Energi AS

Task background:

Dalsfoss hydropower plant is located at Kragerø. In this hydropower plant, there are strict requirements to maintain the water level at the dam between prescribed lower and upper levels. This would have been relatively easier to do if the amount of water flowing into Lake Toke (which is the reservoir) was exactly known. In reality, inflow of water into the lake are calculated using hydrological models, weather forecast etc. and 50 different possible inflows to the lake are predicted every day. The inflow predictions are done for the next 13 days.

Thus the task of controlling the water level cannot be straight forward solved using deterministic MPC. For this we should develop a stochastic MPC which is capable of using all the 50 inflow prediction scenarios and which is also robust in nature. Robust in a sense that no matter which (out of the possible 50) inflow occurs into the system, the MPC should still be able to satisfy the water level concession requirements.

Aim:

The aim of this thesis is to take into account the uncertainties present in a process/plant and develop an MPC that can handle it. An example is the robust MPC or stochastic MPC. In particular, it is of interest to look into non-conservative robust/stochastic MPC. The developed robust MPC should be applied to the Dalsfoss hydropower case study. The mathematical model of the case study (together with full process description) will be provided to the student by the supervisor.

Task description:

The following are the main tasks:

- (i) Detailed literature review on different methods for Robust/Stochastic MPC.
- (ii) Evaluate the possible objective functions for stochastic MPC.
- (iii) Develop stochastic MPC for optimal operation of the Dalsfoss hydropower plant. Real historical data of inflow ensembles will be provided by Skagerak Energy AS.
- (iv) Study about conservativeness and robustness of the developed MPC through detailed simulations.
- (v) Document the work in a report. The report should be technically sound. Presentation of the work.


Student category: IIA and EPE student

The tasks requires that the student is able to program in MATLAB and is interested to learn about advanced control.

Is the task suitable for online students (not present at the campus)? Yes

Practical arrangements: N/A

Signatures:

Supervisor (date and signature):  17.01.2022

Student name (write clearly in all capitalized letters): NABIN K C

Student (date and signature):  17.01.2022

Appendix B – MATLAB files

A list of all the MATLAB script and function files used to produce the results for this thesis is given in Table A.1.

Table A.1. List of MATLAB files

Types	File name	Description
Open loop simulation	Simulator_casadi.m	Model simulation
Deterministic, Level Maximization objective	main_file.m	contains parameters setting, function calling and main loop of MPC simulation
	compute_both.m	evaluate objectives and constraints over the prediction horizon
Deterministic, Reference region tracking objective	compute_both.m	evaluate objectives and constraints
Stochastic MPC	main_file.m	contains parameters setting, function calling and main loop of MPC simulation
	compute_both.m	evaluate objectives and constraints over the prediction horizon
Common files	select_disturbance.m	create deterministic realization inflow
	state_models.m	contains model equations of the process
	update_states.m	Use first input into the process, sliding horizon implementation

main_file.m (deterministic: level maximization)

```

clc
clear
% close all
addpath('C:\aaa._MY_FILES\AUSN\4th semester\casadi-windows-matlabR2016a-v3.5.5');
import casadi.*
%%.....Parameters settings::Start.....%%
%->....Gate opening requirements
hg_min = 0; % [m], minimum gate opening
hg_max = 5.6;%[m], maximum gate opening

%->....Time setting for simulations
sampling_time = 3600;% [s],Sampling time of 1 hour
sim_time = 30*24*3600;% [s],Simulation time of 30 days
timespan = 0:sampling_time:sim_time;
timesteps = length(timespan);

%->....Length of the prediction horizon
Np = 13*24; % 13 days of prediction horizon

%->....Level constraints from April 15 to May 15
x_LRV1 = 55.75; %[m] from april 15 to april 30
x_LRV2 = 58.85; % [m] from may 1st to may 15
x_HRV1 = 60.35; %[m] from april 15 to april 30
x_HRV2 = 59.85; % [m] from may 1st to may 15

%->....Creating array for constraints limit and also needed for plotting
xLRV = zeros(timesteps,1);
xLRV(1:timesteps/2) = x_LRV1;
xLRV(timesteps/2:end) = x_LRV2;

xHRV = zeros(timesteps,1);
xHRV(1:timesteps/2) = x_HRV1;
xHRV(timesteps/2:end) = x_HRV2;

%->....Add extra number of values for simulating until the end of...
%->simulation time
xLRV = [xLRV;ones(Np+1,1).*x_LRV2];
xHRV = [xHRV;ones(Np+1,1).*x_HRV2];

%->....Finding minimum value of xLRV, added to states to find x_M and x_D
x_LRV_min = min(xLRV); %[m]

%->.... Nominal values for MPC cost function parameters
x_R = 0.75; % Location of r_l_i in interval [xLRV,xHRV]
delta_HRV = 0.05; %[m] Safety margin for upper reference region boundary
%Selection of reference region boundaries

```

```

r1 = (1-X_R).*xLRV + X_R.* xHRV;
ru = xHRV - delta_HRV;

%-->.... Nominal values for MPC cost function parameters
X_R = 0.75; % Location of r1_i in interval [xLRV,xHRV]
delta_HRV = 0.05; %[m] Safety margin for upper reference region boundary

%Weight matrices
w_R = 10; %weight on cost outside reference region
w_delU = 1; %weight on change of flood gate opening
w_U = 0.1; % weight on flood gate opening

% Vectors for storing volumetric flow through gates
vgate_each = zeros(timesteps,2);
vgate_total = zeros(timesteps,1);

%%.....Parameters settings::End.....%%

%%.....Symbolic representation for CasADi::Start.....%%

%-->....States representation,two states h1 and h2
h = SX.sym('h',2,1);
st = h ;
n_states = length(st);

%-->....Control action representation,hg1 and hg2
hg = SX.sym('hg',2,1);
controls = hg;
n_controls = length(controls);
n_u_prev = n_controls;

%-->....Process disturbance, volumetric inflow
vi_dot = SX.sym('vi_dot');
n_disturbance = length(vi_dot);

%-->....Process model, rhs term and the output from the models
[hdot,Vg_dot,Vg_dot_total] = State_models(st,controls,vi_dot);

%-->.... Non Linear mapping function; inputs to outputs
f = Function('f',{st,controls,vi_dot},{hdot,Vg_dot,Vg_dot_total});

%-->....Decision variables(control input) and states vector for the whole
%-->prediction horizon
U = SX.sym('U',n_controls,Np ); % Control inputs matrix
H = SX.sym('H',n_states,(Np + 1)); % States matrix

%-->.... Parameter vector to store initial states and disturbance
P = SX.sym('P',n_states + n_u_prev + Np,1); % 2 initial states and Np number of
%-->disturbance for the whole prediction horiozon
%%.....Symbolic representation for CasADi::END.....%%
%%.....Evaluation of objective and constraint::Start.....%%
%-->....Compute objective and constraints over the prediction horizon
[obj,g] = compute_both(U,H,P,Np,sampling_time,x_LRV_min,w_R,w_delU,w_U,f);
%%.....Evaluation of objective and constraint::END.....%%
%%.....IPOPT optimizer setup::Start.....%%
%-->....Define decision/optimization variables

```

```

OPT_variables = [reshape(H,2*(Np+1),1);reshape(U,2*Np,1)];

%->...Define NLP problem object
nlp_prob = struct('f',obj,'x',OPT_variables,'g',g,'p',P);

%->...Options set up for optimizer
opts = struct;
opts.ipopt.max_iter = 1000;
opts.print_time = 0;
opts.ipopt.acceptable_tol = 1e-6;
opts.ipopt.acceptable_obj_change_tol = 1e-6;
opts.ipopt.print_level = 0; % 3,5(default) upto 13
opts.ipopt.fixed_variable_treatment = 'make_constraint';
opts.ipopt.warm_start_init_point = 'yes';

solver = nlpso1('solver','ipopt',nlp_prob,opts);

%%.....IPOPT optimizer setup::END.....%%
%%.....Bounds setup::Start.....%%
args = struct;
%->...equality constraint matrix g(equality constraint) and delta_u

args.lbg = repelem(0,2*(Np+1));
args.ubg = repelem(0,2*(Np+1));

%->...Bounds on gate opening, U
args.lbx(2*(Np+1)+1:1:2*(Np+1)+ 2*Np,1) = 0; % Lower limit of gate opening
args.ubx(2*(Np+1)+1:1:2*(Np+1)+ 2*Np,1) = hg_max;
%Upper bound of gate opening
%%.....Bound setup::Partial END.....%%
%%.....Partial END because the bounds on states are changed in each
%%time step, therefore the bounds are supplied fromt the main loop.
%%.....The simulation loop starts from here.....%%
%-----
%-----

%->... Initialization of control input over the whole prediction horizon
initial_control = [0;0];
U0 = repmat(initial_control,1,Np)';

%->...Initialization of states over the whole prediction horizon
%->...Give initial states between 0 and 4.6, this limit is because it will
%help to bound x_M and x_D inbetween x_LRV and x_HRV
initial_states = [3.8;3.75];
H0 = repmat(initial_states,1,Np+1)';

u_prev = [0;0];
%->...state_history matrix contains the evolution of states over the whole
%simulation time, these states values are obtained from the first optimized
%control input in each time step
state_history(:,1) = initial_states;

%->...u_eachstep contains the changes in control signal over the whole
%simulation time period, these values are first optimal move stored in each
%time step
u_eachstep = [initial_control];

```

```

%->.... Create a large vector storing the value for the disturbance over
%the simulation period
disturbance = select_disturbance();
%-> ...Repeat each element of inflow disturbance(Here,24 times), because
%preceion of inflow from the company are provided after 24 hours, i.e next
%day
generate_disturbance = repelem(disturbance,24).*1;

%->.... Start MPC with with initialization of iterations count
no_iterations = 0;
%->....For optimal solution trajectory starting each timestep
% optimal_traj = []; % Contains the optimal solution trajectory

%->....Start the main loop
for i=1:timesteps

    tic;
    %->....Parametrization vector,P and initial vector args.x0 are changed
    %in each timestep
    args.p = [initial_states;u_prev;generate_disturbance(i:i+Np-1,1)];
    args.x0 = [reshape(H0',2*(Np+1),1);reshape(U0',2*Np,1)];

    %->....Bounds on h1
    % x_M E [xLRV,xHRV]
    % xLRV <= x_M <= xHRV
    %xLRV <= h1+x_LRV_min <= xHRV
    %xLRV-x_LRV_min <= h1 <= xHRV-x_LRV_min

    args.lbx(1:2:2*(Np+1),1) = xLRV(i:i+Np,1) - x_LRV_min;
    args.ubx(1:2:2*(Np+1),1) = xHRV(i:i+Np,1) - x_LRV_min;

%    %->....Bounds on h2
    args.lbx(2:2:2*(Np+1),1) = xLRV(i:i+Np,1)-x_LRV_min;
    args.ubx(2:2:2*(Np+1),1) = xHRV(i:i+Np,1)-x_LRV_min;

    %->.... Call IPOPT solver object in each iterations
    sol = solver('x0',args.x0,'lbx',args.lbx,'ubx',args.ubx,'lbg',...
                args.lbg,'ubg',args.ubg,'p',args.p);

    %->....Extract the control signal which is after states variables in
    % the optimization variable
    u = reshape(full(sol.x(2*(Np+1)+1:end))',2,Np)';
    %Get OPTIMAL solution trajectory
    optimal_traj(:,1:2,no_iterations+1) = reshape(full(sol.x(1:2*(Np+1)))',2,Np+1)';

    %->....Append first move in eachstep
    u_eachstep = [u_eachstep,u(1,:)'];

    %->....Update the states by applying first control move
    [initial_states,U0,Vg_dot,Vg_total] = update_states(sampling_time,...
                initial_states,u,f,generate_disturbance(i));

    u_prev = u(1,:)';
    %->....Storing gate flow in each time step
    Vgate_each(no_iterations+1,:) = full(Vg_dot);

```

```

Vgate_total(no_iterations+1,:) = full(Vg_total);

% ->....Storing updated states in each time step
state_history(:,no_iterations+2) = initial_states;

% ->....Get solution trajectory
H0 = reshape(full(sol.x(1:2*(Np+1))),2,Np+1)';

% ->....Shift trajectory to initialize the next step
H0 = [H0(2:end,:);H0(end,:)];

% ->....Increase the number of iterations
no_iterations = no_iterations + 1

% ->....Record time of completion of each loop
time_eachloop(:,no_iterations) = toc;
end

% ->....Finding average loop completion time
Average_looprun_time = sum(time_eachloop)/timesteps
Total_iteration = no_iterations
total_time = Total_iteration*Average_looprun_time

% ->.... x holds the information both x_M and x_D, the levels we are
% interested on.
x = state_history + x_LRV_min;

% ->....Visualization function call for plotting all the results
plotting_results(x,u_eachstep,Vgate_each,Vgate_total,generate_disturbance,...
                x_LRV,x_HRV,timesteps,time_eachloop,Average_looprun_time,r1,ru)

```

compute_both.m (deterministic: level_maximization)

```

function [obj,g] = compute_both(U,H,P,Np,sampling_time,x_LRV_min,w_R,w_delU,w_U,f)

% ->....Objective function initialization
obj = 0;
% ->....Empty Constraints vector, later we will add the evaluated
% constraints in each step of prediction horizon through for loop
g = [];
% ->....st variable initialization with the optimal states stored on H
st = H(:,1); %Initial states
% ->....Initial condition constraints, the difference is appended in
% each iteration, the concept of multiple shooting adds this equality
% constraint
g = [g; st - P(1:2,1)];
% ->....Disturbance vector is assigned to P vector, these values stored
% in P vector are changed in each iteration fromt the main for loop
Vi_disturbance = P(5:end,1);
u_prev = P(3:4,1);
U_extended = [u_prev,U];
del_con = U_extended(:,2:end) - U_extended(:,1:end-1);

for k = 1:Np

```

```

st = H(:,k);
con = U(:,k);

inflow = vi_disturbance(k,1);

%->....level of Merkebekk to be maximized in the objective function
x_M1 = H(1,k) + x_LRV_min ;
obj = obj + (-w_R*x_M1^2 + del_con(:,k)'.*...
            w_de1U*del_con(:,k) + con'*w_U*con);
st_next = H(:,k+1);

% Integrating the models with explicit runge kutta method
k1 = f(st,con,inflow);
k2 = f(st+k1.*sampling_time/2,con,inflow);
k3 = f(st+k2.*sampling_time/2,con,inflow);
k4 = f(st+k3.*sampling_time,con,inflow);

st_next_predicted = st +sampling_time/6*(k1+2.*k2+2.*k3+k4);

g = [g;
     st_next - st_next_predicted % x(k+1) = f(x(k),u(k)) Equality constraints
     ];
end

end

```

compute_both.m (deterministic: reference region tracking)

```

function [obj,g] = compute_both(U,H,P,Np,sampling_time,...
                               w_R,w_de1U,w_U,f,n_states,n_u_prev,x_LRV_min)

%->....Objective function initialization
obj = 0;
%->....Empty Constraints vector, later we will add the evaluated
%constraints in each step of prediction horizon through for loop
g = [];

%->....st variable initialization with the optimal states stored on H
st = H(:,1); %Initial states

%->....Initial condition constraints, the difference is appended in
%each iteration, the concept of multiple shooting adds this equality
%constraint
g = [g; st - P(1:n_states)];

%->....Disturbance vector is assigned to P vector, these values stored
%in P vector are changed in each iteration fromt the main for loop

u_prev = P(n_states+1:n_states+2,1);

U_extended = [u_prev,U];

```

```

del_con = U_extended(:,2:end) - U_extended(:,1:end-1);

vi_disturbance = P(n_states + n_u_prev + 1:n_states + n_u_prev + Np,1);

r_ll = P(n_states + n_u_prev + Np + 1:n_states + n_u_prev + Np + Np,1);

r_uu = P(n_states + n_u_prev + Np + Np + 1:end,1);

for k = 1:Np

    st = H(:,k);
    con = U(:,k);

    inflow = vi_disturbance(k,1);

    R = min(H(1,k)+x_LRV_min-r_ll(k),0) + max(H(1,k)+x_LRV_min-r_uu(k),0);
    obj = obj + w_R*R^2 + del_con(:,k)'*...
            w_delU*del_con(:,k) + con'*w_U*con;

    st_next = H(:,k+1);

    % Integrating the models with explicit runge kutta method
    k1 = f(st,con,inflow);
    k2 = f(st+k1.*sampling_time/2,con,inflow);
    k3 = f(st+k2.*sampling_time/2,con,inflow);
    k4 = f(st+k3.*sampling_time,con,inflow);

    st_next_predicted = st +sampling_time/6*(k1+2.*k2+2.*k3+k4);

    g = [g;
        st_next - st_next_predicted % x(k+1) = f(x(k),u(k)) Equality constraints
        ];
end

end

```

main_file.m (stochastic MPC)

```

clc
clear
% close all

addpath('C:\aaa.\MY_FILES\AUSN\4th semester\casadi-windows-matlabR2016a-v3.5.5');
import casadi.*
%%.....Parameters settings::Start.....%%

%->....Gate opening requirements
hg_min = 0; % [m], minimum gate opening
hg_max = 5.6;%[m], maximum gate opening

%->....Time setting for simulations
sampling_time = 3600;% [s],Sampling time of 1 hour
sim_time = 30*24*3600;% [s],Simulation time of 30 days
timespan = 0:sampling_time:sim_time;

```



```

timesteps = length(timespan);

%->....Choose the number of ensemble for MPC
no_ensemble = 20;
%->....Length of the prediction horizon
N_days = 13; % Number of days for prediction horizon
Np = N_days*24; % number of steps in prediction horizon
%->....Level constraints from April 15 to May 15
x_LRV1 = 55.75; %[m] from april 15 to april 30
x_LRV2 = 58.85; % [m] from may 1st to may 15
x_HRV1 = 60.35; %[m] from april 15 to april 30
x_HRV2 = 59.85; % [m] from may 1st to may 15
%->....Creating array for constraints limit and also needed for plotting
xLRV = zeros(timesteps,1);
xLRV(1:timesteps/2) = x_LRV1;
xLRV(timesteps/2:end) = x_LRV2;

xHRV = zeros(timesteps,1);
xHRV(1:timesteps/2) = x_HRV1;
xHRV(timesteps/2:end) = x_HRV2;

%->....Add extra number of values for simulating until the end of...
%->simulation time
xLRV = [xLRV;ones(Np+1,1).*x_LRV2];
xHRV = [xHRV;ones(Np+1,1).*x_HRV2];

%->....Finding minimum value of xLRV, added to states to find x_M and x_D
x_LRV_min = min(xLRV); %[m]
%->.... Nominal values for MPC cost function parameters
X_R = 0.75; % Location of r_l_i in interval [xLRV,xHRV]
delta_HRV = 0.05; %[m] Safety margin for upper reference region boundary

%Selection of reference region boundaries
r_l = (1-X_R).*xLRV + X_R.* xHRV;
r_u = xHRV - delta_HRV;

%->.... Nominal values for MPC cost function parameters
X_R = 0.75; % Location of r_l_i in interval [xLRV,xHRV]
delta_HRV = 0.05; %[m] Safety margin for upper reference region boundary

%Weighting matrices for MPC
w_R = repmat([1,1,1,1,1,1,0.9,0.9,0.8,0.8,0.7,0.7],24); %weight given to the level of
Merkebekk
w_deU = repmat([100,100,100,100,100,100,1000,1000,1000,2000,2000,2000,2000],24); %weight on
change of flood gate opening
w_U = repmat([0.1,0.1,0.1,0.5,0.5,0.5,0.95,0.85,0.75,0.75,0.75,0.75,0.75],24); % weight on
flood gate opening
% Vectors for storing volumetric flow through gates
Vgate_each = zeros(timesteps,2);
Vgate_total = zeros(timesteps,1);
%%.....Parameters settings::End.....%%
%%.....Symbolic representation for CasADi::Start.....%%
%->....States representation,two states h1 and h2
h = Sx.sym('h',2,1);
st = h ;
n_states = length(st);

```

```

%->....Control action representation,hg1 and hg2
hg = SX.sym('hg',2,1);
controls = hg;
n_controls = length(controls);
n_u_prev = n_controls;

%->....Process disturbance, volumetric inflow
Vi_dot = SX.sym('vi_dot');
n_disturbance = length(Vi_dot);

%->....Process model, rhs term and the output from the models
[hdot,Vg_dot,Vg_dot_total] = State_models(st,controls,Vi_dot);

%->.... Non Linear mapping function; inputs to outputs
f = Function('f',{st,controls,Vi_dot},{hdot,Vg_dot,Vg_dot_total});

%->....Decision variables(control input) and states vector for the whole
%>prediction horizon
U = SX.sym('U',n_controls,Np); % Control inputs matrix
H = SX.sym('H',n_states,no_ensemble*(Np+1)); % States matrix

%->.... Parameter vector to store initial states and disturbance
P = SX.sym('P',n_states + n_u_prev + no_ensemble*Np,1); % 2 initial states
% , 2 previous control signals and (no_ensemble * Np) number of
% disturbance for the whole prediction horiozon
%%.....Symbolic representation for CasADi::END.....%%
%%.....Evaluation of objective and constraint::Start.....%%
%->....Compute objective and constraints over the prediction horizon
[obj,g] =
compute_both(U,H,P,Np,sampling_time,x_LRV_min,w_R,w_deU,w_U,f,n_states,n_u_prev,no_ensemble)
;
%%.....Evaluation of objective and constraint::END.....%%

%%.....IPOPT optimizer setup::Start.....%%

%->....Define decision/optimization variables
OPT_variables = [reshape(H,no_ensemble*2*(Np+1),1);reshape(U,2*Np,1)];

%->....Define NLP problem object
nlp_prob = struct('f',obj,'x',OPT_variables,'g',g,'p',P);

%->....Options set up for optimizer
opts = struct;
opts.ipopt.max_iter = 800;
opts.print_time = 0;
opts.ipopt.acceptable_tol = 1e-5;
opts.ipopt.acceptable_obj_change_tol = 1e-5;
opts.ipopt.print_level = 0; % 3,5(default) upto 13
opts.ipopt.fixed_variable_treatment = 'make_constraint';
opts.ipopt.warm_start_init_point = 'yes';

solver = nlpso1('solver','ipopt',nlp_prob,opts);

%%.....IPOPT optimizer setup::END.....%%
%%.....Bounds setup::Start.....%%
args = struct;

```

```

%->...equality constraint matrix g(equality constraint)
args.lbg = repelem(0,no_ensemble*2*(Np+1));
args.ubg = repelem(0,no_ensemble*2*(Np+1));

%->...Bounds on gate opening, U
% Lower limit of gate opening
args.lbx(no_ensemble*2*(Np+1)+1:1:no_ensemble*2*(Np+1)+ 2*Np,1) = 0;
%Upper bound of gate opening
args.ubx(no_ensemble*2*(Np+1)+1:1:no_ensemble*2*(Np+1)+ 2*Np,1) = hg_max;

%.....The simulation loop starts from here.....%%
% % -----
% % -----

%->... Initialization of control input over the whole prediction horizon
initial_control = [0;0];
U0 = repmat(initial_control,1,Np)';

%->u_previous value is needed to calculate delta u
u_prev = [0;0];

%->...Initialization of states over the whole prediction horizon
%->...Give initial states between 0 and 4.6, this limit is because it will
%help to bound x_M and x_D inbetween xLRV and xHRV
initial_states = [3.8;3.75];
H0 = repmat(initial_states,1,no_ensemble*(Np+1))';

%->...state_history matrix contains the evolution of states over the whole
%simulation time, these states values are obtained from the first optimized
%control input in each time step
state_history(:,1) = initial_states;

%->...u_eachstep contains the changes in control signal over the whole
%simulation time period, these values are first optimal move stored in each
%time step
u_eachstep = zeros(2,timesteps);

% ->...array for storing each loop execution time
time_eachloop = zeros(timesteps,1);

%->... Create a large vector storing the value for the disturbance over
%the simulation period
disturbance = select_disturbance();

%-> ...Repeat each element of inflow disturbance(Here,24 times), because
%prection of inflow from the company are provided after 24 hours, i.e next
%day
generate_disturbance = repelem(disturbance,24).*1;

%->... Start MPC with with initialization of iterations count
no_iterations = 0;

%->...Days counter initialization, data are accessed for the whole one
%month or 30 days
days_counter = 1;

```

```

%->....first day data accesse, multiplies by a number to realize flood scenario
%and then to see the controllere performance
rep_ensem = bring_ensemble(days_counter,no_ensemble,N_days).*1;
each_step_ensemble = rep_ensem';

%->....Putting all the ensemble as a column array which is sent to MPC for
%control signal optimization
in_one_col = reshape(each_step_ensemble,[],1);

% optimal_traj = zeros(no_ensemble*(Np+1),2,120);
%->.....Start the main loop
for i=1:timesteps

    tic;
    %->....Parametrization vector,P and initial vector args.x0 are changed
    %in each timestep

    args.p = [initial_states;u_prev;in_one_col];
    args.x0 = [reshape(H0',no_ensemble*2*(Np+1),1);reshape(U0',2*Np,1)];

    for m = 1:no_ensemble

        %->....Bounds on h1

        args.lbx((m-1)*2*(Np+1)+1:2:m*2*(Np+1),1) = xLRV(i:i+Np,1) - x_LRV_min;
        args.ubx((m-1)*2*(Np+1)+1:2:m*2*(Np+1),1) = xHRV(i:i+Np,1) - x_LRV_min;

        %->....Bounds on h2
        args.lbx((m-1)*2*(Np+1)+2:2:m*2*(Np+1),1) = 0;
        args.ubx((m-1)*2*(Np+1)+2:2:m*2*(Np+1),1) = inf;

    end

    %->.... Call IPOPT solver object in each iterations
    sol = solver('x0',args.x0,'lbx',args.lbx,'ubx',args.ubx,'lbg',...
                args.lbg,'ubg',args.ubg,'p',args.p);

    %->....After each days(23 steps of simulations), new data are accessed
    % from next day
    if mod(i,24) == 0
        days_counter = days_counter + 1 ;
        rep_ensem = bring_ensemble(days_counter,no_ensemble,N_days).*1;
        each_step_ensemble = rep_ensem';
        in_one_col = reshape(each_step_ensemble,[],1);

    else
        rep_ensem = [rep_ensem(:,2:end),rep_ensem(:,end)];
        each_step_ensemble = rep_ensem';
        in_one_col = reshape(each_step_ensemble,[],1);
    end

    %->....Extract the control signal which is after states variables in
    % the optimization variable

```

```

u = reshape(full(sol.x(no_ensemble*2*(Np+1)+1:end))',2,Np)';

%->...Store first move in eachstep
u_eachstep(:,i) = u(1,:)';

%->...Update the states by applying first control move
[initial_states,U0,Vg_dot,Vg_total] = update_states(sampling_time,...
                                                    initial_states,u,f,generate_disturbance(i));

%->...Update u_previous for next iteration
u_prev = u(1,:)';

%->...Storing gate flow in each time step
Vgate_each(no_iterations+1,:) = full(Vg_dot);
Vgate_total(no_iterations+1,:) = full(Vg_total);

% ->...Storing updated states in each time step
state_history(:,no_iterations+2) = initial_states;

%->...Get optimal trajectories
%   optimal_traj(:,1:2,no_iterations+1) =
reshape(full(sol.x(1:no_ensemble*2*(Np+1)))',2,no_ensemble*(Np+1))'+ x_LRV_min;

%->...Get solution trajectory
H0 = reshape(full(sol.x(1:no_ensemble*2*(Np+1)))',2,no_ensemble*(Np+1))';

%->...Shift trajectory to initialize the next step
H0 = [H0(2:end,:);H0(end,:)];

%->...Increase the number of iterations
no_iterations = no_iterations + 1

%->...Record time of completion of each loop
time_eachloop(i) = toc;

end

% ->...Finding average loop completion time
Average_looprun_time = sum(time_eachloop)/timesteps
Total_iteration = no_iterations;
total_time = Total_iteration*Average_looprun_time

%->... x holds the information both x_M and x_D, the levels we are
% interested on.
x = state_history + x_LRV_min;
% ->...Visualization function call for plotting all the results
plotting_results(x,u_eachstep,Vgate_each,Vgate_total,generate_disturbance,...
                 x_LRV,x_HRV,timesteps,time_eachloop,Average_looprun_time,r1,ru);

```

compute_both.m(stochastic MPC)

```

function [obj_total,g] = compute_both(U,H,P,Np,sampling_time,...
                                     x_LRV_min,w_R,w_delU,w_U,f,n_states,n_u_prev,no_ensemble)

```

```

addpath('C:\aaa._MY_FILES\AUSN\4th semester\casadi-windows-matlabR2016a-v3.5.5');
import casadi.*

%->....Disturbance vector is assigned to P vector, these values stored
%in P vector are changed in each iteration fromt the main for loop

%->....Objective function vector for each of the ensemble
obj_array = SX.zeros(1,no_ensemble);
%->....Empty Constraints vector, later we will add the evaluated
%constraints in each step of prediction horizon through for loop
g = [];

ensemble_iteration = 0;

u_prev = P(3:4,1);

U_extended = [u_prev,U];

del_con = U_extended(:,2:end)-U_extended(:,1:end-1);

for j = 1:no_ensemble

    obj = 0;

    opt_states = H(:,(j-1)*(Np+1) + 1:j*(Np+1));

    st = opt_states(:,1);

    %->....Initial condition constraints, the difference is appended in
    %each iteration, the concept of multiple shooting adds this equality
    %constraint
    g = [g; st - P(1:n_states,1)];

    %->....select each ensemble
    Vi_disturbance = P(n_states + n_u_prev + (j-1)*Np + 1:n_states + n_u_prev + j*Np,1);

    ensemble_iteration = ensemble_iteration + 1;

    sprintf('we are using %d numbered ensemble in the MPC',ensemble_iteration)

    for k = 1:Np

        st = opt_states(:,k);
        con = U(:,k);

        inflow = Vi_disturbance(k,1);

        %->....level of Merkebakk to be maximized in the objective function
        x_M1 = opt_states(1,k) + x_LRV_min ;

        obj = obj + (-w_R(k)*x_M1^2 + del_con(:,k)'.*...
                    w_delU(k)*del_con(:,k) + con'*w_U(k)*con);
    end
end

```

```

st_next = opt_states(:,k+1);

% Integrating the models with explicit runge kutta method
k1 = f(st,con,inflow);
k2 = f(st+k1.*sampling_time/2,con,inflow);
k3 = f(st+k2.*sampling_time/2,con,inflow);
k4 = f(st+k3.*sampling_time,con,inflow);

st_next_predicted = st +sampling_time/6*(k1+2.*k2+2.*k3+k4);

g = [g;
      st_next - st_next_predicted % x(k+1) = f(x(k),u(k),d(k)) Equalityconstr
     ];
end
obj_array(j) = obj;
end

obj_total = 1*sum(obj_array);
end

```

select_disturbance.m(common file)

```

function selected_inflow = select_disturbance()

clear all;
clc;

addpath 'C:\aaa._MY_FILES\AUSN\4th semester\Vi_Forecast_data\2020April_May_NormalData';
dirName = 'C:\aaa._MY_FILES\AUSN\4th semester\Vi_Forecast_data\2020April_May_NormalData';

files = dir(fullfile(dirName,'*.mat'));
files = {files.name};

%->....50 different predicted ensemble
no_ensemble = 50;
no_days = 46;
mean_each_ensemble = zeros(no_ensemble,1);
square_error = zeros(no_ensemble,1);
%->....Array for storing error for each 50 ensemble
selected_inflow = zeros(no_days,1);

for k = 1:no_days

    load(files{k});
    %->....Use all the data except row number 51 as it is control signal
    data = Vi(1:end-1,:);

    %->....Loop for finding mean of each 50 ensemble for 13 days
    for j = 1:no_ensemble
        val = data(j,:);
        mean_each_ensemble(j,:) = sum(val)/numel(val);
    end

    mean_of_mean = sum(mean_each_ensemble)/numel(mean_each_ensemble);

```

```

%->....Loop for comparing mean of each ensemble to mean of mean of
%ensemble and finding error
for m = 1:no_ensemble
    square_error(m) = sqrt((mean_of_mean - mean_each_ensemble(m))^2/...
                            no_ensemble);
end
[min_val,index] = min(square_error);

selected_inflow(k) = data(index,1);

end
end

```

state_models.m (common file)

```

function [h_dot, Vg_dot,Vg_dot_total] = State_models(h,hg,Vi)

%->....Parameters declaration
alpha = 0.05; % Fraction of surface area in compartment 2
beta = 0.02; % Fraction of inflow to compartment 2

C_d = 0.7; % Discharge coefficient, Dalsfos gate
w = [11.6,11]; % width of the flood gates 1 and 2
g = 9.8; % Acceleration due to gravity
K12 = 800; % Inter compartmental flow coefficient

PPP = 10^(-5);

%->....Disturbance assignment to a variable
Vi_dot = Vi;
%->....Volumetric flow through the turbine [m3/sec]
Vt_dot = 36;

%->...Intermediate equations: Algebraic equations
sectional_area = @(h) max((28e6*1.1* max(h,PPP)^(1/10)),1e3);

V12_dot = K12*(h(1)-h(2))*sqrt(max(abs(h(1)-h(2)),PPP));

Vg_dot = [C_d*w(1)*min(hg(1),h(2))*sqrt(2*g*max(h(2),PPP));
          C_d*w(2)*min(hg(2),h(2))*sqrt(2*g*max(h(2),PPP))
          ];

%->....Total flow through both gates
Vg_dot_total = sum(Vg_dot);

% State equations
h_dot = [((1-beta)*Vi_dot-V12_dot)/((1-alpha)*sectional_area(h(1)));
         (beta*Vi_dot-Vt_dot-Vg_dot_total+V12_dot)/(alpha*sectional_area(h(2))) ];

end

```


update_states.m(common file)

```

function [initial_states,u0,Vg_dot,Vg_total] =
update_states(sampling_time,initial_states,u,f,generate_disturbance)

    dis_inflow = generate_disturbance;
    st = initial_states;
    con = u(1,:)' ; % Apply first control move to th model to update the states

    [k1,Vg_dot,Vg_total] = f(st,con,dis_inflow);
    k2 = f(st+k1.*sampling_time/2,con,dis_inflow);
    k3 = f(st+k2.*sampling_time/2,con,dis_inflow);
    k4 = f(st+k3.*sampling_time,con,dis_inflow);
    st = st +sampling_time/6*(k1+2.*k2+2.*k3+k4);

    initial_states = full(st);

    %->...Removing first control and repeating last control
    u0 = [u(2:size(u,1),:);u(size(u,1),:)] ;

end

```

Simulator_casadi.m (Open loop simulation)

```

clc;
clear;
close;
addpath('C:\aaa._MY_FILES\auSN\4th semester\casadi-windows-matlabR2016a-v3.5.5');
import casadi.*

%Define requirements
% hg_max = 5.6;

%Level constraints from April 15 to May 15
x_LRV1 = 55.75; %[m] from april 15 to april 30
x_HRV1 = 60.35; %[m] from april 15 to april 30

x_LRV2 = 58.85; % [m] from may 1st to may 15
x_HRV2 = 59.85; % [m] from may 1st to may 15

Vi_dot = 250; % Inflow variations in the lake Toke [m3/sec]

Vt_dot = 36;

x_LRV_min = 55.75; %[m]

% Time setting for simulations
initial_time = 0;
final_time = 30*86400; % Total simulation period of 13 days
sampling_time = 3600;% Sampling time of 1 hour
t = initial_time:sampling_time:final_time;
number_timesteps = int64((final_time-initial_time)/sampling_time) + 1 ;

```

```

hours = 1:1:number_timesteps;

%%Creating array with constraints for plotting
xLRV = zeros(1,number_timesteps);
xLRV(1:360) = x_LRV1;
xLRV(360:end) = x_LRV2;

xHRV = zeros(1,number_timesteps);
xHRV(1:360) = x_HRV1;
xHRV(360:end) = x_HRV2;

%%Symbolic representation for casadi
%%States representation
h = MX.sym('h',2);
states = [h] ;
n_states = length(states);

%%Control action representation
hg = MX.sym('hg',2,1);
controls = [hg];
n_control = length(controls);

%%Process model
hdot = State_models(states,controls);

% Non Linear mapping function
f = Function('f',{states,controls},{hdot});

% For storing states of the system
% H = DM(n_states,number_timesteps);

H = SX.sym('H',n_states,number_timesteps);

% Control signal over the simulation period
Hg = DM(n_control,number_timesteps);

%%States assignment
initial_states = [58.75;58.70];
H(:,1)= initial_states;
E1_t = zeros(number_timesteps-1,1);
tic;
for k = 1:number_timesteps -1
%   tic;
    if k<=50
        Hg(:,k) = 0;

    elseif k>50 && k<=150
        Hg(:,k) = 0;

    elseif k>150 && k<=340
        Hg(:,k) = 0;

    elseif k>300 && k<=400
        Hg(:,k) = 1.2;

```

```

elseif k>400 && k<=500
    Hg(:,k) = 0.4;

elseif k>500 && k<=700
    Hg(:,k) = 0.4;
else
    Hg(:,k) = 0.4;

end
% Integrating the models with explicit runge kutta method
k1 = f(initial_states,Hg(:,k));
k2 = f(initial_states+k1.*sampling_time/2,Hg(:,k));
k3 = f(initial_states+k2.*sampling_time/2,Hg(:,k));
k4 = f(initial_states+k3.*sampling_time,Hg(:,k));
H(:,k+1) = initial_states +sampling_time/6*(k1+2.*k2+2.*k3+k4);
initial_states = H(:,k+1);

%      E1_t(k,1)=toc;
end

toc;

H = full(evalf(H));
Hg = full(evalf(Hg));

%%Plot results showing the level changes x_M and x_D
subplot(3,1,1);
plot5 = plot(hours,H,'Linewidth',1.05);
hold on;
plot6 = plot(hours,xLRV,hours,xHRV,'Linewidth',1.05);
combined_plot = [plot5;plot6];
legend(combined_plot,'x_M','x_D','xLRV','xHRV');

% xlabel('Time(hours)');
ylabel('Levels[m]');
title('Open loop simulation for x_M and x_D change')
grid on;

subplot(3,1,2);
plot(hours,Hg,'Linewidth',1.05);
ylim([-0.95,5]);
legend('hg1','hg2');
% xlabel('Time(hours)');
ylabel('control signal, hg[m]');
title('Control signal change over the period')
grid on;

subplot(3,1,3)
plot(hours,ones(1,length(hours)).*vt_dot,hours,ones(1,length(hours)).*vi_dot);
ylim([0,300]);
xlabel('Time(hours)');
ylabel('volumetric flow, m^3/s');
legend('vt dot','vi dot');

```