Master's Thesis 2014

Candidate: Robin Evensen

Title: Determenistic Flood Control using MPC on the Kragerø Waterways

Telemark University College
 Faculty of Technology
Kjølnes
3914 Porsgrunn
Norway

Telemark University College
Faculty of Technology
M.Sc. Programme

MASTER'S THESIS, COURSE CODE FMH606

Student: **<Robin Evensen>**

Thesis title: **<Deterministic Flood Control using MPC on the Kragerø Waterways >**

Signature: . . . . . .

Number of pages: <113>

Keywords: **MPC, Flood Control, Lake Toke,**

Kragerø Waterways. . . . . .

. . . . . .

Supervisor:	<Bernt Lie>	sign.:	. . . . .
2nd Supervisor:	< >	sign.:	. . . . .
Censor:	<N/A>	sign.:	. . . . .
External partner:	<Skagerak Energi>	sign.:	. . . . .
Availability:	<Open >		

Archive approval (supervisor signature): sign.: . . . . . **Date :** . . . . . .

Abstract:

The Kragerø Waterways consist of 5 power plants downstream after Lake Toke, Dalsfos is the first of these power plants. As a deterministic flood control of Lake Toke and Dalsfos there has been designed a MPC. The MPC uses a horizon of 10 days with estimated influx of water into the Lake Toke water reservoir. As a preliminary task before the MPC is designed, an assessment of the requirements and constraints of Lake Toke was made. The primary constraint is to keep the level between an upper and lower limit regulated by NVE.

A 2 state model designed by Bjørn Glemmestad was used to define Lake Toke, to further increase the precision of the model, assessment parameters was done. A simulation and validation of the linearized model of Lake Toke was performed to assess the precision, which proved to perform adequate.

A review of the structure of the MPC program is shown along with discussion and explanation of the implementation of constraints and performance requirements. The controller is validated by simulation a flood scenario with disturbances similar to a real flood situation, there was also added a secondary peak to resemble a rainfall. The objective is to keep the level between a lower and upper limit, *lrv* and *hrv*, with a goal to avoid an overshoot over *hrv*. The controller proved to avoid an overshoot when the weight lower limit in the performance index was lower than the upper limit. A weight on the control output also proved to create a more stable control output as well as avoiding an overshoot when the secondary rainfall arrived.

Table of contents

Preface

The master thesis is entitled Deterministic Flood Control using MPC of the Kragerø Waterways. This thesis is part of the education program System and Control Engineering at Telemark University College. The main part of the thesis was carried out at TUC as the work was computer based, the necessary data and information was given by mail or in meetings at TUC.

The thesis is primarily about designing a Model Predictive Controller (MPC) which has a goal to regulate the water level at Lake Toke by controlling the floodgate. The be able to understand the report fully the reader should have general knowledge about Optimal Control or Model predictive control, as well as fundamental understanding of modeling and discretization of models. The code used in this thesis is given is shown in the Appendix, additionally the code along with the necessary measurement data from Skagerak is also given on a CD.

The task description for this thesis can be found in Appendix 1, the task description is also built upon a project during a course at 3.semester which is shown in Appendix 2. There was scheduled a test of a first version of the controller by 1. March 2013. Due to difficulties with other parts of the thesis, I was not able to finish this first version by the date. The supervisor Bernt Lie was able to develop another version of MPC that was implemented before the flood season started. As a result the focus was held on the completion of the MPC controller itself, as well as developing the models as this is an essential part of the controller.

Apart from writing the report, only MATLAB from MathWorks was used as a tool to work on the thesis. As the most crucial part of MPC is Quadratic programming, MATLAB provides an effective and stable function to solve these equations.

I would like to give a big thanks to Bernt Lie for his guidance that helped me push through and finish the work at the parts where my competence came short, what comes to mind is the design of Kalman Filter along with approaches for model development. During the project in 3. semester, several discussions with Anushka Perera helped me at solving several MPC related problems as well as providing me with a linearization method which proved to give good results. This prepared me for this thesis in regards of MPC and possible problems which I could foresee. Ingvar Andreassen at Skagerak which was responsible for this project at Skagerak was able to provide the necessary data used to develop the models needed.

<Porsgrunn, 3.6.2014>

<Robin Evensen>

Nomenclature

hrv	Limit for the highest regulated water level
lrv	Limit for the lowest regulated water level
LSM	Least Square Method
MATLAB	Matrix Laboratory
MPC	Model Predictive Controller
NVE	Norges vassdrags- og energidirektorat
TUC	Telemark University College

1 Introduction

Norway is one of the leading nations in hydro power facilities, with this comes a high quantity of dams that need water level regulation and flood control. Dalsfos is the first of five hydro power plants between Lake Toke and Kilsfjord in Skagerak. Originally Dalsfos is regulated manually by using their experience and a predicted influx of water based on a hydrological model. There has been proposed a fully automated solution to control the flood gate at Dalsfos. A model predictive controller will be used to prevent a predicted flooding situation, in addition to expending as little water as possible from the Lake Toke reservoir to allow it to be used for electricity production. There is an estimate of influx water into Lake Toke denoted $\dot{V}_{in}$ which is calculated from an external firm using an hydrological model, the generated electricity produced from the turbines at Dalsfos is denoted $\dot{W}_e$.

A 2 state nonlinear model is used to simulate the levels at Merkebekk and Dalsfos. There will be done a study on possible improvements of this model by performing sensitivity analysis on the existing parameters and estimating these using a Kalman Filter. Because $\dot{W}_e$ is given as *kWh*, the flow through the turbine needs to be modelled to give the necessary output m^3/s. Both the turbine flow model and the Lake Toke model will be linearized around realistic steady state values and discretized, these will be simulated in realistic situation and validated.

From Skagerak there are regulated limits on the water level, the limits change depending on the date of the year. The thesis will focus on the flooding season which ranges from April to June. There are several constraints on the system which has to be handled by the control system, i.e. the lower limit on the flow is $4\ m^3/s$, and there are also regulations on the rate of change on the flow. To satisfy the constraint a deterministic flood control will be created using MPC. The controller will use a horizon of 10 days with disturbances $\dot{V}_{in}$ and $\dot{W}_e$ given by Skagerak.

There will be an overall review of the structure of the MPC program as well as discussions and reasoning's behind the design of the controller. The basis of the structure which the MPC program was built upon was designed in the 3. semester project, the program has several functions and a structure which can be reused in this thesis [1]. To validate the controller there will be performed simulations on the 2 state nonlinear model. The disturbances used are simulated based on a real flooding situation. The goal of the simulations will be to maintain the constraints and performance requirements described in Chapter 2.3. To tune the MPC, the weights in the performance index will be disabled and/or changed, then the results will be evaluation w.r.t the constraints and performance requirements.

2 System description

This chapter consists of an overall description of Lake Toke and a functional description which is used to get an overview of the problems which the controller should manage.

2.1 Overview of the Kragerø river system

The system mentioned throughout the report is Lake Toke regulated using the hydropower plant Dalsfos, south of Lake Toke. Lake Toke is located next to Drangedal and is a part of a network of rivers and lakes in Telemark, finally ending up in Kilsfjord in Kragerø. Figure 2-1 shows how the entire lake, note that Merkebekk and Dalsfos is two measurement points used as the reference point for the states h_1 and h_2 respectively in the model described in Chapter 3.2. To fulfill environmental demands from NVE, there is strict lower and upper water level boundaries in Lake Toke that needs to be satisfied, denote *lrv* and *hrv*. These boundaries changes depending on the time of the year.

Figure 2-1: Both upper and lower part of lake Toke [2]

From Dalsfos to Kilsfjord, there is a levitation drop of approximate 58 meters. Dalsfos is one of five hydropower plants utilizing a levitation drop of approximate 21 m. See Figure 2-2 to get an overview of the river system from Dalsfos to Kilsfjord. Dalsfos was originally built

with one Francis turbine in 1907. Dalsfos was expanded with 2 more Francis turbines in 1958. [3][1] Dalsfos is also fitted with flood gates, these floodgates are controlled manually by an operator. This is operator is soon in retirement, thus came the need of an automatically controlled floodgate.

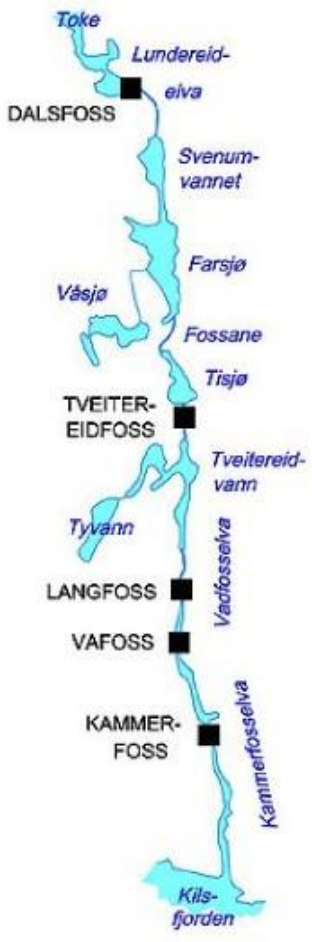


Figure 2-2: The river from Dalsfos to Kilsfjord [4]

2.2 Functional Description

The system of Lake Toke is described as a system with 1 controllable input, 2 disturbances and 2 outputs. A block diagram of this system is shown in Figure 2-3. An explanation of the system objects is in Table 2.1.

[1] In the reference from *Store Norske Leksikon* there is stated that Dalsfos is fitted with 3 Kaplan turbines, this is wrong after a discussion with Ingvar Andreassen in Skagerak Energi.

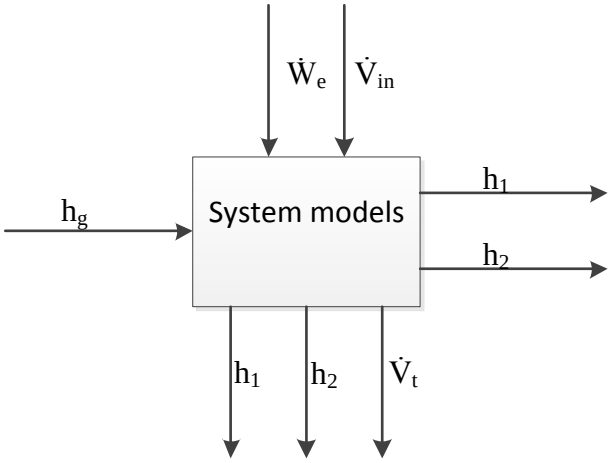


Figure 2-3: A block diagram of Lake Toke and Dalsfos

The level h_1 and h_2 is introduced as both states and inputs, since they are both a measured value as well as estimated values for several time instances in the future. The generated power $\dot{W}_e$ and water influx $\dot{V}_{in}$ is assumed as known disturbances. The system consists of two models, one that express the levels h_1 and h_2, and another that express the turbine flow $\dot{V}_t$.

In reality the flood gate opening consist of three separate gate openings, but for the sake of simplicity the gates are considered as one large gate. This issue is merely a scaling problem, and can be dealt with when a real implementation is due.

Table 2.1: Description of the objects in Figure 2-3

Parameter	Unit	Comment
h_g	m	The flood gate opening *(Controllable input)*
$\dot{W}_e$	kWh	Generated power at Dalsfos *(Disturbance)*
$\dot{V}_{in}$	m^3/s	Influx of water into the system *(Disturbance)*
h_1	m	Water level at Merkebekk *(State, Output)*
h_2	m	Water level at Dalsfos gate *(State, Output)*
$\dot{V}_t$	m^3/s	The flow through the turbine *(State)*

2.3 Problem description

The problem is based on the task description in Appendix A and Appendix B, as well as some mention in an informal meeting with the supervisor Bernt Lie. The main goal of is to control the level h_1 which are subject to restriction of upper and lower boundaries, in addition to keeping the flow through the turbines as low as possible. By reducing the amount of water

through the flood gate, this water can be used to produce electricity instead. Because the economic efficiency the controller is trying to increase is a contradiction to the safety of the environment downstream, there should be some margin of error in case of estimate or measurement errors (i.e. there should be enough time to react to a unpredicted flood). The constraints as well as the performance requirements of the system are discussed in more detail in Chapter 2.3.1 and 2.3.2.

2.3.1 System constraints

The constraints that are mentioned here are is a general representation of the constraint. The technicality of the implementation in a controller is shown in Chapter 4.2. The constraints in this chapter are strict constraints which should be met.

The controller should keep the level of Lake Toke (h_1) within the minimum and maximum regulated water level. There will be situations where these constraints will be broken, but keeping the level within these boundaries should be the main purpose of the controller:

$$h_1 \geq hrv \tag{2-1}$$

$$h_1 \leq lrv \tag{2-2}$$

There is a restriction on the maximum rate of change on the river flow $\dot{V}_O$, denoted $\ddot{V}_O$. This constraint is due to the safety of the people who use the river downstream, to give the public time to react to a sudden increase of flow:

$$\ddot{V}_O \leq \ddot{V}_O^{\max} \tag{2-3}$$

There is a restriction to the minimum allowed flow $\dot{V}_O$, this limit is set to $4\ m^3/s$. Although the constraint might be fulfilled by Skagerak through the turbine flow, there can be situations where there is maintenance on the turbine, resulting in that the constraint has to be fulfilled by releasing water through the flood gates:

$$\ddot{V}_O \geq 4 m^3/s \tag{2-4}$$

2.3.2Performance requirements

This chapter discussed the problems description in Chapter 2.3 and how they can be improved and fulfilled as good as possible.

The known future disturbances is 10 days, therefore the horizon of the system is set to 10 days accordingly the controller should be able to give optimal solutions within these 10 days given that the measurements and disturbances are correct. There are several possible scenarios that the controller needs to handle which can occur during a whole year. Drought, change of *hrv* and *lrv*, but the most importantly the controller needs to satisfy the constraints and performance requirements during a flood season.

To give and indicator to the quality of the model as well as the stability of the controller, the controller should be able to converge to a specified reference point. This also gives an incentive to further improve the previous designed model, as well as designing the flow through the turbine, $\dot{V}_t$. Since model uses 5 variables with a relatively wide range of values during the horizon, the performance of the controller will be highly influenced from the steady state values used to linearize the models. The focus will be held on giving the best result during a flood season, i.e. when the $\dot{V}_{in}$ increases.

To keep the loss of potential economical income as low as possible, the flood gate flow $\dot{V}_g$ should be kept to a minimum, meanwhile keeping the water level h_1 within *lrv* and *hrv*. These two requirements will contradict each other, although there is possible to satisfy the former constraint, the latter should be fulfilled to it upmost potential, i.e. keeping the $\dot{V}_g$ as low as possible. Though it is only a requirement to keep the level within *hrv* and *lrv*, a reference point is also used observe how the controllers performance as well as observing its precision.

3 System models

The system can be sketch and simplified as a two tank system with a flow between them, as seen in Figure 3-1. Here h_1 represents the level at Merkebekk and h_2 at Dalsfos. $\dot{V}_t$ is the flow through the turbines and $\dot{V}_g$ is the flow through the flood gates. $\dot{V}_{in}$ is the estimated collection of water divided amongst the upper and lower compartment. $\dot{V}_{in}$ consists of both water from precipitation and snow melting. At Skagerak, they use an external firm to calculate this $\dot{V}_{in}$ based upon a hydrological model. This $\dot{V}_{in}$ will be calculated 10 days beforehand, and is considered as known in the MPC solution.

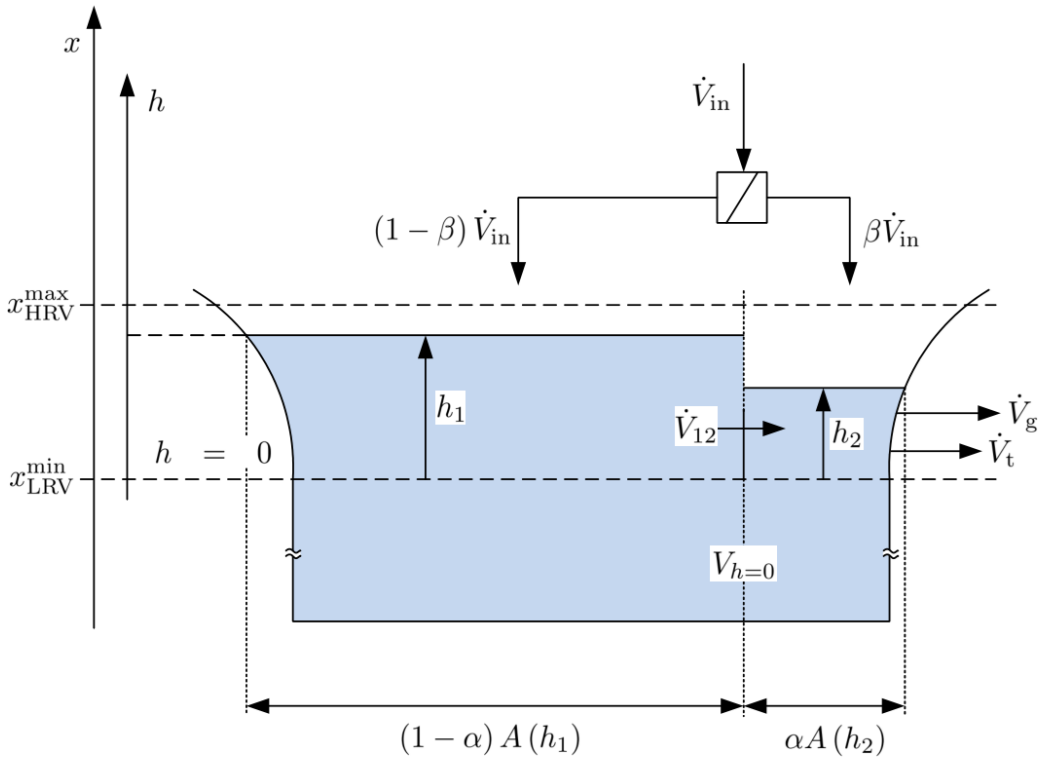


Figure 3-1: Lake Toke sketched visualized as two compartments [5]

The Lake Toke system can be described using mass balance. There has been developed a 2 state nonlinear model at TUC by Bjørn Glemmestad seen in in equation (3-1).

$$\begin{aligned}\frac{dh_1}{dt} &= \frac{1}{(1-\alpha)A(h_1)}\left[(1-\beta)\dot{V}_{in} - \dot{V}_{12}\right] \\ \frac{dh_2}{dt} &= \frac{1}{\alpha A(h_2)}\left[\beta\dot{V}_{in} - \dot{V}_{12} - \dot{V}_t - \dot{V}_g\right]\end{aligned} \tag{3-1}$$

here

$$A(h_x) = \max(28 \times 10^6 \cdot h^{1/10}, 10^3) \tag{3-2}$$

$$\dot{V}_{12} = \lambda(h_1 - h_2)\sqrt{|h_1 - h_2|} \tag{3-3}$$

$\dot{V}_g$ is the outflow through the gate and can be generally described as the equation (3-4).

$$\dot{V}_g = C_D h_g w \sqrt{2gh_2} \tag{3-4}$$

The turbine flow $\dot{V}_t$ is a function of h_2, $\dot{W}_e$ and the level below Dalsfos X_Q, where $\dot{W}_e$ is the generated power and is given by Skagerak. As X_Q isn't measured, it will be estimated in the model.

These parameters will be looked into more in Chapter 3.2.2 to improve its performance. The parameters can be seen in Table 3.1:

Table 3.1: Parameters used in the 2 state nonlinear model

Parameter	Value	Unit	Comment
α	0.05	~	Fraction of surface of the compartment in h_2
β	0.02	~	Fraction of inflow into the compartment of h_2
C_D	1	~	Friction loss factor
λ	800	~	Invented factor
w	11.2	m	Width of flood gate
$h_g^{\max}$	5.6	m	Maximal opening height of gate
$x_{LRV}^{\min}$	55.75	m	Minimal low regulated level value
$x_{HRV}^{\max}$	60.35	m	Maximal high regulated level value

3.1 The Linearizing method

Both the controller and the kalman filter described in Chapter 3.2.2.2 and 3.2.2.3 uses a linearized model to calculate a Kalman Gain. The turbine flow model described in Chapter

3.3 is also linearized. This linearization method was proposed by Anushka Perera, a PhD student at TUC. The linearization method is described generally below, where *f* is the function to be linearized. Equation (3-5) and (3-6) describes the equation which is used to linearize, with a description of the parameters listed in Table 3.2.

$$A = imag(\frac{1}{h} * f(x + h*i*I_x, u)) \tag{3-5}$$

$$B = imag(\frac{1}{h} * f(x, u + h*i*I_u)) \tag{3-6}$$

Table 3.2: Describes the parameters in Equation (3-5) and (3-6)

Parameter	Comment
A	Linearized transition matrix
B	Linearized input matrix
f	Function to linearize
x	Model states
u	Model input
h	A very small number(used a function in MATLAB called *eps*)
i	Imaginary number (√-1)
I	Identity matrix

3.2 The Lake Toke model

This chapter focuses only on the differential equation (3-7), the model for $\dot{V}_t$ is developed in Chapter 3.3.

$$\begin{aligned} \frac{dh_1}{dt} &= \frac{1}{(1-\alpha)A(h_1)}\left[(1-\beta)\dot{V}_{in} - \dot{V}_{12}\right] \\ \frac{dh_2}{dt} &= \frac{1}{\alpha A(h_2)}\left[\beta\dot{V}_{in} - \dot{V}_{12} - \dot{V}_t - \dot{V}_g\right] \end{aligned} \tag{3-7}$$

3.2.1Linearization and simulation of the Lake Toke model

There are performed step changes on the linearized model to observe its performance. The steady states are used to linearize the model are listed in Table 3.3. The steady state values were found experimentally by adjusting the value until the best performance was observed. The code for the simulations is in Appendix 7.

Table 3.3: List of steady state values that are used to linearize the model

Variable	SS Value	Unit	Comment
h_1	h_{1_0}	m	The initial value of the level h_1 is used
h_2	h_{2_0}	m	The initial value of the level h_2 is used
h_g	0.05*5.6	m	5 % of the maximum gate opening
$\dot{V}_{in}$	40	m^3/s	Weight to keep the level h_1 below *hrv*
$\dot{V}_t$	10	m^3/s	Weight to keep the level h_1 above *lrv*

There are done three simulations where they are compared to the nonlinear model described in Chapter 3. Figure 3-2 shows a simulation where there is a large inflow, forcing the level to increase from a low level to a high level. The linear model shows a deviation of 0.5 m higher than the nonlinear model.

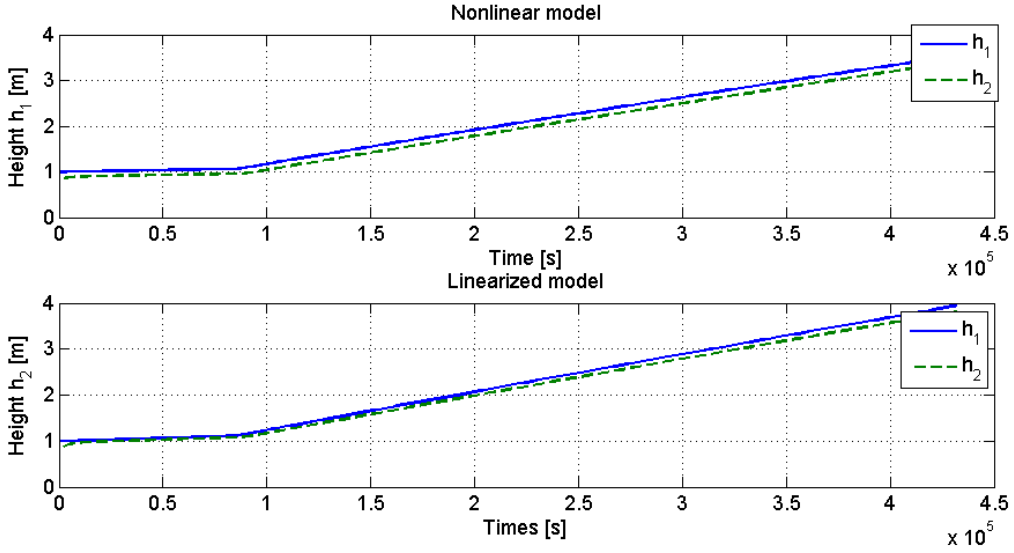


Figure 3-2: The simulation is done over 5 days. A step change is done on the influx $\dot{V}_{in}$ from 50 to 250 m^3/s after 1 day. The $\dot{V}_t$ is kept steady at 24 m^3/s and the h_g of 1%.

When increasing the gate opening h_g the watch how the linearization model handles a descending level on both h_1 and h_2. In Figure 3-3 nonlinear model and linearized model are

compared. One can observe that the level h_1 is almost the same, but the level h_2 in the linearized model drops about 0.5 m lower than the nonlinear model.

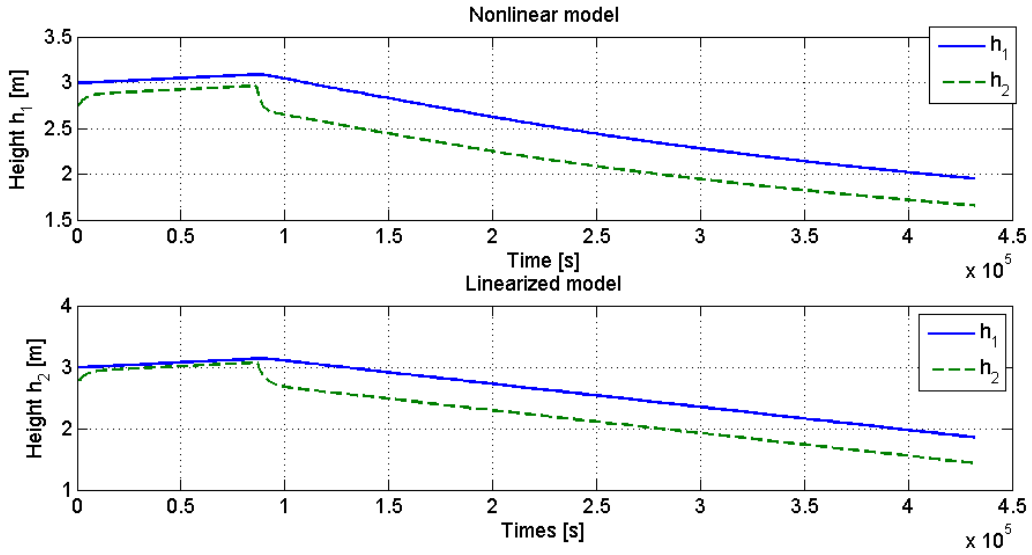


Figure 3-3: Simulation is done over 5 days. A step change on the h_g after 1 day of 2 % to 4 % gate opening. The $\dot{V}_{in}$ is 70 m^3/s and $\dot{V}_t$ is 15 m^3/s. The level starts at h_1 =3 m and h_2 =2.7 m.

Finally there is performed a simulation over 10 days where there is an increase in $\dot{V}_{in}$ from 50 m^3/s to 150 m^3/s after 2 days. Then after 5 days the h_g is increased from 2 % to 5 %. The $\dot{V}_t$ is 15 m^3/s throughout the 10 days. The result is shown in Figure 3-4.

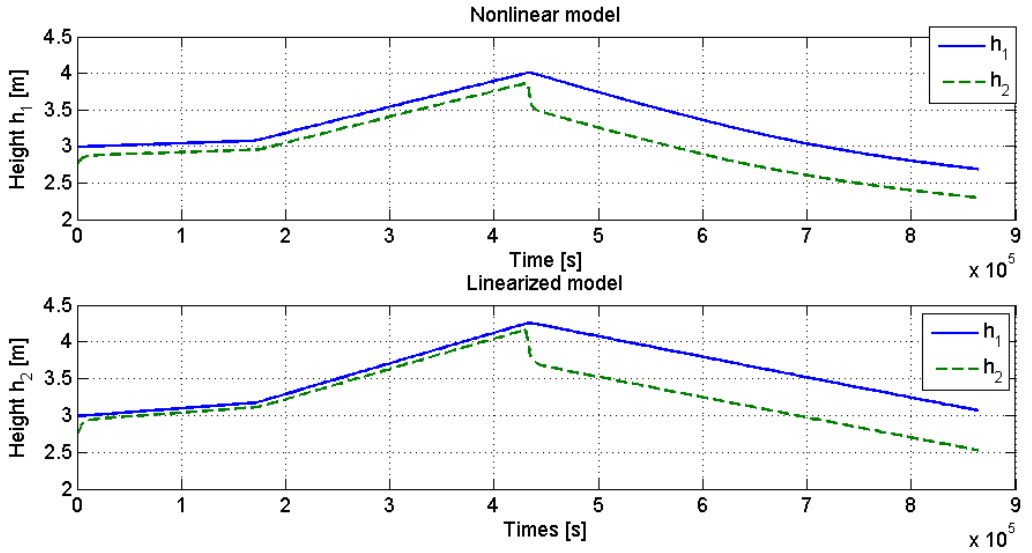


Figure 3-4: A simulation is performed over 10 days with a change in both $\dot{V}_{in}$ and h_g.

Although the model misses by 0.5 m over 10 days after performing these step changes, one important aspect, is that the model should predict an eventual overflow within 10 days. The fact that the linear model shoots higher when the $\dot{V}_{in}$ reaches 150 – 200 m^3/s, creates a safety margin that causes the model to overshoot. Since the model is linearized around the last know measured h_1 and h_2, the precision will increase the further the level reaches steady state.

3.2.2Calibrating the Lake Toke model

The 2 state nonlinear model had several parameters which were chosen experimentally to make to model fit the experimental data. The code for the sensitivity analysis is given in Appendix 8 To see how much impact each parameter have on the model, there was performed a test on each of those parameters individually using the formula in Equation (3-8).

$$S_{x_p} = \frac{f(x_1,...,x_n,x_p) - f(x_1,...,x_n,x_p + \Delta x_p)}{\Delta x_p} \tag{3-8}$$

Here one wants to know the how the parameter x_p affects the model. Δx_p is 5% of x_p. The parameters of interest are α, β, λ and C_D and are listen in Table 3.1. Figure 3-5 shows the effect on h_1 and Figure 3-6 shows the effect on h_2. Since the goal of the sensitivity analysis is to choose which parameters that are the best candidates for parameter estimation. To estimate the parameters a Kalman Filter will be used which is described in Chapter 3.2.2. Because the goal is to create a more stable model, a stable curve is more optimal than an unstable curve. Another thing to consider is also whether the level h_1 or h_2 is most important. Primarily the level h_2 is used to calculate the flow $\dot{V}_g$, but since the most important constraints are connected to the level h_1, the graphs in Figure 3-5 are the once to consider. As one can see in both Figure 3-5 and Figure 3-6, the parameter C_D has the highest impact on level h_1 and h_2. The parameter λ and β has too little impact on the model compared to α. First and foremost the parameters α and C_D will be looked at.

When using the kalman filter on the whole data set in Chapter 3.2.2.2 and 3.2.2.3, the samples from 1 to 1000 made both filters struggle. The reason behind this is unknown, but the dynamics seems to change at around sample 900 – 950 which causes parameter α and C_D to go to 0, resulting in the ODE solver to crash or struggle after these samples. Because of this the test is performed from sample 1000 to sample 3117.

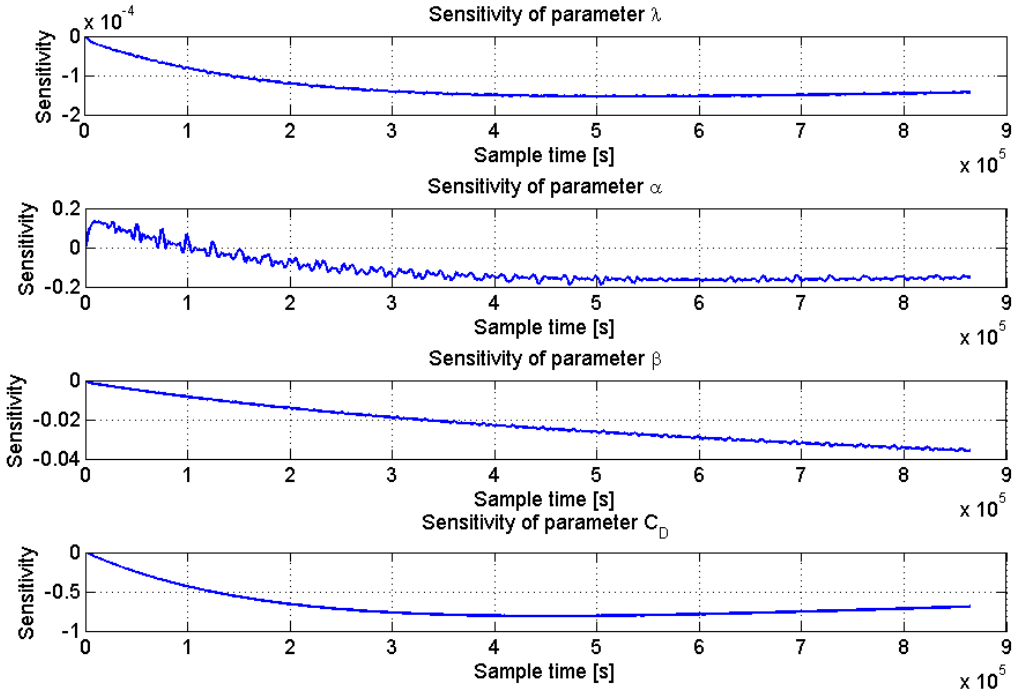


Figure 3-5: Shows the sensitivity of parameter α, β, λ *and* C_D on the level h_1

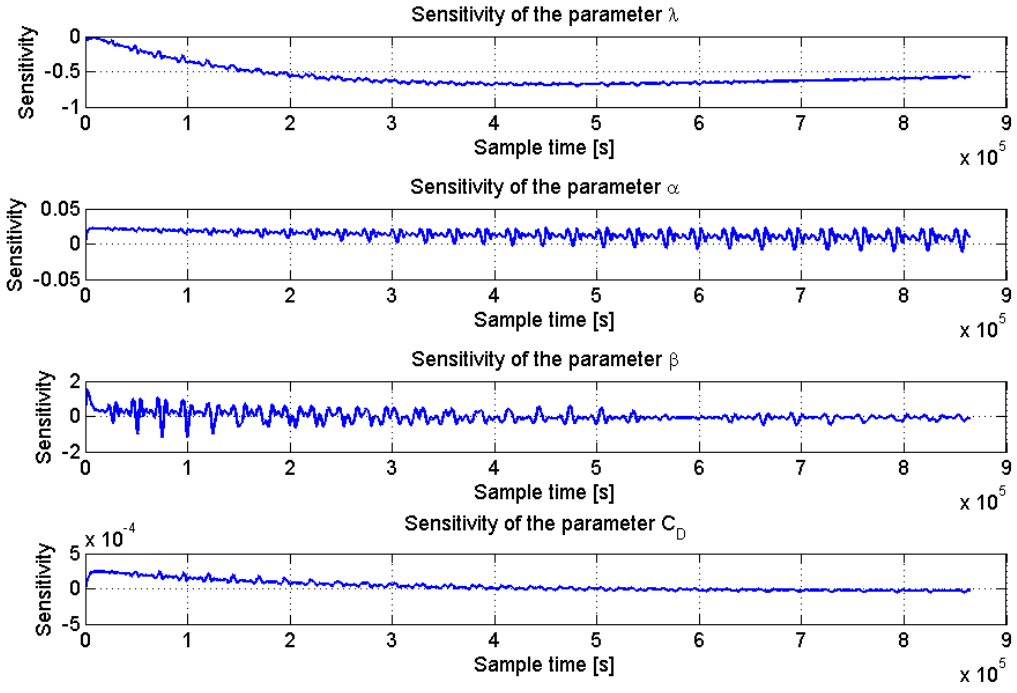


Figure 3-6: Shows the sensitivity of parameter α, β, λ *and* C_D on the level h_2

3.2.2.1Augmenting the model

Before the model can be used in a kalman filter for parameters, the model from Chapter 3 has to be augmented. The code for both the Kalman Filters is given in Appendix 6. The augmented model is:

$$\begin{aligned}
\frac{dh_1}{dt} &= \frac{1}{(1-\alpha)A(h_1)}\left[(1-\beta)\dot{V}_{in} - \dot{V}_{12}(h_1,h_2,\lambda)\right] \\
\frac{dh_2}{dt} &= \frac{1}{\alpha A(h_2)}\left[\beta\dot{V}_{in} - \dot{V}_{12} - \dot{V}_t - \dot{V}_g(h_2,h_g,C_D)\right] \\
\frac{d\alpha}{dt} &= 0 \\
\frac{dC_D}{dt} &= 0
\end{aligned} \tag{3-9}$$

The states α and C_D can be estimated by introducing white noise to the model with suitable amplitude. This gives us the model:

$$\begin{aligned}
\frac{dh_1}{dt} &= \frac{1}{(1-\alpha)A(h_1)}\left[(1-\beta)\dot{V}_{in} - \dot{V}_{12}(h_1,h_2,\lambda)\right] \\
\frac{dh_2}{dt} &= \frac{1}{\alpha A(h_2)}\left[\beta\dot{V}_{in} - \dot{V}_{12} - \dot{V}_t - \dot{V}_g(h_2,h_g,C_D)\right] \\
\frac{d\alpha}{dt} &= w_\alpha \\
\frac{dC_D}{dt} &= w_{C_D}
\end{aligned} \tag{3-10}$$

where w_α and w_{CD} is white noise with a zero mean and a specified variance.

3.2.2.2Time varying kalman filter

Since the model is nonlinear, it can be better to use a time varying kalman filter to compensate for the varying A matrix.

The algorithm for a time-varying extended kalman filter is described below:

- *Deciding initial matrices and parameters*

  $P_{0|0} = diag([1 \quad 10 \quad 0.001 \quad 0.0001])$

  $x_{0|0} = [h_{10|0} \quad h_{20|0} \quad 0.05 \quad 1]$

  - $G = diag([1 \quad 1 \quad 1 \quad 1])$

    $W = diag([1 \quad 10 \quad 0.0001 \quad 0.0001])$

    $V = diag([1 \quad 10])$

Loop

- *Simulation using an ordinary differential equation solver:*
  - $y_{k|k-1} = simulation(f_{k-1|k-1})$
- *Finding the linearized matrices:*
  - $A = \delta A$
  - $G = \delta G$
  - $C = \delta C$
- *Calculating the predicted covariance matrix:*
  - $P_{k|k-1} = AP_{k-1|k-1}A^T + GWG^T$
- *Finding the kalman gain:*
  - $e_{k|k-1} = y_k - y_{k|k-1}$
  - $E_{k|k-1} = CP_{k|k-1}C^T + DVD^T$
  - $K_k = \frac{P_{k|k-1}C^T}{E_{k|k-1}}$
- *Finding the corrected state estimate:*
  - $x_{k|k} = x_{k|k-1} + K_k e_{k|k-1}$
- *Finding the corrected covariance matrix:*
  - $P_{k|k} = P_{k|k-1} - K_k E_{k|k-1} K_k^T$
- *Transition up one time step: k=k-1*

When calibrating the kalman filter by adjusting the variance it proved to have some issues at specific samples, but it proved to that it could be hard to create a stable kalman filter. Since the kalman gain should be a theoretical representation of the system, it is important that the measurements given to the model for linearization is correct. As shown in Chapter 3.2.3, some of the measurement is probably incorrect, thus basing the theoretical model on these measurements used by the kalman filter can give wrong results.

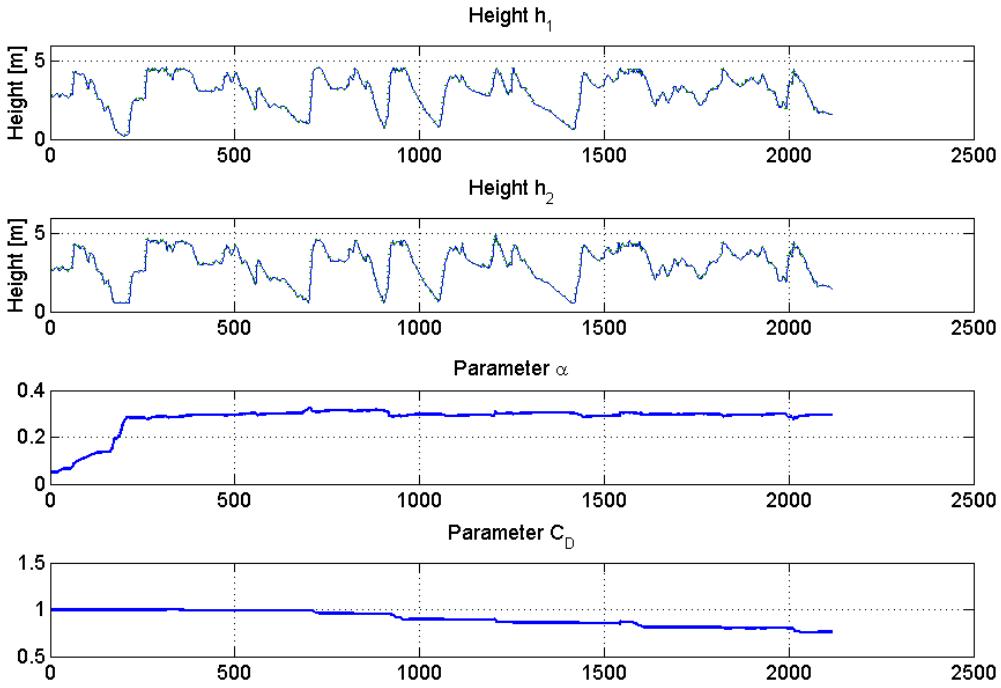


Figure 3-7: Shows the states when using a time varying kalman filter with respect to time [days]

The result shows that α stabilizes around 0.3 from 0.05, but C_D decreases throughout the whole simulation. The C_D parameter represents a coefficient factor on the floodgates, as anything below 0.8 is probably wrong, the results in Figure 3-7 shows that the change on C_D was too unrealistic to be considered. The parameter change is tested in Chapter 3.2.3.

3.2.2.3Steady state kalman filter

A steady state kalman filter was used to create a more stable kalman filter gain since the measurements can create an incorrect kalman gain in a time varying kalman filter. The steady state Kalman Filter algorithm is presented below:

- *Deciding initial matrices and parameters*
  - $G = diag([1 \quad 1 \quad 1 \quad 1])$
  - $W = diag([1 \quad 10 \quad 0.0001 \quad 0.0001])$
  - $V = diag([1 \quad 10])$
- Linearizing the model around steady states values defined in the step above and use it to find the kalman gain

$\dot{V}_{in} = 50$

$\dot{V}_t = 24$

- $hg = 5\%$

$h_1 = 3.5$

$h_2 = 3.2$

$A = \delta A$

- $G = \delta G$

$C = \delta C$

- $K = dlqe$

Loop

- *Simulation using an ordinary differential equation solver:*
  - $y_{k|k-1} = simulation(f_{k-1|k-1})$
- *Finding the corrected state estimate:*
  - $x_{k|k} = x_{k|k-1} + Ke_{k|k-1}$
- *Transition up one time step: k=k-1*

As shown in Figure 3-8, the filter finds obvious measurement errors which were not found by the time varying kalman filter at around sample 200. In Figure 3-8 the initial values of α = 0.05 and C_D = 1 is used. Because the kalman filter in Chapter 3.2.2.2 suggested an α of 0.3, this was tested, giving the result in Figure 3-9.

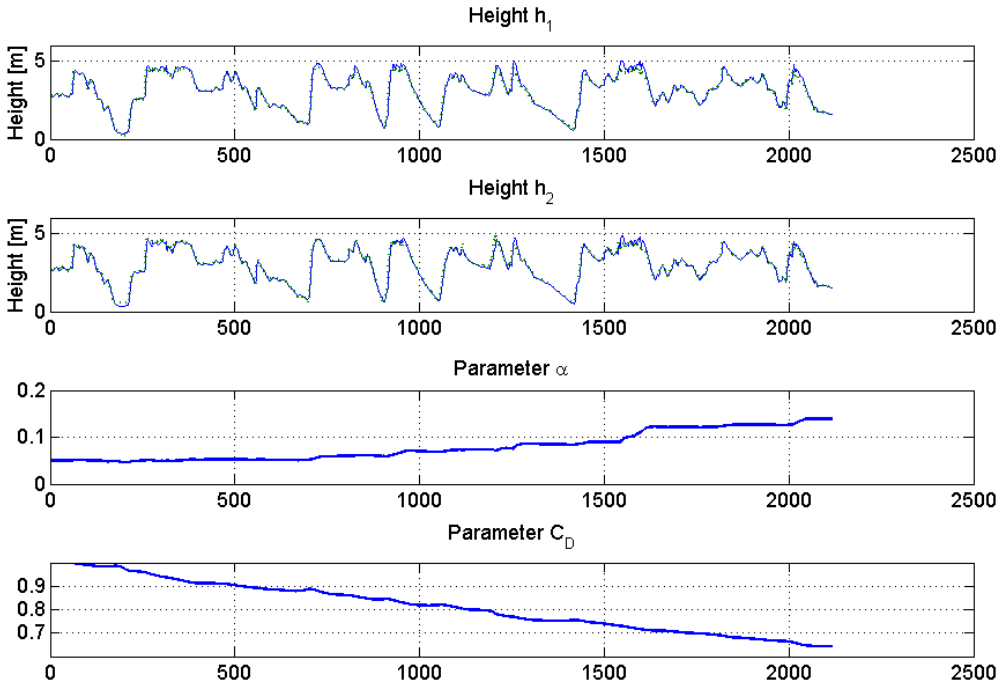


Figure 3-8: Shows the states when using a steady state kalman filter with respect to time [days] and an initial α of 0.05

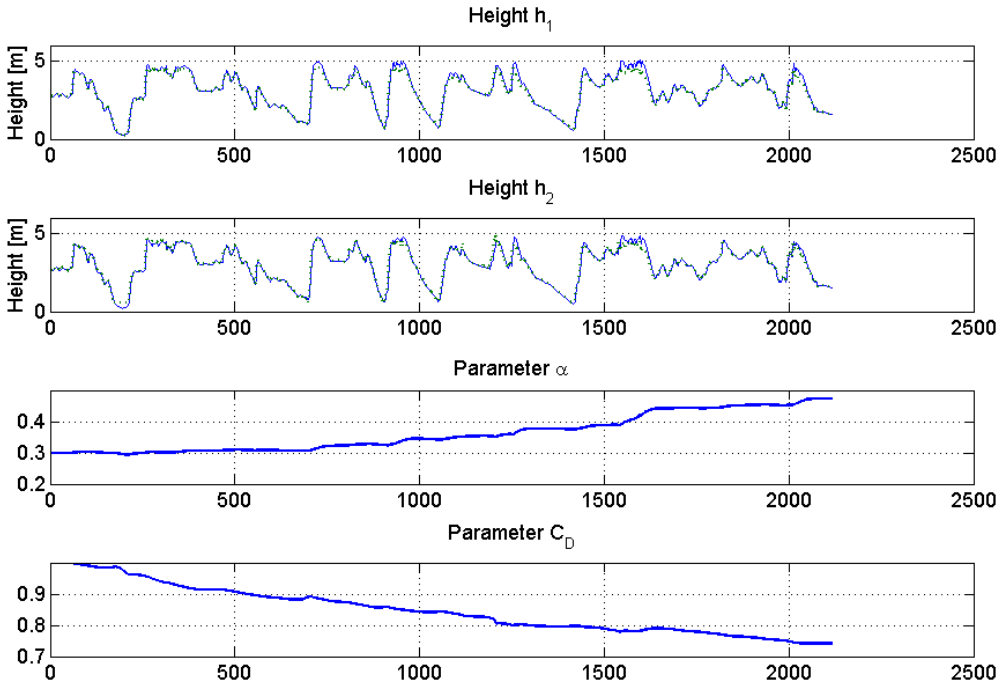


Figure 3-9: Shows the states when using a steady state kalman filter with respect to time [days] and an initial α of 0.3

3.2.3 Results of the parameter estimation on the Lake Toke model

It is shown that the parameter the parameters α, β, λ has not enough impact on the model to form its dynamics sufficiently. The effect is presented in Figure 3-10, Figure 3-11 and Figure 3-12. The parameter C_D is too unstable to make any conclusion from other then that it has a noticeable effect on the model. The parameters are given an overly excessive change and are simulated over 500 days, the goal is to see if there is actually possible to estimate parameters that can have a noticeably effect on the model. The change on the parameters is not noticeable in any of the figures below as the lines are on top of each other. The code used to test the parameters is given in Appendix 9.

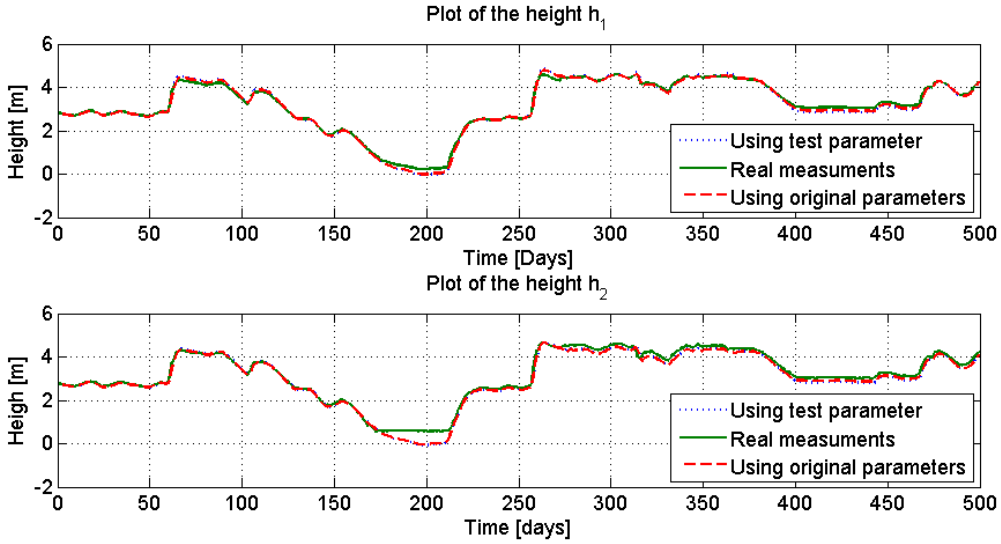


Figure 3-10: Shows the effect on the model changing the parameter α from 0.05 to 0.7

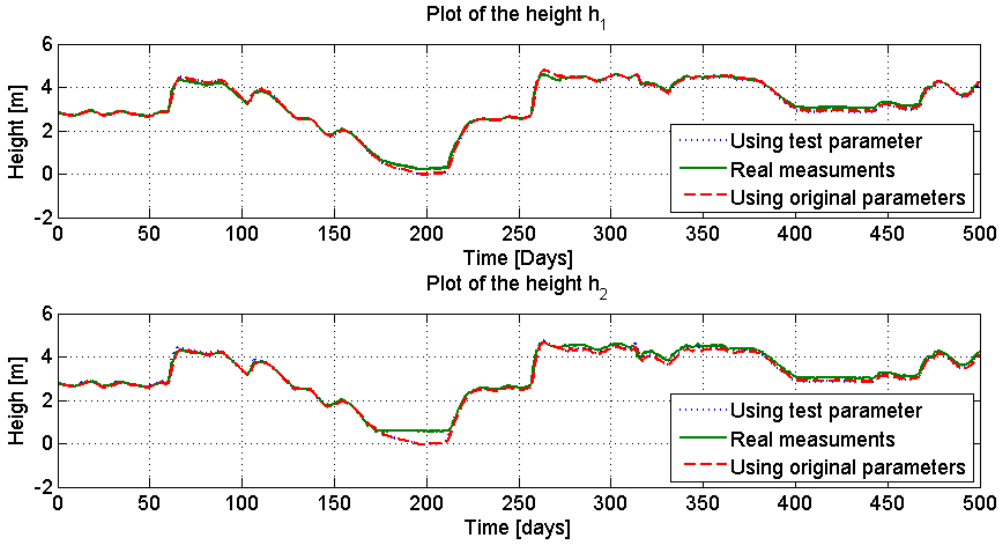


Figure 3-11: Shows the effect on the model changing the parameter β from 0.02 to 0.7

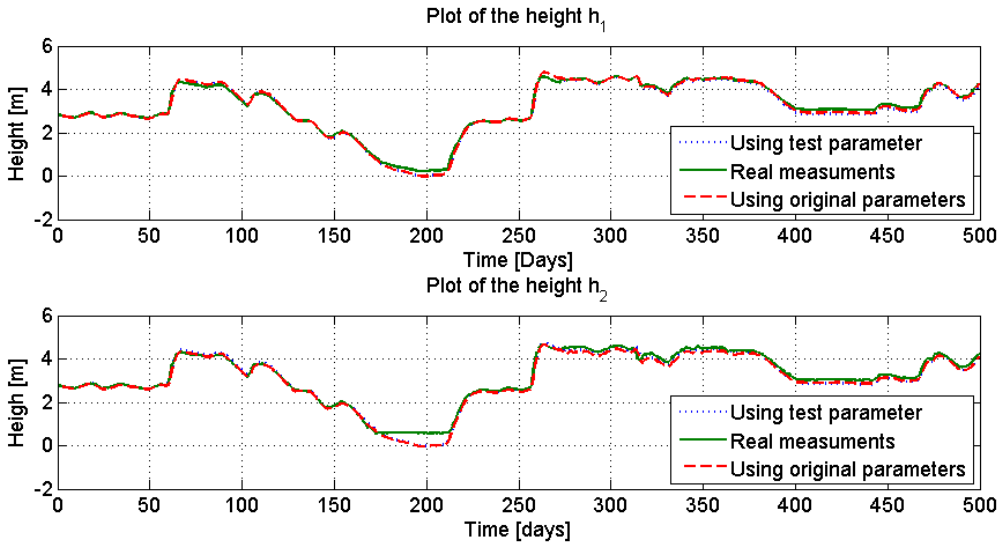


Figure 3-12: Shows the effect on the model changing the parameter λ from 800 to 8000

3.3 Turbine flow model

In order to estimate the flow $\dot{V}_t$, there is assumed a relationship between energy produced, $\dot{W}_e$, and the difference in water level between h_2 and X_Q. The flow is calculated at Skagerak using soft sensor, the same approach should therefore be looked at to calculate V_t. Figure 3-13 shows a spatial description of the turbine system and its respective variables which is used to derive the model for $\dot{V}_t$.

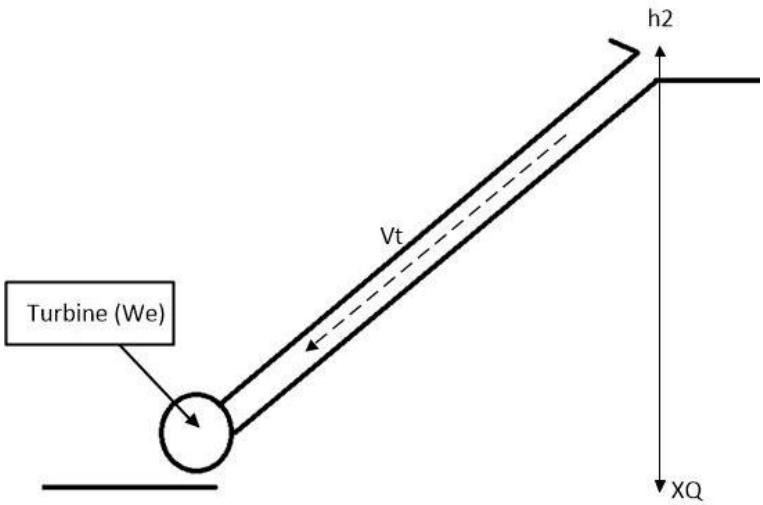


Figure 3-13: Sketch of the turbine system

In order to estimate a flow $\dot{V}_t$, some assumption has been made because of lack of information in the data that is logged from Skagerak. E.g. there is not logged how much the flood gates have been opened, nor if it's being opened at all.

3.3.1 Evaluating the data used to design the turbine flow model

In the data, it is not given if the floodgates have been opened or not. Since the maximum capacity of the turbines is 36 m^3/s, there is assumed that all flows over 36 m^3/s are a result of a floodgate being opened. As a result water flow below 36 m^3/s is $\dot{V}_t$ and the remaining flow over 36 m^3/s is considered as $\dot{V}_g$. There are two different sensors below Dalsfos that measures the level. Figure 3-14 shows the two separate measurements. They should measure approximately the same level, but for unknown reasons there is a sudden drop of 2 meters. Since there is no apparent reason for the level to drop by 2 meters, X_Q is used instead of X_U. The code used to plot Figure 3-14 is given in Appendix 5.

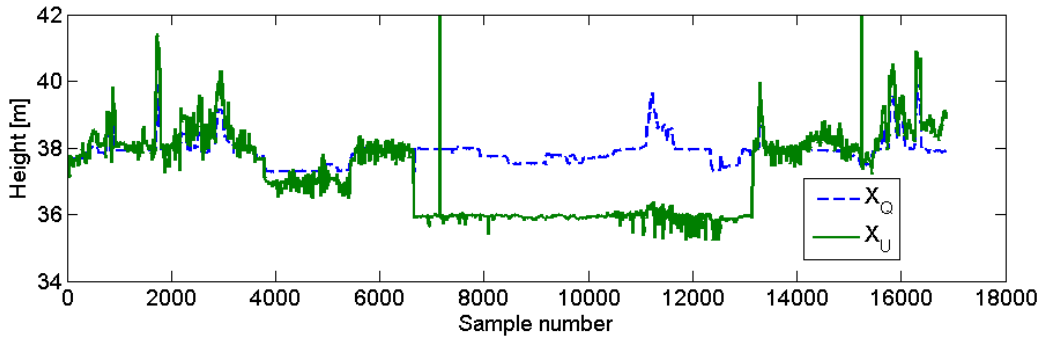


Figure 3-14: Plot showing the difference between X_U and X_Q

There was also some data which is lacking measurements of the level h_2 in 2008, as a result will be excluded from the model. The missing samples are circled in Figure 3-15. The cause of the missing samples may be due to a lot of construction at Dalsfos in the later years.

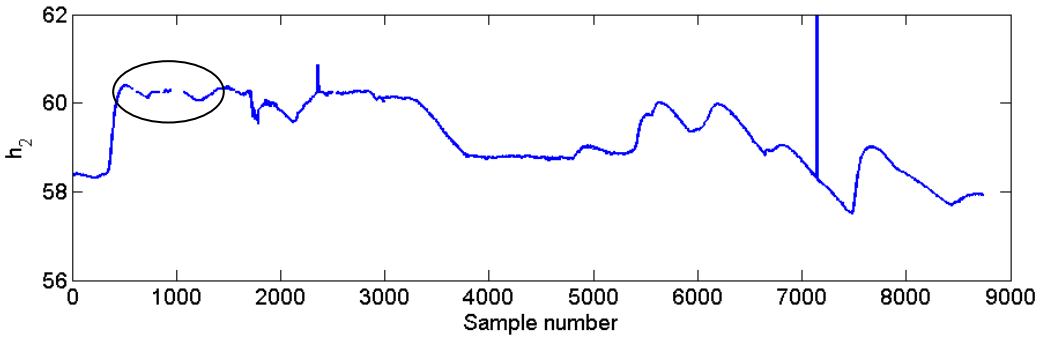


Figure 3-15: Plot of h_2 data samples for 2008

3.3.2 Estimated model

The MATLAB code used to create the turbine flow model is given in Appendix 5. To estimate the model, only the available parameters can be used to estimate the $\dot{V}_t$ flow:

- h_2 – Level above Dalsfos [m]
- $\dot{W}_e$ – Power production [kWh]
- $\dot{V}_g$ – Flow through the flood gate [m^3/s]

A least square method approach is used create the model, instead of choosing a random combination of the above parameters, the method will take into consideration the following general equation for power production:

$$\dot{W}_E = K \times \dot{V}_t \left(h_2 - X_Q \right) \tag{3-11}$$

Since X_Q is not available in the model, X_Q will have to be substituted with another correlated variable. A correlation between $\dot{V}_O$ and X_Q is plotted in Figure 3-16 and examined. There has been used a cubic fit on $\dot{V}_O$ to see the correlation between $\dot{V}_O$ and X_Q[2].

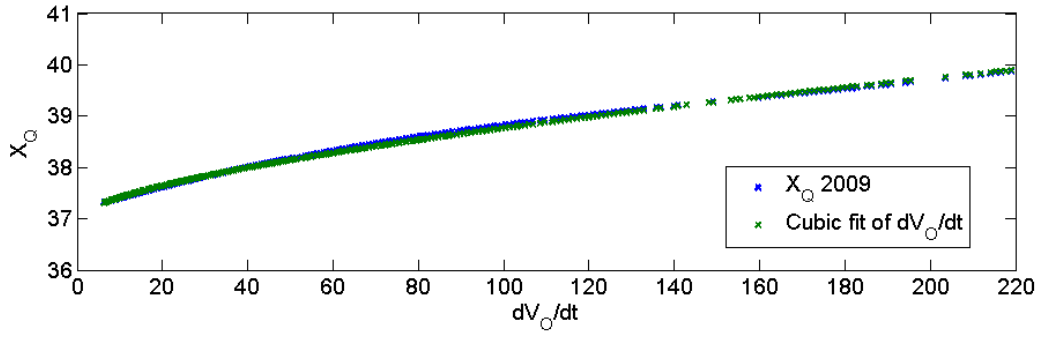


Figure 3-16: Correlation between $\dot{V}_O$ and X_Q

In order to solve this explicitly, $\dot{V}_t$ should be a first order or second order. As a third order or higher equation can give quite a complex answer to the explicit solution. To have a reliable model, an explicit solution is preferred to have control over which roots that should be chosen. The equation to be fitted is:

$$\dot{V}_t = \frac{\dot{W}_e}{(h_2 - X_Q)} \tag{3-12}$$

Replacing X_Q with a fitted equation $X_Q(\dot{V}_t)$:

$$X_Q = c_1 * \dot{V}_t + c_2 \tag{3-13}$$

Although the flow is correlated in the whole range of $\dot{V}_O$, the equation is fitted in the range on the flow from 0 to 36.5 m^3/s. This corresponds to a range on X_Q from 37.3 to 39.3 m. This

[2] The cubic fit was noticed from Bernt Lie in an informal meeting, he mentioned that this information could be exploited to create the turbine flow model.

choice give some error at higher total flow, but the error in flow is a necessary compromise to get an exact $\dot{V}_t$ when $\dot{V}_O$ is lower.

By using Equation (3-12) and (3-13) one gets:

$$c_1 * \dot{V}_t^2 + (c_2 - X_D) * \dot{V}_t = \dot{W}_e \tag{3-14}$$

The equation is solved for $\dot{V}_t$ which gives:

$$\dot{V}_t = c_3 * \frac{-(c_2 - X_D) \pm \sqrt{(c_2 - X_D)^2 - 4c_1\dot{W}_e}}{2c_1} + c_4 \tag{3-15}$$

The least squared method is applied to both roots in Equation (3-15) to calculate the parameters c_3 and c_4, using the data from 2008. Then the model is then validated against the data from 2009. The result from using the *"+ root"* is displayed in Figure 3-17, and *"- root"* is displayed in Figure 3-18.

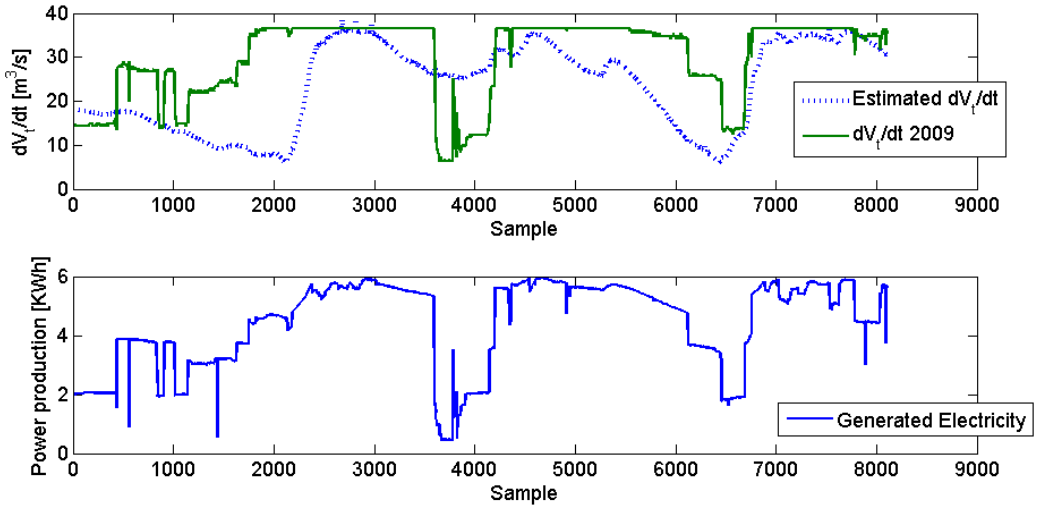


Figure 3-17: Validating the data on samples from 2009, using LSM on equation (3-15),+ root

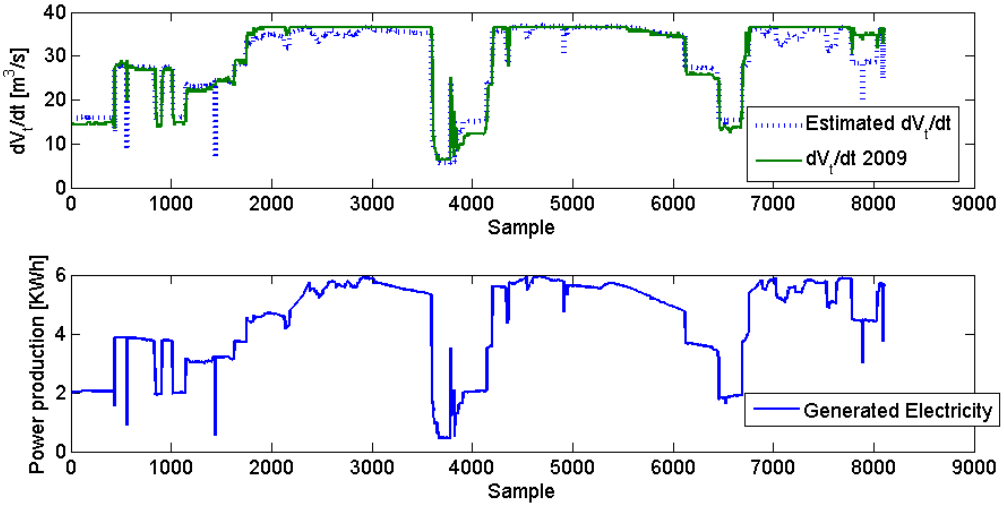


Figure 3-18: Validating the data on samples from 2009, using LSM on equation (3-15),- root

When comparing Figure 3-17 and Figure 3-18, one can see that using the *"- root"*, as shown in Figure 3-18, that this gives a better result. Generally, the model used in Figure 3-18 follows the validation data with some exceptions. Around sample number 8000 there is a sudden drop in the model, while the true value is shown to be 36 m^3/s. The reason for this is probably because one of the turbines where turned off for maintains, as can be seen in the lower plot in Figure 3-18, where the KWh production is lower.

3.3.3 Overview of the turbine flow model

Using the knowledge gained from Chapter 3.3.1 and 3.3.2, one can summarize the model as Equation (3-16) with the parameters defined in Table 3.4.

$$\dot{V}_t = c_3 * \frac{-(c_2 - h_2) \pm \sqrt{(c_2 - h_2)^2 - 4c_1\dot{W}_e}}{2c_1} + c_4 \tag{3-16}$$

Table 3.4: Parameter description of Equation (3-16)

Variable	Value	Unit	Comment
h_2	h_{2_k}	m	The water level h_2
$\dot{W}_e$	$\dot{W}_{e_k}$	kWh	The output of the turbine generator
c_1	*0.0211*	~	Parameter found with the LSM
c_2	*37.1891*	~	Parameter found with the LSM
c_3	*132.0238*	~	Parameter found with the LSM
c_4	*2.8241*	~	Parameter found with the LSM

3.3.4 Linearization and simulation of the turbine flow model

The code used to validate the turbine flow model is given in Appendix 4. Skagerak gives a list of $\dot{W}_e$, hence the level h_2 is the unknown variable to linearize around. As the definition of linearizing a function $F(a,b)$ is $F(a,b) \approx F(a,b) + \Delta f(a,b)$ where $\Delta f(a,b)$ is the linearized function found using the method explained in Chapter 3.1. When applying this to the turbine flow model one gets where only an initial value of h_2 is know:

$$\dot{V}_t(k) = \dot{V}_t(0) + \delta\dot{V}_t(h_2(0), \dot{W}_e(k)) * \left(h_2(0) - h_2(k)\right) \tag{3-17}$$

In Figure 3-19 the linearize model at $h_2 = 3\ m$ the level is linearized at *1 m*. The nonlinear model at $h_2 = 1\ m$ shows the error the $\dot{V}_t$ model would have without the knowledge of the height h_2. From the figure one can see that there will be an error of $4\ m^3/s$ if the level increases from *1* to *3 m* and $\dot{W}_e$ increases from *0* to *5.6 kWh* For comparison the nonlinear model at $h_2 = 3\ m$ shows that the model is nearly linear.

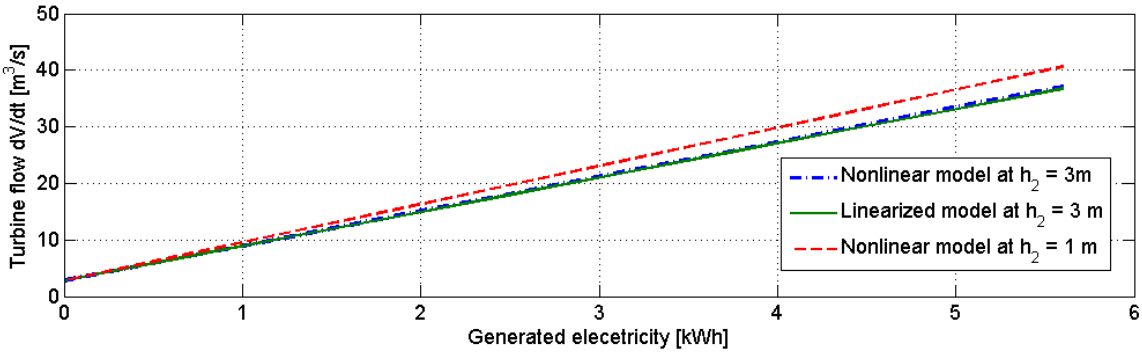


Figure 3-19: Simulated turbine flow to compare the effect the water level h_2

4 Model Predictive Controller

Since solving a QP or LP problem is the core of an MPC, MATLAB is used as this provides the necessary functions as well as being a matrix based programmable language. The MATLAB code for the MPC program is given in Appendix 3. To solve the QP equations, a function called *quadprog* is used to find a feasible solution. Before one can use *quadprog* on a model defined and keep the solution within the constraints, one needs to be redefined the problem as a QP problem. Since *quadprog* uses the equations listen below, the system needs to be redefined so that they fit into the Equations (4-1), (4-2) and (4-3).

$$\min\left(\frac{1}{2}x^T Hx + c^T x, x\right) \tag{4-1}$$

$$A_e x = b_e \tag{4-2}$$

$$A_{ie} x \geq b_{ie} \tag{4-3}$$

To solve the QP problem, Equation (4-1) is minimizes by manipulating the *x* variables. The solution needs to be within Equation (4-3) and on Equation (4-2).The solution is considered unfeasible if the *quadprog* cannot find value for an *x-variable* which doesn't fulfill the constraints. The order which the variables are defined in the MPC program is defined in Equation (4-4) with their respective explanation listed in Table 4.1.

In general there are strict lower and upper limits to h_1, the goal is to see that the controller can keep the system within its constraints without and without becoming unstable.

The performance index *I* needs uses Equation (4-1). Since performance index used by this controller only consists of quadratic terms, the *c* in Equation (4-1) can be ignored.

The soft constraints are explained more detailed in Chapter 4.2.2.1. The variables are chosen inside sometimes to simplify the code, e.g. the *y* variable is exactly the same as the h_1 variable.

To define the variables used to solve the QP problem, and horizon length *N* must be defined. By Skagerak, the available input data is 10 days, thereby giving a horizon length of *N = 10 days*. The data is presumed to be available on an hourly basis, thus giving 240 steps over 10 days.

$$x^T = (u_0^T,...,u_{N-1}^T, h_{1_1}^T, h_{2_1}^T,...,h_{1_N}^T, h_{2_N}^T, e_1^T,...,e_N^T,..., y_1^T,..., y_N^T, S_{\min_1}^T,...,S_{\min_N}^T, S_{\max_1}^T,...,S_{\max N}^T, \dot{V}_{t_0},...,\dot{V}_{t_N}) \tag{4-4}$$

Table 4.1: The variables used in the MPC

Variable	Unit	Comment
u	m	The u represent h_g
h	m	The height h_1 and h_2
e	m	The error from a reference point from h_1
y	m	The output of the system, represents h_1 directly
S_{min}	~	Used as a soft constraint on the lower regulated level
S_{max}	~	Used as a soft constraint on the higher regulated level
$\dot{V}_t$	m^3/s	The turbine flow

4.1 Performance index

There is expected some knowledge about MPC, consequently there is only a brief description of the structure of the program, more detailed code shown in Appendix 3. A performance index used by the MPC is defined in Equation (4-1), only the variables which have a weight other than 0 will be used by the MPC. Although all of the weights are listed in Table 4.2, not all of them will have value given (i.e. some variables in the performance index can be disabled during a simulation) at every test that is performed in Chapter 4.3. Since the weights have no given or predefined value, they are chosen experimentally through the simulations.

$$I = \frac{1}{2} x^T \begin{pmatrix} H_{11} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & H_{22} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & H_{33} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & H_{44} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & H_{55} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & H_{66} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & H_{77} \end{pmatrix} x \qquad (4\text{-}5)$$

The reason why there is a weight on the e variable is to perform simulation where one can test the precision of controller (i.e. no integral error) and to see how well the controller can converge the height to a specified reference point.

Table 4.2: Lists the notations given to the respective weights

Variables	Weight	H_{xx}	Comment
$u_0^T,...,u_{N-1}^T$	$P_0,...,P_{N-1}$	H_{11}	Weight to minimize the flood gate opening
$e_1^T,...,e_N^T$	$Q_1,...,Q_N$	H_{33}	Weight to minimize an error $e_k = r_k - y_k$
$S_{\min_1}^T,...,S_{\min_N}^T$	$Q_{\min_1},...,Q_{\min_N}$	H_{55}	Weight to keep the level h_1 below *hrv*
$S_{\max_1}^T,...,S_{\max N}^T$	$Q_{\max_1},...,Q_{\max_N}$	H_{66}	Weight to keep the level h_1 above *lrv*

4.2 Constraints

To solve a QP problem there must exist a feasible solution that fulfill the constraints described in Equations (4-2) and (4-3). The QP problem exists of equalities which defines what "plane" the solution must exist on. In this MPC program, the equality constraints are used to express model dynamics, e.g. the Lake Toke model and turbine flow model as well as some other variables. The inequality constraints are used the express the boundaries of the system, e.g. upper boundary of the level, or limits on the gate opening h_g. Chapter 4.2.1 and 4.2.2 are simplified explanations which show the structure of the program, the detailed code is in Appendix 3.

4.2.1Equality constraints

The equations listed in Table 4.3 define the equality constraint used in the MPC algorithm. The Equation numbers are used to represent their respective position in the matrix A_e.

Table 4.3: Equations that define the equality constraints used by the MPC program

Equations	Comment	Equation number
$x_{k+1} = Ax_k + Bu_k + M \begin{matrix} \dot{V}_{in_k} \\ \dot{V}_{t_k} \end{matrix}$	*Linearized and discretized state space model of Lake Toke system*	(1)
$y_k = Cx_k$	*Outoput h_1*	(2)
$e_k = r_k - y_k$	*The error based on a reference point*	(3)
$\dot{V}_{t_k} = (h_{2_N} - h_{2_0}) * \dot{V}_{t_k}^{para} + \dot{V}_{t_0}$	*Linearized model of the Turbine flow*	(4)

The equality constraints have is ordered in a specific system which is described below.

Table 4.4: The variables used in the MPC

Matrix A_e	Matrix b_e	Equation number
$A_e = \begin{pmatrix} A_{e,1u} & A_{e,1h} & A_{e,1e} & A_{e,1y} & A_{e,1S_{\min}} & A_{e,1S_{\max}} & A_{e,1\dot{V}_t} \\ A_{e,2u} & A_{e,2h} & A_{e,2e} & A_{e,2y} & A_{e,2S_{\min}} & A_{e,2S_{\max}} & A_{e,2\dot{V}_t} \\ A_{e,3u} & A_{e,3h} & A_{e,3e} & A_{e,3y} & A_{e,3S_{\min}} & A_{e,3S_{\max}} & A_{e,3\dot{V}_t} \\ A_{e,4u} & A_{e,4h} & A_{e,4e} & A_{e,4y} & A_{e,4S_{\min}} & A_{e,4S_{\max}} & A_{e,4\dot{V}_t} \end{pmatrix}$,	$b_e = \begin{matrix} b_{e,1} \\ b_{e,2} \\ b_{e,3} \\ b_{e,4} \end{matrix}$,	(1) (2) (3) (4)

4.2.2Inequality constraints

Inequality constraints are also called hard constraints, compared to the soft constraints in Chapter 4.2.2.1, hard constraints can't be broken. This is important to keep in mind when using hard constraints on variables that the MPC doesn't have full control over, if one is not fully aware of the region which the variable can reach, the solution can become infeasible. A list of the inequality constraints are listed below:

Table 4.5: Equations that define the equality constraints used by the MPC program

Equations	Comment	Equation number
$h_g \leq h_2$	*The gate opening is at the same or lower level as h_2*	(1)
$hg \geq 0$	*The gate opening is always over 0 m*	(2)
$h_1 + S_{\min} \geq h_1^{\min}$	*Defines the lower limit of the level*	(3)
$h_1 - S_{\max} \leq h_1^{\max}$	*Defines the upper limit of the level*	(4)
$\dot{V}_g + \dot{V}_t \geq 4$	*The total flow $\dot{V}_O$ has to be higher than 4 m^3/s*	(5)
$\tilde{\dot{V}}_g + \tilde{\dot{V}}_t \leq V_O^{\max}$	*The change of the flow rate $\dot{V}_O$ cannot exceed a threshold*	(6)

Table 4.6: The variables used in the MPC

Matrix A_e	Matrix b_e	Equation number
$A_{ie}=\begin{pmatrix} A_{ie,1u} & A_{ie,1h} & A_{ie,1e} & A_{ie,1y} & A_{ie,1S_{\min}} & A_{ie,1S_{\max}} & A_{ie,1\dot{V}_t} \\ A_{ie,2u} & A_{ie,2h} & A_{ie,2e} & A_{ie,2y} & A_{ie,2S_{\min}} & A_{ie,2S_{\max}} & A_{ie,2\dot{V}_t} \\ A_{ie,3u} & A_{ie,3h} & A_{ie,3e} & A_{ie,3y} & A_{ie,3S_{\min}} & A_{ie,3S_{\max}} & A_{ie,3\dot{V}_t} \\ A_{ie,3u} & A_{ie,3h} & A_{ie,3e} & A_{ie,3y} & A_{ie,3S_{\min}} & A_{ie,3S_{\max}} & A_{ie,3\dot{V}_t} \\ A_{ie,3u} & A_{ie,3h} & A_{ie,3e} & A_{ie,3y} & A_{ie,3S_{\min}} & A_{ie,3S_{\max}} & A_{ie,3\dot{V}_t} \\ A_{ie,3u} & A_{ie,3h} & A_{ie,3e} & A_{ie,3y} & A_{ie,3S_{\min}} & A_{ie,3S_{\max}} & A_{ie,3\dot{V}_t} \end{pmatrix},$	$b_{ie}=\begin{matrix} b_{ie,1} \\ b_{ie,1} \\ b_{ie,1} \\ b_{ie,1} \\ b_{ie,1} \\ b_{ie,1} \end{matrix},$	(1) (2) (3) (4) (5) (6)

The reason why the upper and lower limit needs to be a soft constraint in Table 4.5 is because one it is a goal to keep the level under the *hrv*, but in extreme circumstances the level might flow over. E.g. if $\dot{V}_{in}$ was underestimated causing a sudden flood, the level can go over *hrv*.

In general in this system, if the constraint is controlled directly through the flood gate h_g, there can be used a hard constraint

The constraints are discussed in Chapter 2.3.1. Above the constraints are represented in a way which is applicable into MPC.

4.2.2.1Soft constraints

Soft constraints are the same and inequality constraints except that they can be broken without giving an infeasible solution. The general definition for a soft constraint for a lower boundary is Equation (4-6), and for the upper boundary Equation (4-7).

$$y_{\min} - S_{\min_k} \leq y_k \tag{4-6}$$

$$y_{\max} + S_{\max_k} \geq y_k \tag{4-7}$$

4.3 Simulation of the MPC

The simulations are based on fictitious values for $\dot{V}_{in}$ and $\dot{W}_e$ and are 60 days simulations each. The disturbances are divided into 8 steps which are equally divided over those 70 days, 60 days with 10 extra days since the horizon is 10 days. The disturbances in between the 8

steps are calculated using a *spline* function in MATLAB which creates a quadratic function between the 8 steps. To separate the simulations from each other, they are presented in separate subchapters where there is an explanation to purpose of each test as well as interesting observations is pointed out. Each simulation is presented with graph of both the control output plotted with the water level at Dalsfos h_2 and a plot of the water levels h_1 and h_2. There is also a plot of the disturbances $\dot{V}_{in}$ and $\dot{W}_e$. Most simulations are simulations where there are similar conditions as the simulation in Chapter 4.3.1, this is to see how controller reacts to either extreme situations or how some parameters or weights affect the controller.

4.3.1Observation of a realistic flood outcome

There is performed a simulation to observe a realistic response, the result is shown in Figure 4-1 with the initial water level of $h_1 = 1m$ and $h_2 = 0.8m$. There is up both upper and lower boundary as well as including the flood gate h_g to the performance index. The initial values are chosen out the statistic data given by Skagerak, it shows that before the flood season h_1 is normally *1 m* above *lrv*. The $\dot{V}_{in}$ is also chosen with its amplitude and gradient according to the data from Skagerak. The disturbance is shown in Figure 4-2. The result shows that it is able converge itself to *hrv* with a slight overshoot. One can observe that the boundary of the $\ddot{V}_O \leq \ddot{V}_O^{max}$ constraint is quickly reached, as this the value of $\ddot{V}_O^{max}$ is not given by Skagerak or the supervisor the an value is chosen such that the effect of the constraint is visible.

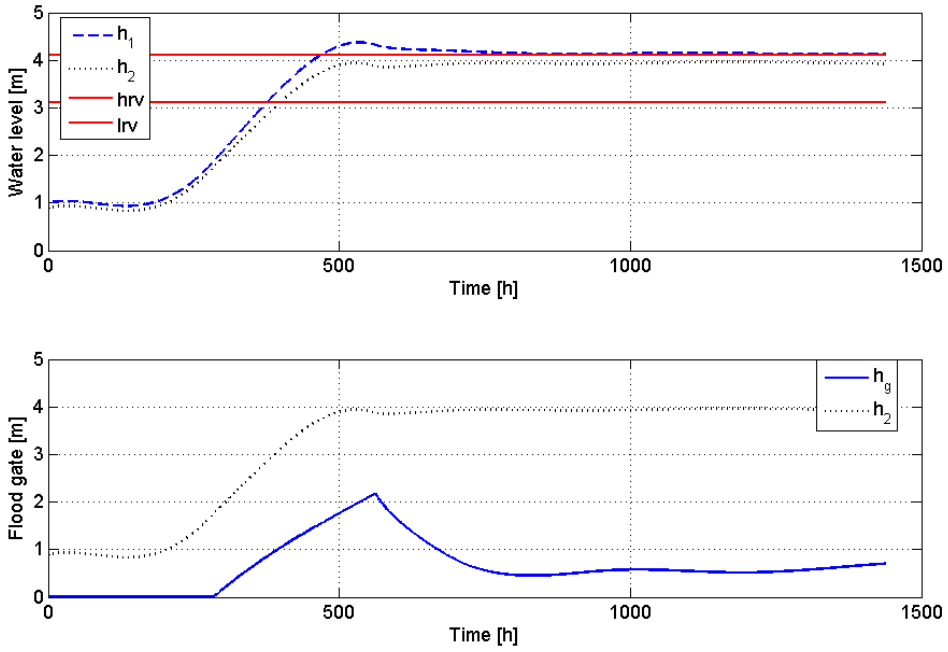


Figure 4-1: Simulation performed to observe a flood season with the following initial states and parameters: $h_1(0) = 1$ m and $h_2(0) = 0.8m$, $Q_{min} = 10$, $Q_{max} = 10$, $P = 1$, $\ddot{V}_O^{max} = 0.7m^3/s.^2$

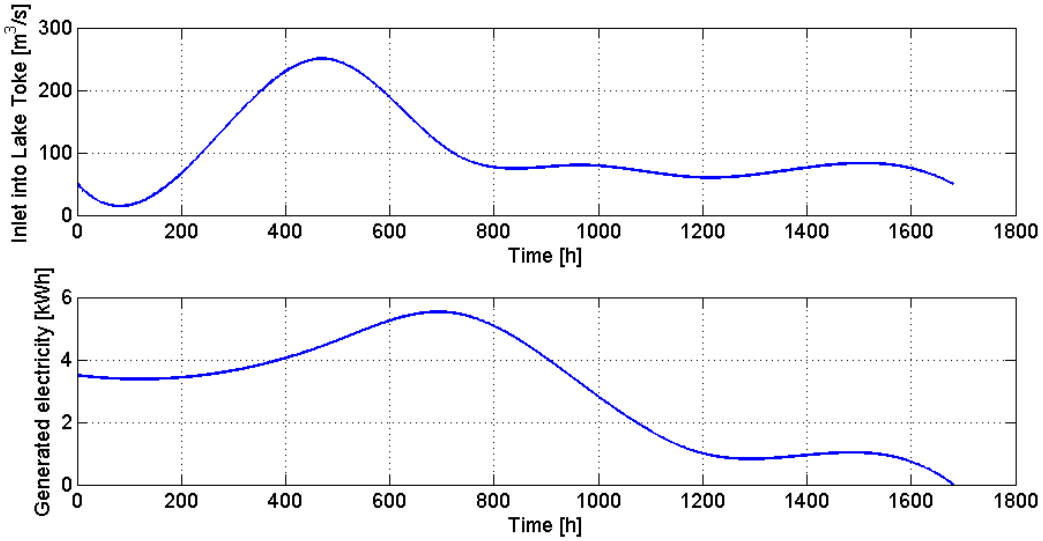


Figure 4-2: Disturbances used in the simulation presented in Figure 4-1.

4.3.2Large weight on the h_g variable in the performance index

If the ratio of weight on the control output h_g and the upper and lower boundary *lrv* and *hrv* is around 1, e.g. the weights in this simulation is set to *10* for both *P*, Q_{min} and Q_{max}, the controller will create a relatively large overshot. The overshot is unnecessary as the conditions for the simulations are exactly the same except for the weights. The result is shown in Figure 4-3 and the disturbances in Figure 4-4.

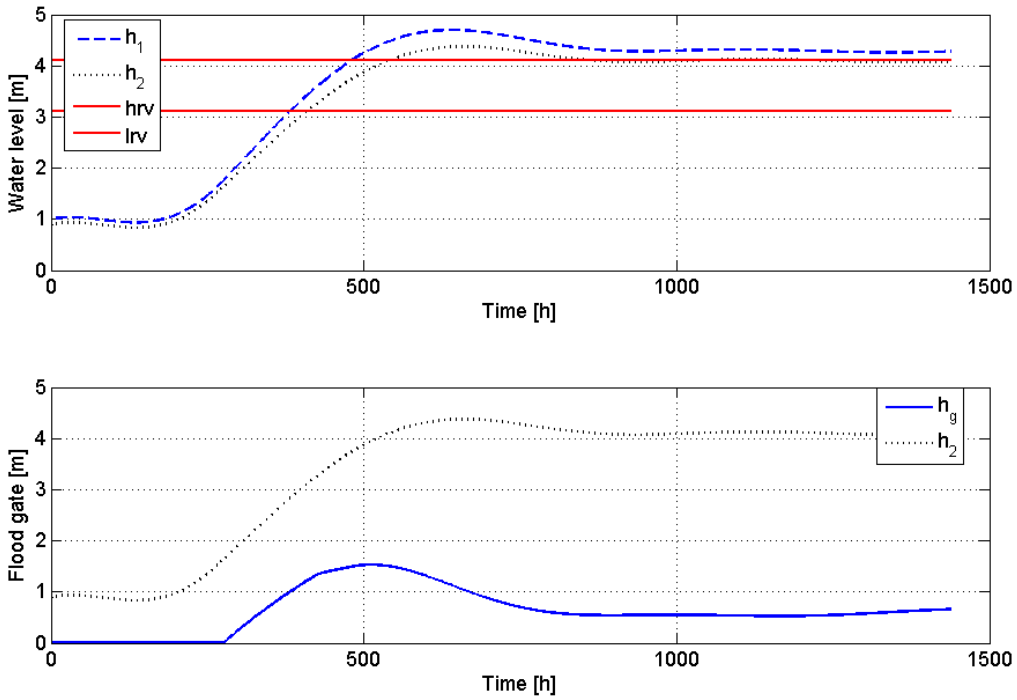


Figure 4-3: Simulation performed to observe a flood season with the following initial states and parameters: $h_1(0) = 1$ m and $h_2(0) = 0.8$m, $Q_{min} = 10$, $Q_{max} = 10$, $P = 10$, $\ddot{V}_O^{max} = 0.7 m^3/s.^2$

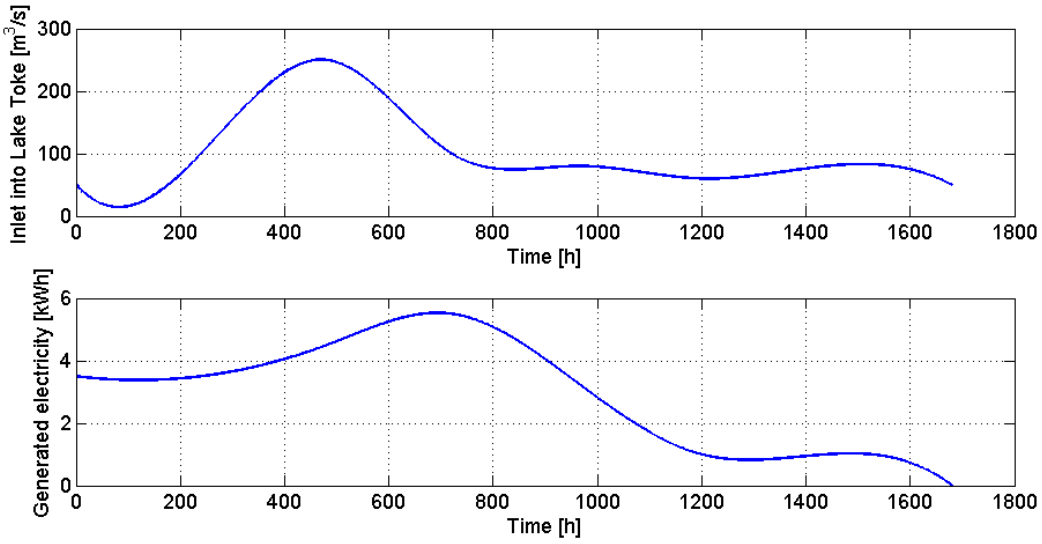


Figure 4-4: Disturbances used in the simulation presented in Figure 4-3.

4.3.3Disabling the $\ddot{V}_O \leq \ddot{V}_O^{max}$ constraint and/or the weight *P*

This simulation was primarily done to see how the weight *P* affects the response from the controller, see let the controller choose the optimal h_g the constraint $\ddot{V}_O \leq \ddot{V}_O^{max}$ was also

disabled. The result from this is shown in Figure 4-5 with the disturbances Figure 4-7. From the figure one can see that the controller chooses keep the level in the middle of *lrv* and *hrv*.

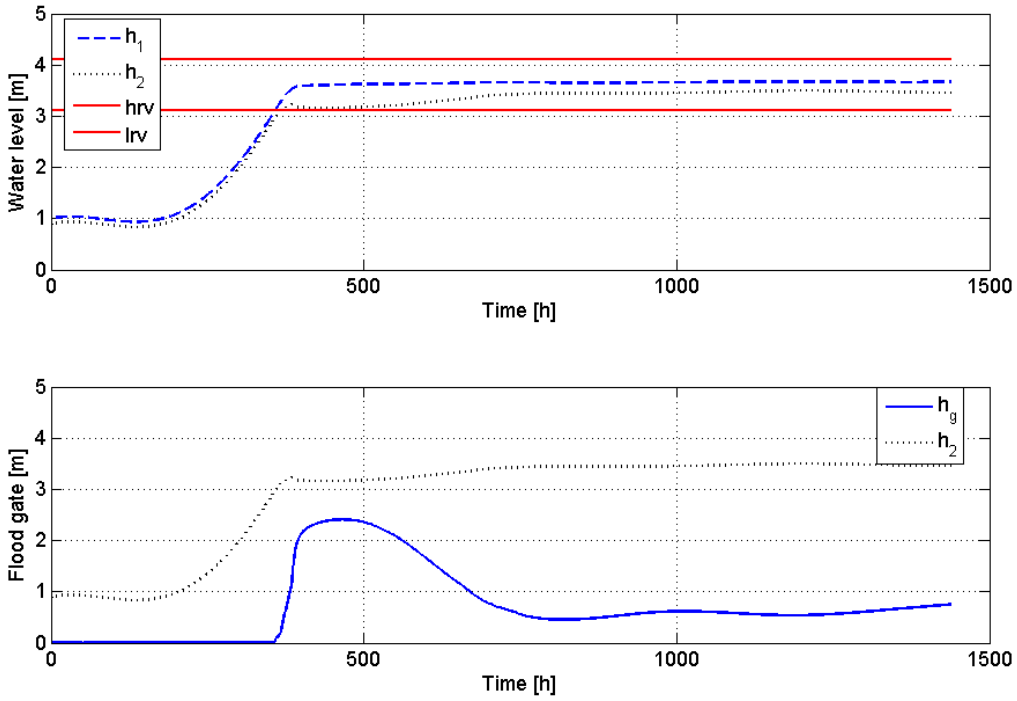


Figure 4-5: Simulation performed to observe a flood season with the following initial states and parameters: $h_1(0)$ = 1 m and $h_2(0)$ = 0.8m, Q_{min} = 10, Q_{max} = 10, P = 0, $\ddot{V}_O^{max}$ =1000 $m^3/s.^2$

Comparing the results from Figure 4-5 and Figure 4-6 where the constraint $\ddot{V}_O^{max}$ is set to *0.7* $m^3/s.^2$, the result shows a slight overshot. The level h_1 will stabilizes in both simulations in the middle of *lrv* and *hrv*. A reason why the level stabilizes in the middle of *lrv* and *hrv* might be because the soft constraint variables S_{min} and S_{max} are given very small values (e.g. 10^{-9} values) even though they are within the constraints, the values are increased slightly when nearing the boundaries.

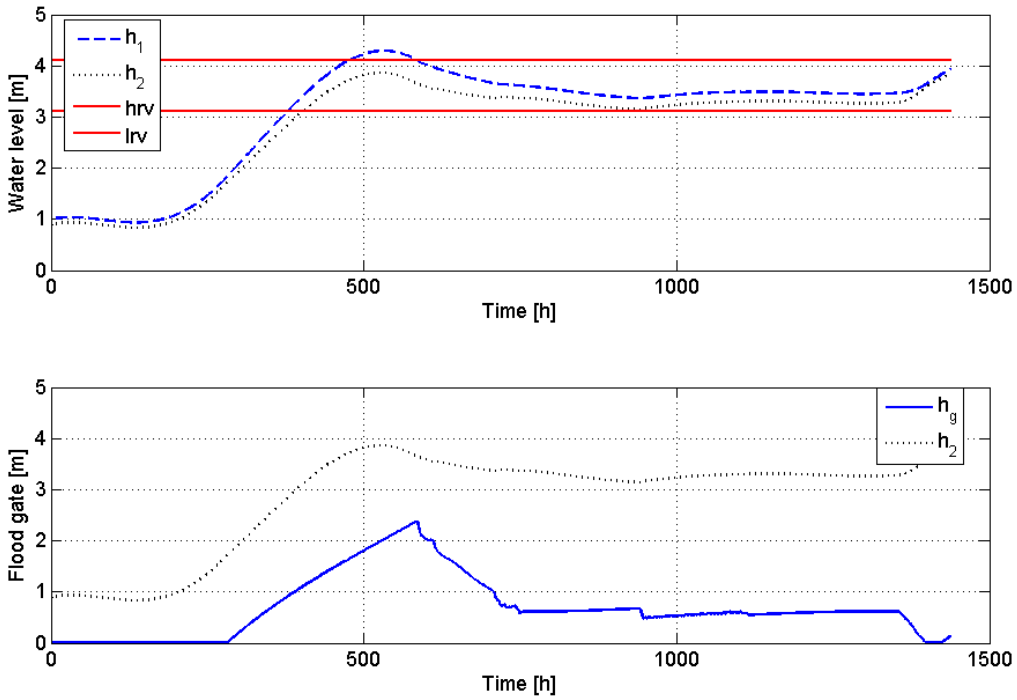


Figure 4-6: Simulation performed to observe a flood season with the following initial states and parameters: $h_1(0) = 1$ m and $h_2(0) = 0.8$m, $Q_{min} = 10$, $Q_{max} = 10$, $P = 0$, $\ddot{V}_O^{max} = 0.7\ m^3/s.^2$

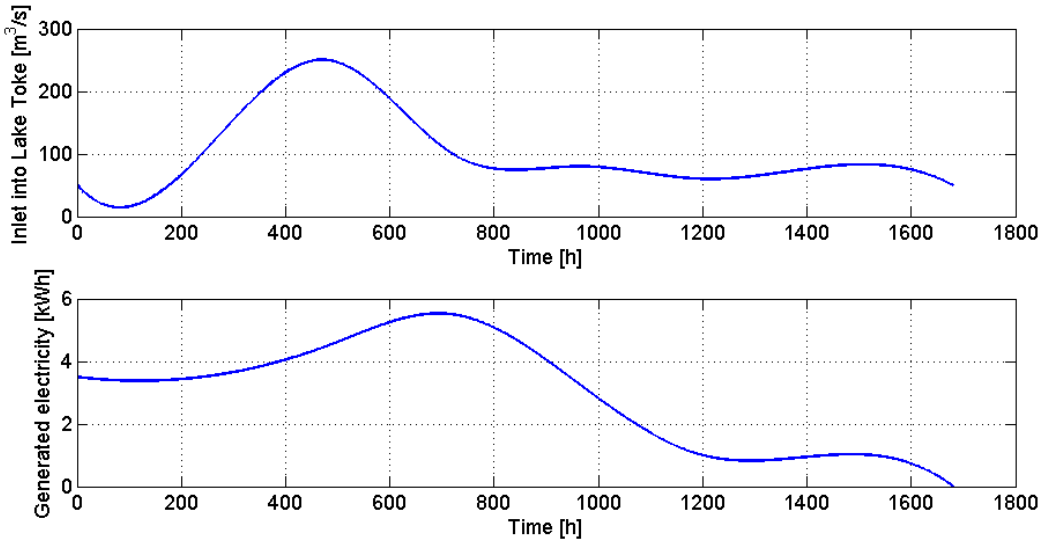


Figure 4-7: Disturbances used in the simulation presented in Figure 4-8.

4.3.4Disabling the $\ddot{V}_O \leq \ddot{V}_O^{max}$ constraint

Since the $\ddot{V}_O^{max}$ might be larger than what was used in the simulation in Chapter 4.3.1. When practically turning the constraint off by increasing $\ddot{V}_O^{max}$ to $1000m^3/s^2$ the controller increases faster as expected, but as shown in the simulation in Chapter 4.3.3 the weight P is causing the difference on the effect from

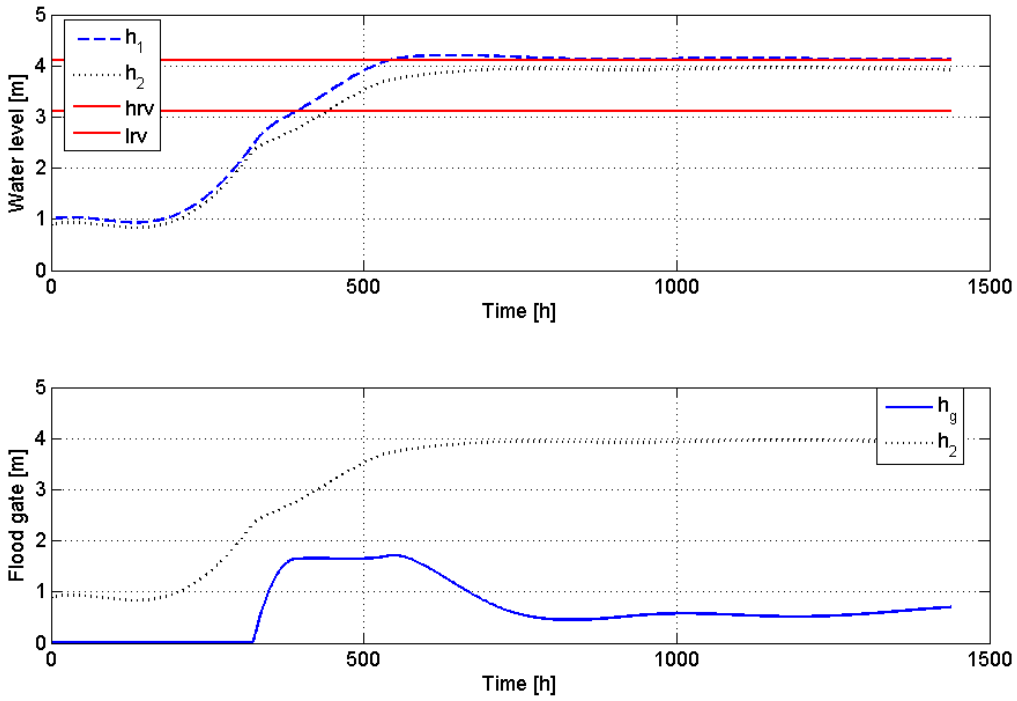


Figure 4-8: Simulation performed to observe a flood season with the following initial states and parameters: $h_1(0) = 1m$ and $h_2(0) = 0.8m$, $Q_{min} = 10$, $Q_{max} = 10$, $P = 1$, $\ddot{V}_O^{max} = 1000\ m^3/s^2$.

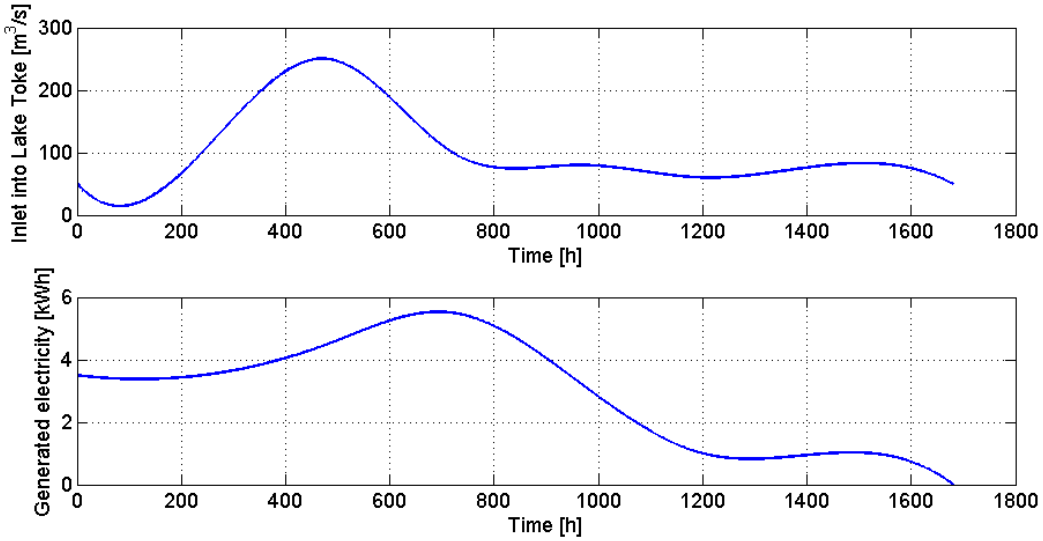


Figure 4-9: Disturbances used in the simulation presented in Figure 4-8.

4.3.5Disabling the lower boundary *lrv*

To disable the lower boundary *lrv* the boundary is set to *0 m*. What is shown in Figure 4-10 is that the controller handles an overshoot better when there the level is under *lrv*. This makes sense since the weight Q_{min} and Q_{max} is both *10*, and the level starts at *3 m* below *lrv*, the level needs to reach the lower limit because fast which also creates an overshoot. The weight Q_{min} can be used as an adjustment based on what Skagerak sees as most important, e.g. is it more important to reach the level *lrv* faster or is it more important to keep the level below *hrv*.

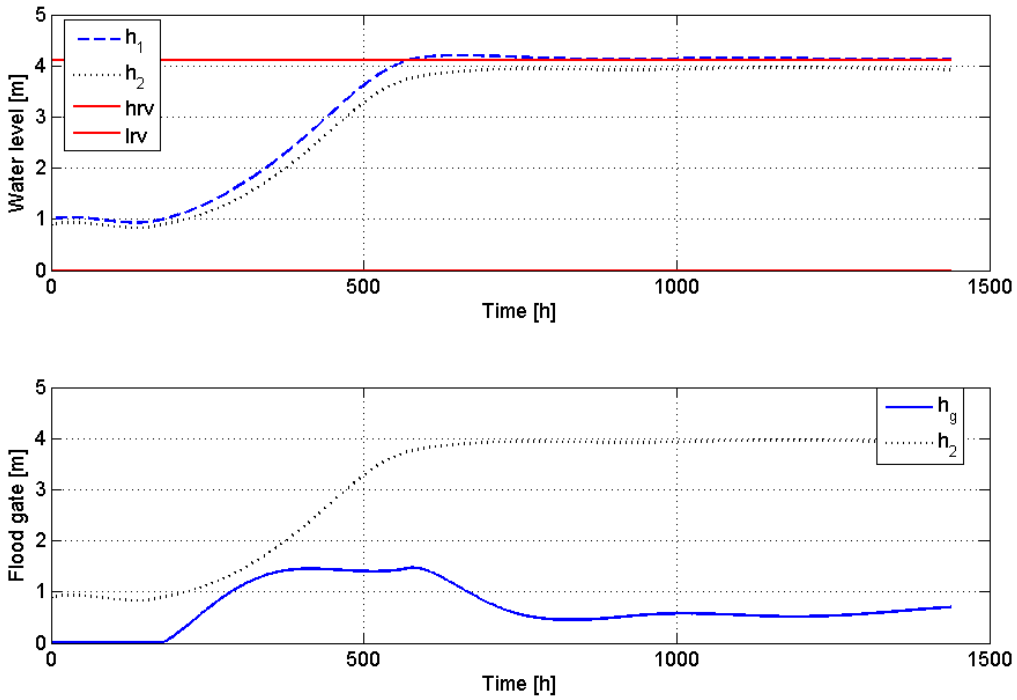


Figure 4-10: Simulation performed to observe a flood season with the following initial states and parameters: $h_1(0)$ = 1m and $h_2(0)$ = 0.8m, Q_{min} = 10, Q_{max} = 10, P = 0, $\ddot{V}_O^{max}$ =0.7 m^3/s^2(lrv is set to 0 m).

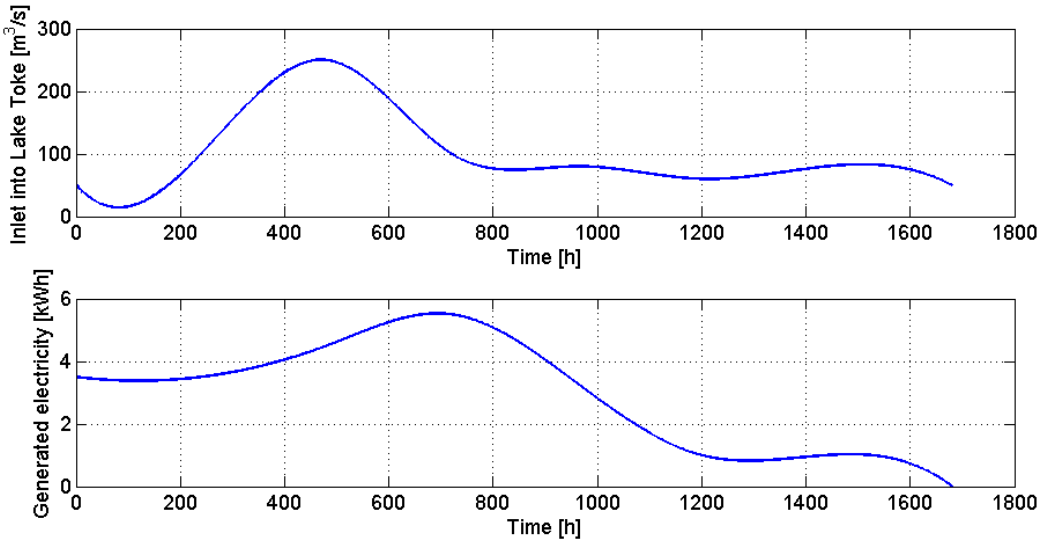


Figure 4-11: Disturbances used in the simulation presented in Figure 4-10.

4.3.6Using the results from the other simulations

Using the results from the previous simulations, and with assumptions about what Skagerak emphasizes. As mentioned in Chapter 2.3.2, they probably emphasizes the ability to handle a

flooding situation as addition to using as little water as possible from the Lake Toke reservoir. In all of the simulations below, the weight on Q_{min} is adjusted to *1*, Q_{max} is still *10*.

4.3.6.1Using a low Qmin and disabling the weight P

The level should probably be in between *lrv* and *hrv* to keep a healthy safety margin such that the controller has time to react to a sudden rainfall outside the flood season. There should also be a focus on avoiding an overshoot if possible, the importance of reaching *lrv* is probably less vital. Though the water flow through the gate is important to keep as low as possible, the weight *P* on h_g is removed in the simulation showed in Figure 4-12 with the disturbances in Figure 4-13.

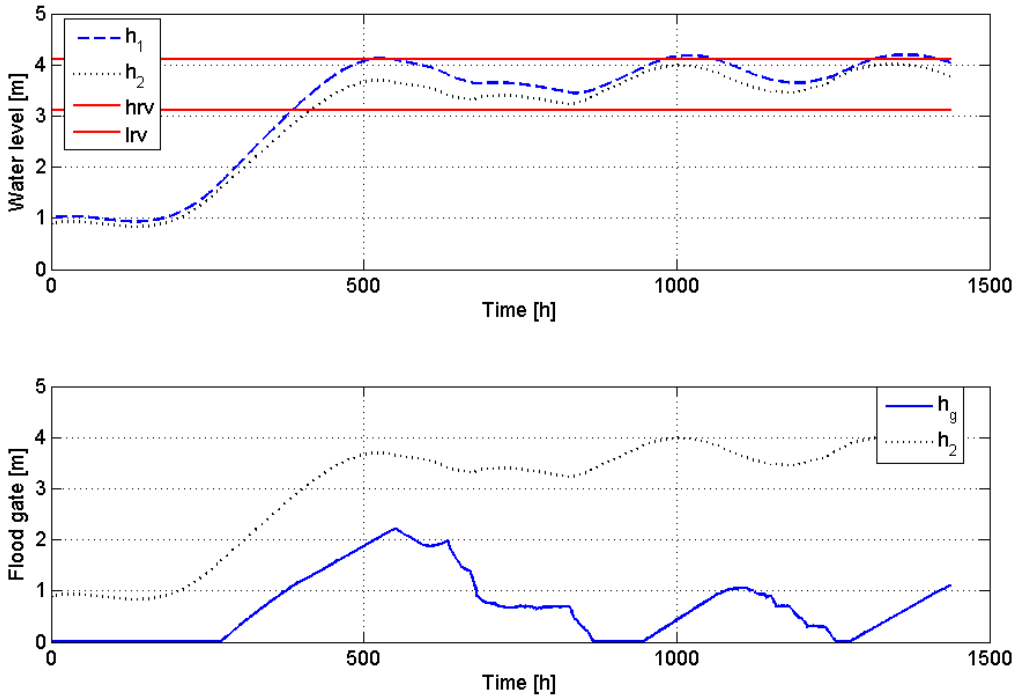


Figure 4-12: Simulation performed to observe a flood season with the following initial states and parameters: $h_1(0)$ = 1m and $h_2(0)$ = 0.8m, Q_{min} = 1, Q_{max} = 10, P = 0, $\ddot{V}_O^{max}$ =0.7 m^3/s^2.

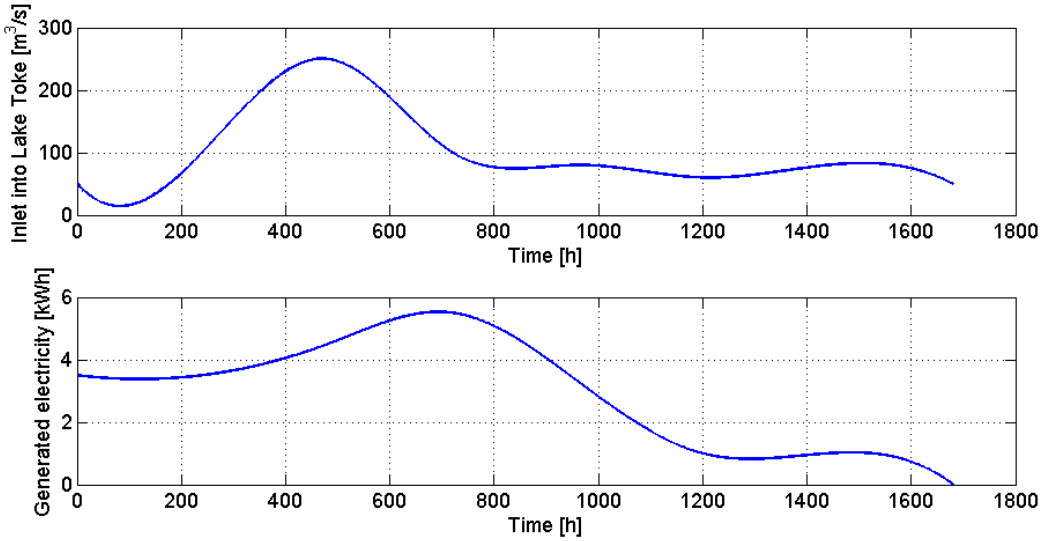


Figure 4-13: Disturbances used in the simulation presented in Figure 4-12.

4.3.6.2Simulations with a sudden rainfall after the flood season

To see how the controller handles a sudden rainfall after the flooding season, there is added a secondary peak in $\dot{V}_{in}$ shown in Figure 4-16. The goal of this simulation is to prevent an overshoot after the simulated rainfall. There are done two simulations, one with a weight *P* of *0* shown in Figure 4-14 and one with a weight *P* of *1*.

The Figure 4-14 showed a slight overshoot, the controller applies its maximum control output within the constraints but cannot prevent an overshot.

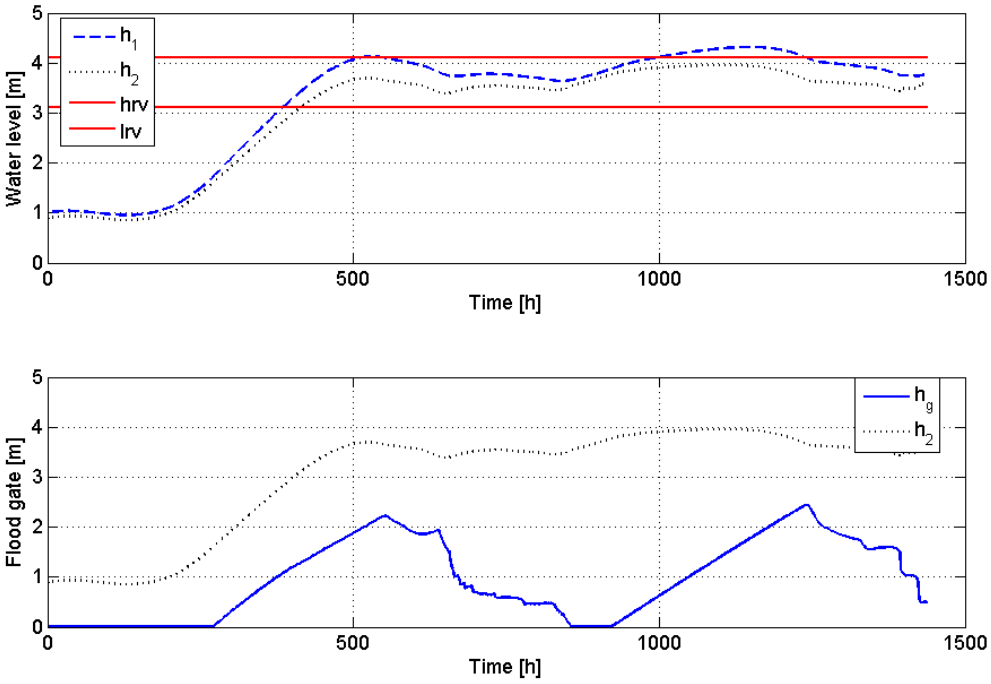


Figure 4-14: Simulation performed to observe a flood season with the following initial states and parameters: $h_1(0) = 1m$ and $h_2(0) = 0.8m$, $Q_{min} = 1$, $Q_{max} = 10$, $P = 0$, $\ddot{V}_O^{max} = 0.7\ m^3/s^2$.

To comparison the simulation with and without a weight on *P*, the result with a *P* of *1* is shown in Figure 4-15. This result shows no overshoot after the rainfall. What can be observed from the control output h_g is that the h_g is more stable compared to not having a weight on *P*. The controller should in theory be more effective at releasing less water through the flood gate, the fact that there was no overshoot in Figure 4-15 might be because the gate was already closed in Figure 4-14 before the rainfall, when in Figure 4-15 the flood gate had a slight opening.

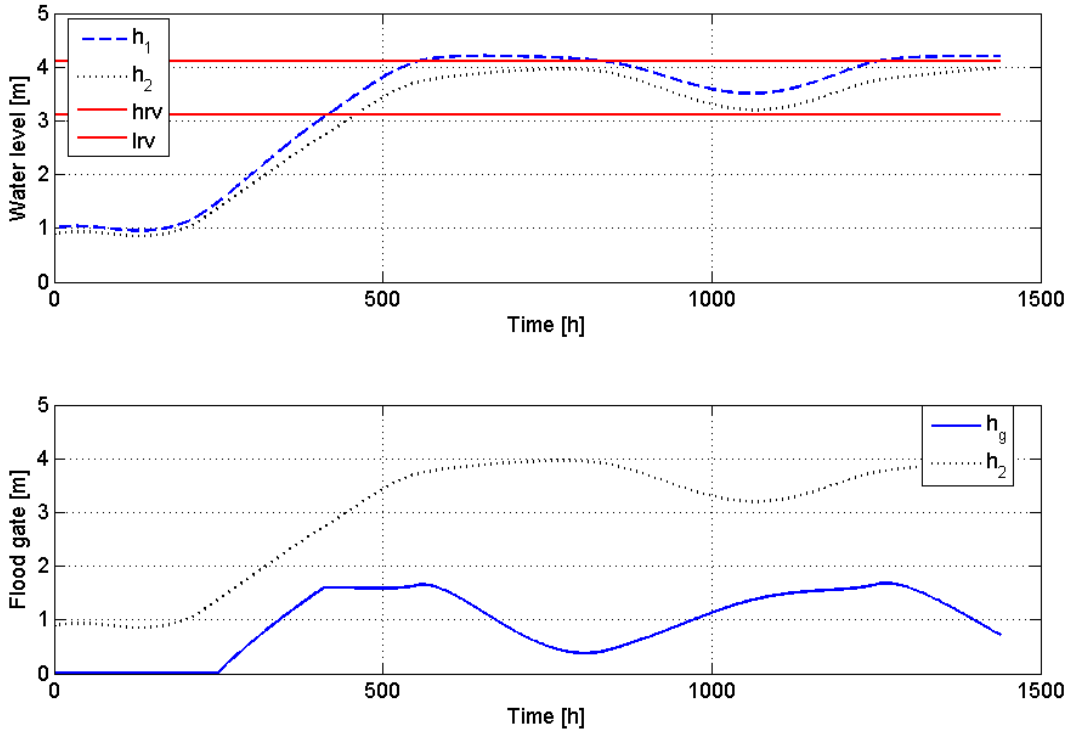


Figure 4-15: Simulation performed to observe a flood season with the following initial states and parameters: $h_1(0) = 1m$ and $h_2(0) = 0.8m$, $Q_{min} = 1$, $Q_{max} = 10$, $P = 1$, $\ddot{V}_O^{max} = 0.7\ m^3/s^2$.

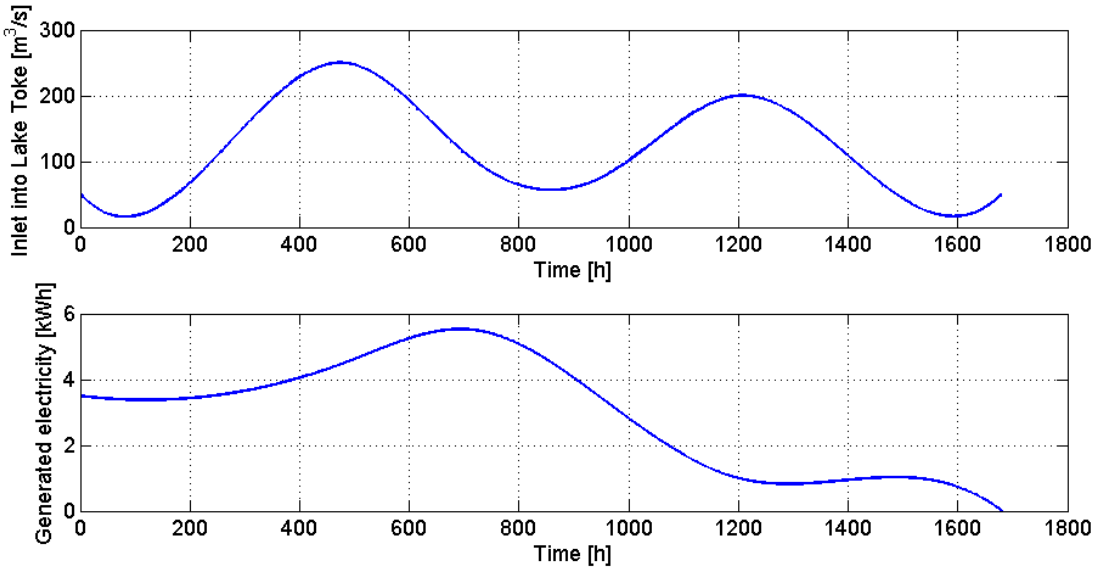


Figure 4-16: Disturbances used in the simulation presented in Figure 4-14 and Figure 4-15.

5 Discussion

This chapter includes discussions about problems that occurred and what might be the cause of these problems. There will also be suggestions for future work, i.e. obvious improvements after the MPC program was designed and tested.

5.1 Parameter estimation and Kalman Filter

In Chapter 3.2.2 there was attempt at adjusting the parameters to create a better model to increase the precision of the control system. It was discovered that the several of the parameters had very little impact on the model, and that the inflow $\dot{V}_{in}$ and the outflow $\dot{V}_g$ was influencing the model substantially more than the other parameters. To create a model it is essential for a parameter estimation that the flow measurement, level measurement and estimated inflow is correct. There should also be a log of how much the gate is opened, not just the total outflow $\dot{V}_O$. It is also shown in Chapter 3.2.2.3 that there are a lot of obvious measurement errors. The level seems to stagnate, when in fact the Kalman Filter showed that the level should continue decreasing. In order to estimate a better model, better samples has to be made such, and perhaps to have enough adjustment options, design a new model or introduce new parameters.

Another application of a more precise model is that this can be used in a Kalman Filter to give a warning whenever the level diverges away from its natural path.

5.2 Suggestions for the MPC

It is a possibility to use only compute hourly controller output for the first day, and then compute a single controller output for the remaining 9 days. This could be used as a tuning option to force the MPC to reach a steady state faster. This could also cause a problem as the controller output's amplitude is a function of the height h_2, thus the controller will only be able to give a control output that is the lowest of the modeled level h_2 during these 9 days.

During the simulations done in Chapter 4.3.6.2 the control output h_g seemed more stable and prevented an overshot better than the other simulations when there was a weight on h_g variable in the performance index. The only problem with this was that it only kept the level h_1 stable on the upper level *hrv*. One way to counter this can be to introduce another upper limit lower than *hrv*, this limit should have a lower weight than the weight on *hrv*. Doing this might create some margin rather than having the level right on the boundary at all time.

It showed that controller gave less overshot when disabling the *lrv*, this lower limit should maybe be disabled until the level reaches higher than *lrv* to give better result.

There hard constraint $\ddot{V}_O \leq \ddot{V}_O{}^{max}$ should maybe be a soft constraint since there is no control over the $\dot{V}_t$, in this thesis there is assumed that Skagerak will adjust the $\dot{V}_t$ according to their regulations on $\ddot{V}_O{}^{max}$.

6 Conclusion

The objective of this thesis is to study the possibility of a deterministic flood control to control the water level at Lake Toke using MPC.

A 2 state nonlinear model designed by Bjørn Glemmestad has undergone a parameter estimation to further improve the model in Chapter 3.2.2. The parameters α, β, C_D and λ was found intuitively when the model was designed. A sensitivity analysis showed that the dynamics of the model was dominated by the influx $\dot{V}_{in}$ and the outflow $\dot{V}_g$, still a parameter estimation was performed using a Kalman Filter. The time varying Kalman filter showed in effective as the Kalman Gain was based on the measurements used in the linearization, this cause the Kalman Filter to receive errors as the model was unreliable. The steady state Kalman Filter proved more effective, but the results showed that there is no change in any of the parameters α, β, C_D and λ that will improve the model.

To implement the nonlinear model into MPC the model was linearized around operation points. To assess the linearized models, simulations were performed in Chapter 3.2.1. Simulations where performed with both ascending water level as well as descending water level using steady states at initial value to see how precise the linearized model predicts the water level over 5 days, the linearization was validated against the nonlinear model of Lake Toke. The simulation will start with an initial value of *1 m* and end at *3 m* when ascending or vice versa when descending. When the linearized model was simulated with an ascending slope the level hit approximately *0.5 m* higher, and when simulating with a descending slope the linearized model proved to be relatively accurate. The fact that the linearization hit higher when ascending can be concluded as a healthy safety margin to prevent a flood during the flooding season, as the controller will foresee the flood to happen before it actually happens. The turbine flow was also modelled as a function of the level at Dalsfos h_2 and the generated electricity $\dot{W}_e$. The level h_2 proved to influence the model by *4* m^3/s when the level differentiated by *2 m*.

To evaluate the controller several simulations was performed with a disturbances similar to a realistic flooding scenario. A fixed horizon of 10 days was used. To test the limits of the controllers as well as tuning it there was performed tests by varying the upper and lower limits weights Q_{min} and Q_{max} as well as the weight on the flood gate P. The simulations proved that the Q_{min} should be smaller than Q_{max} and that a weight on P not only preserves the reservoir at Lake Toke more, but also created a more stable control output and less overshoot. One issue that the P weight was causing was that the level tends to stagnate around the upper limit. To counter this there could be added several upper limits like a hierarchy to fine tune the controller to stabilize between *lrv* and *hrv*.

References

1. RE, RA, TØ. Study the control of the Kragerø waterways in Lake Toke using MPC. [A report written on a project task during a MPC course]. In press 2013.

2. Toke [Internet]. Wikipedia (2013). Cited 29.01.2014. Available from: http://no.wikipedia.org/wiki/Toke.

3. Rosvold KA. Dalsfos Kraftverk (2013). Cited 29.01.2014. Available from: http://snl.no/Dalsfos_kraftverk.

4. Vassdraget og regulering [Internet]. Kragerø Vassdraget (2013). Cited 29.01.2014. Available from: http://www.kragerovassdraget.no/kragerovassdraget/Vassdraget-og-regulering/cid/27654/.

5. Lie B. Predictive Controle with Implementation. [A project task at College University of Telemark]. In press 2013.

Appendices

Appendix 1: Task description

Appendix 2: Project description 3. Semester 2013

Appendix 3: MPC

Appendix 4: Testing of the turbine flow model

Appendix 5: Designing the turbine flow model

Appendix 6: Kalman Filter

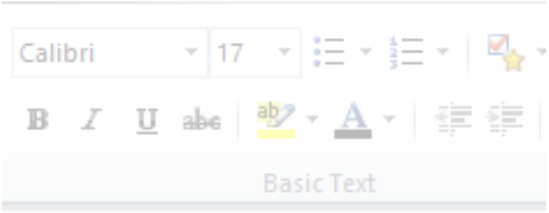
Appendix 7: Simulations of the linearized Lake Toke model

Appendix 8: Testing the sensitivity of the parameters in the Lake Toke model

Appendix 9: Testing the effect of the change on parameters

Appendix 1

Telemark University College
Faculty of Technology



FMH606 Master's Thesis

Title: Deterministic Flood Control using MPC of the Kragerø Waterways

TUC supervisor: Bernt Lie, prof., Telemark University College

External partner: Skagerak Energi, contact: Ingvar Andreassen

Task description:
The following tasks should be carried out:

1. A functional description should be given of a planned Kragerø Waterways flood control system, and the quality of the information should be ascertained.
2. Necessary measurements/information for managing floods in Lake Toke should be described (level measurements, current and future inflow predictions, current and future turbine production flow, etc.). Necessary computer programs for reading data from Skagerak Energi's computer system into MATLAB should be developed, together with programs for writing data back from MATLAB to Skagerak Energi's computer system.
3. A dynamic model of the relevant water levels at Lake Toke should be developed. The model should be validated against experimental/historic data, and an assessment of the accuracy of the model should be given. Methods for on-line updating of the model based on measurements should be developed (parameter estimation, state estimation).
4. Managing floods should be posed as an MPC problem, and an MPC solution should be developed and tested based on proposed deterministic inflow from a hydrological model and production flow through the turbine.
5. A MATLAB program should be developed for testing out MPC for flood management.
6. The work that has been carried out should be documented in a master thesis.

Task background:
Five hydro power stations in the Kragerø Waterways, starting at the Dalsfoss hydro power station, receive their water from Lake Toke in Telemark. The catchment of Lake Toke covers ca. 1156 km2; the surface of the lake itself covers 32 km2. The lake holds some 150 million m3 of water, and the annual average flow out of the lake is ca. 24 m3/s; the residence time of the lake is thus ca. 72 days, which is relatively little in a hydro power context. The Dalsfoss hydro power turbines can maximally utilize 36 m3/s; with a higher flow rate than this, the water must be allowed to bypass the turbine, which implies a lost opportunity from a hydro

Address: Kjølnes ring 56, NO-3918 Porsgrunn, Norway. **Phone:** 35 57 50 00. **Fax:** 35 55 75 47.

power point of view – it would be advantageous to use the buffer capacity of the lake to smooth out some variations in the flow. With a relatively heavy rainfall of 10 mm/h, this implies ca. 90 m3/s of water hitting the lake surface and ca. 3400 m3/s of water hitting the catchment. The Dalsfoss hydro power station will be built out to allow for a maximal capacity of 960 m3/s flood bypass. Normally, a flow of 300 m3/s is considered a dramatic flood. The main spring flood starts in April each year; floods are caused by snow melting and rain, and hydrological models are used to describe the complex flow through the catchment and into the lake.

The operation of the hydro power station at Dalsfoss is strictly constrained by maximally and minimally allowed levels – these constraints change during the year. The flow out of Lake Toke is also constrained. In addition, the operation is constrained by economic considerations.

At the moment, the floodgates are operated by a specialist operator who is approaching retirement. It is thus of interest to develop an automatic system for controlling the flood gates. In an initial attempt, it is reasonable to develop a dynamic model including a mass balance for Lake Toke in combination with a hydrological model. The hydrological model will describe the flow into Toke, and will be provided by Skagerak Energi. The model will be used in a Model based Predictive Control setting, and the initial goal is to compute a proposed flood gate opening off-line. This proposed flood gate opening will then be evaluated by a specialist operator; this is necessary in a trial period due to the consequences if anything goes wrong. This approach will lead to manual closed loop. In a later work, it is of interest to close the loop automatically, but this will not be done right away.

Thus, a solution should be based on receiving hydrological predictions from Skagerak Energi together with production plans. This should be used as input to a dynamic model of the lake level, and an MPC controller should be used for on-line computation of flood gate opening, which should then be passed back to Skagerak Energi. A short term plan is that this solution should be tested during the spring flood of 2014, following this MSc theses in the spring of 2014.

References:

Gøthesen, D.-K., Haile, H.K., Khare, B.B., Kuznetsov, A., Njoku, I.O., Rabchuk, K.V. (2013). "Flood Control using MPC of the Kragerø Waterways", MSc project, Telemark University College, Porsgrunn.

Jørgensen, J.B., Capolei, A., Völcker, C. (2013). «Introduction to Economic MPC». Presentation at Nordic Process Control Workshop, Oulu, Finland, August 22-23.

Thoresen, Hege Marie (2011). "Control and optimisation of "Kragerø-vassdraget"", MSc thesis, Telemark University College, Porsgrunn.

Student category:
Reserved for Robin Evensen

Appendix 2

Group Project 2013
SCE 4106 Predictive Control with Implementation

Bernt Lie
Telemark University College, Porsgrunn, Norway

October 2, 2013

Contents

1 Introduction

This group project is a mandatory project in course *SCE 4106 Predictive Control with Implementation*, and the grade counts 30% in the final grade of the course; the final test counts 70% in the final grade. You, the students of the course are required to set up your own groups of 3-4 students. You will have to hand in a group report and present your work as a group in class according to the course schedule published in Fronter. It is required to contribute in the group work in order to pass, and the grades may be adjusted individually according to the understanding you show both in the project report, and in the project presentation.

The group project is related to an on-going project between Skagerak Energi in Porsgrunn, and Telemark University College, and deals with flood management of the Kragerø waterways. A third semester project group also works with this topic; obviously, they will have an advantage in that they will also work with the project in their project group. But more will be expected of this group, so it should be relatively fair. Also, they don't know much more about the topics in this course project than the other students, so you start more or less on equal foot.

In the sequel, a simple model of the water accumulated in Lake Toke is given, and a simple performance index is given. The group task is to study the control of the waterways using MPC. The nonlinear model should be linearized, and a discrete time QP formulation of the MPC problem should be given. Next, a QP based MPC solution should be used to study the control performance of the Toke model. Important issues are centered around:

- developing a discrete time, linear model
- formulating the optimization problem as a quadratic (linear?) problem
- understanding the modification of the controller to go from optimal control to predictive control
- studying possible choices of horizon length in the controller, and weight "matrices" in the performance index
- studying the importance of hard vs. soft constraints

In an appendix, a simple overview of the flood management problem is given, with specific information about Lake Toke and the Kragerø waterways.

2 Model description

2.1 Model development

The model is based on simple mass balance, and assuming constant density. A general mass balance

$$\frac{dm}{dt} = \dot{m}_{\mathrm{i}} - \dot{m}_{\mathrm{o}}$$

with constant density, $m = \rho V$, $\dot{m} = \rho \dot{V}$, leads to

$$\frac{dV}{dt} = \dot{V}_{\mathrm{i}} - \dot{V}_{\mathrm{o}}.$$

The relationship between volume and level is

$$dV = A\left(h\right) dh.$$

Thus, we get

$$\frac{dh}{dt} = \frac{1}{A\left(h\right)} \left(\dot{V}_{\mathrm{i}} - \dot{V}_{\mathrm{o}} \right).$$

In hydro power systems, the mapping $h \rightarrow A$ is known as the *filling curve*. For Lake Toke at large, the filling curve is given as

$$A\left(h\right) = \max\left(28 \times 10^6 \cdot h^{1/10}, 10^3\right), \tag{1}$$

where h [m] is the water level above the datum line given by $x = x_{\mathrm{LRV}}^{\min}$, the lowest regulated value; $x_{\mathrm{LRV}}^{\min} = 55.75\,\mathrm{m}$ above sea level.

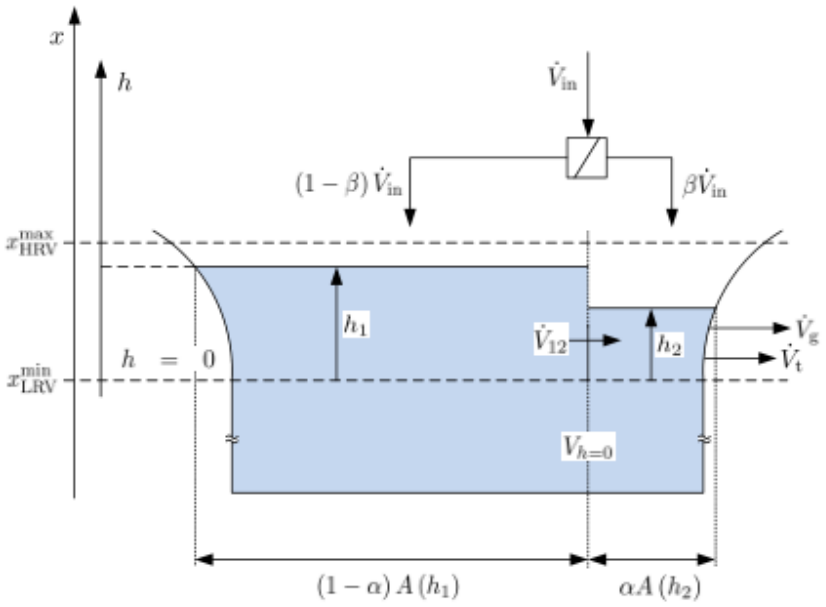


Figure 1: Possible layout of model for Lake Toke.

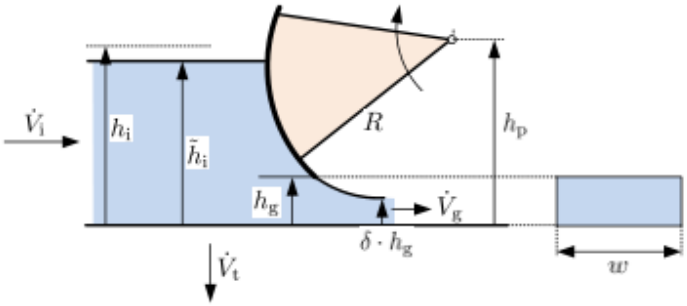


Figure 2: Sketch of free-flood gate. After [3].

The geometry of a possible model of Lake Toke is indicated in fig. 1. The lake is split into two compartments, where the main part of the inflow, $(1-\beta)\,\dot{V}_{\text{in}}$, flows into the upper compartment which has index 1, while the minor part of the inflow, $\beta\dot{V}_{\text{in}}$ flows into the lower compartment. The flow between the two compartments is given by $\dot{V}_{12}$:

$$\dot{V}_{12} = 800\,(h_1 - h_2)\,\sqrt{|h_1 - h_2|}, \tag{2}$$

an expression which is found by calibration to rather uncertain data. The outflow from the lower compartment goes partially through the hydro power turbine, $\dot{V}_{\text{t}}$, and when needed through a radial flood gate, $\dot{V}_{\text{g}}$.

A radial flood gate is sketched in fig. 2. Figure 2 indicates a free-flow gate, and a model

of such a gate is typically given by [3]

$$\dot{V} = \delta h_{\mathrm{g}} \cdot w \sqrt{\frac{2g\left(h_{\mathrm{i}} - \delta h_{\mathrm{g}}\right)}{1+\xi}} \tag{3}$$

where δ represents the contraction coefficient for the flow through the gate opening and ξ represents the energy loss on the upstream side of the gate as well as the effects of non-uniform velocity distribution in the system. h_{i} is the level upstream from boundary effects ($\tilde{h}_{\mathrm{i}}$) close to the gate; h_{p} is the pinion height. The model presumes that $h_{\mathrm{i}} - \delta h_{\mathrm{g}} \geq 0$.

A more classical model is that of [1]:

$$\dot{V} = C_{\mathrm{D}} h_{\mathrm{g}} w \sqrt{2g h_{\mathrm{i}}}, \tag{4}$$

where the discharge coefficient C_{D} is a function of the ratio $h_{\mathrm{i}}/h_{\mathrm{p}}$. A possible mapping from $\frac{h_{\mathrm{i}}}{h_{\mathrm{p}}} \rightarrow C_{\mathrm{D}}$ in a free-flow gate could be a second order polynomial approximately going through the points $(0.32, 0.55)$, $(0.82, 0.65)$, and $(1.52, 0.68)$, see fig. 2 in [1]. The model presumes that $h_{\mathrm{i}} \geq 0$.

A proposed gate model for the Dalsfos gate is

$$\dot{V}_{\mathrm{g}} = \begin{cases} h_{\mathrm{g}} w \sqrt{2g h_2}, & h_2 \geq h_{\mathrm{g}} \\ 0, & h_2 < 0 \end{cases} \tag{5}$$

where h_2 is the level in the lower compartment. Here, $h_{\mathrm{g}} \in \left[0, h_{\mathrm{g}}^{\max}\right]$ and it is presumed that $h_2 > h_{\mathrm{g}}$; alternatively we could write $h_{\mathrm{g}} = u_{\mathrm{g}} h_{\mathrm{g}}^{\max}$, where $u_{\mathrm{g}} \in [0, 1]$. The model for the Dalsfos gate can be questioned: it does not seem reasonable that there should be no losses in efficiency, etc.; see e.g. eq. 4 where $C_{\mathrm{D}} \in [0.55, 0.68]$. Furthermore, the model should also cover the possibility that $h_2 < 0$ and that $h_2 \leq h_{\mathrm{g}}$.

In [2], a free-flow weir is described as

$$\dot{V}_{\mathrm{g}} = C'_{\mathrm{D}} h_{\mathrm{i}} w \sqrt{2g h_{\mathrm{i}}}.$$

Note the change in the model as compared to eq. 5, where basically h_{g} has been replaced by h_{i} (i.e. h_2), and the weir does not have any control input. To regain control, it is necessary to reduce h_{g} such that $h_{\mathrm{g}} \leq h_{\mathrm{i}}$ (here: h_2).

2.2 Model summary

In summary, the following is a possible two-compartment model of Lake Toke:

$$\begin{aligned} \frac{dh_1}{dt} &= \frac{1}{(1-\alpha) A(h_1)} \left[(1-\beta) \dot{V}_{\mathrm{in}} - \dot{V}_{12} \right] \\ \frac{dh_2}{dt} &= \frac{1}{\alpha A(h_2)} \left[\dot{V}_{12} - \dot{V}_{\mathrm{t}} - \dot{V}_{\mathrm{g}} \right] \end{aligned}$$

where the filling curve $A(h)$ is given as

$$A(h) = \max\left(28 \times 10^6 \cdot h^{1/10}, 10^3\right),$$

Table 1: Parameters for the Lake Toke model.

Parameter	Value	Unit	Comment
α	0.05	–	Fraction of surface area in compartment 2
β	0.02	–	Fraction of inflow to compartment 2
w	11.2	m	Width of gate
$h_{\mathrm{g}}^{\max}$	5.6	m	Maximal opening height of gate
$x_{\mathrm{LRV}}^{\min}$	55.75	m	Minimal low regulated level value
$x_{\mathrm{HRV}}^{\max}$	60.35	m	Maximal high regulated level value

Table 2: Operating conditions for validating the Lake Toke model.

Quantity	Value	Unit	Comment
$h_1\,(t=0)$	2.5	m	Initial level, compartment 1
$h_1\,(t=0)$	2.5	m	Initial level, compartment 2
$\dot{V}_{\mathrm{in}}$	–	m^3/s	Inlet flow jumps from $24\,\mathrm{m}^3/\mathrm{s}$ to $200\,\mathrm{m}^3/\mathrm{s}$ after $25\times10^3\,\mathrm{s}$
h_{g}	–	m	Gate opening jumps from 0 m to $h_{\mathrm{g}}^{\max}$ after $50\times10^3\,\mathrm{s}$
$\dot{V}_{\mathrm{t}}$	24	m^3/s	Volumetric flow through turbines

the inter-compartment flow $\dot{V}_{12}$ is given as

$$\dot{V}_{12} = 800\,(h_1 - h_2)\,\sqrt{|h_1 - h_2|},$$

and the gate flow $\dot{V}_{\mathrm{g}}$ is given as

$$\dot{V}_{\mathrm{g}} = \begin{cases} h_{\mathrm{g}} w \sqrt{2g h_2}, & h_2 \geq h_{\mathrm{g}} \\ h_2 w \sqrt{2g h_2}, & h_{\mathrm{g}} \geq h_2 \geq 0 \\ 0, & h_2 < 0 \end{cases} .$$

The model is only valid for $(h_1, h_2) \geq (0, 0)$, where the levels refer to levels above the datum level given by $x_{\mathrm{LRV}}^{\min}$, the minimal low regulated level value. To find the true altitude above sea level of the water levels, these are thus $h_1 + x_{\mathrm{LRV}}^{\min}$ and $h_2 + x_{\mathrm{LRV}}^{\min}$, respectively. Parameters for the model are given in Table 1.

Operating conditions for validating the model are given in Table 2.

The dynamics of the system is rather slow. It is thus reasonable to use e.g. hours or days as time unit, with the necessary conversion factor in the model. Figure 3 indicates the level variations with parameters and operational conditions as specified above. Figure 4 indicates the step change in input/disturbance $\dot{V}_{\mathrm{in}}$ as well as the inter compartmental flow $\dot{V}_{12}$ and the gate flow $\dot{V}_{\mathrm{g}}$. Note that in steady state, $\dot{V}_{\mathrm{g}}$ is less than $\dot{V}_{\mathrm{in}}$ — the difference obviously is due to the constant turbine flow $\dot{V}_{\mathrm{t}}$. Figure 5 indicates the applied gate opening.

The output of the system, i.e. the quantity that is sought *controlled*, is h_1, the level in the main compartment — the level at Merkebekk.

The model given here is quite crude. It is possible that better model fit could be achieved by including a third compartment for upper Toke, with an inter compartment flow through Straume. This is not relevant for the current project in course Predictive Control with Implementation, though.

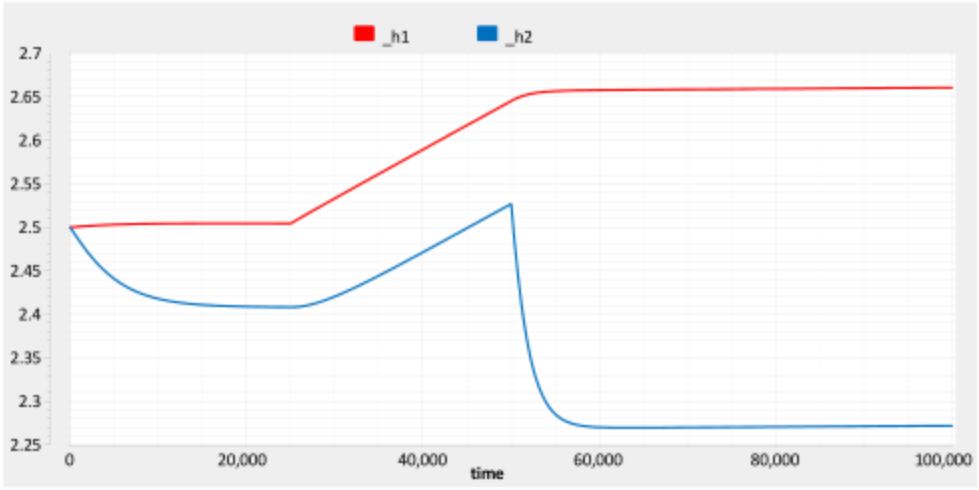


Figure 3: Levels h_1 and h_2 in the two compartments with the given parametes and operational conditions. Time in seconds.

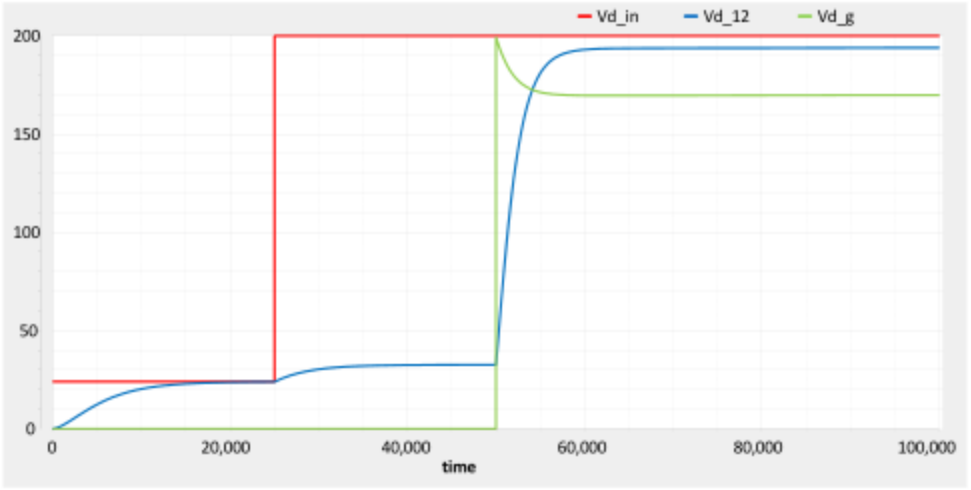


Figure 4: Disturbance flow $\dot{V}_{\text{in}}$, with resulting inter compartmental flow $\dot{V}_{12}$ and gate flow $\dot{V}_{\text{g}}$. Note that in steady state, $(1-\beta)\,\dot{V}_{\text{in}} = \dot{V}_{12}$, while $\dot{V}_{\text{in}} = \dot{V}_{\text{g}} + \dot{V}_{\text{t}}$. Time in seconds.

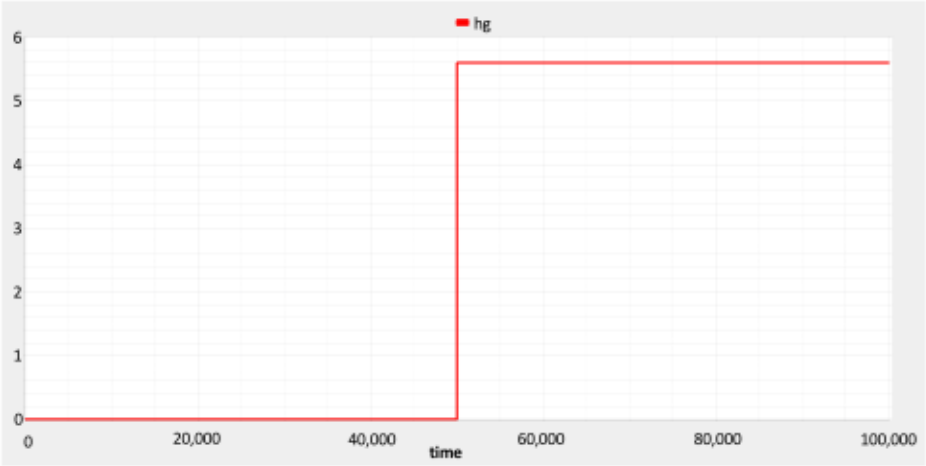


Figure 5: Gate opening h_g. Time in seconds.

3 Performance index

The concession conditions for operating the Kragerø waterways power plants includes the following:

1. The total water flow below Dalsfos power plant ($\dot{V}_t + \dot{V}_g$) must under no conditions be less that $4\,\mathrm{m}^3/\mathrm{s}$.
2. It is a requirement that the level of Lake Toke at Merkebekk (i.e. compartment 1) must lie in the interval $x \in [x_{\mathrm{LRV}}, x_{\mathrm{HRV}}]$, or converted to the h variable: $h_1 \in [0, x_{\mathrm{HRV}} - x_{\mathrm{LRV}}]$. However, the condition of $\dot{V}_t + \dot{V}_g \geq 4\,\mathrm{m}^3/\mathrm{s}$ supersedes this requirement.
3. In reality, the regulation values x_{LRV} and x_{HRV} vary during the year. It thus may make sense to use x_1 and x_2 as variables instead of h_1 and h_2, or better: use the lowest value of x_{LRV} over the year as the datum line.
4. There may be some constrains on $\dot{V}_t$ wrt. avoiding cavitation in the turbine. However, in this project, we will assume that $\dot{V}_t$ is a disturbance and that someone else (i.e. economists) has decided on the flow through the turbine.

A traditional performance index for the operation would be in the form:

$$I = \int_0^{t_h} \left[e^2(t) + p(t)\,\dot{u}^2(t) \right] dt$$

where t_h is the horizon, $e = r - h_1$ is the control deviation, and $p(t)$ is the weight on the control signal slope. It is convenient to instead use a discrete time index

$$I = \sum_{i=1}^{N} \left[e_i^2 + p_i (u_i - u_{i-1})^2 \right]$$

where $e_i = r_i - h_{1,i}$.

Here, the reference signal could e.g. lie in the midst of the regulation interval, e.g. $r_i = \frac{x_{\text{HRV}} - x_{\text{LRV}}}{2}$.

A more economically based performance index could be to avoid discharging water through the gate. Thus, the performance index should be:

$$I = \sum_{i=1}^{N} \dot{V}_{\text{g},i}$$

where it is given that $\dot{V}_{\text{g},i} \geq 0$. This performance index doesn't easily lend itself to being formulated as a QP problem, but instead indicates that one should use an LP formulation — where the linear equality constraints must come from a linearization of the model.

4 Task description

The following tasks should be considered.

1. Make sure you understand the model given. If you find it erroneous or lacking, you should correct it and/or expand it.

2. Implement the model in MATLAB, and validate the model/implementation through simulation, e.g. by comparing your MATLAB results to those in figs. 3–5. Also carry out step response tests on the model, i.e. inject steps in $\dot{V}_{\text{in}}$, h_{g} and $\dot{V}_{\text{t}}$ (other than those indicated above) and observe the step responses in h_1 and h_2. Preferably use "day" as time unit.

3. Find a linearized model around steady state[1]. Find a linearized discrete time model by (i) finding continuous time linear model $\frac{d}{dt}\delta x = A_{\text{c}}\delta x + B_{\text{c}}\delta u$, $\delta y = C\delta x$, and then (ii) use MATLAB function `c2d()` to get the discrete time model $\delta x_{k+1} = A\delta x_k + B\delta u_k$ and $\delta y_k = C\delta x_k$. You may e.g. use 1 h as discretization time, or 4 h, or something like this.

4. Choose as reference value
$$r_k = \frac{x_{\text{HRV}}^{\max} - x_{\text{LRV}}^{\min}}{2},$$
and form the control deviation
$$e_k = r_k - y_k$$
where $y_k = \delta h_1$. Use the following quadratic performance index:
$$I = \sum_{i=1}^{N} \left[e_i^2 + p_i \left(u_i - u_{i-1} \right)^2 \right].$$
Choose N such that it is equivalent to, say, 10 d.

[1] Since the model is relatively simple, it is relatively simple to find an analytic linearized continuous time model.

Rephrase the above performance index and linear discrete time model as a Quadratic Program which fits into MATLAB's specification of a QP.

5. Study the optimal control of the model with the given performance index. Experiment with changes in the inflow, e.g. $\dot{V}_{\text{in}}$ increasing to $200\,\text{m}^3/\text{s}$, and dropping to $0\,\text{m}^3/\text{s}$. How does the optimal controller handle the operation of the system? Inject the control input into both the linearized model, and the full nonlinear model. Discuss differences in behavior.

   (**Hint**: Assume that you know perfectly the real states of the system; in this project, including a state estimator is not important.)

6. Introduce MPC, and study a few scenarios of how the MPC handles the problem. Apply the MPC to both the linearized model and to the full nonlinear model. Discuss differences in behavior.

   (**Hint**: Assume that you perfectly know the real states of the system.)

7. Discuss, without doing it, how you could handle the full case where you include constraints on the control input (either h_{g} or u_{g}). Furthermore, discuss how you would handle the case of constraints in h_1 and h_2, e.g. by given time varying values for $x_{\text{LRV}} \geq x_{\text{LRV}}^{\min}$ and $x_{\text{HRV}} \leq x_{\text{HRV}}^{\max}$. Finally, discuss how you would handle the case of a modified performance index based on minimizing the loss/bypass of water through the gate.

A System Description

Five hydro power stations in the Kragerø Waterways, starting at the Dalsfos hydro power station, receive their water from Lake Toke in Telemark. Figure 6 shows the location of Lake Toke in Telemark County, Norway. The catchment of Lake Toke covers ca. $1156\,\text{km}^2$; the surface of the lake itself covers $32\,\text{km}^2$. Figure **??** shows Lake Toke with Upper Toke and Lower Toke with the Rørholt Fjord.[2] The lake holds some $150 \times 10^6\,\text{m}^3$ of water, and the annual average flow out of the lake is ca. $24\,\text{m}^3/\text{s}$; the residence time is relatively little in a hydro power context. The Dalsfos hydro power turbines can maximally utilize $36\,\text{m}^3/\text{s}$; with a higher flow rate than this, the water must be allowed to bypass the turbine, which implies a lost opportunity from a hydro power point of view – it would be advantageous to use the buffer capacity of the lake to smooth out some variations in the flow. With a relatively heavy rainfall of $10\,\text{mm/h}$, this implies ca. $90\,\text{m}^3/\text{s}$ of water hitting the lake surface and ca. $3210\,\text{m}^3/\text{s}$ of water hitting the catchment. The Dalsfos hydro power station will be built out to allow for a maximal capacity of $960\,\text{m}^3/\text{s}$ flood bypass. Normally, a flow of $300\,\text{m}^3/\text{s}$ is considered a dramatic flood. The main spring flood starts in April each year; floods are caused by snow melting and rain, and hydrological models are used to describe the complex flow through the catchment and into the lake.

[2] There is another, considerably larger Lake Tokke in Upper Telemark.

Figure 6: Location of Lake Toke in Telemark County.

Figure 7: Map displaying the shape of Lake Toke, Lower Telemark.

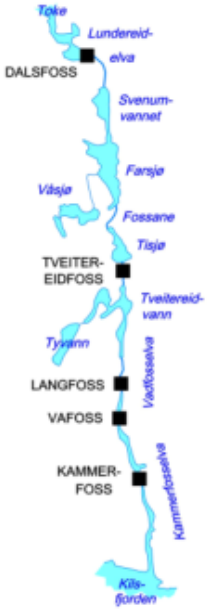


Figure 8: Kragerø Waterways from Lake Toke to the Kils Fjord.

The operation of the hydro power station at Dalsfos is strictly constrained by maximally and minimally allowed levels – these constraints change during the year. The level requirements are for site Merkebekk, before the final ca. 5 km river from the main part of Lake Toke down to the pondage of the Dalsfos hydro power station. The flow out of Lake Toke is also constrained to a minimum of $4\,\mathrm{m}^3/\mathrm{s}$; this requirement has higher ranking than the required levels in Lake Toke. Figure 8 shows the Kragerø Waterways from Lake Toke to the Kils Fjord east of Kragerø. In addition, the operation is constrained by economic considerations.

At the moment, the floodgates are operated by a specialist operator who is approaching retirement. It is thus of interest to develop an automatic system for controlling the flood gates. In an initial attempt, it is reasonable to develop a dynamic model including a mass balance for Lake Toke in combination with a hydrological model. The hydrological model will describe the flow into Toke, and will be provided by Skagerak Energi. The model will be used in a Model based Predictive Control setting, and the initial goal is to compute a proposed flood gate opening off-line. This proposed flood gate opening will then be evaluated by a specialist operator; this is necessary in a trial period due to the consequences if anything goes wrong. This approach will lead to manual closed loop. In a later work, it is of interest to close the loop automatically, butt his will not be done right away.

Thus, a solution should be based on receiving hydrological predictions from Skagerak Energi together with production plans. This should be used as input to a dynamic model of the lake level, and an MPC controller should be used for on-line computation of flood gate opening, which should then be passed back to Skagerak Energi. A short term plan is that

this solution should be tested during the spring flood of 2014, following this project work as well as 2 MSc theses in the spring of 2014.

References

[1] Buyalski, C.T. (1983). "Canal Radial Gate Discharge Algorithms and their Use". *Proceedings, ASCE 1983 Speciality Conference, Irrigation and Drainage Division: "Advances in Irrigation and Drainage: Surviving External Pressures"*. Jackson, Wyoming, July 20–22, 1983.

[2] Malaterre, P.-O., and Baume, J.-P. (1998). "Modeling and regulation of irrigation canals: existing applications and ongoing researches". Uknown publication, but attributed to (ISBN?) 0-7803-4778-1/98, IEEE.

[3] Wahl, T.L. (2004). "Issues and Problems with Calibration of Canal Gates". *Proceedings, World Water & Environmental Resources Congress*, Salt Lake City, UT, June 27 -July 1, 2004.

Appendix 3

This program which runs the MPC on model of Lake Toke consists of 8 scripts:

Script 1: *runMPC.m*
Script 2: *build_matrices.m*
Script 3: *Vinsimulator.m*
Script 4: *Wesimulator.m*
Script 5: *Vtnsimulator.m*
Script 6: *linearize_model_imag.m*
Script 7: *MPCmodel.m*
Script 8: *nonlinmodel.m*
Script 9: *plotwithh2inhgplot.m*
Script10: *plotwref0.m*

Script 1(*runMPC.m)*:

```
clc

% Initial states
x0 = [2.5,2.41]';

%maximum opening of the gate
hgmax=5.6;

% Horizon
N=24*10;

%Lowest and highest regulated level
lrv=55.75;
hrv=59;

% Number of sample points, timestep is set to 1 hour
n = 24*60;

%weight of variable e
Q=1;

%Weight of variable u
P=1;

%Weight on Smin
Qmin = eps;
%Weight on Smax
Qmax = 10;

%Highest allowed change in flow
Vmax=25;

%reference point for the height i h1
r0 = 2.3;

%reference for upper and lower limit in h1
Ymin=0;
Ymax=3;

%configurating options for quadprog
```

```
opts = optimset('Algorithm','interior-point-convex', 'Display', 'off');

%initializing matrices for plots
yplot=zeros(1,n);
uplot=zeros(1,n);
x2plot=zeros(1,n);
Vinplot=zeros(1,n);
eplot=zeros(1,n);

%initialize time step for non linear model update
tspan=[1 60*60];

% Initial u and Vt
u0 = 0;
Vt0 = 5;

%Prepare disturbance matrices
Vin=Vinsimulator();

% Vt=ones(1,length(Vin))*24;
We=Wesimulator();

%Defining sizes of system matrices
nx=2;
nu=1;
ny=1;

for i = 1:n
    i

    if(i==10)
       i;
    end
    %Steady states
    hss=[x0(1); x0(2)];
    hgss=0.05*5.6;
    Vinss=40;
    Vtss=10;

    %System matrices used by the controller
    [A,B,C,D,M]=linearize_model_imag(60*60, [hss(1)
hss(2)]',hgss,Vinss,Vtss);
  k=1;
  %Creating matrice with Vin and Vt
  for(j=i:i+N-1)
      dmat(k:k+1)=M*[Vin(j);0];
      k=k+2;
  end

  %The Vin at time t
  Vint=Vin[5];

  %Finding the parameters to calculate the flow Vt
  Vtpara=linearize_Vt_imag(x0(2)+lrv,We(i+1:i+N-1));

    %calculate area
    Ah1=max(28*1000000*x0(1)^(1/10),1000);
    Ah2=max(28*1000000*x0(2)^(1/10),1000);

    % Present state and measurements
```

```
  x = x0;
  y = C*x0;

    [H, f, Ae, Aineq] = build_matrices(A,B,C,M,P,Q,N,nx,ny,nu,Qmin,Qmax,
Vtpara,x0(2));


  % Calculate present u
  rN = r0*ones(N,1);

  %Building the b on the equality matrix
  b1=[A*x0+dmat(1:2)';zeros(nx*(N-1),1)+dmat(3:N*2)'];
  b2=[zeros(ny*N,1)];
  b3=[rN];
  b4=[[Vtmodel(x0(2)+lrv,We[5])],[-Vtpara.*x0(2)+Vtmodel(x0(2)+lrv,
We(i+1:i+N-1))]]';

  be=[b1;b2;b3;b4];
  bineq = [x0(2),zeros(1,N*2*nu-1), ones(1,N)*-Ymin, ones(1,N)*Ymax, (-
4)*ones(1,N), [Vmax+Vt0+u0*(11.2*sqrt(2*9.81*x0(2))), Vmax*ones(1,N-1)]];
  U0 = quadprog(H,f,Aineq,bineq,Ae,be,[],[],[],opts);
  u0 = U0(1:nu);
  u = u0;

  %Saving Vt0 for next iteration
  Vt0=Vtmodel(x0(2)+lrv,We[5]);

  %update the non linear model to simulate a real step
  mymodel = @(t,x) MPCmodel(t,x,Vtmodel(x0(2)+lrv,We[5]), Vint,u);
  [time,xlist]=ode45(mymodel,tspan,x);

x0=[xlist(length(xlist(:,1)),1),xlist(length(xlist(:,1)),2)]';
%filling plot
% Vinplot(1,i)=Vin;
x2plot(1,i)=x0(2);
x1plot(1,i)=x0(1);
hgplot(1,i)=u;
eplot(1,i)=U0(N*3+1);

end


ref=r0*ones(1,n);
t=[0:1:n-1];
figure(1)
subplot(2,1,1)
plot(t,x1plot,t,x2plot,t,ref);
subplot(2,1,2)
plot(t,hgplot);
```

Script 2(*build_matrices.m)*:

```
function [H,f, Ae, Aineq] = build_matrices(A,B,C,M,P,Q,N, nx, ny, nu,
Qmin,Qmax, Vtpara,h2)

%number of vt variables and high and low soft limit
nvt=1;
```

```
ns=2;

%Quadratic cost function
H1 = diag(P*ones(1,N));
H2 = zeros(N*nx,N*nx);
H3 = zeros(N,N);
% H3 = diag(Q*ones(1,N)); %This part is for the Q
H4 = zeros(N,N);
H5 = diag(Qmin*ones(1,N));
H6 = diag(Qmax*ones(1,N));
H7 = zeros(N*nvt,N*nvt);
H = blkdiag(H1,H2,H3,H4,H5,H6,H7);

f = zeros(1,(nu+nx+ny+ny+ns+nvt)*N);

%Equality constraints
Ae11 = -kron(eye(N,N),B);
Ae12 = kron(eye(N,N),eye(nx,nx)) - kron(diag(ones(N-1,1),-1),A);
Ae16 = kron(eye(N,N),-M(:,2));
Ae22 = -kron(eye(N,N),C);
Ae24 = eye(N*ny,N*ny);
Ae33 = eye(N*ny,N*ny);
Ae34 = eye(N*ny,N*ny);
Ae42 = kron(diag(Vtpara'.*ones(N-1,1),-1),[0 -1]);
Ae46 = eye(N*nvt,N*nvt);


Ae = [Ae11,Ae12,zeros(nx*N,ny*N),zeros(nx*N,ny*N), zeros(nx*N,2*N), Ae16
%x(k+1)=Ax[3]+Bu[3]+M[3]
    zeros(ny*N,nu*N),Ae22,zeros(ny*N,ny*N),Ae24, zeros(N,ns*N),
zeros(nvt*N,nvt*N)     %y[3]=Cx[3]
    zeros(ny*N,nu*N),zeros(ny*N,nx*N),Ae33,Ae34, zeros(N,ns*N),
zeros(nvt*N,nvt*N)     %e[3]=r[3]-y[3]
    zeros(ny*N,nu*N),Ae42, zeros(ny*N,ny*N), zeros(ny*N,ny*N),
zeros(N,ns*N), Ae46];   %Vt[3]=(h2(N)-h2(0))*Vtpara + Vtmodel


% Inequality constraints
Aineq11= eye(N,N);
Aineq12= -kron(diag(ones(N-1,1),-1),[0 1]);
Aineq21= -eye(N,N);
Aineq34= -eye(N,N);
Aineq35= -eye(N,N);
Aineq44= eye(N,N);
Aineq46= -eye(N,N);
Aineq51= -1*(11.2*sqrt(2*9.81*h2))*eye(N,N);
Aineq56= -eye(N,N);
Aineq61= 11.2*sqrt(2*9.81*h2)*eye(N,N) - 11.2*sqrt(2*9.81*h2)*diag(ones(N-
1,1),-1);
Aineq66= eye(N,N) - diag(ones(N-1,1),-1);



Aineq = [Aineq11, Aineq12, zeros(N,ny*N),zeros(N,ny*N), zeros(N,2*N),
zeros(nvt*N,nvt*N)             %hg-h2<=0
        Aineq21, zeros(N,nx*N),zeros(N,ny*N),zeros(N,ny*N), zeros(N,2*N),
zeros(nvt*N,nvt*N)         %-hg<=0
        zeros(N,N), zeros(N,nx*N), zeros(N,ny*N),Aineq34, Aineq35,
zeros(N,N), zeros(nvt*N,nvt*N)    %-y-Smin<=-ymin
        zeros(N,N), zeros(N,nx*N), zeros(N,ny*N),Aineq44, zeros(N,N),
Aineq46, zeros(nvt*N,nvt*N)    %y-Smax<=-Ymax
```

```
        Aineq51, zeros(N,nx*N), zeros(N,ny*N), zeros(N,ny*N), zeros(N,2*N),
Aineq56                  %-hg<=(-4+Vt)/(w*sqrt(2*g*h2))
        Aineq61, zeros(N,nx*N), zeros(N,ny*N), zeros(N,ny*N), zeros(N,2*N),
Aineq66];                %Vg(k+1)-Vg[3]+Vt(k+1)-Vt[3]<=Vomax

End
```

Script 3(*Vinsimulator.m)*:

```
function [ Vin ] = Vinsimulator()

y(1)=30;
y(2)=50;
y(3)=150;
y(4)=200;
y(5)=210;
y(6)=150;
y(7)=80;
y(8)=80;

x=1:240:60*24+10*24+1;
Y=y;
xx=1:1:60*24+10*24+1;

yy = spline(x,Y,xx);

Vin=yy;
end
```

Script 4(*Wesimulator.m)*:

```
function [ Vt ] = Wesimulator()

XQpara(1)=0.0211;
XQpara(2)=37.1891;

Vtpara(1)=132.0238;
Vtpara(2)=2.8241;



y(1)=3.5;
y(2)=3.5;
y(3)=4.5;
y(4)=5.5;
y(5)=3.3;
y(6)=1;
y(7)=1;
y(8)=0;

x=1:240:60*24+10*24+1;
Y=y;
xx=1:1:60*24+10*24+1;

yy = spline(x,Y,xx);
```

```
Vt=yy;

end
```

Script 5(*Vtnsimulator.m)*:

```
function [ yVt ] = Vtmodel(XD, We )

XQpara(1)=0.0211;
XQpara(2)=37.1891;

Vtpara(1)=132.0238;
Vtpara(2)=2.8241;

yVt = Vtpara(1).*(-(XQpara(2)-XD)-sqrt((XQpara(2)-XD).^2-
4*XQpara(1).*We))./(2.*XQpara(1)) + Vtpara(2);


end
```

Script 6(*linearize_model_imag.m)*:

```
function [A,B,C,D,M] = linearize_model_imag(Ts, xss, hgss, Vinss, Vtss)

nx = length(xss);

Ix = eye(nx,nx);

h = sqrt(eps);

A = zeros(nx,nx);
i = sqrt(-1);

for k = 1:nx
A(:,k) = (1/h)* imag(nonlinmodel(xss+h*i*Ix(:,k),hgss,Vinss, Vtss));
end

Btemp = zeros(nx,1);
Btemp(:,1) = (1/h)*imag(nonlinmodel(xss,hgss+h*i,Vinss, Vtss));
Btemp(:,2) = (1/h)*imag(nonlinmodel(xss,hgss,Vinss+i*h, Vtss));
Btemp(:,3) = (1/h)*imag(nonlinmodel(xss,hgss,Vinss, Vtss+i*h));

% M = zeros(nx,1);
% M(:,1) = (1/h)*imag(nonlinmodel(xss,hgss,Vinss+i*h, Vtss));
% M(:,2) = (1/h)*imag(nonlinmodel(xss,hgss,Vinss, Vtss+i*h));

C = [1 0];

D = 0;

sys=ss(A,Btemp,C,D);
disc_sys=c2d(sys,Ts);

A=disc_sys.a;
```

```
Btemp=disc_sys.b;
C=disc_sys.c;
D=disc_sys.d;

%Separating know disturubance from input
B=Btemp(:,1);
M=[Btemp(:,2) Btemp(:,3)];

End
```

Script 7(*MPCmodel.m)*:

```
function [dx_dt]= MPCmodel(t,x,Vt, Vin, hg)
dx_dt=zeros(length(x),1);

a=0.05;
B=0.02;
w=11.2;
g=9.81;
A1=max(28*1000000*x(1)^(1/10),1000);
A2=max(28*1000000*x(2)^(1/10),1000);
V12=800*(x(1)-x(2))*sqrt(abs(x(1)-x(2)));

if(x(2)>=hg)
    Vg=hg*w*sqrt(2*g*x(2));
elseif(hg>=x(2) && x(2)>=0)
    Vg=x(2)*w*sqrt(2*g*x(2));
elseif(x(2)<0)
    Vg=0;
end


dx_dt(1) = 1/((1-a)*A1)*((1-B)*Vin-V12);
dx_dt(2) = 1/[3]*(V12-Vt-Vg);

%transpose dx_dt so it is a column vector
% dx_dt = dx_dt';
return
```

Script 8(*nonlinmodel.m)*:

```
function [dx_dt]= nonlinmodel(x,hg, Vin,Vt)

dx_dt=zeros(length(x),1);

omega=11.2;
a=0.05;
B=0.02;
w=11.2;
g=9.81;
Cd=1;
A1=max(28*1000000*x(1)^(1/10),1000);
A2=max(28*1000000*x(2)^(1/10),1000);

V12=800*(x(1)-x(2))*sqrt(abs(x(1)-x(2)));

%The flow Vg is replaced with
Vg=Cd*hg*omega*sqrt(2*g*x(2));
```

```
% if(x(2)>=hg)
%     Vg=hg*w*sqrt(2*g*x(2));
% elseif(hg>=x(2) && x(2)>=0)
%     Vg=x(2)*w*sqrt(2*g*x(2));
% elseif(x(2)<0)
%     Vg=0;
% end

dx_dt(1) = 1/((1-a)*A1)*((1-B)*Vin-V12);
dx_dt(2) = 1/[3]*(V12-Vt-Vg+B*Vin);

end
```

Script 9(*plotwithh2inhgplot.m)*:

```
function plotwithh2inhgplot(X1, YMatrix1, YMatrix2)
%CREATEFIGURE(X1, YMATRIX1, YMATRIX2)
%  X1:  vector of x data
%  YMATRIX1:  matrix of y data
%  YMATRIX2:  matrix of y data

%  Auto-generated by MATLAB on 28-May-2014 22:35:33

% Create figure
figure1 = figure('PaperType','<custom>','PaperSize',[29.7 15],...
    'InvertHardcopy','off',...
    'Color',[1 1 1]);

% Create subplot
subplot1 =
subplot(2,1,1,'Parent',figure1,'FontWeight','light','YGrid','on','XGrid','o
n',...
    'FontSize',13);
ylim(subplot1,[0 5]);
box(subplot1,'on');
hold(subplot1,'all');

% Create multiple lines using matrix input to plot
plot1 = plot(X1,YMatrix1,'Parent',subplot1,'LineWidth',2);
set(plot1(1),'LineStyle','--','DisplayName','h_1');
set(plot1(2),'LineStyle',':','DisplayName','h_2','Color',[0 0 0]);
set(plot1(3),'DisplayName','hrv');
set(plot1(4),'Color',[1 0 0],'DisplayName','lrv');

% Create xlabel
xlabel('Time [h]','FontWeight','light','FontSize',13);

% Create ylabel
ylabel('Water level [m]','FontWeight','light','FontSize',13);

% Create subplot
subplot2 =
subplot(2,1,2,'Parent',figure1,'FontWeight','light','YGrid','on','XGrid','o
n',...
    'FontSize',13);
ylim(subplot2,[0 5]);
box(subplot2,'on');
hold(subplot2,'all');
```

```
% Create multiple lines using matrix input to plot
plot2 = plot(X1,YMatrix2,'Parent',subplot2,'LineWidth',2);
set(plot2(1),'DisplayName','h_g');
set(plot2(2),'LineStyle',':','DisplayName','h_2','Color',[0 0 0]);

% Create xlabel
xlabel('Time [h]','FontWeight','light','FontSize',13);

% Create ylabel
ylabel('Flood gate [m]','FontWeight','light','FontSize',13);

% Create legend
legend1 = legend(subplot1,'show');
set(legend1,...
    'Position',[0.147783473615092 0.757862341626222 0.06484375
0.123188405797101]);

% Create legend
legend2 = legend(subplot2,'show');
set(legend2,...
    'Position',[0.818229166666667 0.351449275362319 0.05859375
0.0717391304347826]);
```

Script 10(*plotwref0.m)*:

```
function plotwref0(X1, YMatrix1, Y1)
%CREATEFIGURE(X1, YMATRIX1, Y1)
%  X1:  vector of x data
%  YMATRIX1:  matrix of y data
%  Y1:  vector of y data

%  Auto-generated by MATLAB on 28-May-2014 22:14:39

% Create figure
figure1 = figure('PaperType','<custom>','PaperSize',[29.7 15],...
    'InvertHardcopy','off',...
    'Color',[1 1 1]);

% Create subplot
subplot1 = subplot(2,1,1,'Parent',figure1,'FontWeight','light',...
    'FontSize',13);
%% Uncomment the following line to preserve the X-limits of the axes
% xlim(subplot1,[0 1500]);
% Uncomment the following line to preserve the Y-limits of the axes
ylim(subplot1,[0 5]);
%% Uncomment the following line to preserve the Z-limits of the axes
% zlim(subplot1,[-1 1]);
box(subplot1,'on');
hold(subplot1,'all');

% Create multiple lines using matrix input to plot
plot1 = plot(X1,YMatrix1,'Parent',subplot1,'LineWidth',2);
set(plot1(1),'LineStyle','--','DisplayName','h_1');
set(plot1(2),'LineStyle',':','DisplayName','h_2','Color',[0 0 0]);
set(plot1(3),'DisplayName','hrv');
set(plot1(4),'Color',[1 0 0],'DisplayName','lrv');

% Create xlabel
```

```
xlabel('Time [h]','FontWeight','light','FontSize',13);

% Create ylabel
ylabel('Water level [m]','FontWeight','light','FontSize',13);

% Create subplot
subplot2 = subplot(2,1,2,'Parent',figure1,'FontWeight','light',...
    'FontSize',13);
%% Uncomment the following line to preserve the X-limits of the axes
% xlim(subplot2,[0 1500]);
% Uncomment the following line to preserve the Y-limits of the axes
ylim(subplot2,[0 5]);
%% Uncomment the following line to preserve the Z-limits of the axes
% zlim(subplot2,[-1 1]);
box(subplot2,'on');
hold(subplot2,'all');

% Create plot
plot(X1,Y1,'Parent',subplot2,'LineWidth',2,'DisplayName','h_g');

% Create xlabel
xlabel('Time [h]','FontWeight','light','FontSize',13);

% Create ylabel
ylabel('Flood gate [m]','FontWeight','light','FontSize',13);

% Create legend
legend1 = legend(subplot2,'show');
set(legend1,...
    'Position',[0.160416666666667 0.383852691218131 0.05859375
0.0500472143531634]);

% Create legend
legend2 = legend(subplot1,'show');
set(legend2,...
    'Position',[0.159244791666668 0.767705382436265 0.06484375
0.16052880075543]);
```

Appendix 4

This appendix consists of the MATLAB code which run the test of the turbine flow model, the code consists of 4 scripts.

Script 1: *TestVtlin.m*
Script 2: *linearize_Vt_imag.m*
Script 3: *Vtmodel.m*
Script 4: *simuplot.m*

Script 1(*TestVtlin.m)*:

```
lrv=55.75;

x2=1;
XD=x2+lrv;
n=20;
We=linspace(0,5.6,10);

%Since the level is not expected to be less then the lrv, the model is
%linearized with a x2 from 0 to 2 if the level is lower then 1


Vtpara=linearize_Vt_imag(XD,We);

Vtlin = @(XDdt, Wedt, XD, We)  Vtpara*XDdt + Vtmodel(XD,We);
% Vtlin = @(XDdt, Wedt, XD, We)  Vtmodel(XD,We);

y1=Vtmodel(XD+2,We)';
y2=Vtlin(2,0,XD, We)';
y3=Vtmodel(XD,We)';
x=0:1:length(y1)-1;

% plot(We,y1,We,y2,We,y3)

simuplot(We, [y1 y2 y3])

%# centimeters units
X = 29.7;                   %# A4 paper size
Y = 10;                   %# A4 paper size
xMargin = 1;                %# left/right margins from page borders
yMargin = 1;                %# bottom/top margins from page borders
xSize = X - 2*xMargin;      %# figure size on paper (widht & hieght)
ySize = Y - 2*yMargin;      %# figure size on paper (widht & hieght)

%# figure size on screen (50% scaled, but same aspect ratio)
set(gcf, 'Units','centimeters', 'Position',[5 5 xSize ySize])

%# figure size printed on paper
set(gcf, 'PaperUnits','centimeters')
set(gcf, 'PaperSize',[X Y])
set(gcf, 'PaperPosition',[xMargin yMargin xSize ySize])
```

```
set(gcf, 'PaperOrientation','portrait')
iptsetpref('ImshowBorder','tight');
set(gca,'LooseInset',get(gca,'TightInset'))
```

Script 2(*linearize_Vt_imag.m)*:

```
function [Vtpara] = linearize_Vt_imag(XD,We)

h = sqrt(eps);

i = sqrt(-1);

Vtpara = (1/h)* imag(Vtmodel(XD+h*i,We));


end
```

Script 3(*Vtmodel.m)*:

```
function [ yVt ] = Vtmodel(XD, We )

XQpara(1)=0.0211;
XQpara(2)=37.1891;

Vtpara(1)=132.0238;
Vtpara(2)=2.8241;

yVt = Vtpara(1).*(-(XQpara(2)-XD)-sqrt((XQpara(2)-XD).^2-
4*XQpara(1).*We))./(2.*XQpara(1)) + Vtpara(2);


end
```

Script 4(*simuplot.m)*:

```
function simuplot(X1, YMatrix1)
%CREATEFIGURE(X1, YMATRIX1)
%  X1:  vector of x data
%  YMATRIX1:  matrix of y data

%  Auto-generated by MATLAB on 27-May-2014 11:03:46

% Create figure
figure1 = figure('PaperType','<custom>','PaperSize',[29.7 10],...
    'InvertHardcopy','off',...
    'Color',[1 1 1]);

% Create axes
axes1 = axes('Parent',figure1,'YGrid','on','XGrid','on',...
    'FontWeight','light',...
    'FontSize',13);
%% Uncomment the following line to preserve the X-limits of the axes
% xlim(axes1,[0 9]);
%% Uncomment the following line to preserve the Y-limits of the axes
```

```
% ylim(axes1,[0 45]);
%% Uncomment the following line to preserve the Z-limits of the axes
% zlim(axes1,[-1 1]);
box(axes1,'on');
hold(axes1,'all');

% Create multiple lines using matrix input to plot
plot1 = plot(X1,YMatrix1,'Parent',axes1,'LineWidth',2);
set(plot1(1),'DisplayName','Nonlinear model at h_2 = 3m');
set(plot1(2),'DisplayName','Linearized model at h_2 = 3 m');
set(plot1(3),'DisplayName','Nonlinear model at h_2 = 1 m','LineWidth',0.5);

% Create xlabel
xlabel('Generated elecetricity [kWh]','FontWeight','light','FontSize',13);

% Create ylabel
ylabel('Turbine flow dV/dt [m^3/s]','FontWeight','light','FontSize',13);

% Create legend
legend(axes1,'show');
```

Appendix 5

This MATLAB code creates the turbine flow model, the code consists of 5 scripts which. Script 1 needs to be executed before script 2.

Script 1: *LSM1.m*
Script 2: *LSM2.m*
Script 3: *plotcode.m*
Script 4: *remove_flow_to_XQ.m*
Script 5: *removeNaN.m*
Script 6: *split_vt_vg.m*
Script 7: *VovsXQ.m*
Script 8: *plotVovsXQ.m*

Script 1(*LSM1.m)*:

```
clear all
close all
clc

load Vo08
load Vo09
load We08
load We09
load XQ08
load XQ09
load XD08
load XD09

removeNaN;

Outlier=6870;
Vo08(Outlier) = [];
XD08(Outlier) = [];
We08(Outlier) = [];
XQ08(Outlier) = [];

Outlier=6492;
Vo09(Outlier) = [];
XD09(Outlier) = [];
We09(Outlier) = [];
XQ09(Outlier) = [];

split_vt_vg;
remove_flow_to_XQ;



%==========================================================================
%Using XQ = Vt + K
Als = [Votemp ones(1,length(Votemp))'];
yls = XQtemp';

xls=inv(Als'*Als)*Als'*yls';
XQpara=xls;
yXQ = @(Vo) xls(1).*Vo + xls(2);
```

```
figure
plot(Vt09,yXQ(Vt09),'x',Votemp,XQtemp,'x')
xlim([0 40])
ylim([37 38])

Vttest=(-(XQpara(2)-XD09)-sqrt((XQpara(2)-XD09).^2-
4*XQpara(1).*We09))/(2*XQpara(1));
figure
plot(We09,Vo09,'x',We09,Vttest,'x')
ylim([0.017 0.26])
```

Script 2(*LSM2.m)*:

```
close all

%
%========================================================================
% %Using Vt = We + K
Als = [(-(XQpara(2)-XD08)-sqrt((XQpara(2)-XD08).^2-
4*XQpara(1).*We08))/(2*XQpara(1)) ones(1,length(We08))'];
yls = Vt08';

xls=inv(Als'*Als)*Als'*yls';
yVt = @(XD, We, Vg)  xls(1)*(-(XQpara(2)-XD)-sqrt((XQpara(2)-XD).^2-
4*XQpara(1).*We))/(2*XQpara(1)) + xls(2);

% plot(Vo08,XQ08,'x');
figure
plot(We09,yVt(XD09,We09, Vg09),'x',We09,Vt09,'x')
% xlim([])
% ylim([0 300])
x=0:1:length(We09)-1;

plotcode(x,[yVt(XD09,We09,Vg09) Vt09],We09)

%# centimeters units
X = 29.7;                   %# A4 paper size
Y = 15;                   %# A4 paper size
xMargin = 1;                %# left/right margins from page borders
yMargin = 1;                %# bottom/top margins from page borders
xSize = X - 2*xMargin;      %# figure size on paper (widht & hieght)
ySize = Y - 2*yMargin;      %# figure size on paper (widht & hieght)

%# figure size on screen (50% scaled, but same aspect ratio)
set(gcf, 'Units','centimeters', 'Position',[5 5 xSize ySize])

%# figure size printed on paper
set(gcf, 'PaperUnits','centimeters')
set(gcf, 'PaperSize',[X Y])
set(gcf, 'PaperPosition',[xMargin yMargin xSize ySize])
set(gcf, 'PaperOrientation','portrait')
iptsetpref('ImshowBorder','tight');
set(gca,'LooseInset',get(gca,'TightInset'))
% set(gca,'position',[0 0 1 1],'units','normalized')

% figure
```

```
% subplot(2,1,1)
% plot(x,yVt(XD09,We09,Vg09),x,Vt09)
% subplot(2,1,2)
% plot(x,We09)
```

Script 3(*plotcode.m)*:

```
function plotcode(X1, YMatrix1, Y1)
%CREATEFIGURE(X1, YMATRIX1, Y1)
%  X1:  vector of x data
%  YMATRIX1:  matrix of y data
%  Y1:  vector of y data

%  Auto-generated by MATLAB on 03-Mar-2014 14:13:57

% Create figure
figure1 = figure;

% Create subplot
subplot1 = subplot(2,1,1,'Parent',figure1);
box(subplot1,'on');
hold(subplot1,'all');

% Create multiple lines using matrix input to plot
plot1 = plot(X1,YMatrix1,'Parent',subplot1,'LineWidth',2);
set(plot1(1),'DisplayName','Estimated dV_t/dt');
set(plot1(2),'DisplayName','dV_t/dt 2009');

% Create xlabel
xlabel('Sample');

% Create ylabel
ylabel('dV_t/dt [m^3/s]');

% Create subplot
subplot2 = subplot(2,1,2,'Parent',figure1);
box(subplot2,'on');
hold(subplot2,'all');

% Create plot
plot(X1,Y1,'Parent',subplot2,'LineWidth',2);

% Create xlabel
xlabel('Sample');

% Create ylabel
ylabel('Power production [KWh]');

% % Create legend
% legend1 = legend(subplot1,'show');
% set(legend1,...
%     'Position',[0.289299536373768 0.657752401779758 0.127029608404967
0.178807947019868]);
```

Script 4(*remove_flow_to_XQ.m)*:

```
% rows_to_remove09 = any(Vo09>=36, 2);
% XQ09(rows_to_remove09,:) = [];
% XD09(rows_to_remove09,:) = [];
% Vo09(rows_to_remove09,:) = [];
% We09(rows_to_remove09,:) = [];


rows_to_remove08 = any(Vo08>=36.5, 2);
XQtemp=XQ08;
XQtemp(rows_to_remove08,:) = [];
Votemp=Vo08;
Votemp(rows_to_remove08,:) = [];
```

Script 5(*removeNaN.m)*:

```
rows_to_remove = any(isnan(XD08), 2);
XD08(rows_to_remove,:) = [];
Vo08(rows_to_remove,:) = [];
We08(rows_to_remove,:) = [];
XQ08(rows_to_remove,:) = [];
```

Script 6(*split_vt_vg.m)*:

```
Vt08=Vo08;
Vt09=Vo09;

Vg08=zeros(length(Vo08),1);
Vg09=zeros(length(Vo09),1);

rows = any(Vo08>=36.5, 2);
Vt08(rows,:) = 36.5;

rows = any(Vo09>=36.5, 2);
Vt09(rows,:) = 36.5;

rows = any(Vo08>36.5, 2);
Vg08(rows,:) = Vo08(rows)-Vt08(rows);

rows = any(Vo09>36.5, 2);
Vg09(rows,:) = Vo09(rows)-Vt09(rows);
```

Script 7(*VovsXQ.m)*:

```
close all
%=========================================================================
%Using XQ = V0^3 + Vo.^2 + Vo + K
Als = [Vo08.^3 Vo08.^2 Vo08 ones(1,length(Vo08))'];
yls = XQ08';

xls=inv(Als'*Als)*Als'*yls';
XQpara=xls;
yXQ = @(Vo) xls(1).*Vo.^3 + xls(2).*Vo.^2 + xls(3)*Vo + xls(4);

% testplot(Vo09,[XQ09 yXQ(Vo09)])
plotVovsXQ(Vo09,[yXQ(Vo09) XQ09])
```

Script 8(*plotVovsXQ.m)*:

```
function plotVovsXQ(X1, YMatrix1)
%CREATEFIGURE(X1, YMATRIX1)
%  X1:  vector of x data
%  YMATRIX1:  matrix of y data

%  Auto-generated by MATLAB on 05-Mar-2014 19:21:50

% Create figure
figure1 = figure();

% Create axes
axes1 = axes('Parent',figure1,'FontSize',14);
box(axes1,'on');
hold(axes1,'all');

% Create multiple lines using matrix input to plot
plot1 =
plot(X1,YMatrix1,'Parent',axes1,'Marker','x','LineStyle','none','Linewidth'
,2);
set(plot1(1),'DisplayName','X_Q 2009');
set(plot1(2),'DisplayName','Cubic fit of dV_O/dt');

% Create xlabel
xlabel('dV_O/dt','FontSize',14);

% Create ylabel
ylabel('X_Q','FontSize',14);
legend(axes1,'show');

xlim([0 220]);
ylim([36 41])

%# centimeters units
X = 29.7;                   %# A4 paper size
Y = 10;                   %# A4 paper size
xMargin = 1;                %# left/right margins from page borders
yMargin = 1;                %# bottom/top margins from page borders
xSize = X - 2*xMargin;      %# figure size on paper (widht & hieght)
ySize = Y - 2*yMargin;      %# figure size on paper (widht & hieght)

%# figure size on screen (50% scaled, but same aspect ratio)
set(gcf, 'Units','centimeters', 'Position',[5 5 xSize ySize])

%# figure size printed on paper
set(gcf, 'PaperUnits','centimeters')
set(gcf, 'PaperSize',[X Y])
set(gcf, 'PaperPositionMode', 'manual')
set(gcf, 'PaperPosition',[xMargin yMargin xSize ySize])
set(gcf, 'PaperOrientation','portrait')
% iptsetpref('ImshowBorder','tight');
% set(gca,'LooseInset',get(gca,'TightInset'))
```

Appendix 6

This MATLAB code runs the both the steady state kalman filter (*sskalmanfilter.m*) and the time varying kalman filter (*tvkalmanfilter.m*).

Script 1: *sskalmanfilter.m*
Script 2: *tvkalmanfilter.m*
Script 3: *auglinearization.m*
Script 4: *fig1.m*
Script 5: *fig2.m*

Script 1(*sskalmanfilter.m)*:

```
close all
clear all
load Data

%Gain matrix for the estimated noise
G=diag([0 0 1 1]);
H=diag([1 1]);
W=diag([10 100 0.0001 0.001]);
V=diag([100 100]);

%Timestep on the nonlinear model [s]
tode=0:60:60*60*24;

%Parameters
hgmax=5.6;
Tslin=60*60*24;
hgmin=0.01*hgmax;
lrv=55.75;


%initializing matrices
yk=zeros(2,1);
xkk_1=zeros(4,1);
xk_1k_1=zeros(4,1);
xkk=zeros(4,1);

%Defining data length
tstart=1000;
tend=length(Data(:,1));
% tend=1000;
deltat=tend-tstart;
newdata=Data(tstart:tend,:);

%Initial steady state values
h1=newdata(1,3)-lrv;
h2=newdata(1,2)-lrv;
% if(h1<=0)
%     h1=0.01;
% end
% if(h2<=0)
%     h2=0.01;
```

```
% end
% if(newdata(1,4)*hgmax<hgmin)
%    hgss=hgmin;
% else
%     hgss=newdata(1,4)*hgmax;
% end
hgss=newdata(1,4)*hgmax;
Vinss=newdata(1,1);
Vtss=newdata(1,5);

%Choosing initial value for parameters of interest
x3=0.05;
x4=1;

x0=[h1 h2 x3 x4]';
xk_1k_1=x0;

yk(1)=newdata(1,3)-lrv;
yk(2)=newdata(1,2)-lrv;

%Plot newdata
xplot=0:1:deltat-1;
xkkplot(:,1)=x0;
ekkplot=zeros(4,deltat);
ekplot=zeros(2,deltat);

%Log the sum of error change
xec=zeros(4,1);

%Calculate steady state kalman gain

xssaug=[3.5 3.2 x3 x4]';
Vinssaug=50;
Vtssaug=24;
hgssaug=0.05*hgmax;

[A, B, C, D]=auglinearization(Tslin, xssaug, hgssaug, Vinssaug, Vtssaug, G,
H);
[Kk,Pp,Pc,E]=dlqe(A,G,C,W,V);

% for(i=2:length(newdata(:,1)))
for(i=2:deltat)

    if(i==412)
       i;
    end

%Progagation step:
%Predicting measurment estimate for k+1
pred_model=@(t,x) augnonlinmodel(t,x, hgss, Vinss, Vtss);
[t,ynonlin]=ode45(pred_model,tode,[xk_1k_1(1) xk_1k_1(2) xk_1k_1(3)
xk_1k_1(4)]);
% [t,ynonlin]=ode45(pred_model,t,xc);
xkk_1(1)=real(ynonlin(length(ynonlin(:,1)),1));
xkk_1(2)=real(ynonlin(length(ynonlin(:,1)),2));
xkk_1(3)=real(ynonlin(length(ynonlin(:,1)),3));
xkk_1(4)=real(ynonlin(length(ynonlin(:,1)),4));


%Measurment update:
```

```
ykk_1=C*xkk_1;
ekk_1=yk-ykk_1;

%Logging the error
ekplot(:,i-1)=ekk_1;

%Corrected measurment estimate
xkk=xkk_1+Kk*ekk_1;

%Logging kalman change effect
ekkplot(:,i-1)=Kk*ekk_1;
xec=xec+Kk*ekk_1;

% if(xkk(3)<0.8)
%    xkk(3)=0.8;
% end

%plots
xkkplot(:,i)=xkk;

%================================
%Shifting up one time instance
yk(1)=newdata(i,3)-lrv;
yk(2)=newdata(i,2)-lrv;
Vtss=newdata(i,5);
% if(newdata(i,4)*hgmax<hgmin)
%    hgss=hgmin;
% else
%     hgss=newdata(i,4)*hgmax;
% end
hgss=newdata(i,4)*hgmax;
Vinss=newdata(i,1);

xk_1k_1=xkk;
if(xk_1k_1(1)<=0)
   xk_1k_1(1)=0.01;
elseif(xk_1k_1(2)<=0)
   xk_1k_1(2)=0.01;
end

i

end



% figure(1)
% subplot(4,1,1)
% plot(xplot,xkkplot(1,:),xplot,newdata(1:deltat,3)-lrv)
% grid on
%
% subplot(4,1,2)
% plot(xplot,xkkplot(2,:),xplot,newdata(1:deltat,2)-lrv)
% grid on
%
% subplot(4,1,3)
% plot(xplot,xkkplot(3,:))
% grid on
% % ylim([-0.5 2])
%
```

```
% subplot(4,1,4)
% plot(xplot,xkkplot(4,:))
% grid on

fig1(xplot,[xkkplot(1,:)' (newdata(1:deltat,3)-lrv)], [xkkplot(2,:)'
newdata(1:deltat,2)-lrv], xkkplot(3,:), xkkplot(4,:));

%# centimeters units
X = 29.7;                   %# A4 paper size
Y = 20;                   %# A4 paper size
xMargin = 1;                %# left/right margins from page borders
yMargin = 1;                %# bottom/top margins from page borders
xSize = X - 2*xMargin;      %# figure size on paper (widht & hieght)
ySize = Y - 2*yMargin;      %# figure size on paper (widht & hieght)

%# figure size on screen (50% scaled, but same aspect ratio)
set(gcf, 'Units','centimeters', 'Position',[5 5 xSize ySize])

%# figure size printed on paper
set(gcf, 'PaperUnits','centimeters')
set(gcf, 'PaperSize',[X Y])
set(gcf, 'PaperPosition',[xMargin yMargin xSize ySize])
set(gcf, 'PaperOrientation','portrait')
iptsetpref('ImshowBorder','tight');
set(gca,'LooseInset',get(gca,'TightInset'))

% figure(2)
% subplot(4,1,1)
% plot(xplot,ekkplot(1,:))
% grid on
%
% subplot(4,1,2)
% plot(xplot,ekkplot(2,:))
% grid on
%
% subplot(4,1,3)
% plot(xplot,ekkplot(3,:))
% grid on
%
% subplot(4,1,4)
% plot(xplot,ekkplot(4,:))
% grid on

fig2(xplot, ekkplot(1,:), ekkplot(2,:), ekkplot(3,:), ekkplot(4,:))

%# centimeters units
X = 29.7;                   %# A4 paper size
Y = 20;                   %# A4 paper size
xMargin = 1;                %# left/right margins from page borders
yMargin = 1;                %# bottom/top margins from page borders
xSize = X - 2*xMargin;      %# figure size on paper (widht & hieght)
ySize = Y - 2*yMargin;      %# figure size on paper (widht & hieght)

%# figure size on screen (50% scaled, but same aspect ratio)
set(gcf, 'Units','centimeters', 'Position',[5 5 xSize ySize])

%# figure size printed on paper
set(gcf, 'PaperUnits','centimeters')
set(gcf, 'PaperSize',[X Y])
set(gcf, 'PaperPosition',[xMargin yMargin xSize ySize])
set(gcf, 'PaperOrientation','portrait')
```

```
iptsetpref('ImshowBorder','tight');
set(gca,'LooseInset',get(gca,'TightInset'))

figure(3)
subplot(2,1,1)
plot(xplot,ekplot(1,:))
grid on

subplot(2,1,2)
plot(xplot,ekplot(2,:))
grid on

xec
```

Script 2(*tvkalmanfilter.m)*:

```
clear all
load Data

%Gain matrix for the estimated noise
G=diag([1 1 1 1]);
H=diag([1 1]);
W=diag([1 10 0.0001 0.0001]);
V=diag([1 10]);

%Timestep on the nonlinear model [s]
t=0:60:60*60*24;

%Parameters
hgmax=5.6;
Tslin=60*60*24;
hgmin=0.001*hgmax;
lrv=55.75;


%initializing matrices
yk=zeros(2,1);
xkk_1=zeros(4,1);
xk_1k_1=zeros(4,1);
xkk=zeros(4,1);

%Defining data length
tstart=1000;
tend=length(Data(:,1));
% tend=100;
deltat=tend-tstart;
newdata=Data(tstart:tend,:);

%Initial steady state values
h1=newdata(1,3)-lrv;
h2=newdata(1,2)-lrv;
if(h1<=0)
    h1=0.01;
end
if(h2<=0)
    h2=0.01;
end
if(newdata(1,4)*hgmax<hgmin)
   hgss=hgmin;
else
```

```
    hgss=newdata(1,4)*hgmax;
end
Vinss=newdata(1,1);
Vtss=newdata(1,5);


x0=[h1,h2,0.05,1]';
xk_1k_1=x0;

yk(1)=newdata(1,3)-lrv;
yk(2)=newdata(1,2)-lrv;

%Defining initial value of the autocovariance matrix
P0=diag([1 10 0.001 0.0001]);
Pk_1k_1=P0;

%Plot newdata
xplot=0:1:deltat-1;
xkkplot(:,1)=x0;
ekkplot=zeros(4,deltat);
ekplot=zeros(2,deltat);

%Log the sum of error change
xec=zeros(4,1);


% for(i=2:length(newdata(:,1)))
for(i=2:deltat)

%Progagation step:
%Predicting measurment estimate for k+1
pred_model=@(t,x) augnonlinmodel(t,x, hgss, Vinss, Vtss);
[t,ynonlin]=ode45(pred_model,t,[xk_1k_1(1) xk_1k_1(2) xk_1k_1(3)
xk_1k_1(4)]);
% [t,ynonlin]=ode45(pred_model,t,xc);
xkk_1(1)=real(ynonlin(length(ynonlin(:,1)),1));
xkk_1(2)=real(ynonlin(length(ynonlin(:,1)),2));
xkk_1(3)=real(ynonlin(length(ynonlin(:,1)),3));
xkk_1(4)=real(ynonlin(length(ynonlin(:,1)),4));

%Find linearized model
[A, B, C, D]=auglinearization(Tslin, xk_1k_1, hgss, Vinss, Vtss, G, H);



%Calculate predicted autocovariance for k+1
Pkk_1=A*Pk_1k_1*A' + B*W*B';

%Measurment update:
ykk_1=C*xkk_1;
ekk_1=yk-ykk_1;
Zkk_1=Pkk_1*C';
Ekk_1=C*Pkk_1*C' + D*V*D';

%Logging the error
ekplot(:,i-1)=ekk_1;


%Find Kalman gain
```

```
Kk=Zkk_1/(Ekk_1);

%Corrected measurment estimate
xkk=xkk_1+Kk*ekk_1;

%Logging kalman change effect
ekkplot(:,i-1)=Kk*ekk_1;
xec=xec+Kk*ekk_1;

% if(xkk(3)<0.8)
%    xkk(3)=0.8;
% end
Pkk=Pkk_1 - Kk*Ekk_1*Kk';


% %Test if observable
% Ob = obsv(A,C);
% if(rank(obsv(A,C))<4)
%     rank(obsv(A,C))
%     pause;
% end

%Check for positive definitness
if(~all(eig(Pkk) > 0))
    Pkk
    pause;
end

if(~all(eig(Pkk_1) > 0))
    Pkk_1
    pause;
end

%plots
xkkplot(:,i)=xkk;

%================================
%Shifting up one time instance
yk(1)=newdata(i,3)-lrv;
yk(2)=newdata(i,2)-lrv;
Vtss=newdata(i,5);
if(newdata(i,4)*hgmax<hgmin)
   hgss=hgmin;
else
    hgss=newdata(i,4)*hgmax;
end
Vinss=newdata(i,1);

xk_1k_1=xkk;
% if(xk_1k_1(1)<=0)
%    xk_1k_1(1)=0.01;
% elseif(xk_1k_1(2)<=0)
%    xk_1k_1(2)=0.01;
% end
Pk_1k_1=Pkk;

i

end
```

```
% figure(1)
% subplot(4,1,1)
% plot(xplot,xkkplot(1,:),xplot,newdata(1:deltat,3)-lrv)
% grid on
%
% subplot(4,1,2)
% plot(xplot,xkkplot(2,:),xplot,newdata(1:deltat,2)-lrv)
% grid on
%
% subplot(4,1,3)
% plot(xplot,xkkplot(3,:))
% grid on
% % ylim([-0.5 2])
%
% subplot(4,1,4)
% plot(xplot,xkkplot(4,:))
% grid on

fig1(xplot,[xkkplot(1,:)' (newdata(1:deltat,3)-lrv)], [xkkplot(2,:)'
newdata(1:deltat,2)-lrv], xkkplot(3,:), xkkplot(4,:));

%# centimeters units
X = 29.7;                   %# A4 paper size
Y = 20;                   %# A4 paper size
xMargin = 1;                %# left/right margins from page borders
yMargin = 1;                %# bottom/top margins from page borders
xSize = X - 2*xMargin;      %# figure size on paper (widht & hieght)
ySize = Y - 2*yMargin;      %# figure size on paper (widht & hieght)

%# figure size on screen (50% scaled, but same aspect ratio)
set(gcf, 'Units','centimeters', 'Position',[5 5 xSize ySize])

%# figure size printed on paper
set(gcf, 'PaperUnits','centimeters')
set(gcf, 'PaperSize',[X Y])
set(gcf, 'PaperPosition',[xMargin yMargin xSize ySize])
set(gcf, 'PaperOrientation','portrait')
iptsetpref('ImshowBorder','tight');
set(gca,'LooseInset',get(gca,'TightInset'))

% figure(2)
% subplot(4,1,1)
% plot(xplot,ekkplot(1,:))
% grid on
%
% subplot(4,1,2)
% plot(xplot,ekkplot(2,:))
% grid on
%
% subplot(4,1,3)
% plot(xplot,ekkplot(3,:))
% grid on
%
% subplot(4,1,4)
% plot(xplot,ekkplot(4,:))
% grid on

fig2(xplot, ekkplot(1,:), ekkplot(2,:), ekkplot(3,:), ekkplot(4,:))
```

```
%# centimeters units
X = 29.7;                   %# A4 paper size
Y = 20;                   %# A4 paper size
xMargin = 1;                %# left/right margins from page borders
yMargin = 1;                %# bottom/top margins from page borders
xSize = X - 2*xMargin;      %# figure size on paper (widht & hieght)
ySize = Y - 2*yMargin;      %# figure size on paper (widht & hieght)

%# figure size on screen (50% scaled, but same aspect ratio)
set(gcf, 'Units','centimeters', 'Position',[5 5 xSize ySize])

%# figure size printed on paper
set(gcf, 'PaperUnits','centimeters')
set(gcf, 'PaperSize',[X Y])
set(gcf, 'PaperPosition',[xMargin yMargin xSize ySize])
set(gcf, 'PaperOrientation','portrait')
iptsetpref('ImshowBorder','tight');
set(gca,'LooseInset',get(gca,'TightInset'))

figure(3)
subplot(2,1,1)
plot(xplot,ekplot(1,:))
grid on

subplot(2,1,2)
plot(xplot,ekplot(2,:))
grid on

%# centimeters units
X = 29.7;                   %# A4 paper size
Y = 20;                   %# A4 paper size
xMargin = 1;                %# left/right margins from page borders
yMargin = 1;                %# bottom/top margins from page borders
xSize = X - 2*xMargin;      %# figure size on paper (widht & hieght)
ySize = Y - 2*yMargin;      %# figure size on paper (widht & hieght)

%# figure size on screen (50% scaled, but same aspect ratio)
set(gcf, 'Units','centimeters', 'Position',[5 5 xSize ySize])

%# figure size printed on paper
set(gcf, 'PaperUnits','centimeters')
set(gcf, 'PaperSize',[X Y])
set(gcf, 'PaperPosition',[xMargin yMargin xSize ySize])
set(gcf, 'PaperOrientation','portrait')
iptsetpref('ImshowBorder','tight');
set(gca,'LooseInset',get(gca,'TightInset'))

xec
```

Script 3(*auglinearization.m)*:

```
function [A,B,C,D] = auglinearization(Ts, xss, hgss, Vinss, Vtss, G, H)

nx = length(xss);

Ix = eye(nx,nx);
t = 0;

h = sqrt(eps);
```

```
A = zeros(nx,nx);
i = sqrt(-1);

for k = 1:nx
A(:,k) = (1/h)* imag(augnonlinmodel(t,xss+h*i*Ix(:,k),hgss,Vinss, Vtss));
end

% B = zeros(nx,1);
% B(:,1) = (1/h)*imag(augnonlinmodel(t,xss,hgss+h*i,Vinss, Vtss));
% B(:,2) = (1/h)*imag(augnonlinmodel(t,xss,hgss,Vinss+h*i, Vtss));

C = [1 0 0 0;0 1 0 0];

d=0;

sys=ss(A,G,C,d);
disc_sys=c2d(sys,Ts);

A=disc_sys.a;
B=disc_sys.b;
C=disc_sys.c;
D=H;


end
```

Script 3(*fig1.m)*:

```
function fig1(X1, YMatrix1, YMatrix2, Y1, Y2)
%CREATEFIGURE(X1, YMATRIX1, YMATRIX2, Y1, Y2)
%  X1:  vector of x data
%  YMATRIX1:  matrix of y data
%  YMATRIX2:  matrix of y data
%  Y1:  vector of y data
%  Y2:  vector of y data

%  Auto-generated by MATLAB on 05-May-2014 14:34:52

% Create figure
figure1 = figure('PaperType','<custom>','PaperSize',[29.7 20]);

% Create subplot
subplot1 = subplot(4,1,1,'Parent',figure1);
box(subplot1,'on');
grid(subplot1,'on');
hold(subplot1,'all');

% Create multiple lines using matrix input to plot
plot1 = plot(X1,YMatrix1,'Parent',subplot1);
set(plot1(2),'LineWidth',2,'LineStyle',':');

% % Create xlabel
% xlabel('Sample number');

% Create ylabel
ylabel('Height [m]');
```

```
% Create title
title('Height h_1');

% Create subplot
subplot2 = subplot(4,1,2,'Parent',figure1);
box(subplot2,'on');
grid(subplot2,'on');
hold(subplot2,'all');

% Create multiple lines using matrix input to plot
plot2 = plot(X1,YMatrix2,'Parent',subplot2);
set(plot2(2),'LineWidth',2,'LineStyle',':');

% % Create xlabel
% xlabel('Sample number');

% Create ylabel
ylabel('Height [m]');

% Create title
title('Height h_2');

% Create subplot
subplot3 = subplot(4,1,3,'Parent',figure1);
box(subplot3,'on');
grid(subplot3,'on');
hold(subplot3,'all');

% Create plot
plot(X1,Y1,'Parent',subplot3,'LineWidth',2);

% % Create xlabel
% xlabel('Sample number');

% Create title
title('Parameter \alpha');

% Create subplot
subplot4 = subplot(4,1,4,'Parent',figure1);
box(subplot4,'on');
grid(subplot4,'on');
hold(subplot4,'all');

% Create plot
plot(X1,Y2,'Parent',subplot4,'LineWidth',2);

% % Create xlabel
% xlabel('Sample number');

% Create title
title('Parameter C_D');
```

Script 3(*fig2.m)*:

```
function fig2(X1, Y1, Y2, Y3, Y4)
%CREATEFIGURE(X1, Y1, Y2, Y3, Y4)
%  X1:  vector of x data
%  Y1:  vector of y data
%  Y2:  vector of y data
```

```
%  Y3:  vector of y data
%  Y4:  vector of y data

%  Auto-generated by MATLAB on 05-May-2014 15:03:05

% Create figure
figure1 = figure;

% Create subplot
subplot1 = subplot(4,1,1,'Parent',figure1);
box(subplot1,'on');
grid(subplot1,'on');
hold(subplot1,'all');

% Create plot
plot(X1,Y1,'Parent',subplot1,'LineWidth',1);

% Create title
title('Height h_1');

% Create subplot
subplot2 = subplot(4,1,2,'Parent',figure1);
box(subplot2,'on');
grid(subplot2,'on');
hold(subplot2,'all');

% Create plot
plot(X1,Y2,'Parent',subplot2,'LineWidth',1);

% Create title
title('Height h_1');

% Create subplot
subplot3 = subplot(4,1,3,'Parent',figure1);
box(subplot3,'on');
grid(subplot3,'on');
hold(subplot3,'all');

% Create plot
plot(X1,Y3,'Parent',subplot3,'LineWidth',1);

% Create title
title('Parameter \alpha');

% Create subplot
subplot4 = subplot(4,1,4,'Parent',figure1);
box(subplot4,'on');
grid(subplot4,'on');
hold(subplot4,'all');

% Create plot
plot(X1,Y4,'Parent',subplot4,'LineWidth',1);

% Create title
title('Parameter C_D');
```

Appendix 7

This is the MATLAB code which run simulations of the linearized model on Lake Toke. The code consists of 5 scripts.

Script 1: *simulation.m*
Script 2: *ssmodel.m*
Script 3: *linearized_model_imag.m*
Script 4: *modelusedforlin.m*
Script 5: *plotsime.m*

Script 1(*simulation.m)*:

```
close all;

hgmax=5.6;
t=[0 10*24*60*60];
x0=[3,2.7]';
Vin1=50;
Vin2=150;
Vint12=2*24*60*60;
hg1=0.02*hgmax;
hg2=0.5*hgmax;
hgt12=5*24*60*60;

[A,B,C,D,M]=linearize_model_imag(x0, 0.01*hgmax, 50,15);

statespacemodel = @(t,x) ssmodel(t,x,Vin1,Vint12,Vin2,hg1,hgt12,hg2,A,B,M);

origmodel = @(t,x) nonlinmodel(t,x,Vin1,Vint12,Vin2,hg1,hgt12,hg2);

[time1,hss]=ode45(statespacemodel,t,x0);

[time2,horig]=ode45(origmodel,t,x0);


plotsimu(time2,[horig(:,1) horig(:,2)],time1, [hss(:,1) hss(:,2)]);

%# centimeters units
X = 29.7;                   %# A4 paper size
Y = 15;                   %# A4 paper size
xMargin = 1;                %# left/right margins from page borders
yMargin = 1;                %# bottom/top margins from page borders
xSize = X - 2*xMargin;      %# figure size on paper (widht & hieght)
ySize = Y - 2*yMargin;      %# figure size on paper (widht & hieght)

%# figure size on screen (50% scaled, but same aspect ratio)
set(gcf, 'Units','centimeters', 'Position',[5 5 xSize ySize])

%# figure size printed on paper
set(gcf, 'PaperUnits','centimeters')
set(gcf, 'PaperSize',[X Y])
set(gcf, 'PaperPosition',[xMargin yMargin xSize ySize])
set(gcf, 'PaperOrientation','portrait')
```

```
iptsetpref('ImshowBorder','tight');
set(gca,'LooseInset',get(gca,'TightInset'))
```

Script 2(*ssmodel.m)*:

```
function [ dxdt ] = ssmodel(t,x,Vin1,Vint12,Vin2,hg1,hgt12,hg2, A, B, M )
dx_dt=zeros(length(x),1);

if(t<=Vint12)
   Vin=Vin1;
else
    Vin=Vin2;
end

if(t<=hgt12)
hg=hg1;
else
    hg=hg2;
end

dxdt=A*x+B*hg+M*Vin;


end
```

Script 3*(linearize_model_imag.m):*

```
function [A,B,C,D,M] = linearize_model_imag(xss, hgss, Vinss, Vtss)

nx = length(xss);

Ix = eye(nx,nx);

h = sqrt(eps);

A = zeros(nx,nx);
i = sqrt(-1);

for k = 1:nx
A(:,k) = (1/h)* imag(modelusedforlin(xss+h*i*Ix(:,k),hgss,Vinss, Vtss));
end

Btemp = zeros(nx,1);
Btemp(:,1) = (1/h)*imag(modelusedforlin(xss,hgss+h*i,Vinss, Vtss));
Btemp(:,2) = (1/h)*imag(modelusedforlin(xss,hgss,Vinss+i*h, Vtss));

% M = zeros(nx,1);
% M(:,1) = (1/h)*imag(nonlinmodel(xss,hgss,Vinss+i*h, Vtss));
% M(:,2) = (1/h)*imag(nonlinmodel(xss,hgss,Vinss, Vtss+i*h));

C = [1 0];

D = 0;


%Separating know disturubance from input
```

```
B=Btemp(:,1);
M=Btemp(:,2);

end
```

Script 4*(modelusedforlin.m):*

```
function [dx_dt]= modelusedforlin(x,hg,Vin,Vt)
dx_dt=zeros(length(x),1);

a=0.05;
B=0.02;
w=11.2;
g=9.81;
A1=max(28*1000000*x(1)^(1/10),1000);
A2=max(28*1000000*x(2)^(1/10),1000);
V12=800*(x(1)-x(2))*sqrt(abs(x(1)-x(2)));

if(x(2)>=hg)
    Vg=hg*w*sqrt(2*g*x(2));
elseif(hg>=x(2) && x(2)>=0)
    Vg=x(2)*w*sqrt(2*g*x(2));
elseif(x(2)<0)
    Vg=0;
end


dx_dt(1) = 1/((1-a)*A1)*((1-B)*Vin-V12);
dx_dt(2) = 1/(a*A2)*(V12-Vt-Vg);

%transpose dx_dt so it is a column vector
% dx_dt = dx_dt';
return
```

Script 5*(plotsimu.m):*

```
function plotsimu(X1, YMatrix1, X2, YMatrix2)
%CREATEFIGURE(X1, YMATRIX1, X2, YMATRIX2)
%  X1:  vector of x data
%  YMATRIX1:  matrix of y data
%  X2:  vector of x data
%  YMATRIX2:  matrix of y data

%  Auto-generated by MATLAB on 21-May-2014 23:27:00

% Create figure
figure1 = figure('PaperType','<custom>','PaperSize',[29.7 15],...
    'InvertHardcopy','off',...
    'Color',[1 1 1]);

% Create subplot
subplot1 = subplot(2,1,1,'Parent',figure1,'YGrid','on','XGrid','on',...
    'FontWeight','light',...
    'FontSize',13);
%% Uncomment the following line to preserve the X-limits of the axes
% xlim(subplot1,[0 450000]);
```

```
% Uncomment the following line to preserve the Y-limits of the axes
ylim(subplot1,[2 4.5]);
%% Uncomment the following line to preserve the Z-limits of the axes
% zlim(subplot1,[-1 1]);
box(subplot1,'on');
hold(subplot1,'all');

% Create multiple lines using matrix input to plot
plot1 = plot(X1,YMatrix1,'Parent',subplot1,'LineWidth',2);
set(plot1(1),'DisplayName','h_1');
set(plot1(2),'LineStyle','--','DisplayName','h_2');

% Create xlabel
xlabel('Time [s]','FontWeight','light','FontSize',13);

% Create ylabel
ylabel('Height h_1 [m]','FontWeight','light','FontSize',13);

% Create title
title('Nonlinear model','FontWeight','light','FontSize',13);

% Create subplot
subplot2 = subplot(2,1,2,'Parent',figure1,'YGrid','on','XGrid','on',...
    'FontWeight','light',...
    'FontSize',13);
%% Uncomment the following line to preserve the X-limits of the axes
% xlim(subplot2,[0 450000]);
% Uncomment the following line to preserve the Y-limits of the axes
ylim(subplot2,[2 4.5]);
%% Uncomment the following line to preserve the Z-limits of the axes
% zlim(subplot2,[-1 1]);
box(subplot2,'on');
hold(subplot2,'all');

% Create multiple lines using matrix input to plot
plot2 = plot(X2,YMatrix2,'Parent',subplot2,'LineWidth',2);
set(plot2(1),'DisplayName','h_1');
set(plot2(2),'LineStyle','--','DisplayName','h_2');

% Create xlabel
xlabel('Times [s]','FontWeight','light','FontSize',13);

% Create ylabel
ylabel('Height h_2 [m]','FontWeight','light','FontSize',13);

% Create title
title('Linearized model','FontWeight','light','FontSize',13);

% Create legend
legend1 = legend(subplot1,'show');
set(legend1,...
    'Position',[0.849913194444444 0.828743961352657 0.0390625
0.0956521739130435]);

% Create legend
legend(subplot2,'show');
```

Appendix 8

This MATLAB code runs the sensitivity test on the parameters in the nonlinear Lake Toke model. The code consists of 4 scripts.

Script 1: *simulation.m*
Script 2: *nonlinmodel.m*
Script 3: *h1plot.m*
Script 4: *h2plot.m*

Script 1*(simulation.m):*

```
close all;

hgmax=5.6;
tmaks=60*60*24*10;
t=[1:10:tmaks];
x0=[2.7,2.5]';
Vt=24;
Vin=20;
u=1;
hg=60/100*hgmax;
V12para=800;
a=0.05;
B=0.02;
Cd=1;


%Calculating the sensitivity of V12para
mymodel1 = @(t,x) nonlinmodel(t,x,Vt,Vin, hg, V12para, a, B, Cd);
mymodel2 = @(t,x) nonlinmodel(t,x,Vt,Vin, hg, V12para+V12para*0.05, a, B,
Cd);

[time,h1]=ode45(mymodel1,t,x0);
[time2,h2]=ode45(mymodel2,t,x0);

h1SV12para=(h2(:,1)-h1(:,1))/(V12para*0.05);
h2SV12para=(h2(:,2)-h1(:,2))/(V12para*0.05);

%Calculating the sensitivity of a
mymodel1 = @(t,x) nonlinmodel(t,x,Vt,Vin, hg, V12para, a, B, Cd);
mymodel2 = @(t,x) nonlinmodel(t,x,Vt,Vin, hg, V12para, a+a*0.05, B, Cd);

[time,h1]=ode45(mymodel1,t,x0);
[time2,h2]=ode45(mymodel2,t,x0);

h1Sa=(h2(:,1)-h1(:,1))/(a*0.05);
h2Sa=(h2(:,2)-h1(:,2))/(a*0.05);

%Calculating the sensitivity of B
mymodel1 = @(t,x) nonlinmodel(t,x,Vt,Vin, hg, V12para, a, B, Cd);
mymodel2 = @(t,x) nonlinmodel(t,x,Vt,Vin, hg, V12para, a, B+B*0.05, Cd);

[time,h1]=ode45(mymodel1,t,x0);
```

```
[time2,h2]=ode45(mymodel2,t,x0);

h1SB=(h2(:,1)-h1(:,1))/(B*0.05);
h2SB=(h2(:,2)-h1(:,2))/(B*0.05);

%Calculating the sensitivity of Cd
mymodel1 = @(t,x) nonlinmodel(t,x,Vt,Vin, hg, V12para, a, B, Cd);
mymodel2 = @(t,x) nonlinmodel(t,x,Vt,Vin, hg, V12para+V12para*0.05, a, B,
Cd+Cd*0.05);

[time,h1]=ode45(mymodel1,t,x0);
[time2,h2]=ode45(mymodel2,t,x0);

h1SCd=(h2(:,1)-h1(:,1))/(Cd*0.05);
h2SCd=(h2(:,2)-h1(:,2))/(Cd*0.05);

h1plot(t, h1SV12para,h1Sa,h1SB,h1SCd)
%# centimeters units
X = 29.7;                   %# A4 paper size
Y = 20;                   %# A4 paper size
xMargin = 1;                %# left/right margins from page borders
yMargin = 1;                %# bottom/top margins from page borders
xSize = X - 2*xMargin;      %# figure size on paper (widht & hieght)
ySize = Y - 2*yMargin;      %# figure size on paper (widht & hieght)

%# figure size on screen (50% scaled, but same aspect ratio)
set(gcf, 'Units','centimeters', 'Position',[5 5 xSize ySize])

%# figure size printed on paper
set(gcf, 'PaperUnits','centimeters')
set(gcf, 'PaperSize',[X Y])
set(gcf, 'PaperPosition',[xMargin yMargin xSize ySize])
set(gcf, 'PaperOrientation','portrait')
iptsetpref('ImshowBorder','tight');
set(gca,'LooseInset',get(gca,'TightInset'))

h2plot(t, h2SV12para,h2Sa,h2SB,h2SCd)
%# centimeters units
X = 29.7;                   %# A4 paper size
Y = 20;                   %# A4 paper size
xMargin = 1;                %# left/right margins from page borders
yMargin = 1;                %# bottom/top margins from page borders
xSize = X - 2*xMargin;      %# figure size on paper (widht & hieght)
ySize = Y - 2*yMargin;      %# figure size on paper (widht & hieght)

%# figure size on screen (50% scaled, but same aspect ratio)
set(gcf, 'Units','centimeters', 'Position',[5 5 xSize ySize])

%# figure size printed on paper
set(gcf, 'PaperUnits','centimeters')
set(gcf, 'PaperSize',[X Y])
set(gcf, 'PaperPosition',[xMargin yMargin xSize ySize])
set(gcf, 'PaperOrientation','portrait')
iptsetpref('ImshowBorder','tight');
set(gca,'LooseInset',get(gca,'TightInset'))
```

Script 2*(nonlinmodel.m):*

```
function [dx_dt]= nonlinmodel(t,x,Vt, Vin,hg, V12para, a, B, Cd)
dx_dt=zeros(length(x),1);
%a function which returns a rate of change vector
% if(t<=Vint12)
%    Vin=Vin1;
% else
%     Vin=Vin2;
% end

% if(t<=hgt12)
% hg=hg1;
% else
%     hg=hg2;
% end
% Vt=24;

% a=0.05;
% B=0.02;
w=11.2;
g=9.81;
A1=max(28*1000000*x(1)^(1/10),1000);
A2=max(28*1000000*x(2)^(1/10),1000);
V12=V12para*(x(1)-x(2))*sqrt(abs(x(1)-x(2)));

if(x(2)>=hg)
    Vg=Cd*hg*w*sqrt(2*g*x(2));
elseif(hg>=x(2) && x(2)>=0)
    Vg=Cd*x(2)*w*sqrt(2*g*x(2));
elseif(x(2)<0)
    Vg=0;
end


dx_dt(1) = 1/((1-a)*A1)*((1-B)*Vin-V12);
dx_dt(2) = 1/(a*A2)*(B*Vin+V12-Vt-Vg);

%transpose dx_dt so it is a column vector
% dx_dt = dx_dt';
return
```

Script 3*(h1plot.m):*

```
function h1plot(X1, Y1, Y2, Y3, Y4)
%CREATEFIGURE(X1, Y1, Y2, Y3, Y4)
%  X1:  vector of x data
%  Y1:  vector of y data
%  Y2:  vector of y data
%  Y3:  vector of y data
%  Y4:  vector of y data

%  Auto-generated by MATLAB on 30-Apr-2014 15:14:50

% Create figure
figure1 = figure;

% Create subplot
subplot1 = subplot(4,1,1,'Parent',figure1,'YGrid','on','XGrid','on');
box(subplot1,'on');
hold(subplot1,'all');
```

```
% Create plot
plot(X1,Y1,'Parent',subplot1,'LineWidth',2);

% Create xlabel
xlabel('Sample time [s]');

% Create ylabel
ylabel('Sensitivity');

% Create title
title('Sensitivity of parameter \lambda');

% Create subplot
subplot2 = subplot(4,1,2,'Parent',figure1,'YGrid','on','XGrid','on');
box(subplot2,'on');
hold(subplot2,'all');

% Create plot
plot(X1,Y2,'Parent',subplot2,'LineWidth',2);

% Create xlabel
xlabel('Sample time [s]');

% Create ylabel
ylabel('Sensitivity');

% Create title
title('Sensitivity of parameter \alpha');

% Create subplot
subplot3 = subplot(4,1,3,'Parent',figure1,'YGrid','on','XGrid','on');
box(subplot3,'on');
hold(subplot3,'all');

% Create plot
plot(X1,Y3,'Parent',subplot3,'LineWidth',2);

% Create xlabel
xlabel('Sample time [s]');

% Create ylabel
ylabel('Sensitivity');

% Create title
title('Sensitivity of parameter \beta');

% Create subplot
subplot4 = subplot(4,1,4,'Parent',figure1,'YGrid','on','XGrid','on');
box(subplot4,'on');
hold(subplot4,'all');

% Create plot
plot(X1,Y4,'Parent',subplot4,'LineWidth',2);

% Create xlabel
xlabel('Sample time [s]');

% Create ylabel
```

```
ylabel('Sensitivity');

% Create title
title('Sensitivity of parameter C_D');
```

Script 4*(h2plot.m):*

```
function h2plot(X1, Y1, Y2, Y3, Y4)
%CREATEFIGURE(X1, Y1, Y2, Y3, Y4)
%  X1:  vector of x data
%  Y1:  vector of y data
%  Y2:  vector of y data
%  Y3:  vector of y data
%  Y4:  vector of y data

%  Auto-generated by MATLAB on 30-Apr-2014 14:53:28

% Create figure
figure1 = figure;

% Create subplot
subplot1 = subplot(4,1,4,'Parent',figure1,'YGrid','on','XGrid','on');
box(subplot1,'on');
hold(subplot1,'all');

% Create plot
plot(X1,Y1,'Parent',subplot1,'LineWidth',2);

% Create xlabel
xlabel('Sample time [s]');

% Create ylabel
ylabel('Sensitivity');

% Create title
title('Sensitivity of the parameter C_D');

% Create subplot
subplot2 = subplot(4,1,3,'Parent',figure1,'YGrid','on','XGrid','on');
box(subplot2,'on');
hold(subplot2,'all');

% Create plot
plot(X1,Y2,'Parent',subplot2,'LineWidth',2);

% Create xlabel
xlabel('Sample time [s]');

% Create ylabel
ylabel('Sensitivity');

% Create title
title('Sensitivity of the parameter \beta');

% Create subplot
```

```
subplot3 = subplot(4,1,2,'Parent',figure1,'YGrid','on','XGrid','on');
box(subplot3,'on');
hold(subplot3,'all');

% Create plot
plot(X1,Y3,'Parent',subplot3,'LineWidth',2);

% Create xlabel
xlabel('Sample time [s]');

% Create ylabel
ylabel('Sensitivity');

% Create title
title('Sensitivity of the parameter \alpha');

% Create subplot
subplot4 = subplot(4,1,1,'Parent',figure1,'YGrid','on','XGrid','on');
box(subplot4,'on');
hold(subplot4,'all');

% Create plot
plot(X1,Y4,'Parent',subplot4,'LineWidth',2);

% Create xlabel
xlabel('Sample time [s]');

% Create ylabel
ylabel('Sensitivity');

% Create title
title('Sensitivity of the parameter \lambda');
```

Appendix 9

This MATLAB code runs a simulation to see if the effect of changing the parameters. The code consists of 4 scripts.

Script 1: *testpara2.m*
Script 2: *originalmodel.m*
Script 3: *nonlinmodel.m*
Script 4: *testparaplot.m*

Script 1*(testpara2.m):*

```
close all
clear all
load Data

hgmax=5.6;
lrv=55.75;

tspan=[0 60*60*24]
x=zeros(2,1);

%defining timespan
tstart=1000;
% tend=length(Data(:,1));
tend=1500;
newdata=Data(tstart:tend,:);
deltat=tend-tstart;

x0=[newdata(1,3)-lrv newdata(1,2)-lrv]';


for(i=1:length(newdata(:,1)))

    i

    %testing new parameters
    if(i<2)
        x=x0;
    else
        x(1)=h1kp1;
        x(2)=h2kp1;
    end
    xplot(i,1)=x(1);
    xplot(i,2)=x(2);

    hg=newdata(i,4)*hgmax;
    Vin=newdata(i,1);
    Vt=newdata(i,5);
    model=@(t,x) nonlinmodel(t,x, hg, Vin, Vt);
    [t,y]=ode45(model,tspan,x);

    h1kp1=y(length(y(:,1)),1);
    h2kp1=y(length(y(:,1)),2);
```

```
end
for(i=1:length(newdata(:,1)))
    i
%using original parameters as a comparison
    if(i<2)
        xorig=x0;
    else
        xorig(1)=h1origkp1;
        xorig(2)=h2origkp1;
    end
    xorigplot(i,1)=xorig(1);
    xorigplot(i,2)=xorig(2);

    hg=newdata(i,4)*hgmax;
    Vin=newdata(i,1);
    Vt=newdata(i,5);
    model=@(t,x) originalmodel(t,x, hg, Vin, Vt);
    [t,y]=ode45(model,tspan,xorig);

    h1origkp1=y(length(y(:,1)),1);
    h2origkp1=y(length(y(:,1)),2);

end

%Logging the error
xerrortest=xplot-[newdata(:,3) newdata(:,2)];
MSEtest1=mean(abs(xerrortest(:,1)))
MSEtest2=mean(abs(xerrortest(:,2)))

xerrororig=xorigplot-[newdata(:,3) newdata(:,2)];
MSEorig1=mean(abs(xerrororig(:,1)))
MSEorig2=mean(abs(xerrororig(:,2)))

tplot=0:1:deltat;

testparplot(tplot,[xplot(:,1) newdata(:,3)-lrv xorigplot(:,1)],[xplot(:,2)
newdata(:,2)-lrv xorigplot(:,2)]);

%# centimeters units
X = 29.7;                   %# A4 paper size
Y = 15;                   %# A4 paper size
xMargin = 1;                %# left/right margins from page borders
yMargin = 1;                %# bottom/top margins from page borders
xSize = X - 2*xMargin;      %# figure size on paper (widht & hieght)
ySize = Y - 2*yMargin;      %# figure size on paper (widht & hieght)

%# figure size on screen (50% scaled, but same aspect ratio)
set(gcf, 'Units','centimeters', 'Position',[5 5 xSize ySize])

%# figure size printed on paper
set(gcf, 'PaperUnits','centimeters')
set(gcf, 'PaperSize',[X Y])
set(gcf, 'PaperPosition',[xMargin yMargin xSize ySize])
set(gcf, 'PaperOrientation','portrait')
iptsetpref('ImshowBorder','tight');
set(gca,'LooseInset',get(gca,'TightInset'))
```

Script 2*(originalmodel.m):*

```
function [dx_dt]= originalmodel(t,x,hg, Vin,Vt)
% tstep=60*60;
% hg=hgv((t+tstep)/tstep);
% Vin=Vinv((t+tstep)/tstep);
% Vt=Vtv((t+tstep)/tstep);
dx_dt=zeros(length(x),1);

omega=11.2;
% %new parameters
% a=0.18;
% B=0.4;

%old parameters
a=0.05;
B=0.02;

w=11.2;
g=9.81;
Cd=1;
A1=max(28*1000000*x(1)^(1/10),1000);
A2=max(28*1000000*x(2)^(1/10),1000);

% A1=2.8e7*1.1* x(1)^0.1;
% A2=2.8e7*1.1* x(2)^0.1;

V12=800*(x(1)-x(2))*sqrt(abs(x(1)-x(2)));
Vg=Cd*hg*omega*sqrt(2*g*x(2));

% if(x(2)>=hg)
%     Vg=hg*w*sqrt(2*g*x(2));
% elseif(hg>=x(2) && x(2)>=0)
%     Vg=x(2)*w*sqrt(2*g*x(2));
% elseif(x(2)<0)
%     Vg=0;
% end

dx_dt(1) = 1/((1-a)*A1)*((1-B)*Vin-V12);
dx_dt(2) = 1/(a*A2)*(V12-Vt-Vg+B*Vin);

end
```

Script 3*(nonlinmodel.m):*

```
function [dx_dt]= nonlinmodel(t,x,hg, Vin,Vt)
% tstep=60*60;
% hg=hgv((t+tstep)/tstep);
% Vin=Vinv((t+tstep)/tstep);
% Vt=Vtv((t+tstep)/tstep);
dx_dt=zeros(length(x),1);

omega=11.2;
% %new parameters
% a=0.18;
% B=0.4;
```

```
%old parameters
a=0.05;
B=0.7;
w=11.2;
g=9.81;
Cd=1;
A1=max(28*1000000*x(1)^(1/10),1000);
A2=max(28*1000000*x(2)^(1/10),1000);

% A1=2.8e7*1.1* x(1)^0.1;
% A2=2.8e7*1.1* x(2)^0.1;

V12=800*(x(1)-x(2))*sqrt(abs(x(1)-x(2)));
Vg=Cd*hg*omega*sqrt(2*g*x(2));

% if(x(2)>=hg)
%     Vg=hg*w*sqrt(2*g*x(2));
% elseif(hg>=x(2) && x(2)>=0)
%     Vg=x(2)*w*sqrt(2*g*x(2));
% elseif(x(2)<0)
%     Vg=0;
% end

dx_dt(1) = 1/((1-a)*A1)*((1-B)*Vin-V12);
dx_dt(2) = 1/(a*A2)*(V12-Vt-Vg+B*Vin);

end
```

Script 3*(testparplot.m):*

```
function testparplot(X1, YMatrix1, YMatrix2)
%CREATEFIGURE(X1, YMATRIX1, YMATRIX2)
%  X1:  vector of x data
%  YMATRIX1:  matrix of y data
%  YMATRIX2:  matrix of y data

%  Auto-generated by MATLAB on 06-May-2014 16:45:33

% Create figure
figure1 = figure;

% Create subplot
subplot1 = subplot(2,1,1,'Parent',figure1);
box(subplot1,'on');
grid(subplot1,'on');
hold(subplot1,'all');

% Create multiple lines using matrix input to plot
plot1 = plot(X1,YMatrix1,'Parent',subplot1,'LineWidth',2);
set(plot1(1),'LineStyle',':','DisplayName','Using test parameter');
set(plot1(2),'DisplayName','Real measuments');
set(plot1(3),'LineStyle','--','DisplayName','Using original parameters');

% Create xlabel
xlabel('Time [Days]');

% Create ylabel
ylabel('Height [m]');
```

```
% Create title
title('Plot of the height h_1');

% Create subplot
subplot2 = subplot(2,1,2,'Parent',figure1);
box(subplot2,'on');
grid(subplot2,'on');
hold(subplot2,'all');

% Create multiple lines using matrix input to plot
plot2 = plot(X1,YMatrix2,'Parent',subplot2,'LineWidth',2);
set(plot2(1),'LineStyle',':','DisplayName','Using test parameter');
set(plot2(2),'DisplayName','Real measuments');
set(plot2(3),'LineStyle','--','DisplayName','Using original parameters');

% Create xlabel
xlabel('Time [days]');

% Create ylabel
ylabel('Heigh [m]');

% Create title
title('Plot of the height h_2');

% Create legend
legend1 = legend(subplot1,'hide');
set(legend1,'Location','SouthEast');

% Create legend
legend2 = legend(subplot2,'hide');
set(legend2,'Location','SouthEast');
```