ISSN 0280–5316
ISRN LUTFD2/TFRT--5661--SE

Predictive Control of Irrigation Channels

Pia Eklund
Marianne Tufvesson

Department of Automatic Control
Lund Institute of Technology
February 2001

Department of Automatic Control **Lund Institute of Technology** **Box 118** **SE-221 00 Lund Sweden**	*Document name* MASTER THESIS *Date of issue* February 2001 *Document Number* ISRN LUTFD2/TFRT--5661--SE
Author(s) Pia Eklund and Marianne Tufvesson	*Supervisor* Erik Weyer, The University of Melbourne Björn Wittenmark *Sponsoring organisation*

Title and subtitle

Predictive Control of Irrigation Channels

Abstract

The aim of this project is to make a predictive controller of a part of an irrigation channel. The channel is called "The Haughton Main Channel" (HMC) and is situated in Queensland. Irrigation channels are used by farmers all around the year and it is important that they deliver water on demand. The project has been made in cooperation with the Department of Electrical and Electronic Engineering at the University of Melbourne and the company Rubicon Systems Australia.

The goals of the project are to keep the water level constant in the main channel despite of disturbances (when farmers are taking water from the channel), and also to move the gates as little as possible because of the limited power of the solar cells and we do not want to cause waves in the channels

To be able to reach these goals it is important to have a good model for the channel that explains its physical behaviour in mathematical terms. When you have the mathematical functions you can make a predictive controller with help from "Model Predictive Control" (MPC) theories. The criterion for MPC is the same as for linear quadratic control (LQ-control), for more details see chapter three. The control law is computed over a certain time horizon. In every time step an optimization is made and a new control law for that time step is calculated. The time horizon is then moved one time step and a new optimization is made. The resulting control law is in general time varying and cannot be expressed in closed form.

In this report we will explain how we have developed a predictive controller and the different tools we have been using. Our result is a predictive controller that controls two pools in the Haughton Main Channel.

Key words

Classification system and/ or index terms (if any)

Supplementary bibliographical information

ISSN and key title 0280–5316		*ISBN*
Language English	*Number of pages* 88	*Recipient's notes*
Security classification		

Preface

This report is a Master Thesis written by Pia Eklund and Marianne Tufvesson at the University of Melbourne, Australia, in the second half of year 2000. The project is a part of the education Master of Science in Electrical Engineering at Lund Institute of Technology, Sweden. It is made in cooperation with Rubicon Systems Australia and the department of Electrical and Electronic Engineering at the University of Melbourne in Australia.

We would like to take this opportunity to thank our supervisor in Australia, Dr. Erik Weyer at the department of Electrical and Electronic Engineering, the University of Melbourne, and our supervisor in Sweden, professor Björn Wittenmark at the department of Automatic Control, Lund Institute of Technology, who both have helped us to make this project possible. We would also like to thank professor Iven Mareels at the University of Melbourne, who helped us arrange the stay in Australia.

Pia Eklund and Marianne Tufvesson

Contents

1 Introduction 9
1.1 Irrigation . . . 9
1.2 The problem . . . 10
1.3 Outline . . . 10

2 Description of the System 11
2.1 Physical constraints . . . 12
2.2 Offtakes . . . 12
2.2.1 Offtake in pool eight . . . 12
2.2.2 Offtake in pool nine . . . 13
2.3 The model . . . 14
2.4 Our model . . . 14
2.5 State-space representation . . . 15
2.5.1 Representation with linear model and nonlinear constraints with sampling-time one . . . 16
2.5.2 Representation with linear model and nonlinear constraints with sampling-time two . . . 17
2.5.3 Representation with nonlinear model and linear constraints with sampling-time one . . . 19
2.5.4 Representation with nonlinear model and linear constraints with sampling-time two . . . 20
2.6 Summary . . . 21

3 Predictive Control 21
3.1 Model Predictive Control . . . 21
3.1.1 Simple example of MPC . . . 21
3.2 Optimization strategy . . . 22
3.3 Summary . . . 24

4 Optimization 24
4.1 MATLAB programming . . . 24
4.1.1 Conclusion . . . 25
4.2 Implementation in C . . . 25
4.2.1 Description of the program . . . 26
4.2.2 Up and running . . . 27
4.3 Simulation model . . . 27
4.4 Interface with Simulink model . . . 29
4.5 Summary . . . 30

5 The Performance of the Predictive Controller 31
5.1 Tuning of parameters . . . 31
5.2 Performance tests . . . 34
5.2.1 Disturbance in pool eight . . . 35
5.2.2 Disturbance in pool nine . . . 38
5.2.3 Initial water level in pool eight below error tolerance . . . 41
5.2.4 Initial water level in pool nine below error tolerance . . . 47
5.2.5 Initial water level in pool eight above error tolerance . . . 52
5.2.6 Initial water level in pool nine above error tolerance . . . 54

5.2.7 Summary 59
5.3 Practical implementation 59
5.3.1 Computational requirements 59
5.3.2 Communication 59

6 Conclusions **60**
6.1 Results 60
6.2 Future work 60

A Derivatives used in MATLAB - fmincon **61**
A.1 Derivatives of $h_8(k)$ - $h_{10}(k)$ with respect to Δ $h_8(k)$- Δ $h_{10}(k)$. 61
A.2 Calculation of the gradient 61
A.2.1 Derivatives of $\hat{y}_9$ 61
A.2.2 Derivatives of $\hat{y}_{10}$ 62
A.2.3 Derivatives of $\hat{h}_3$ 63
A.2.4 Derivatives of the constraints 64

B Derivatives in C-program, version one **70**
B.1 Derivatives of the cost function 70
B.2 Derivatives of the transitions 70
B.3 Derivatives of the constraints 71

C Derivatives in C-program, version two **72**
C.1 Derivatives of the cost function 72
C.2 Derivatives of the transitions 72
C.3 Derivatives of the constraints 73

D Derivatives in C-program, version three **75**
D.1 Derivatives of the cost function 75
D.2 Derivatives of the transitions 75
D.3 Derivatives of the constraints 76

E Costder **78**

F Nonlcon **83**

G LQR-test **89**

1 Introduction

1.1 Irrigation

In earliest times the purely agricultural communities tended to settle down in the better-watered regions, i.e. in areas of highest or most reliable rainfall and along the main rivers. There is, however, evidence that irrigation was practiced from prehistoric times (see [1]). Egypt claims to have had the world's oldest dam, built 5000 years ago to store water for drinking and irrigation. Basin irrigation introduced in the Nile valley around 3000 B.C. still plays an important part in the Egyptian agriculture.

During the industrial revolution of Europe, people moved from the countryside to the cities and the population started to grow. Many irrigation projects were set up in Egypt, India, Pakistan, and Iraq by British engineers to find out how to meet the growing needs of water, both for the growing cities and for arid land.

Aridity is not unnatural. The arids now existing in the world are far too vast to have been created wholly by human beings (see [2]) . Of course man-made deserts do exist—parts of the earth's surface where the intervention of man has prevented plant growth. But their total extent is probably small in relation to the areas of arid land for which the forces of nature alone are responsible. The countries or continents that have arid land are Africa, the Middle-East, the United States, Australia, some parts in South-America and the north of India. In these countries and continents there is a great need for irrigation on farms for crops and animals.

Water has now become a very scarce resource in many parts of the world because of the increasing population. We have taken for granted the seemingly endless supplies of water and access to water has been the key to agriculture and industrialization. A lot of water is used in agriculture and the number is growing. To grow 320 tons of food on one square mile ($\approx 2.6\ km^2$) of land, more than one million tons of water may be needed. Studies, see [4], have shown that building large new dams and river diversions is becoming costly and environmentally damaging. In most cases it is better to use the water already supplied more efficiently. It is therefore very important to reduce the wastages of water through, for example, better control of irrigation channels. A lot of water in the channels is wasted because of lack of efficient control. The channels are often overdimensioned because of lack of knowledge when they were built and causes much water to go wasted. The water delivering system is not perfect and to make it better we have to minimize the water losses, but we also have to deliver water to the farmers on demand. In this thesis we have tried to develop a predictive controller for a specific part of the Haughton Main Channel, HMC, to see if this kind of controller would be suitable for irrigation channels. The HMC is situated in Queensland, on the east coast of Australia. The HMC gets its water from the Burdekin Dam in north Queensland. For more information see [3].

1.2 The problem

This project is a part of the Rubicon project. Rubicon Systems Australia [5] is a systems integration company which supplies specialist operational technology to the water industry. Erik Weyer, our supervisor, is working together with Rubicon to help them develop better methods to reduce the losses of the irrigation channels.

The goals of the project are:

- Keep the water level constant in the main channel despite of disturbances
- Move the gates as little as possible

The water level should be kept on a constant level so there always is enough water for the farmers. The gates should be moved as little as possible because of the limited power from the solar cells. Movements of the gates can also cause waves in the channel and we do not want that because the model does not include wave effects.

To reach these goals the first thing to do is to find a good model for the channels. We want to describe the channel and its behaviour with mathematical functions. The next thing to do is to find the function to be optimized. The function to be minimized is the error and the last head over gate. In our case the error is the difference between the real water level and the desired water level. The error and the head over gate should then be minimized by MPC (Model Predictive Control) theories, and this is done in the chapter called Predictive Control. MPC gives high performance control systems that can operate without intervention for long periods of time which is needed in irrigation systems.

1.3 Outline

In this report a predictive controller is developed. We start by giving a description of the system in chapter 2. The physical description of the channel was supplied to us by our supervisor Erik Weyer. We have made the state-space formulation and the formulation of the problem. In chapter 3 we give an overview of Model Predictive Control, the control strategy that we have been using. Chapter 4 shows the different optimization tools that have been used. The final optimization tool that we implemented our problem in, is a program made by James Wettenhall. The performance of the predictive controller is described in chapter 5. Here we have made different simulations to see how the controller reacts in different situations. The simulation model that we have been using is made by [E. Weyer, E. Lim]. We have made some changes in the model to suit our problem. Chapter 6 completes the report with conclusions and suggested future work.

2 Description of the System

An irrigation channel consists of many gates and between two gates there is a stretch called a pool. The water level in the pool is controlled by the gates located at both ends of the pool. We have considered gate eight, nine and ten of the HMC. The measurements available are the water level upstream and downstream of each gate and the gate position. The height of water above a gate is called the head over gate and can be computed from the upstream water level and the gate position. The upstream water level is referred to as y_i and the head over gate as h_i, see figure 1. The water levels are all measured in mAHD (meters Australian Height Datum). This is a level with a reference to a certain point in Australia which is set to be zero.

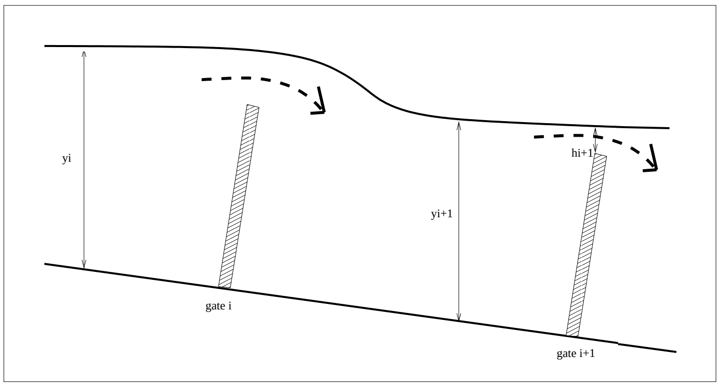

Figure 1: The Channel

The gates are overshot gates, which means that the water is moving over the gates. In some places undershot gates, when the water goes under the gate, are used but not in this case. The gate opening increases by rotating the whole gate around the lower edge axis. The gate position is represented by p in meter as shown in figure 2. When the gate is completely closed the value of p is zero.

The desired water levels are referred to as y_{9sp} and y_{10sp} and the values are:

$y_{9sp} = 26.50$ mAHD
$y_{10sp} = 23.85$ mAHD

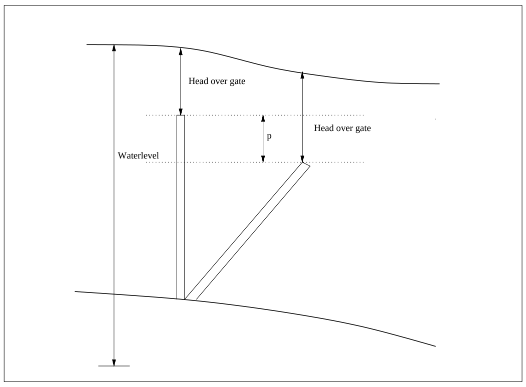

Figure 2: The Gate

2.1 Physical constraints

It is just possible to move the gate between two values. These values are different from gate to gate. The different limitations are shown below in meter:

$$0 \leq p_{08} \leq 2.005$$
$$0 \leq p_{09} \leq 1.474$$
$$0 \leq p_{010} \leq 1.487$$

We also have a limitation of how much the gates can change in every time step, Δp_{08}, Δp_{09} and Δp_{010}. There is a rate limit of the gate movements and this limit is ± 0.4 meter per time step. Another limitation that is not actually a physical limitation is that we do not want the head over gate to be negative. This limitation will also decrease the movements of the gates. The shapes and dimensions of the pools are shown in figure 3 and 4.

2.2 Offtakes

When a farmer takes out water from the channel we refer to it as an offtake. We show below how these are calculated for pool eight and nine.

2.2.1 Offtake in pool eight

In the real pool eight there exist some offtakes that we have to take into consideration. The maximum offtake is 30 mega liters per day but this offtake is a bit too small to see the influence on the pool so we decided to try an offtake of 530 mega liters per day instead. This offtake is implemented as a disturbance in the model. To be able to implement the disturbance we calculated the difference in water level per minute, Δy, that corresponds to the difference in the flow. The amount of 530 mega liters turned out to be a good value to see reactions on the pool.

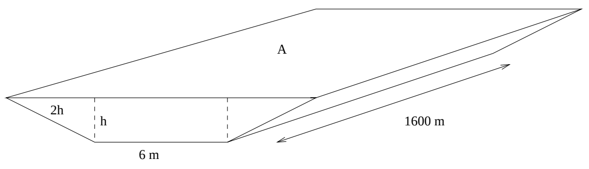

Figure 3: the shape of pool eight

The difference in the water level is so small so we assume the area is constant over this interval. The area times Δy corresponds to the volume per minute.

Calculations:

$A = 1600 \cdot (6 + 2 \cdot 2h)$
$h = 1.6m \Longrightarrow A = 19840m^2$

change in volume per minute:

$\Delta V = A \cdot \Delta y$

With the flow disturbance 540 mega liters per day Δy becomes:

$\Delta y = \frac{530 \cdot 10^3}{60 \cdot 24} / 19840 \approx 0.01855m$

2.2.2 Offtake in pool nine

Even if there does not exist any offtakes in pool nine it is possible that somebody wants to implement an offtake later on. In this case we counted with an offtake of 300 mega liters per day. The calculation for this pool is shown below:

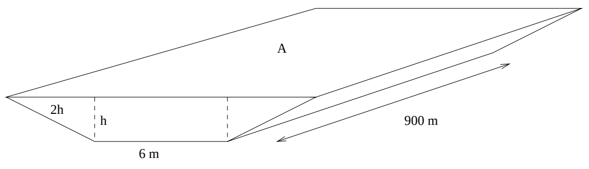

Figure 4: the shape of pool nine

The area times Δy corresponds to the volume per minute.

$A = 900 \cdot (6 + 2 \cdot 2h)$
$h = 1.6m \Longrightarrow A = 11160m^2$

change in volume per minute:

$\Delta V = A \cdot \Delta y$

With the flow disturbance 300 mega liters per day Δy becomes:

$\Delta y = \frac{300 \cdot 10^3}{60 \cdot 24}/11160 \approx 0.01867m$

2.3 The model

Prediction and control of irrigation channels require models. The water level is the controlled variable, so it is clear that a model for control should have the water level as output. As the gate position is the variable that is changeable this is considered as the input signal.

In most irrigation channels the channel is modeled by the St. Venant equations, see [18]. These equations are nonlinear and are derived from mass and momentum balances and are given by:

$$\frac{\partial A}{\partial t} + \frac{\partial Q}{\partial x} = 0$$

$$\frac{\partial Q}{\partial t} + (\frac{gA}{B} - \frac{Q^2}{A^2}) \cdot \frac{\partial A}{\partial x} + \frac{2Q}{A} \cdot \frac{\partial A}{\partial x} + gAS_f - gA\vec{S} = 0$$

where
A is the cross section area of the channel
B is the width of the water surface
Q is the flow
g is the gravity
S_f is the friction slope
$\vec{S}$ is the mean bed slope

The St Venant equations are known to be good from laboratory experiments but the real world is different from the ideal laboratory environment. In our model the problems with the St Venant equations are that the friction slope is unknown, the equations do not model the flow over the gate and it is also difficult to use the equations for prediction and control. However, the St Venant equations together with some equations for the flow across the gates are usually taken as starting point for developing models for control.

2.4 Our model

Using system identification [6], [7], we are able to obtain models that are in agreement with the physical reality and also useful for predictive control.

If the behaviour of the overshot gate is approximately the same as a sharp crested weir, then the flow over the gate can be written as

$$Q(t) = ch^{3/2}(t) \qquad (1)$$

where Q is the flow, h is the head over gate and c is an unknown parameter. c incorporates the effect of the discharge coefficients introduced in order to compensate for the incorrect assumptions made in the derivation in the flow equation.

If a simplified mass balance is considered and the volume in the pool is considered to be proportional to the water level the mass balance would be:

$$\dot{y}_9(t) = \tilde{c}_1 h_8^{3/2}(t) + \tilde{c}_2 h_9^{3/2}(t) = \tilde{c}_1 h_8^{3/2}(t) + \tilde{c}_2 (y_9(t) - p_9(t))^{3/2} \qquad (2)$$

$y_9(t)$ is the water level upstream of gate ten, $h_8(t)$ the head over gate eight, $h_9(t)$ the head over gate nine and $p_9(t)$ the position of gate nine. There is a time delay between the two gates and taking this into account and introducing an Euler approximation for the derivatives means that the following equation ([6]) can be obtained:

$$y_9(t+1) = y_9(t) + c_1 h_8^{3/2}(t-\tau) + c_2 (y_9(t) - p_2(t))^{3/2} \qquad (3)$$

The model for the stretch of the channel from gate eight to gate ten can be modeled in a similar way and the result is:

$$y_9(t+1) = y_9(t) + c_1 h_8^{3/2}(t-\tau_8) + c_2 h_9(t)^{3/2} \qquad (4)$$

$$y_{10}(t+1) = y_{10}(t) + c_3 h_9^{3/2}(t-\tau_9) + c_4 h_{10}(t)^{3/2} \qquad (5)$$

The waves in the channel are not represented in this model as more complex models are needed for this which are unnecessary complex for controller design.

The relation between the gate position, the water level and the head over gate is as follows:

$h_i(t) = y_i(t) + \frac{p_i(t)}{1000} - g_i$

where $h_i(t)$ is the head over gate (m), $y_i(t)$ the upstream water level (m), $p_i(t)$ the gate position (mm) and g_i is the adjustment factor.

2.5 State-space representation

To be able to do Model Predictive Control we need a mathematical model for the system. The water level in the next time step can be described by:

$y_9(k+1) = y_9(k) + c_1 h_8^{3/2}(k-\tau_8) + c_2 h_9^{3/2}(k)$
$y_{10}(k+1) = y_{10}(k) + c_3 h_9^{3/2}(k-\tau_9) + c_4 h_{10}^{3/2}(k)$

where

$\tau_8 = 6$ min
$\tau_9 = 4$ min

For one minute sampling period
$c_1 = 0.0208$
$c_2 = -0.0278$

$c_3 = 0.0650$
$c_4 = -0.0660$

For two minutes sampling period
$c_1 = 0.0416$
$c_2 = -0.0554$
$c_3 = 0.1488$
$c_4 = -0.134$

The water level in the next time step, $y(k+1)$, is the water level in the previous time step, $y(k)$, plus the flow into the pool minus the flow out from the pool. τ_8 and τ_9 are the time delays for the different pools, k is the time step, h is the head over the gate and c is a constant that compensates for incorrect assumptions made in the derivation in the flow equation 1. c is determined by system identification, see [6] and [7]. τ_8 and τ_9 have to be adjusted for the sampling period. We must be able to express the time delay τ in k because we want to replace the time delays with states. We have been using different sampling periods, one and two minutes because a sampling time of two minutes reduced our model and it was easier for our optimization tool to handle this. We have also been using two different models, one linear and one nonlinear. In the case of a linear model we have put the nonlinearity on the constraints instead. The nonlinear model with linear constraints turned out to be the best representation for the optimization tool. This model is shown in chapter 2.5.4.

The following equation which shows the difference in flow was also used:

$h_i^{3/2}(k+1) = h_i^{3/2}(k) + \Delta x_i(k)$

Here $\Delta x_i(k)$ is the difference in the head over gate risen to three halves between two time steps and is used as our input signal for the system.

To take care of the time delays new states were introduced. This is shown in the representation below.

2.5.1 Representation with linear model and nonlinear constraints with sampling-time one

In this state-space model the sampling-time is one minute. The calculation is shown below:

$z_0(k) = h_8^{3/2}(k)$
$z_0(k+1) = h_8^{3/2}(k+1) = h_8^{3/2}(k) + \Delta x_8(k) = z_0(k) + \Delta x_8(k)$
$z_1(k+1) = z_0(k)$
$z_2(k+1) = z_1(k) = z_0(k-1)$
$z_3(k+1) = z_2(k) = z_0(k-2)$
$z_4(k+1) = z_3(k) = z_0(k-3)$
$z_5(k+1) = z_4(k) = z_0(k-4)$
$z_6(k+1) = z_5(k) = z_0(k-5)$

$s_0(k+1) = h_9^{3/2}(k+1) = h_9^{3/2}(k) + \Delta x_9(k) = s_0(k) + \Delta x_9(k)$
$s_1(k+1) = s_0(k)$

$s_2(k+1) = s_1(k) = s_0(k-1)$
$s_3(k+1) = s_2(k) = s_0(k-2)$
$s_4(k+1) = s_3(k) = s_0(k-3)$

$w_0(k+1) = h_{10}^{3/2}(k+1) = h_{10}^{3/2}(k) + \Delta x_{10}(k) = w_0(k) + \Delta x_{10}(k)$

$y_9(k+1) = y_9(k) + c_1 h_8^{3/2}(k-6) + c_2 h_9^{3/2}(k) = y_9(k) + c_1 z_6(k) + c_2 s_0(k)$
$y_{10}(k+1) = y_{10}(k) + c_3 h_9^{3/2}(k-4) + c_4 h_{10}^{3/2}(k) = y_{10}(k) + c_3 s_4(k) + c_4 w_0(k)$

Written in matrix form this becomes:

$$\begin{bmatrix} z_0(k+1) \\ z_1(k+1) \\ z_2(k+1) \\ z_3(k+1) \\ z_4(k+1) \\ z_5(k+1) \\ z_6(k+1) \\ s_0(k+1) \\ s_1(k+1) \\ s_2(k+1) \\ s_3(k+1) \\ s_4(k+1) \\ w_0(k+1) \\ y_9(k+1) \\ y_{10}(k+1) \end{bmatrix} = \begin{bmatrix} 1&0&0&0&0&0&0&0&0&0&0&0&0&0&0 \\ 1&0&0&0&0&0&0&0&0&0&0&0&0&0&0 \\ 0&1&0&0&0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&1&0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&1&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&1&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&1&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&1&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&1&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&1&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&1&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0&0&1&0&0 \\ 0&0&0&0&0&0&c_1&c_2&0&0&0&0&0&1&0 \\ 0&0&0&0&0&0&0&0&0&0&c_3&c_4&0&0&1 \end{bmatrix} \begin{bmatrix} z_0(k) \\ z_1(k) \\ z_2(k) \\ z_3(k) \\ z_4(k) \\ z_5(k) \\ z_6(k) \\ s_0(k) \\ s_1(k) \\ s_2(k) \\ s_3(k) \\ s_4(k) \\ w_0(k) \\ y_9(k) \\ y_{10}(k) \end{bmatrix} + \begin{bmatrix} 1&0&0 \\ 0&0&0 \\ 0&0&0 \\ 0&0&0 \\ 0&0&0 \\ 0&0&0 \\ 0&0&0 \\ 0&1&0 \\ 0&0&0 \\ 0&0&0 \\ 0&0&0 \\ 0&0&0 \\ 0&0&1 \\ 0&0&0 \\ 0&0&0 \end{bmatrix} \begin{bmatrix} \Delta x_8(k) \\ \Delta x_9(k) \\ \Delta x_{10}(k) \end{bmatrix}$$

The nonlinear constraints are written on the form g(k) ≥ 0

$$\begin{aligned} g_0(k) &= z_0^{2/3}(k) - y_8 + 27.693 \\ g_1(k) &= -(z_0^{2/3}(k) - y_8 + 27.693 - 2.005) \\ g_2(k) &= s_0^{2/3}(k) - y_9(k) + 26.545 \\ g_3(k) &= -(s_0^{2/3}(k) - y_9(k) + 26.545 - 1.474) \\ g_4(k) &= w_0^{2/3}(k) - y_{10}(k) + 24.018 \\ g_5(k) &= -(w_0^{2/3}(k) - y_{10}(k) + 24.018 - 1.487) \\ g_6(k) &= z_0(k) \\ g_7(k) &= s_0(k) \\ g_8(k) &= w_0(k) \end{aligned}$$

2.5.2 Representation with linear model and nonlinear constraints with sampling-time two

In the next state-space model the sampling time,l, is two minutes. Now the calculations become:

$z_0(kl) = h_8^{3/2}(kl)$
$z_0(kl+l) = h_8^{3/2}(kl+l) = h_8^{3/2}(kl) + \Delta x_8(kl) = z_0(kl) + \Delta x_8(kl)$
$z_1(kl+l) = z_0(kl)$
$z_2(kl+l) = z_1(kl) = z_0(kl-l)$
$z_3(kl+l) = z_2(kl) = z_0(kl-2l)$

$l = 2 \Longrightarrow$

$$\begin{aligned}
&z_0(2k) = h_8^{3/2}(2k)\\
&z_0(2k+2) = z_0(2k) + \Delta x_8(2k)\\
&z_1(2k+2) = z_0(2k)\\
&z_2(2k+2) = z_1(2k)\\
&z_3(2k+2) = z_2(2k)
\end{aligned}$$

$$\begin{aligned}
&s_0(kl+l) = h_9^{3/2}(kl+l) = h_9^{3/2}(kl) + \Delta x_9(kl) = s_0(kl) + \Delta x_9(kl)\\
&s_1(kl+l) = s_0(kl)\\
&s_2(kl+l) = s_1(kl) = s_0(kl-l)
\end{aligned}$$

$l = 2 \Longrightarrow$

$$\begin{aligned}
&s_0(2k+2) = s_0(2k) + \Delta x_9(2k)\\
&s_1(2k+2) = s_0(2k)\\
&s_2(2k+2) = s_1(2k)
\end{aligned}$$

$$w_0(kl+l) = h_{10}^{3/2}(kl+l) = h_{10}^{3/2}(kl) + \Delta x_{10}(kl) = w_0(kl) + \Delta x_{10}(kl)$$

$l = 2 \Longrightarrow$

$$w_0(2k+2) = w_0(2k) + \Delta x_{10}(2k)$$

$$\begin{aligned}
&y_9(kl+l) = y_9(kl) + c_1 h_8^{3/2}(kl-6) + c_2 h_9^{3/2}(kl) = y_9(kl) + c_1 z_3(kl) + c_2 s_0(kl)\\
&y_{10}(kl+l) = y_{10}(kl) + c_3 h_9^{3/2}(kl-4) + c_4 h_{10}^{3/2}(kl) = y_{10}(kl) + c_3 s_2(kl) + c_4 w_0(kl)
\end{aligned}$$

$l = 2 \Longrightarrow$

$$\begin{aligned}
&y_9(2k+2) = y_9(2k) + c_1 z_3(2k) + c_2 s_0(2k)\\
&y_{10}(2k+2) = y_{10}(2k) + c_3 s_2(2k) + c_4 w_0(2k)
\end{aligned}$$

Written in matrix form this becomes:

$$\begin{bmatrix} z_0(2k+2)\\ z_1(2k+2)\\ z_2(2k+2)\\ z_3(2k+2)\\ z_4(2k+2)\\ z_5(2k+2)\\ z_6(2k+2)\\ s_0(2k+2)\\ s_1(2k+2)\\ s_2(2k+2)\\ s_3(2k+2)\\ s_4(2k+2)\\ w_0(2k+2)\\ y_9(2k+2)\\ y_{10}(2k+2) \end{bmatrix} = \begin{bmatrix} 1&0&0&0&0&0&0&0&0&0\\ 1&0&0&0&0&0&0&0&0&0\\ 0&1&0&0&0&0&0&0&0&0\\ 0&0&1&0&0&0&0&0&0&0\\ 0&0&0&0&1&0&0&0&0&0\\ 0&0&0&0&1&0&0&0&0&0\\ 0&0&0&0&0&1&0&0&0&0\\ 0&0&0&0&0&0&0&1&0&0\\ 0&0&0&c_1&c_2&0&0&0&1&0\\ 0&0&0&0&0&0&c_3&c_4&0&1 \end{bmatrix} \begin{bmatrix} z_0(2k)\\ z_1(2k)\\ z_2(2k)\\ z_3(2k)\\ s_0(2k)\\ s_1(2k)\\ s_2(2k)\\ w_0(2k)\\ y_9(2k)\\ y_{10}(2k) \end{bmatrix} + \begin{bmatrix} 1&0&0\\ 0&0&0\\ 0&0&0\\ 0&0&0\\ 0&1&0\\ 0&0&0\\ 0&0&0\\ 0&0&1\\ 0&0&0\\ 0&0&0 \end{bmatrix} \begin{bmatrix} \Delta x_8(2k)\\ \Delta x_9(2k)\\ \Delta x_{10}(2k) \end{bmatrix}$$

The nonlinear constraints are written on the form g(2k) ≥ 0

$$\begin{aligned}
g_0(2k) &= z_0^{2/3}(2k) - y_8 + 27.693 \\
g_1(2k) &= -(z_0^{2/3}(2k) - y_8 + 27.693 - 2.005) \\
g_2(2k) &= s_0^{2/3}(2k) - y_9(2k) + 26.545 \\
g_3(2k) &= -(s_0^{2/3}(2k) - y_9(2k) + 26.545 - 1.474) \\
g_4(2k) &= w_0^{2/3}(2k) - y_{10}(2k) + 24.018 \\
g_5(2k) &= -(w_0^{2/3}(2k) - y_{10}(2k) + 24.018 - 1.487) \\
g_6(2k) &= z_0(2k) \\
g_7(2k) &= s_0(2k) \\
g_8(2k) &= w_0(2k)
\end{aligned}$$

2.5.3 Representation with nonlinear model and linear constraints with sampling-time one

In this representation we have introduced the gate position as the first state as this is the variable we want as input to the system.

$x_0(k) = p_8(k)$
$x_0(k+1) = p_8(k) + \Delta p_8(k) = x_0(k) + u_0(k)$
$x_1(k+1) = h_8(k) = y8 + x_0(k) - g_8$
$x_2(k+1) = h_8(k-1) = x_1(k)$
$x_3(k+1) = h_8(k-2) = x_2(k)$
$x_4(k+1) = h_8(k-3) = x_3(k)$
$x_5(k+1) = h_8(k-4) = x_4(k)$
$x_6(k+1) = h_8(k-5) = x_5(k)$
$x_7(k+1) = p_9(k) + \Delta p_9(k) = x_7(k) + u_1(k)$
$x_8(k+1) = h_9(k) = y_9(k) + p_9(k) - g_9 = x_{13}(k) + x_7(k) - g_9$
$x_9(k+1) = h_9(k-1) = x_8(k)$
$x_{10}(k+1) = h_9(k-2) = x_9(k)$
$x_{11}(k+1) = h_9(k-3) = x_{10}(k)$
$x_{12}(k+1) = p_{10}(k) + u_2(k) = x_{12}(k) + u_2(k)$
$x_{13}(k+1) = y_9(k) + c_1 \cdot h_8^{3/2}(k-\tau_1) + c_2 \cdot h_9^{3/2}(k) = x_{13}(k) + c_1 \cdot x_6^{3/2}(k) + c_2 \cdot (x_{13}(k) + x_7(k) - g_9)^{3/2}$
$x_{14}(k+1) = y_{10}(k) + c_3 \cdot h_9^{3/2}(k-\tau_2) + c_4 \cdot h_{10}^{3/2}(k) = x_{14}(k) + c_3 \cdot x_{11}^{3/2}(k) + c_4 \cdot (x_{14}(k) + x_{12}(k) - g_{10})^{3/2}$

Simplified this becomes:

$x_0(k) = p_8(k)$
$x_0(k+1) = x_0(k) + u_0(k)$
$x_1(k+1) = y8 + x_0(k) - g_8$
$x_2(k+1) = x_1(k)$
$x_3(k+1) = x_2(k)$
$x_4(k+1) = x_3(k)$
$x_5(k+1) = x_4(k)$
$x_6(k+1) = x_5(k)$
$x_7(k+1) = x_7(k) + u_1(k)$
$x_8(k+1) = x_{13}(k) + x_7(k) - g_9$
$x_9(k+1) = x_8(k)$
$x_{10}(k+1) = x_9(k)$

$x_{11}(k+1) = x_{10}(k)$
$x_{12}(k+1) = x_{12}(k) + u_2(k)$
$x_{13}(k+1) = x_{13}(k) + c_1 \cdot x_6^{3/2}(k) + c_2 \cdot (x_{13}(k) + x_7(k) - g_9)^{3/2}$
$x_{14}(k+1) = x_{14}(k) + c_3 \cdot x_{11}^{3/2}(k) + c_4 \cdot (x_{14}(k) + x_{12}(k) - g_{10})^{3/2}$

The constraints are written on the form $g(k) \geq 0$

$$\begin{aligned}
g_0(k) &= x_0(k) \\
g_1(k) &= 2.005 - x_0(k) \\
g_2(k) &= x_7(k) \\
g_3(k) &= 1.474 - x_7(k) \\
g_4(k) &= x_{12}(k) \\
g_5(k) &= 1.487 - x_{12}(k)
\end{aligned}$$

2.5.4 Representation with nonlinear model and linear constraints with sampling-time two

$x_0(2k) = p_8(2k)$
$x_0(2k+2) = p_8(2k) + \Delta p_8(2k) = x_0(2k) + u_0(2k)$
$x_1(2k+2) = h_8(2k) = y8 + x_0(2k) - g_8$
$x_2(2k+2) = h_8(2k-2) = x_1(2k)$
$x_3(2k+2) = h_8(2k-4) = x_2(2k)$
$x_4(2k+2) = h_8(2k-6) = x_3(2k)$
$x_5(2k+2) = p_9(2k) + \Delta p_9(2k) = x_5(2k) + u_1(2k)$
$x_6(2k+2) = h_9(2k) = y_9(2k) + p_9(2k) - g_9 = x_{10}(2k) + x_5(2k) - g_9$
$x_7(2k+2) = h_9(2k-2) = x_6(2k)$
$x_8(2k+2) = h_9(2k-4) = x_7(2k)$
$x_9(2k+2) = p_{10}(2k) + u_2(2k) = x_9(2k) + u_2(2k)$
$x_{10}(2k+2) = y_9(2k) + c_1 \cdot h_8^{3/2}(2k-\tau_1) + c_2 \cdot h_9^{3/2}(2k) = x_{10}(2k) + c_1 \cdot x_6^{3/2}(2k) + c_2 \cdot (x_{10}(2k) + x_5(2k) - g_9)^{3/2}$
$x_{11}(2k+2) = y_{10}(2k) + c_3 \cdot h_9^{3/2}(2k-\tau_2) + c_4 \cdot h_{10}^{3/2}(2k) = x_{11}(2k) + c_3 \cdot x_8^{3/2}(2k) + c_4 \cdot (x_{11}(2k) + x_9(2k) - g_{10})^{3/2}$

Simplified this becomes:

$x_0(2k) = p_8(2k)$
$x_0(2k+2) = x_0(2k) + u_0(2k)$
$x_1(2k+2) = y8 + x_0(2k) - g_8$
$x_2(2k+2) = x_1(2k)$
$x_3(2k+2) = x_2(2k)$
$x_4(2k+2) = x_3(2k)$
$x_5(2k+2) = x_5(2k) + u_1(2k)$
$x_6(2k+2) = x_{10}(2k) + x_5(2k) - g_9$
$x_7(2k+2) = x_6(2k)$
$x_8(2k+2) = x_7(2k)$
$x_9(2k+2) = x_9(2k) + u_2(2k)$
$x_{10}(2k+2) = x_{10}(2k) + c_1 \cdot x_6^{3/2}(2k) + c_2 \cdot (x_{10}(2k) + x_5(2k) - g_9)^{3/2}$
$x_{11}(2k+2) = x_{11}(2k) + c_3 \cdot x_8^{3/2}(2k) + c_4 \cdot (x_{11}(2k) + x_9(2k) - g_{10})^{3/2}$

The constraints are written on the form $g(2k) \geq 0$

$$\begin{aligned}
g_0(2k) &= x_0(2k)\\
g_1(2k) &= 2.005 - x_0(2k)\\
g_2(2k) &= x_5(2k)\\
g_3(2k) &= 1.474 - x_5(2k)\\
g_4(2k) &= x_9(2k)\\
g_5(2k) &= 1.487 - x_9(2k)
\end{aligned}$$

2.6 Summary

The model we finally used in our optimization is the nonlinear model with linear constraints and sampling-time two minutes. Two minutes sampling-time reduces the states for the model and makes it easier for the optimization to handle the problem. We have also found out that simple and linear constraints are preferable as they give a faster solution from the optimization program.

3 Predictive Control

3.1 Model Predictive Control

Model Predictive Control (MPC) (see [10]) is a family of controllers in which there is a direct use of an explicit and separately identifiable model. Control design methods based on the MPC concept have found wide acceptance in industrial applications and have been studied by academia. MPC have flexible constraints capabilities and is applicable to non-linear systems and therefore would be a good method to solve our problem with.

The name "Model Predictive Control" arises from the manner in which the control law is computed. At the present time k the behaviour of the process over a time horizon p is considered as seen in figure 5 . Using a model the process response to changes in the manipulated variable is predicted. The moves of the manipulated variables are selected such that the predicted response has certain desirable characteristics. Only the first computational change in the manipulated variable is implemented. At the time $k + 1$ the computation is repeated with the time horizon moved by one time interval. The resulting control law is in general time varying and cannot be expressed in closed form.

MPC is an attractive tool when there is a need for control of rapid changes. There are though some areas where there is a need for more research. When working with non-linear systems there are uncertainties in the constraint handling that are unanswered. Also large-scale applications can be a problem as the algorithms supplied for constraint handling are not efficient enough. When there exists constraints on both the input and the output it is possible that the system is unable to find a feasible solution when a disturbance pushes the process outside the usual operating region.

3.1.1 Simple example of MPC

MPC has a close relationship with linear quadratic control, LQ-control. They are both based on a state-space representation of the model and the cost-function of the model. The difference between MPC and LQ-control is that in MPC you can take care of constraints. This is not possible in LQ-control.

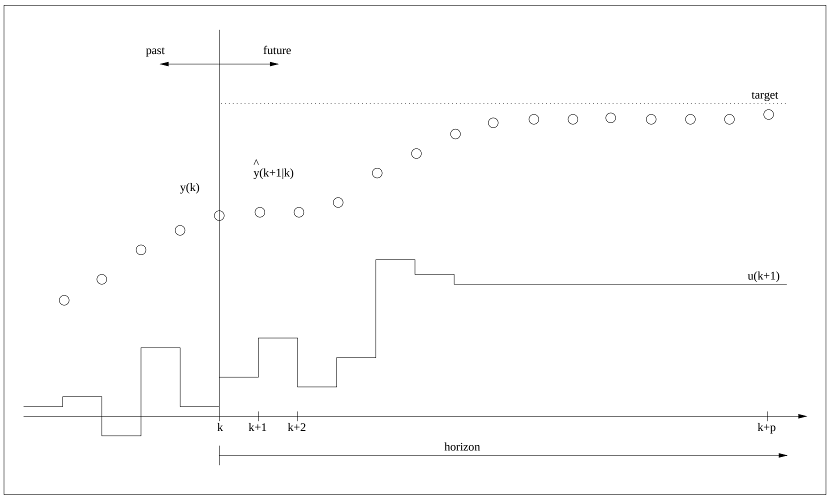

Figure 5: The “moving horizon” approach of Model Predictive Control

The model can be written on state-space form as:

$$x(k+1) = A(k)x(k) + B(k)\Delta u(k) \tag{6}$$

where $A(k)$ and $B(k)$ are time-varying matrices. The loss function for the system can be written as:

$$J = \frac{1}{2} \cdot \sum_{k=1}^{\infty} y^\top(k)Qy(k) + \sum_{k=1}^{\infty} \Delta u^\top(k)R\Delta u(k) \tag{7}$$

The design criteria we will use is a way of weighting the magnitude of the states and control signals. This is done by the weighting matrices Q and R, which are symmetric positive matrices. The optimal control problem is now defined to be finding the admissible control signal that minimizes the loss function of 7 when the model is described by the model of 6. This optimal control problem is solved for a certain horizon and only the first computational change is implemented. In the next time step the optimization is repeated with the time horizon moved by one time interval. This will result in a control law that is different in every time step and cannot be expressed in closed looped form as in the linear quadratic case.

3.2 Optimization strategy

To find the most optimal controller is the same as to find the control signal that minimizes the loss function. To learn more about predictive control and optimization we read [11], [16] and [17]. The loss function for the system would be:

$$J = \tfrac{1}{2} \cdot \textstyle\sum_{k=1}^{\infty} \hat{y}^\top(k)Q\hat{y}(k) + \sum_{k=1}^{\infty} \Delta u^\top(k)R\Delta u(k)$$

The function $\hat{y}(k)$ represents the error and the last head over gate. The error is the difference between the real water level and the desired water level. We tolerate a difference between these two in ± 5 centimetres (see figure 6). This interval we want to have because we do not want to move the gates if it is not necessary. A difference of five centimetres from the set point is too small to take into account. We want to minimize the last head over gate because the water that goes over this gate is often wasted. Another variable that we want to minimize is the gate movement because we do not want to use more power than necessary. Movements of the gates can also cause waves in the channel and our model does not take wave effects into account. Q and R are the weighting matrices described in the chapter called Model Predictive Control and are the design parameters in our controller.

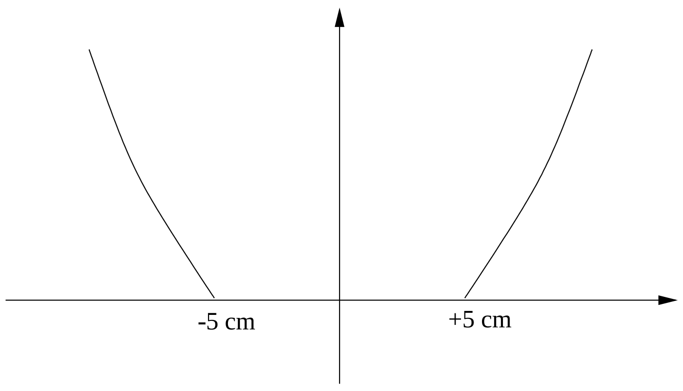

Figure 6: The Penalty function

This means that the function can be written:

$$\hat{y}_9(k) = \begin{cases} 0, & |y_9(k) - y_{9sp}| \leq 0.05 \\ (y_9(k) - y_{9sp} - 0.05), & y_9(k) - y_{9sp} \geq 0.05 \\ (-0.05 - y_9(k) + y_{9sp}), & y_9(k) - y_{9sp} \leq -0.05 \end{cases}$$

$$\hat{y}_{10}(k) = \begin{cases} 0, & |y_{10}(k) - y_{10sp}| \leq 0.05 \\ (y_{10}(k) - y_{10sp} - 0.05), & y_{10}(k) - y_{10sp} \geq 0.05 \\ (-0.05 - y_{10}(k) + y_{10sp}), & y_{10}(k) - y_{10sp} \leq -0.05 \end{cases}$$

The loss function for this problem can then be written as:

$$\begin{aligned} J \quad = \quad & \frac{1}{2}\sum_{k=1}^{n} \begin{bmatrix} \hat{y}_9(k) \\ \hat{y}_{10}(k) \\ h_{10}(k) \end{bmatrix}^{\top} \begin{bmatrix} q_1 & 0 & 0 \\ 0 & q_2 & 0 \\ 0 & 0 & q_3 \end{bmatrix} \begin{bmatrix} \hat{y}_9(k) \\ \hat{y}_{10}(k) \\ h_{10}(k) \end{bmatrix} + \\ & \frac{1}{2}\sum_{k=1}^{n} \begin{bmatrix} \Delta u_8(k) \\ \Delta u_9(k) \\ \Delta u_{10}(k) \end{bmatrix}^{\top} \begin{bmatrix} r_1 & 0 & 0 \\ 0 & r_2 & 0 \\ 0 & 0 & r_3 \end{bmatrix} \begin{bmatrix} \Delta u_8(k) \\ \Delta u_9(k) \\ \Delta u_{10}(k) \end{bmatrix} \end{aligned}$$

$$\Longrightarrow$$

$$\begin{aligned} J &= \frac{1}{2}\sum_{k=1}^{n}(\hat{y}_9^2(k)\cdot q_1 + \hat{y}_{10}^2(k)\cdot q_2 + h_{10}^2(k)\cdot q_3) + \\ &\quad \frac{1}{2}\sum_{k=1}^{n}(\Delta u_8^2(k)\cdot r_1 + \Delta u_9^2(k)\cdot r_2 + \Delta u_{10}^2(k)\cdot r_3) \end{aligned}$$

where $q_1,\ q_2,\ q_3 \geq 0$ and $r_1,\ r_2,\ r_3 > 0$. These are the tunable parameters in the controller. $\hat{y}_9(k)$ is the error of the water level in pool eight, $\hat{y}_{10}(k)$ is the error of the water level in pool nine and $h_{10}(k)$ is the head over gate ten that we want to minimize. $\Delta u_8(k)$ is the gate movement of gate eight, $\Delta u_9(k)$ is the gate movement of gate nine and $\Delta u_{10}(k)$ the gate movement for the last gate, gate number ten.

3.3 Summary

In this chapter we have been given an overview of Model Predictive Control (MPC), the optimization tool we have been using. A simple example has been given and our optimization strategy has been explained. In our problem we have a penalty function that allows the water levels to vary around set point with ± 5 centimetres. In this interval the error in the loss function is set to be zero. We tolerate this variation in the water levels because we do not want to move the gates more than necessary. The loss function is:

$$\begin{aligned} J &= \frac{1}{2}\sum_{k=1}^{n}(\hat{y}_9^2(k)\cdot q_1 + \hat{y}_{10}^2(k)\cdot q_2 + h_{10}^2(k)\cdot q_3) + \\ &\quad \frac{1}{2}\sum_{k=1}^{n}(\Delta u_8^2(k)\cdot r_1 + \Delta u_9^2(k)\cdot r_2 + \Delta u_{10}^2(k)\cdot r_3) \end{aligned}$$

where $q_1,\ q_2,\ q_3 \geq 0$ and $r_1,\ r_2,\ r_3 > 0$. These are the tunable parameters in the controller. $\hat{y}_9(k)$ is the error of the water level in pool eight, $\hat{y}_{10}(k)$ is the error of the water level in pool nine and $h_{10}(k)$ is the head over gate ten that we want to minimize. $\Delta u_8(k)$ is the gate movement of gate eight, $\Delta u_9(k)$ is the gate movement of gate nine and $\Delta u_{10}(k)$ the gate movement for the last gate, gate number ten.

4 Optimization

4.1 MATLAB programming

To minimize the loss function some different methods have been tried out. The problem we want to implement is rather large and we need some kind of computational tool to be able to minimize it. The first thing we tried was the Optimization Toolbox in MATLAB, see [13]:

- MATLAB - constr

  The first method we tried in MATLAB was constr. constr finds the constrained minimum of a function of several variables. This function has now been replaced in MATLAB by fmincon.

  This method we used to get started, to get familiar with our problem and to learn more about the Optimization Toolbox in MATLAB. constr does not handle nonlinear behaviour so we started with a linear model of the problem. constr worked but was quite slow. As it was replaced by another function we thought that method would be better.

- MATLAB - fmincon

  fmincon makes the same thing as constr but is more efficient and can handle nonlinear behaviour. The gradient of the function was provided (see appendix E), as were the derivatives of the constraints (see appendix F). The calculations are shown in appendix A. In fmincon you can choose to run with large-scale algorithm or the medium-scale algorithm depending on the size of your problem. Certain conditions must be met to be able to use the large-scale algorithm; the gradient must be provided, only upper and lower bounds may be specified, or only linear equality constraints must exist. Unfortunately our constraints are inequalities and this means that we had to choose to use the medium-scale method instead. As our problem is of a large scale it could be hard for fmincon to give a feasible solution in short time, as we demand.

  fmincon was tried out with the medium-scale method. The iteration was slow, an optimization time of less than one minute was desired and our optimization with fmincon took over 30 minutes with a horizon of 30 minutes.

  This behaviour could depend on many things. Of course it would be better to be able to use the large-scale algorithm. Another reason for being slow could be our starting-values of the control-vector. If they are too far away from the real values it could take a long time before fmincon finds a solution. If we could find starting-values closer to the solution it would probably go faster.

- MATLAB - lsqnonlin

  The function lsqnonlin solves a nonlinear least-squares problem, this means our problem without the constraints. Outputs from lsqnonlin would probably give better starting values of the control-vector (see appendix G). With these values we got fmincon to take only 12 minutes for the optimization routine to evaluate its value with a time horizon of 50 minutes.

4.1.1 Conclusion

The result with MATLAB was not good enough and our conclusion was that MATLAB was too slow for our problem. We had too find another optimization tool to solve our large-scale problem with.

4.2 Implementation in C

To be able to make a faster evaluation than the functions in MATLAB could do we considered to find a program made in Fortran, C++ or C. These languages are the most common languages to write optimization code in. After some searching on the WWW and at the university one program was found that might be our solution. A student named James Wettenhall had made a program [12] which could take care of constrained nonlinear optimal control problems, exactly the kind of problem we had.

4.2.1 Description of the program

The program [12] is implemented in C except for one part which is in Fortran. It uses a linear search method to find the optimal solution to a constrained problem depending of several variables. A primal-dual interior point algorithm has been used together with a differential dynamic programming algorithm to be able to solve this problem. The primal-dual interior point algorithm we used is based on approximate complementary and is referred to as ETTZ. The differential dynamic programming (DDP) is a technique which approximates both the cost function and the transition equations by quadratic functions of a differential perturbation away from the current estimate of the solution as determined by the previous iteration.

We had to formulate our own problem in the program. To do that we had to write some functions. The functions were:

cost evaluates the cost function

cost_deriv the first derivatives of the cost function

cost_2ndderiv the second derivatives of the cost function

transition_f the transition function for the model

transition_deriv the first derivatives of the transitions

transition_2ndderiv the second derivatives of the transitions

ineqconstraint the constraints of the gate positions and the head over gates (written on form greater or equal to zero)

ineqconstraint_deriv the first derivatives of the constraints

ineqconstraint_2ndderiv the second derivatives of the constraints

As we already had our problem formulated it was not too hard to reformulate it. The main difference was that we now had to take the derivatives and the second derivatives with respect to the state parameters and the control parameters just in one time step, i.e we could see them as independent variables. The calculations are made in appendix B, C and D. The program itself calculates the derivatives in all other necessary time steps. We also had to set some constants in the program. These constants were:

NumStages the number of time stages

dim_u the number of components in the control vector

dim_x the number of components in the state vector

ineqs_per_stage the number of inequality constraints per time stage

uhi an array containing the upper bounds for the control

ulo an array containing the lower bounds for the control

xhi an array containing the upper bounds for the state

xlo an array containing the lower bounds for the state

x_initial an array containing the initial conditions for the state

initial_guess_u an array containing an initial guess for the optimal control

tol an error tolerance used by the optimization routine

MAX_ITER maximum number of iterations of the optimization routine

initial_mu initial value of the barrier parameter

iterations_between_decreasing_mu steps between the barrier parameter mu

mu_reduction_strategy if zero the strategy ETTZ is used, if one GOW is used

slow_version_of_ETTZ how fast the barrier parameter is decreased

4.2.2 Up and running

As the program is not commercial and has not been tested for so many different cases we could not be sure how well it should work for our problem. The program did not work correctly in the beginning. There were some small bugs in the program that had not been discovered before which made the program not work for our large-scale problem. It took a while for us to find these bugs and we had a lot of contact with the author of the program, James Wettenhall. It was also hard to understand what actually was happening sometimes because the program is rather large and we did not have any user's manual. Finally we got the program running and could get results. But the program crashed when we used a larger horizon than 20 minutes. The problem was hard to find. We knew that the more complicated our constraints were the harder it was for the program to find a feasible solution. We decided to rewrite our problem. We rewrote our constraints to be linear and transmitted the nonlinearity to the transition-variables.

Now the time horizon was exceeded to be 50, but the norm was too large so this was not good enough either. The norm is a parameter in the program that tells us how close we are to the optimal solution. It is also possible to sample our problem with a sampling-time of two minutes instead of one. The program seems to have problems with large scaled problems so we rewrote our problem again, this time to a sample time with two minutes. This reduces our transitions and gives a less number of evaluations. The program handled this problem formulation even better. But we still had one problem. In some cases we got the answer NaN (Not a Number). It took a while for us to find what caused the problem, but finally we did. In the program a division by zero was made. We did not know what to do about this problem, and neither did James, the author of the program. We tried to increase the number of iterations between decreasing mu to get around this problem and fortunately it worked. The norm was though quite large for a large horizon and we did not know if it was large because the program gave a bad solution or if the program was close to the constraints or bounds. If it was the case that we were close to the constraints the MPC would still work and give good results, if it was the first case we would have to change something to make the program better. After a while we understood that it was the program which gave a bad solution and we determined to decrease the horizon to find a smaller norm. We could now finally let the program calculate the optimal solution for 40 time steps which in this case means 80 minutes with a very low norm and we decided to be satisfied with that. The optimization is also very fast which is desired.

4.3 Simulation model

To be able to try out our control design we have a simulation model [E. Weyer, E. Lim]. This model is made in Simulink, the simulation environment in MATLAB. The simulation model is different from our model. In the Simulink model wave effects are taken into account and this is not done in our model.

Factors involved in the control of the irrigation channel is, for example, available water in reservoirs/rivers, required water for crops and demanded patterns. These factors makes it necessary to have a model for long time planning (see [9]). The model consists of gate eight to twelve of the Haughton Main Channel (HMC). Data is taken from system identification experiments on the channel on the 29th of September 1999 [8] and the model itself is made with system identification tools. The model is considered to be a good approximation of the real channel.

We modified this model to suit our problem with help from [14]. The model included a feedback controller which was taken away in our version as we were going to build a predictive controller ourselves. For our problem we just need two pools, pool eight and pool nine so the model was reduced to consist of only two pools. To be able to run the model for just one time step, we had to replace the time-delay blocks with transition functions. When replacing the time delays with transition functions we can get different initial values for the time delays which is not possible with a time-delay block. We used the following state-space representations for the time-delays:

Time-delay six:

$$x(2k+2) = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} x(2k) + \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} u(2k)$$

$$y(2k) = \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix} x(2k)$$

$$\Longrightarrow y(2k) = h^{3/2}(2k-6)$$

Time-delay four:

$$x(2k+2) = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} x(2k) + \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} u(2k)$$

$$y(2k) = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} x(2k)$$

$$\Longrightarrow y(2k) = h^{3/2}(2k-4)$$

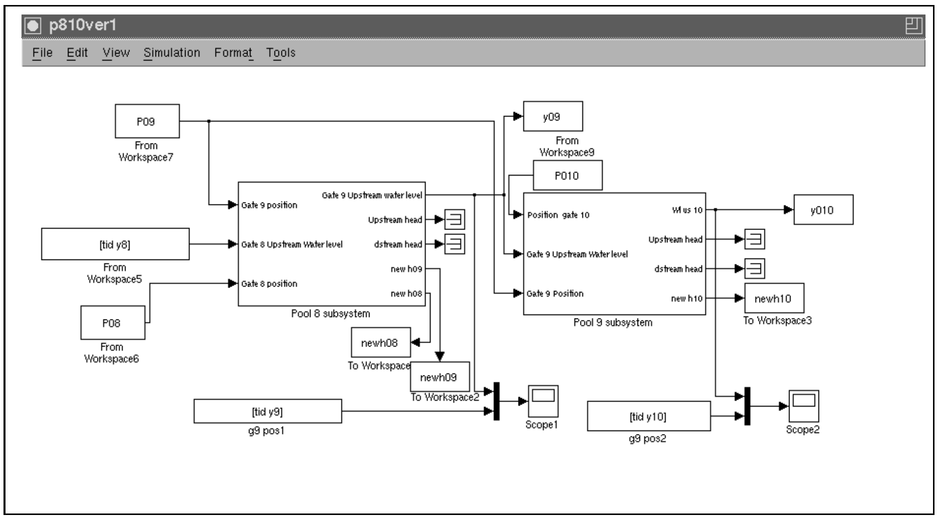

Figure 7: The Simulink model

In the model we also changed the integrators to be transfer functions to avoid setting initial values. Inputs to the model are the three gate positions, *P08*, *P09* and *P010*, which are collected from workspace in MATLAB. Outputs are the upstream water levels of gate nine and ten, *y09* and *y010*. We assume that the water level in pool seven is constant, as this pool is not included in our model.

4.4 Interface with Simulink model

To be able to do MPC, which means many iterations of the simulation in not too long time we had to make an interface between our C-program and MATLAB—Simulink. After some reading we decided to call MATLAB from our C-program, run one simulation in Simulink from MATLAB and then return the results to the C-program. We did this by using the MATLAB Engine, see [15]. The MATLAB Engine Library is a set of routines that allows you to call MATLAB from your own C or Fortran programs, thereby employing MATLAB as a computation engine. The MATLAB engine operates by running in the background as a separate process from your own program. On UNIX, the operative system we use, the engine library communicates with the MATLAB engine using pipes. To do the simulation and optimization efficient we would like to do the following in one run of our C-program:

- To get the right starting values for the Simulink model we want to run the simulation for a while until we get steady state. The gate positions are here constant values.
- Run one optimization in the C-program with one initial guess on the states and one on the control signal.
- Deliver the result from the optimization of the gate positions to MATLAB, i.e. the values at x[10] and x[11] in every time step.
- Run one simulation in Simulink for two minutes with the gate positions from the optimization.
- Deliver the result of the water levels in pool nine and ten from the simulation in time step two back to the optimization routine.
- Run the optimization again, this time with the water levels in pool nine and ten from the simulation as initial values at x[10] and x[11]. Initial values on the rest of the states are set to be the optimal solution from the previous run of the optimization shifted one step i.e we start the simulation with the result from time step two from the previous run. The initial guess on the control signal is the optimal control solution from the previous run.
- Repeat all the points, except point one and two, until we are in the last time step.

To be able to do this we wrote two for-loops in our C-program. First of all we start MATLAB from our C-program. To get the Simulink model to the right initial values we send constant values for P08, P09 and P010 to the model, and run the simulation in the first for-loop ten times. The constant values for P08, P09 and P010 were tuned out to be values so the water level is the desired in steady state. The model is ready to start run the actual simulations. The new for-loop is now entered. This runs the simulation in Simulink and then the optimization, passing the needed variables between the two programs as many times as wanted. In every run of the simulation, needed results are saved.

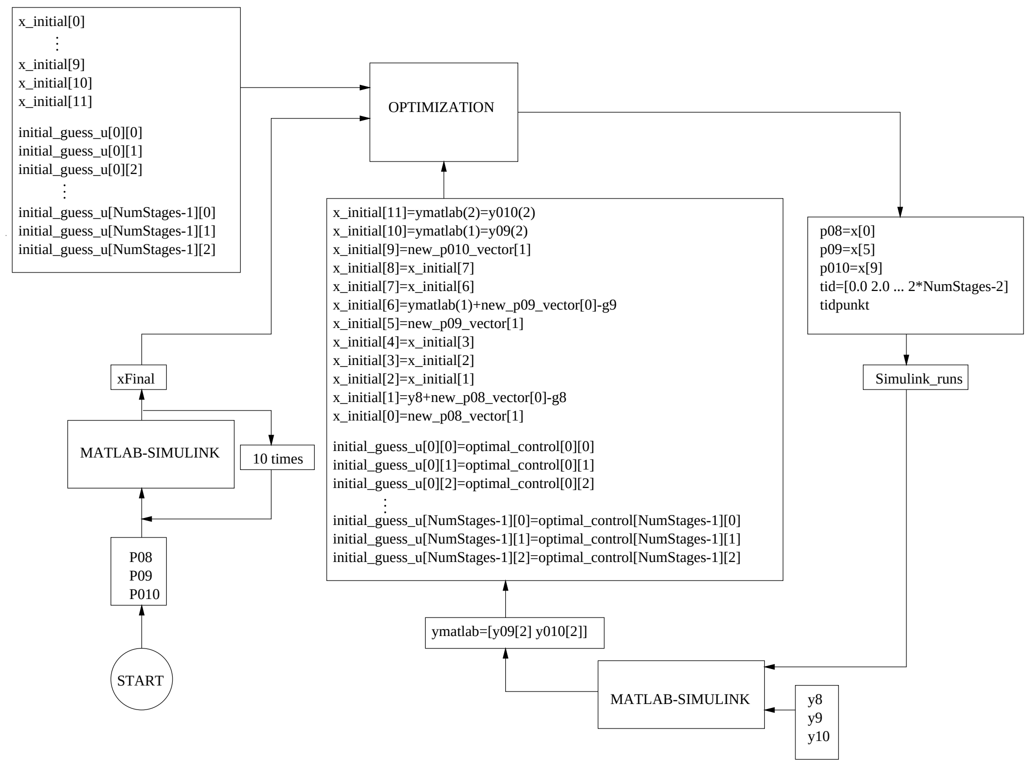
x_initial[0]
⋮
x_initial[9]
x_initial[10]
x_initial[11]
initial_guess_u[0][0]
initial_guess_u[0][1]
initial_guess_u[0][2]
⋮
initial_guess_u[NumStages-1][0]
initial_guess_u[NumStages-1][1]
initial_guess_u[NumStages-1][2]
OPTIMIZATION
x_initial[11]=ymatlab(2)=y010(2)
x_initial[10]=ymatlab(1)=y09(2)
x_initial[9]=new_p010_vector[1]
x_initial[8]=x_initial[7]
x_initial[7]=x_initial[6]
x_initial[6]=ymatlab(1)+new_p09_vector[0]-g9
x_initial[5]=new_p09_vector[1]
x_initial[4]=x_initial[3]
x_initial[3]=x_initial[2]
x_initial[2]=x_initial[1]
x_initial[1]=y8+new_p08_vector[0]-g8
x_initial[0]=new_p08_vector[1]
initial_guess_u[0][0]=optimal_control[0][0]
initial_guess_u[0][1]=optimal_control[0][1]
initial_guess_u[0][2]=optimal_control[0][2]
⋮
initial_guess_u[NumStages-1][0]=optimal_control[NumStages-1][0]
initial_guess_u[NumStages-1][1]=optimal_control[NumStages-1][1]
initial_guess_u[NumStages-1][2]=optimal_control[NumStages-1][2]
p08=x[0]
p09=x[5]
p010=x[9]
tid=[0.0 2.0 ... 2*NumStages-2]
tidpunkt
Simulink_runs
xFinal
MATLAB-SIMULINK
10 times
P08
P09
P010
START
ymatlab=[y09[2] y010[2]]
MATLAB-SIMULINK
y8
y9
y10

Figure 8: Running the program

4.5 Summary

We made the conclusion that the optimization toolbox in MATLAB is not suitable for our large problem as it is too slow. The C-program that we used was able to do the optimizations much faster. The communication between MATLAB and the C-program worked well and made the optimization easy and fast. Our time horizon is not as large as we desired but the program could not handle a larger one.

The finally problem formulation for the optimization is shown below.

Representation with nonlinear model and linear constraints with sampling-time two:

$$
\begin{aligned}
&x_0(2k) = p_8(2k)\\
&x_0(2k+2) = x_0(2k) + u_0(2k)\\
&x_1(2k+2) = y8 + x_0(2k) - g_8\\
&x_2(2k+2) = x_1(2k)\\
&x_3(2k+2) = x_2(2k)\\
&x_4(2k+2) = x_3(2k)\\
&x_5(2k+2) = x_5(2k) + u_1(2k)\\
&x_6(2k+2) = x_{10}(2k) + x_5(2k) - g_9\\
&x_7(2k+2) = x_6(2k)\\
&x_8(2k+2) = x_7(2k)\\
&x_9(2k+2) = x_9(2k) + u_2(2k)
\end{aligned}
$$

$x_{10}(2k+2) = x_{10}(2k) + c_1 \cdot x_6^{3/2}(2k) + c_2 \cdot (x_{10}(2k) + x_5(2k) - g_9)^{3/2}$
$x_{11}(2k+2) = x_{11}(2k) + c_3 \cdot x_8^{3/2}(2k) + c_4 \cdot (x_{11}(2k) + x_9(2k) - g_{10})^{3/2}$

The loss-function for the system:

$$\begin{aligned} J &= \frac{1}{2}\sum_{k=1}^{n}(\hat{y}_9^2(k)\cdot q_1 + \hat{y}_{10}^2(k)\cdot q_2 + h_{10}^2(k)\cdot q_3) + \\ &\quad \frac{1}{2}\sum_{k=1}^{n}(\Delta u_8^2(k)\cdot r_1 + \Delta u_9^2(k)\cdot r_2 + \Delta u_{10}^2(k)\cdot r_3) \end{aligned}$$

where $q_1,\ q_2,\ q_3 \geq 0$ and $r_1,\ r_2,\ r_3 > 0$. These are the tunable parameters in the controller. $\hat{y}_9(k)$ is the error of the water level in pool eight, $\hat{y}_{10}(k)$ is the error of the water level in pool nine and $h_{10}(k)$ is the head over gate ten that we want to minimize. $\Delta u_8(k)$ is the gate movement of gate eight, $\Delta u_9(k)$ is the gate movement of gate nine and $\Delta u_{10}(k)$ the gate movement for the last gate, gate number ten.

The constraints of the gate positions are shown below in meter:

$$0 \leq p_{08} \leq 2.005$$
$$0 \leq p_{09} \leq 1.474$$
$$0 \leq p_{010} \leq 1.487$$

Other limitations (in meter):

$$-0.4 < \Delta p_i < 0.4$$
$$h_i > 0$$

5 The Performance of the Predictive Controller

5.1 Tuning of parameters

It is possible to tune different parameters in the model to get better results of the MPC. The changeable parameters are the weighting parameters, Q and R, in the cost function. Q determines how large influence the water levels and the last head over gate will have in the cost calculations. R determines how large weight the different gate movements should have and they can have different values dependent on how important they are. For example, if we set R to be a large value, it will “cost” a lot to move the variables influenced by R and they will therefore not move very much in comparison with the other values in the cost function. Q and R are symmetric positive semidefinite matrices and are in our problem of size 3×3, which here means they consists of three changeable variables each.

The first water level in our model, y_8, is changeable and can be tuned. It is limited by the adjustment factor, which is 27.693 mAHD, and should not be larger than this. We want to have the possibility to close off the flow into pool eight. After some experiments we got y_8 to be 27.60 mAHD. This value seemed to satisfy both the optimization routine and the simulation model.

Before an optimization run is started, it is important that the simulation model is in steady-state to take away incorrect values from the time-delays in the simulation model. Tests have also shown that a starting-point close to the set points gives better values of the water levels and takes away a lot of initial disturbances. The set points for the water levels are 26.50 mAHD for y_8 and 23.85 mAHD for y_9.

When we started to do MPC we had the horizon in the optimization to be 40 time steps, this means 80 minutes and run the MPC-horizon for 800 minutes. With a horizon of 40 time steps we found a small norm and also a good solution where we finally could see reactions on load disturbances and other tests. Figure 9 shows the upstream water levels in pool eight and nine. Q is here set to I and R to 400·I (I is here the identity matrix). The water levels have some small initial disturbances but goes quite fast to stable values. Figure 10 shows the gate positions of gate eight, nine and ten when the water levels goes to a stable value. The gate-positions should move as little as possible and also have upper and lower constraints. The most they can move between every interval is 0.4 meter. Here we can see that the gates are closing (goes to zero) just as we want. In figure 11 we can see the corresponding head over gate. The head over gate should be as small as possible over gate ten to minimize the water losses. Notice that we have different scales in different figures.

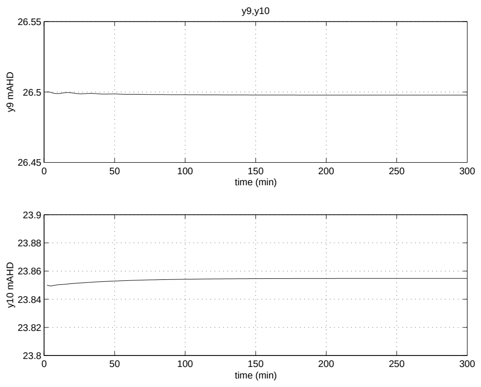

Figure 9: The water level is stable

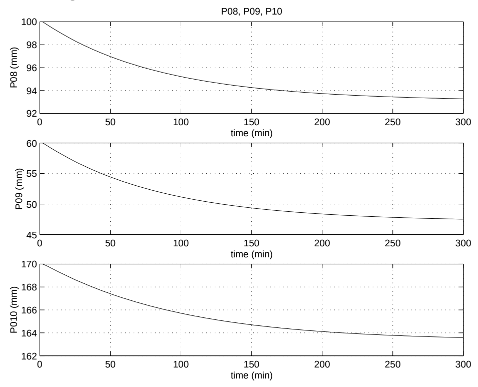

Figure 10: The gates are closing

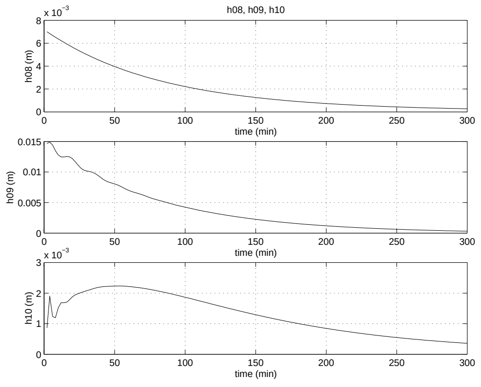

Figure 11: The head over gates are getting smaller

5.2 Performance tests

To be able to see the performance of the predictive controller we do a stability analysis to see how the controller will react on load disturbances (offtakes) and a start above and under its usual operating region (too much or too little water in the pool). These are cases that can happen in the reality and it is therefore important to do these tests on the model.

As the horizon is only 80 minutes, we had to make sure that the results made sense. A test was made, and this is shown in figure 12 and 13. Here one optimization has been made followed by another one with initial water levels and initial $\Delta p : s$ from the first optimization. If the second optimization gives an answer that not correspond and follow the first one, then the optimization with a horizon of 80 minutes is probably not good enough. In this case we can see that the result is quite good, the simulations seems to smooth out. Our conclusion from this test is that a horizon of 80 minutes is probably good enough. Notice that the scale is different in different figures.

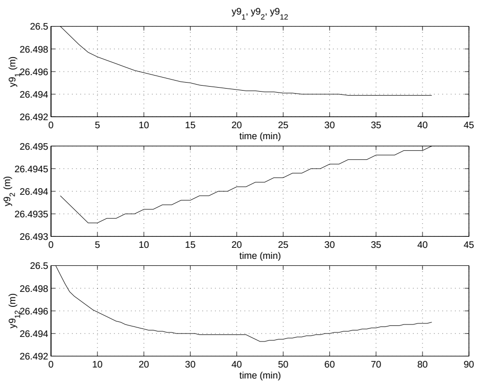

Figure 12: Two optimizations of y_9 after each other, then put together in the last diagram

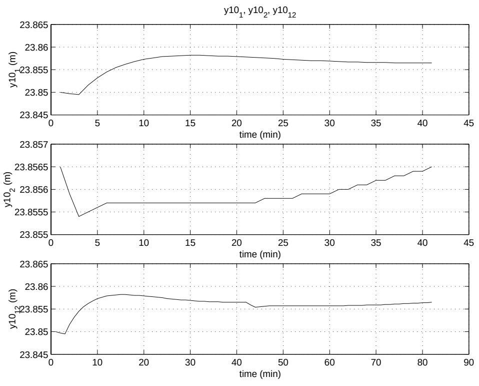

Figure 13: Two optimizations of y_{10} after each other, then put together in the last diagram

5.2.1 Disturbance in pool eight

When the parameters were tuned to be the best possible it was time to see how our model took care of disturbances. We implemented an offtake as a change in the water level in our Simulink model in form of a step change subtracted from the water level. The step is turned on in time step 100 minutes and turned off in time step 170 minutes. The calculations for the offtake can be seen in chapter 2.2.1.

Figure 14 has got a disturbance in form of an offtake of 530 mega liter on pool eight. Different values of R were tried out, and the best results were given with R set to be 500·I, see figure 14. The water level in pool eight goes fast inside the operating region, as desired, after the load diturbance is turned off after 170 minutes. The water level in pool nine goes first to a too high value after the load disturbance is turned off. This behavior can be explained by the time constant in the pool, it takes a while before the water level in pool nine is stable. The gate position can be seen in figure 15 and the head over gates in figure 16 and h_{10} is trying to be as small as possible.

Now let the optimization program know that we have a disturbance in pool eight. We introduced this known disturbance in the transition functions in the C-program. Head over gate ten is minimized around 0.1 meter. This means that we want to minimize around a constant flow over the last gate. The loss function now becomes:

$$\begin{aligned} J &= \frac{1}{2}\sum_{k=1}^{n}(\hat{y}_9^2(k)\cdot q_1 + \hat{y}_{10}^2(k)\cdot q_2 + (h_{10}(k)-0.1)^2\cdot q_3) + \\ &\quad \frac{1}{2}\sum_{k=1}^{n}(\Delta u_8^2(k)\cdot r_1 + \Delta u_9^2(k)\cdot r_2 + \Delta u_{10}^2(k)\cdot r_3) \end{aligned}$$

If the behaviour is better than without a known disturbance can be discussed (see figure 17, 18 and 19). Gate eight and nine start to open to prepare for the disturbance in pool eight, with the result that the water level in pool nine starts to increase. Pool nine has for that reason a water level above the error tolerance before the disturbance but goes to a final value inside the tolerance. An expected behaviour of the water level in pool eight would be an increased water level before the offtake. This does not happen, and an explanation can be that pool eight is reacting slower than pool nine and it takes longer time to increase the water level in pool eight. Unfortunately, we get a too large water level in pool eight but the level never goes that low as in the offtake without a known disturbance.

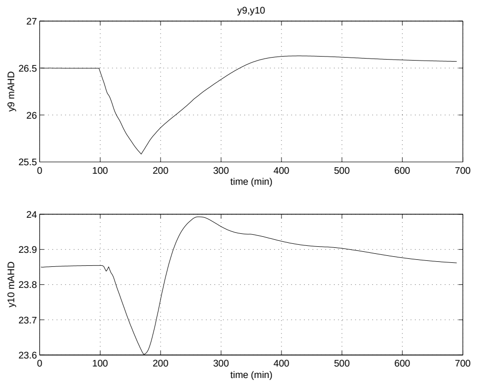

Figure 14: The water levels with a disturbance in pool eight, and a weight R of 500

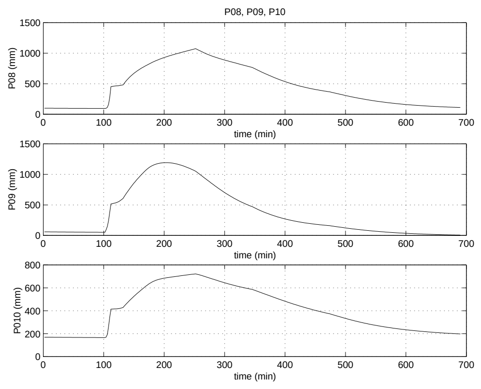

Figure 15: The gate positions with a disturbance in pool eight, and a weight R of 500

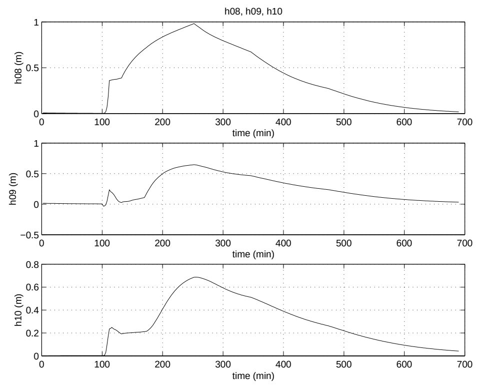

Figure 16: The head over gates with a disturbance in pool eight, and a weight R of 500

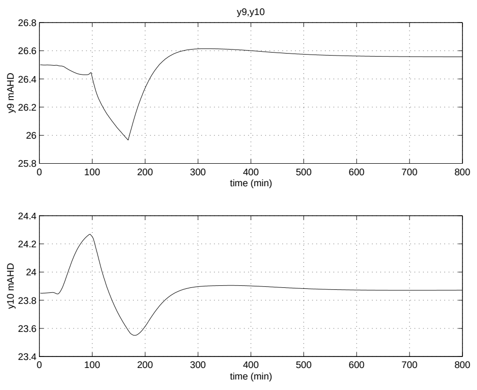

Figure 17: The water levels with a known disturbance in pool eight

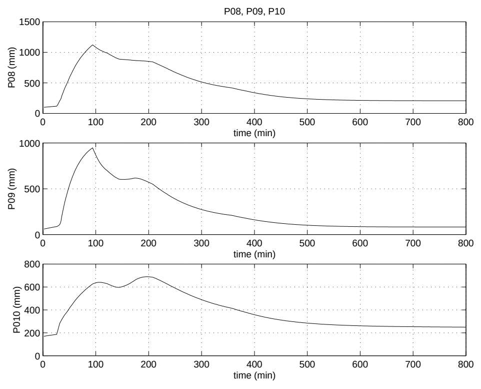

Figure 18: The gate positions with a known disturbance in pool eight

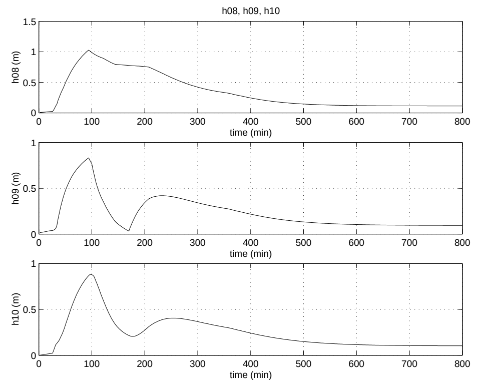

Figure 19: The head over gates with a known disturbance in pool eight

5.2.2 Disturbance in pool nine

A disturbance is now introduced in pool nine as a step change that starts in time step 50 and stops in time step 170 minutes. The calculations for the offtake can be seen in chapter 2.2.2.

A load of 300 mega liters is taken of the pool nine. The responds is good (see figure 20), the water level in pool eight is just above the operating region and the water level in pool nine is just below. The behaviour of the gates and the head over gates can be seen in figure 21 and 22.

A known disturbance is now introduced in pool nine. The minimization of h_{10} is here made around 0.1 meter. This means that we want to minimize around a constant flow over the last gate. The result is shown in figure 23, 24 and 25. The water levels have a better behaviour than without a known disturbance. As we can see the water level in pool nine does not go as low in this case compared to an unknown disturbance. We also notice that this water level increases from the beginning before the disturbance to prepare for the offtake. Also the water level in pool eight starts to increase before the disturbance is put into action. When we look at the head over gate ten we can see that it strives to go to 0.1 meter as desired.

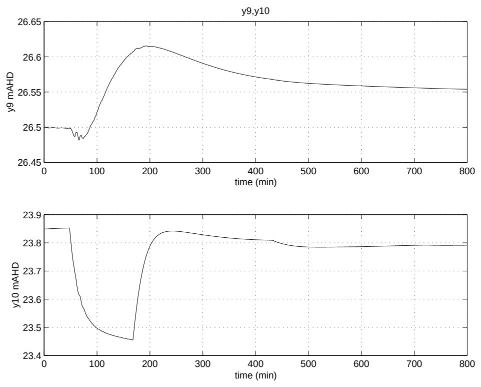

Figure 20: The water levels with a disturbance in pool nine

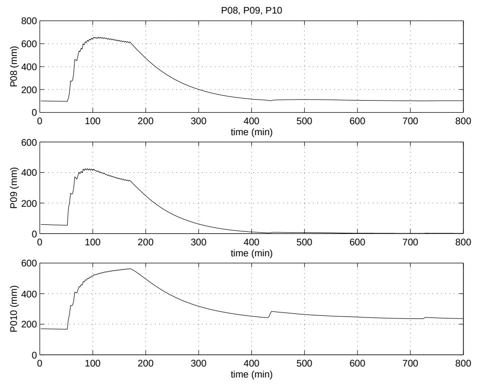

Figure 21: The gates are opening to let more water into the pools

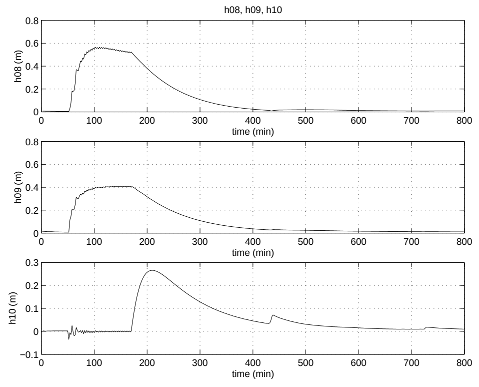

Figure 22: The head over gates are increasing because of the disturbance

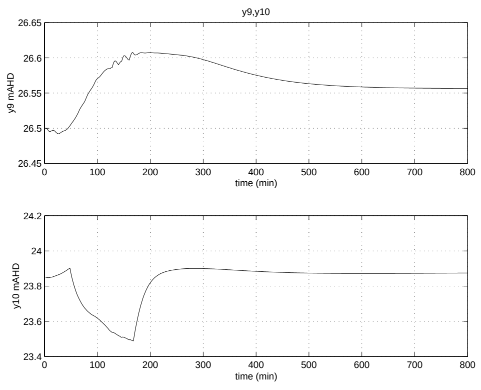

Figure 23: The water levels with a known disturbance in pool nine

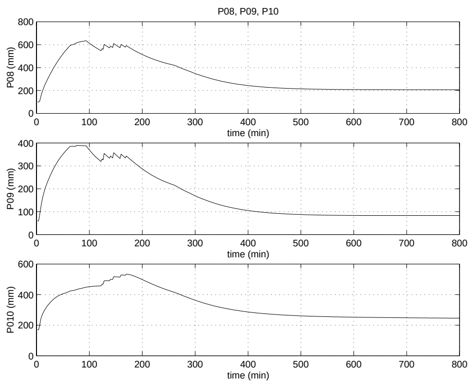

Figure 24: The gate positions with a known disturbance in pool nine

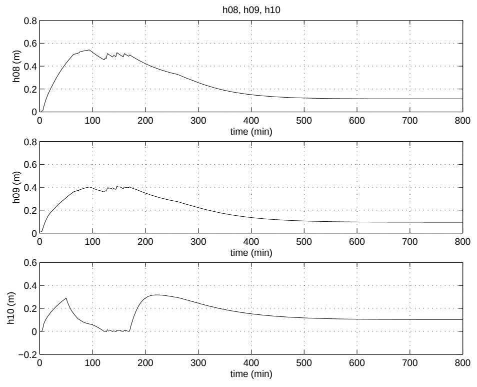

Figure 25: The head over gates with a known disturbance in pool nine

5.2.3 Initial water level in pool eight below error tolerance

First we run a simulation when the water level in pool eight is around 0.1 meter below the error tolerance. The values of p are the values used to get the Simulink model in steady state.

$p_{08} = 10$ mm
$p_{09} = 140$ mm
$p_{010} = 170$ mm

Values used in the simulations (m):
$x\_initial[0] = 0.1$
$x\_initial[1] = 0.0000000001$
$\vdots$
$x\_initial[4] = 0.0000000001$
$x\_initial[5] = 0.15$
$x\_initial[6] = 0.0000000001$
$\vdots$
$x\_initial[8] = 0.0000000001$
$x\_initial[9] = 0.17$
$x\_initial[10] = y_9 = 26.405$ mAHD
$x\_initial[11] = y_{10} = 23.851$ mAHD

$x\_initial[0]$	initial guess for gate position for gate eight
$x\_initial[1] \ldots x\_initial[4]$	initial guess for head over gate eight
$x\_initial[5]$	initial guess for gate position for gate nine
$x\_initial[6] \ldots x\_initial[8]$	initial guess for head over gate nine
$x\_initial[9]$	initial guess for gate position for gate ten
$x\_initial[10]$	initial guess for water level in pool eight
$x\_initial[11]$	initial guess for water level in pool nine
Comment	We want the head over gates to be as small as possible without getting to close to the condition that they should be greater than zero. Therefore we let the initial values of the head over gates to be very close to zero.

Figure 26: Explanation of the *x_initial* values

run	r_1	r_2	r_3	q_1	q_2	q_3	comments
1	400	400	400	1	1	0.1	put in a condition that h_{10} should minimizes around 0.1 m
2	400	400	400	1	1	0.0	the condition on h_{10} does not have any effect because q_3 is zero.

It seems like the optimization program has problems when we are close to the error tolerance. We have to increase iterations_between_decreasing_mu to three to get it to run at all. Even after that the solution is not perfect but the norm is low which is good. Pool eight is quite long, 1.6 km, and the time delay is also long. These two factors make the reactions in pool eight slow. The expected response will just be to open gate eight to let more water into pool eight until the water level is inside the error tolerance. Why this does not happen is hard to explain. When looking at the water levels in figure 27 we can see that the controller has some problems in the beginning to rise the water level in pool eight, and it takes a while before it actually starts increasing the level. When it finally does that it increase the level a bit too much. But when the water level is above the error tolerance we can see that it actually starts decrease again towards the set point. We can also see that the water level in pool nine starts increasing. This depends on that we have a minimization around 0.1 meter for h_{10} and gate ten does not want to open more because of that. When looking at the head over gates (figure 29) we can see that h_{10} starts increase to get rid of water in pool nine while this water level is to high. When the water level in pool eight starts to go into the tolerance we can see that h_{10} starts to close because it wants to be around 0.1 meter and this is a good behaviour. It would be good if the controller was a bit faster and if the water level in pool eight does not increase that much, but we can see that the controller reacts and in a correct way.

In run two q_3 is put to be zero. This results in that the minimization does not take any consideration about minimizing h_{10}. As we can see in figure 30 the water level in pool eight has a better final value than for run one, but unfortunately the water level in pool nine is worse. The water level in pool eight is going to a value inside the error-tolerance. Gate p_{10}, see figure 31, is slowly continuing to open and will probably make the water level in pool nine a bit lower. We also notice that the head over gates are quite large in figure 32. Of course it can be discussed which is the best solution, but if it is important that h_{10} is quite small run one is the best one.

The next test to do is to start the simulation when the water level in pool eight is around 0.01 meter below the error tolerance. The following values are used:

$p_{08} = 10$ mm
$p_{09} = 100$ mm
$p_{010} = 200$ mm

Values used in the simulations (m):

$x\_initial[0] = 0.1$
$x\_initial[1] = 0.0000000001$
$\vdots$
$x\_initial[4] = 0.0000000001$
$x\_initial[5] = 0.1$
$x\_initial[6] = 0.0000000001$
$\vdots$
$x\_initial[8] = 0.0000000001$
$x\_initial[9] = 0.2$
$x\_initial[10] = y_9 = 26.447$ mAHD
$x\_initial[11] = y_{10} = 23.82$ mAHD

These values are explained in figure 26.

run	r_1	r_2	r_3	q_1	q_2	q_3	comments
3	400	400	400	1	1	0.1	put in a condition that h_{10} should minimizes around 0.1 m
4	400	400	400	1	1	0.0	the optimization program did not handle this minimization

As we can see in figure 33 the controller has problems to control the water levels even when we start 0.01 meter below the error tolerance. The water level in pool eight starts to decrease for about 100 minutes before it changes to increase instead. This may depend on that h_{10} strives to go to 0.1 m. The minimization is a balance between the different factors that we want to minimize and sometimes they compete against each other. As we can see in figure 35 h_{10} is around 0.2 meter when it is stable. The water level in pool nine is a bit too high, about 5 millimetre. Why does the controller not just open gate ten and let some water out? Again this can depend on that the minimization of the different factors are competing against each other. We want the error of the water level to be minimized but we also want the head over gate ten to be minimized. If we let more water out, i.e. open gate ten this will result in a higher head over gate as well. When looking at the results from the optimization we also have to take into consideration that the optimization is just running in a time-horizon of 40 steps i.e 80 minutes and this may be too low to get an optimal solution.

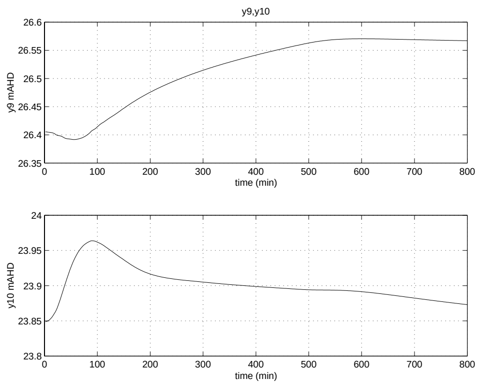

Figure 27: The water levels for run 1, initial water level about 1 dm below error tolerance

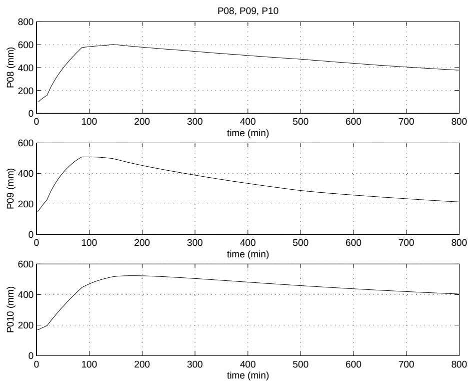

Figure 28: The gate positions for run 1

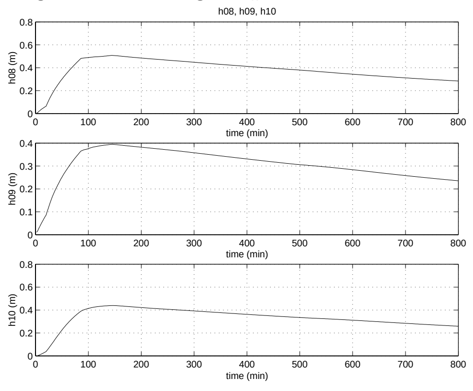

Figure 29: The head over gates for run 1

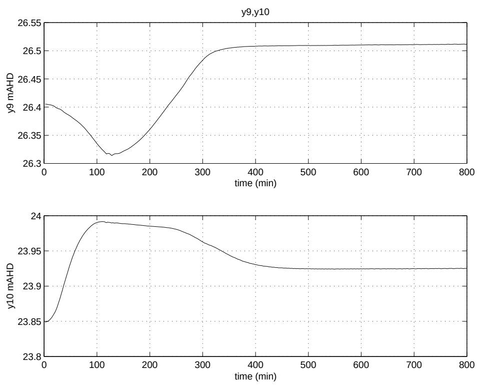

Figure 30: The water levels for run 2, initial water level about 1 dm below set point

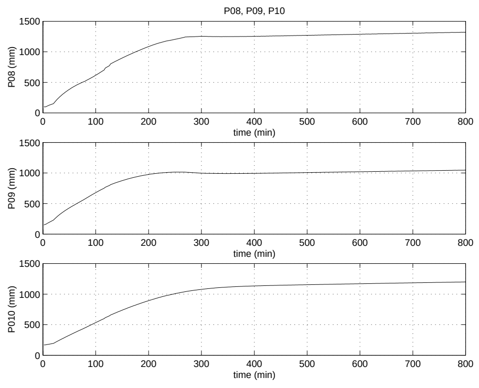

Figure 31: The gate positions for run 2

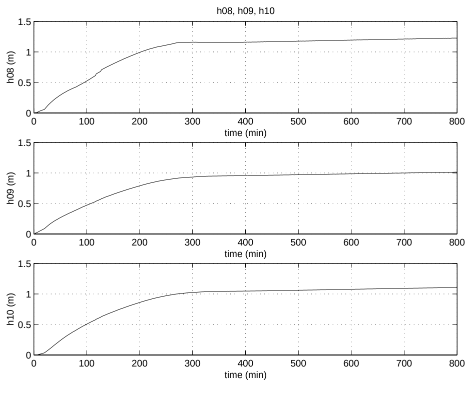

Figure 32: The head over gates for run 2

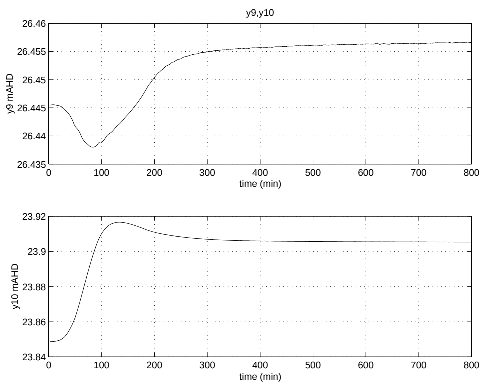

Figure 33: The water levels for run 3, initial water level slightly below error tolerance

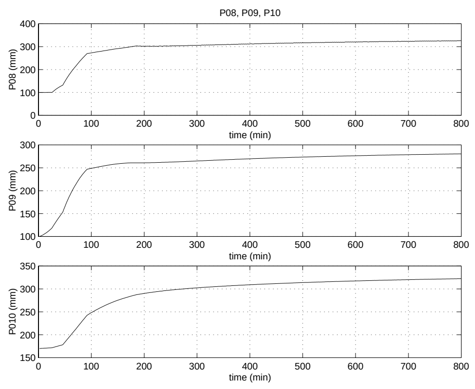

Figure 34: The gate positions for run 3

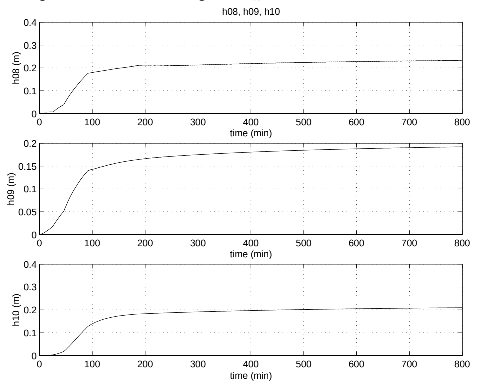

Figure 35: The head over gates for run 3

5.2.4 Initial water level in pool nine below error tolerance

Next test is to start with the water level in pool nine just under the error tolerance. The values of p are the values used to get the Simulink model in steady state.

$p_{08} = 30$ mm
$p_{09} = 40$ mm
$p_{010} = 230$ mm

Values used in the simulations (m):
$x\_initial[0] = 0.1$
$x\_initial[1] = 0.0000000001$
$\vdots$
$x\_initial[4] = 0.0000000001$
$x\_initial[5] = 0.05$
$x\_initial[6] = 0.0000000001$
$\vdots$
$x\_initial[8] = 0.0000000001$
$x\_initial[9] = 0.24$
$x\_initial[10] = y_9 = 26.502$ mAHD
$x\_initial[11] = y_{10} = 23.788$ mAHD

These values are explained in figure 26.

run	r_1	r_2	r_3	q_1	q_2	q_3	comments
1	400	400	400	1	1	0.1	put in a condition that h_{10} should minimizes around 0.1 m
2	400	400	400	1	1	0.0	q_3 is zero so the minimization around 0.1 m for h_{10} does not have any effect

If we take a look in figure 36 we can see that the controller reacts in a correct way. It starts increasing the water level in pool nine and this is a correct behaviour when that water level is too low. We can see that the water level goes to a nice value around set point. The water level in pool eight also decreases in the beginning but the level stays inside the error tolerance. It does not look as it goes to a certain value but it is inside the limit. One explanation for this is that pool eight is longer than pool nine and it needs more time to stabilize the water level.

When looking at the gate positions in figure 37 we can see that they start to open. This behaviour is correct because we want the water level in pool nine to increase. To reach this goal we have to open gate eight and nine. Gate ten opens because we want the head over gate ten to be minimized around 0.1 meter and to achieve that the gate has to open.

The head over gates, see figure 38, increase because we want more water in pool nine. We also see that the head over gate ten strives against 0.1 meter.

In run two q_3 is put to be zero. This results in that the minimization does not take any consideration about minimizing h_{10}. It seems like the water levels becomes more stable in this case. The head over gate ten becomes larger. The controller also have larger problems in the beginning than in run one. We make the conclusion that run one is better than run two. Figures for run two are not shown in the report because run one was much better.

We also want to see how the controller reacts when the water level is around one decimetre below the error tolerance in pool nine. The following values are used in this simulations:

p_{08}=40 mm
p_{09}=40 mm
p_{010}=330 mm

Values used in the simulations (m):
$x\_initial[0] = 0.1$
$x\_initial[1] = 0.0000000001$
$\vdots$
$x\_initial[4] = 0.0000000001$
$x\_initial[5] = 0.05$
$x\_initial[6] = 0.0000000001$
$\vdots$
$x\_initial[8] = 0.0000000001$
$x\_initial[9] = 0.350$
$x\_initial[10] = y_9 = 26.502$ mAHD
$x\_initial[11] = y_{10} = 23.687$ mAHD

These values are explained in figure 26.

run	r_1	r_2	r_3	q_1	q_2	q_3	comments
3	400	400	400	1	1	0.1	put in a condition that h_{10} should minimizes around 0.1 m
4	400	400	400	1	1	0.0	q_3 is zero so the minimization around 0.1 m for h_{10} does not have any effect

The water level in pool eight, see figure 39, starts increase immediately because we need more water for pool nine. It is a bit strange that the water level in pool nine starts to decrease in the beginning. This behaviour can be explained with that the head over gate ten wants to be minimized around 0.1 meter. But why the head over gate ten does not stop increase when it reaches 0.1 meter is strange. After a while the water level starts increase and finally it reaches a water level inside the error tolerance. The water level in pool eight increases a bit too much, but it seems like it strives to go against the set point.

In the next simulation (run four) q_{10} is set to zero. That means that we do not have any minimization on h_{10}. If we look at the water levels for this run, see figure 42, we can see that the final value for them are better than for run three. It is even more strange that gate ten starts to open in this case because we do not have any desire that it should go to 0.1 meter this time.

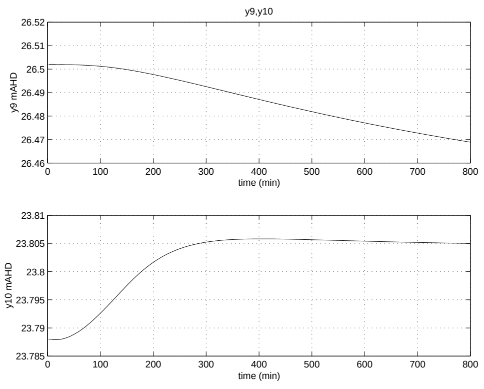

Figure 36: The water levels for run 1, initial water level slightly below error tolerance

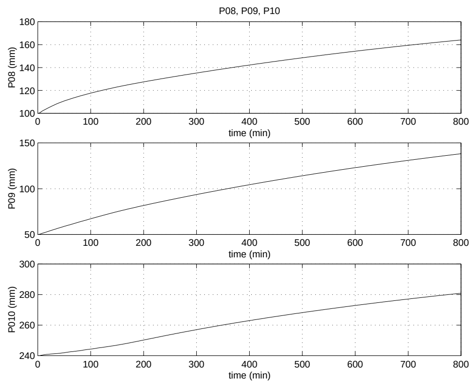

Figure 37: The gate positions for run 1

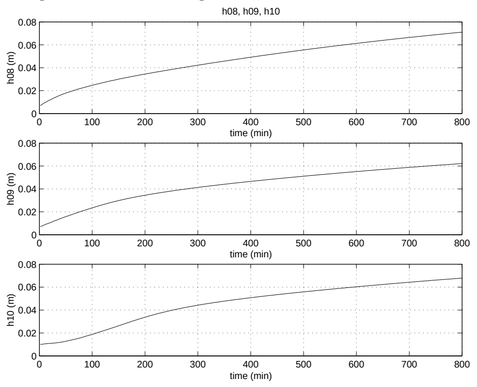

Figure 38: The head over gates for run 1

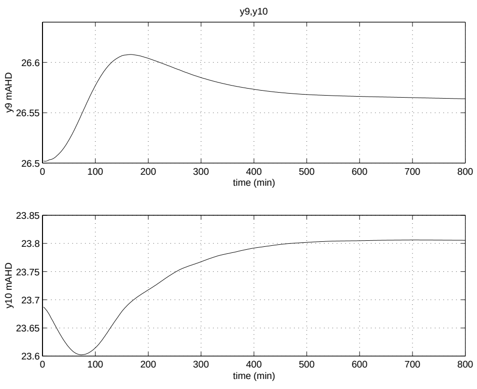

Figure 39: The water levels for run 3, initial water level about 1 dm below error tolerance

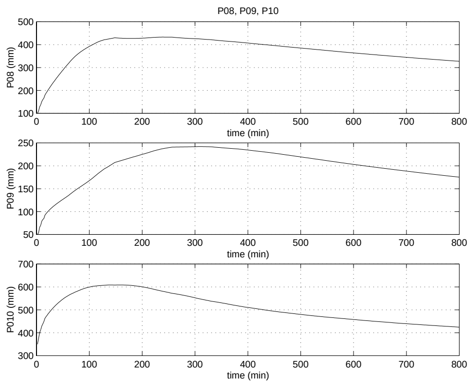

Figure 40: The gate positions for run 3

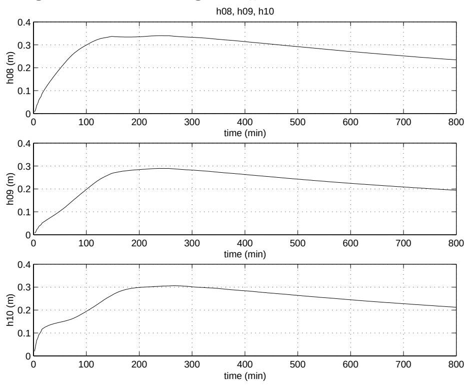

Figure 41: The head over gates for run 3

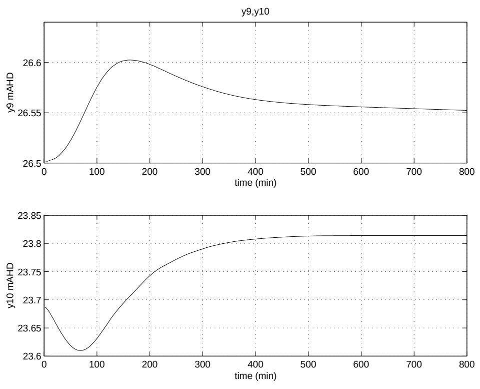

Figure 42: The water levels for run 4, initial water level about 1 dm below error tolerance

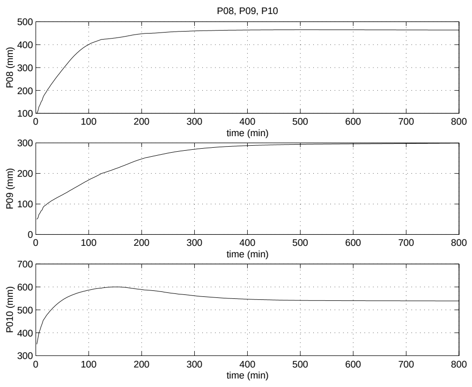

Figure 43: The gate positions for run 4

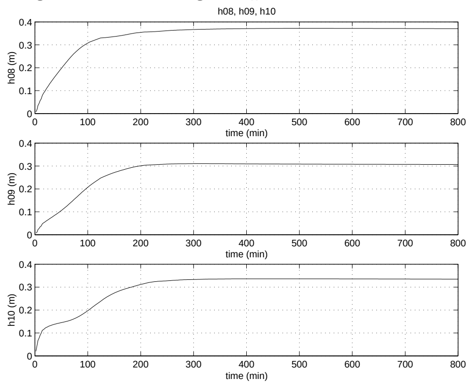

Figure 44: The head over gates for run 4

5.2.5 Initial water level in pool eight above error tolerance

In this test we put the water level about 0.01 meter above the error tolerance in pool eight. The results are good and can be seen in figure 45, 46 and 47. The values of p are the values used to get the Simulink model in steady state.

p_{08}=210 mm
p_{09}=80 mm
p_{010}=270 mm

$x\_initial[0] = 0.1$
$x\_initial[1] = 0.0000000001$
$\vdots$
$x\_initial[8] = 0.0000000001$
$x\_initial[9] = 0.17$
$x\_initial[10] = y_9 = 26.562$ mAHD
$x\_initial[11] = y_{10} = 23.854$ mAHD

These values are explained in figure 26.

run	r_1	r_2	r_3	q_1	q_2	q_3	comments
1	400	400	400	1	1	0.1	h_{10} is minimized around 0.0 meter

p_{08}=320 mm
p_{09}=80 mm
p_{010}=370 mm

$x\_initial[0] = 0.1$
$x\_initial[1] = 0.0000000001$
$\vdots$
$x\_initial[8] = 0.0000000001$
$x\_initial[9] = 0.17$
$x\_initial[10] = y_9 = 26.652$ mAHD
$x\_initial[11] = y_{10} = 23.853$ mAHD

These values are explained in figure 26.

run	r_1	r_2	r_3	q_1	q_2	q_3	comments
2	400	400	400	1	1	0.1	h_{10} is minimized around 0.0 meter

With an initial water level starting about one decimetre above the error tolerance in pool eight, the behaviour is the same as for they starting 0.01 meter above. The result can be seen in figure 48.

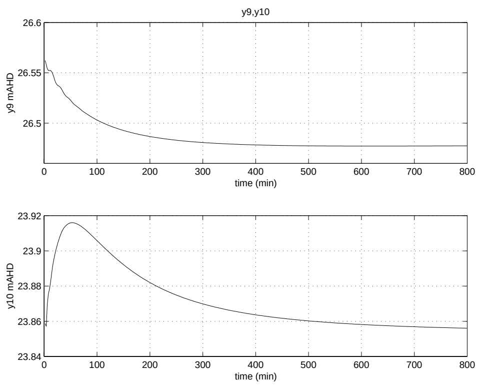

Figure 45: The initial water level in pool eight slightly above the error tolerance

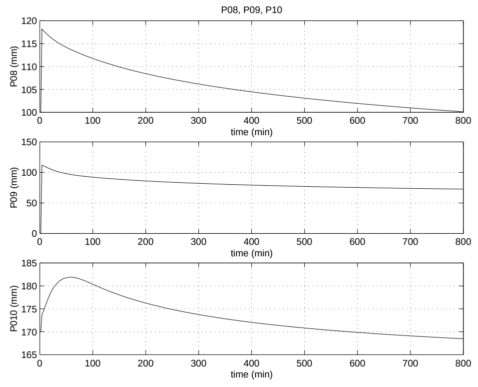

Figure 46: The gate movements with an initial water level slightly above the error tolerance

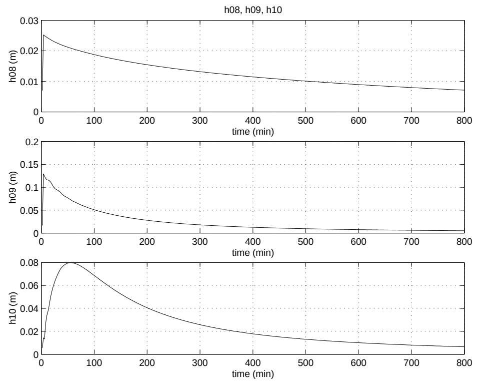

Figure 47: The head over gates with an initial water level slightly above the error tolerance

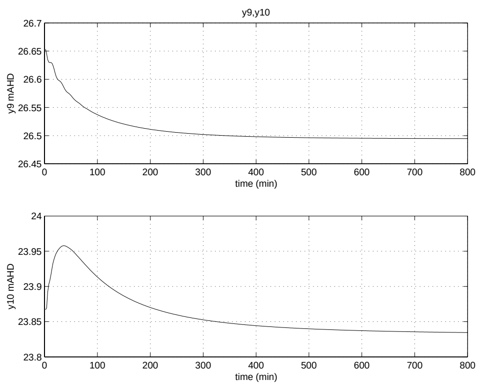

Figure 48: The initial water level in pool eight above the error tolerance

5.2.6 Initial water level in pool nine above error tolerance

The final test we did was to put the water level in pool nine above the error tolerance. Values used in the first tests are shown below:

p_{08}=40 mm
p_{09}=40 mm
p_{010}=330 mm

Values used in the simulations (m):
$x\_initial[0] = 0.1$
$x\_initial[1] = 0.0000000001$
$\vdots$
$x\_initial[4] = 0.0000000001$
$x\_initial[5] = 0.05$
$x\_initial[6] = 0.0000000001$
$\vdots$
$x\_initial[8] = 0.0000000001$
$x\_initial[9] = 0.12$
$x\_initial[10] = y_9 = 26.505$ mAHD
$x\_initial[11] = y_{10} = 23.908$ mAHD

These values are explained in figure 26.

run	r_1	r_2	r_3	q_1	q_2	q_3	comments
1	400	400	400	1	1	0.1	put in a condition that h_{10} should minimizes around 0.1 m
2	400	400	400	1	1	0.0	q_3 is zero so the minimization around 0.1 m for h_{10} does not have any effect

As we can see in figure 49, 50 and 51 the result from run one is not particularly good. The water level in pool nine never succeed to reach the error tolerance within 800 minutes. In run two the results are much better as you can see in figure 52, 53 and 54. Both the water levels are inside the error tolerance in the end and they also look stable. Even if the condition that h_{10} should be minimized is taken away in this run we can see that the head over gate ten is small. Why run two is better than run one is a bit hard to explain but it could be that the minimization has too many variables to minimize and it is difficult for the tool to do the minimization. The next thing to do should be to run simulations with the initial water level in pool nine about one decimetre above the error tolerance. We tried to run this but the optimization did not handle this situation (we got NaN immediately). We ran simulations with the water level around five centimetres above the error tolerance instead. Values used are shown below:

p_{08}=135 mm
p_{09}=80 mm
p_{010}=105 mm

Values used in the simulations (m):
$x\_initial[0] = 0.1$
$x\_initial[1] = 0.0000000001$
$\vdots$
$x\_initial[4] = 0.0000000001$
$x\_initial[5] = 0.05$
$x\_initial[6] = 0.0000000001$
$\vdots$
$x\_initial[8] = 0.0000000001$
$x\_initial[9] = 0.07$
$x\_initial[10] = y_9 = 26.500$ mAHD
$x\_initial[11] = y_{10} = 23.951$ mAHD

These values are explained in figure 26

run	r_1	r_2	r_3	q_1	q_2	q_3	comments
3	400	400	400	1	1	0.1	put in a condition that h_{10} should minimizes around 0.0 m

The optimization manages to run the problem without getting NaN but the solution is not good, see figure 55, 56 and figure 57. The expected behaviour would be to close gate eight and nine and open gate ten to get rid of the water in pool nine. As we can see this is not happening. Both gate eight and nine start to open and this gives an even higher water level in the beginning for pool nine than we had before. The water level in pool eight is decreasing in the beginning but then starts increasing and goes to a value just outside the error tolerance. Even if the behaviour is not good we can see that the water level in pool eight goes to a stable value. The water level in pool nine decreases after about 50 minutes so its final value is inside the error tolerance in the end. The bad behaviour could depend on initial disturbances as can be seen in the beginning of the water levels.

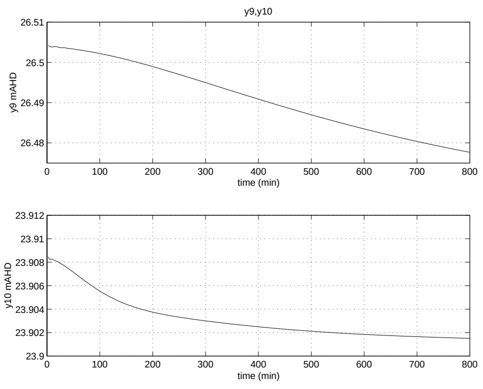

Figure 49: The water levels for run 1, initial water level slightly above error tolerance

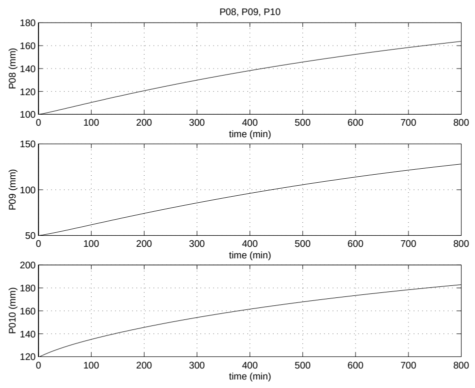

Figure 50: The gate positions for run 1

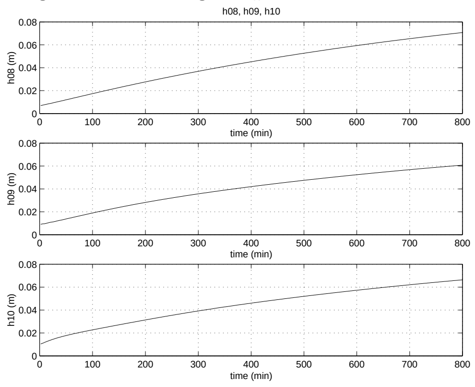

Figure 51: The gate positions for run1

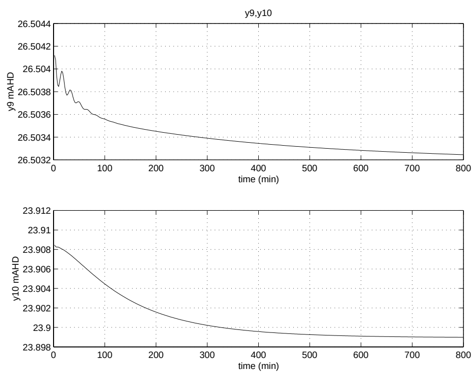

Figure 52: The water levels for run 2, initial water level slightly above error tolerance

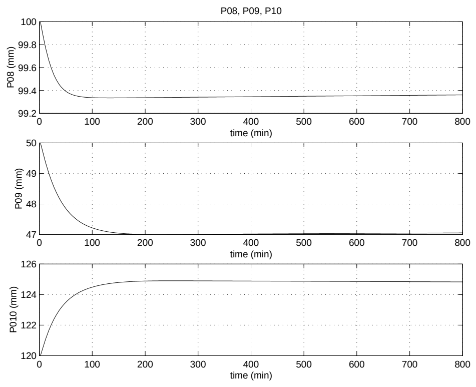

Figure 53: The gate positions for run 2

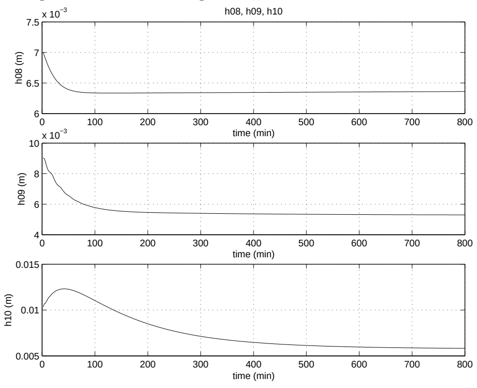

Figure 54: The head over gates for run 2

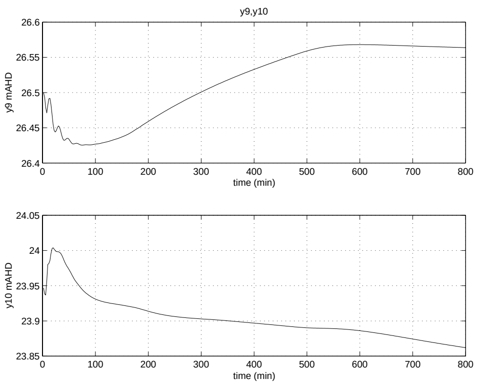

Figure 55: The water levels for run 3, initial water level about 1 dm above error tolerance

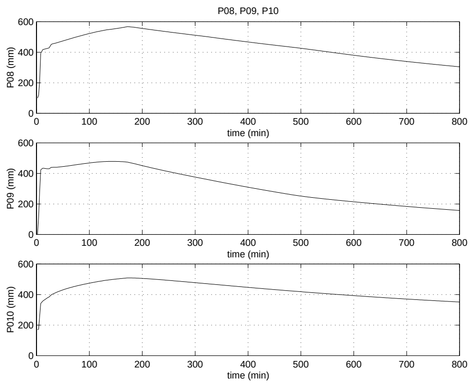

Figure 56: The gate positions for run 3

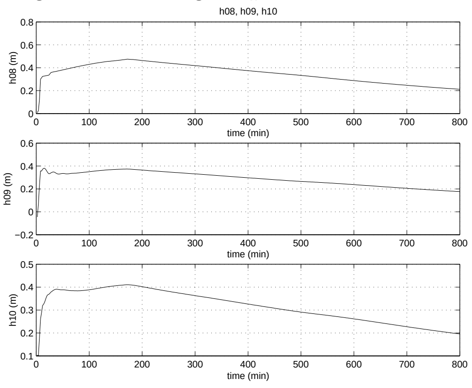

Figure 57: The head over gates for run 3

5.2.7 Summary

The controller reacts in a good way when the water levels are around set point from the beginning. We made simulations when we started both above and below the error tolerance and we also made simulations with disturbances to the pools. These simulations showed that the optimization tool probably is not the best one. In many cases it would be better to have a larger horizon but the optimization program did not manage to handle this. It also seems like the optimization tool wants the gate positions to follow each other and this gives strange results in some cases. This behaviour is hard to explain but it might be the short horizon in the optimization program that causes trouble. In general it is easier for the optimization program to handle disturbances in pool nine than in pool eight. This can be explained by that pool eight is longer than pool nine and because of that it takes longer time to see changes in pool eight.

The tunable parameters for our controller are the weighting matrices Q and R. It is a bit unexpected to have R, the weight on the gate movements, as large as we have, in most cases around $400 \cdot I$. It is expected to have higher weight on the error in the water levels and last head over gate, the Q matrix, instead. q_1 and q_2 are in most cases set to be one and q_3 to be 0.1, where q_1 and q_2 are the weights on the error in the water levels and q_3 is the weight on the last head over gate. These values we choose to make the optimization routine work in a satisfying way.

5.3 Practical implementation

5.3.1 Computational requirements

In the real world it is important that the optimization goes fast. The optimization must of course go faster than the water level is sampled, in our case this means two minutes. Our controller is satisfying in this point of view, one optimization in the program normally takes less than one minute, in most cases just a few seconds. The whole MPC takes 30-70 minutes and the predicted horizon is 800 minutes (13 hours) so this is also a good result.

5.3.2 Communication

In most irrigation systems the irrigators must order their water requirements in advance to enable the supply strategy to be determined. In this channel an interactive voice response system has been integrated. It is a Rubicon production database which enables computer controlled voice based interaction between the water production staff and irrigators. The water ordering, scheduling and confirmation transaction necessary for each and every irrigator order is now performed using computer assembled voice messages and telephone tones with minimal operator intervention.

To be able to supply enough water for the farmers there is a communication between some quantities in the channel and a central computer. The water levels in the channel are measured with a ultra sonic sensor. The gate positions are also measured. The values of these variables are sent to the central computer. The optimization is made in the computer and new gate positions are sent back to the gates.

6 Conclusions

6.1 Results

The result is satisfying in some aspects. The controller works fine when the initial water levels are around set points. A better behaviour is desired when we add disturbances to the system and also when the initial water levels are above and below the error tolerance.

The largest problem was to find an optimization tool for the minimization of the cost function. Our minimization problem is quite large and MATLAB was too slow. We decided to try a C-program that a student at the University of Melbourne had written, James Wettenhall. We finally got the optimization to work but just for a time horizon of 80 minutes. This can explain some of the strange results in the optimization. We found out that the results of the optimization depends very much of the initial conditions. It also seems like the optimizer wants the gate positions to follow each other and this is a strange behaviour that we found hard to explain. It might come from the short horizon in the optimization program. To get a better optimization a longer time horizon is needed. We have also learnt that it is preferable to have simple and linear constraints (and move the nonlinearities to the transfer functions) as this makes the optimization easier and faster. With a better optimization tool the result had probably been even better.

The tunable parameters for the controller are the weighting-matrices Q and R. It was hard to make the optimization program work for a large weight on the error and the last head over gate as we wanted to have. To be able to get results from the optimization program we have a small weight on the error and the head over gate ten, and a large weight on the gate movements.

Another result to notice is that the control of disturbances in pool eight takes much longer time to stabilize than disturbances in pool nine. It depends on that pool eight is longer than pool nine and have a longer time delay. The controller is satisfying when we start around the set point but a better behaviour when we add disturbances is desired.

6.2 Future work

The most important thing to do in the future is to find a better optimization tool to be able to increase the horizon of the optimization. We have been studying two pools in the channel, pool eight and pool nine, and one thing to continue with is to take more pools into consideration. Much work has been done on irrigation channels to make them better, but there is still a lot to do. The model we have used does not take wave-effects in the channel into account and another thing to do is to increase the model so it takes the wave-effects into account as well.

A Derivatives used in MATLAB - fmincon

A.1 Derivatives of $h_8(k)$ - $h_{10}(k)$ with respect to Δ $h_8(k)$- Δ $h_{10}(k)$

$h_8(k) = h_8(k-1) + \Delta h_8(k)$
$h_9(k) = h_9(k-1) + \Delta h_9(k)$
$h_{10}(k) = h_{10}(k-1) + \Delta h_{10}(k)$

$h_8(k) = h_8(0) + \sum_{k=1}^{n} \Delta h_8(k)$
$h_9(k) = h_9(0) + \sum_{k=1}^{n} \Delta h_9(k)$
$h_{10}(k) = h_{10}(0) + \sum_{k=1}^{n} \Delta h_{10}(k)$

$h_i(-6) = const$
$\vdots$
$h_i(0) = const$
$h_i(1) = h_i(0) + \Delta h_i(1)$
$h_i(2) = h_i(1) + \Delta h_i(2) = h_i(0) + \Delta h_i(1) + \Delta h_i(2)$
$h_i(3) = h_i(2) + \Delta h_i(3) = h_i(0) + \Delta h_i(1) + \Delta h_i(2) + \Delta h_i(3)$
$\vdots$
$h_i(n) = h_i(0) + \Delta h_i(1) + \ldots + \Delta h_i(n)$

for i=1,2,3

$\frac{\partial h_i(-6)}{\partial \Delta h_i(j)} = 0$
$\vdots$
$\frac{\partial h_i(j-1)}{\partial \Delta h_i(j)} = 0$

$\frac{\partial h_i(j)}{\partial \Delta h_i(j)} = 1$
$\vdots$
$\frac{\partial h_i(n)}{\partial \Delta h_i(j)} = 1$

for i=1,2,3 and j=1 ... n

A.2 Calculation of the gradient

A.2.1 Derivatives of $\hat{y}_9$

$\hat{y}_9(k+1) = y_9(k) + c_1 h_8(k-6) + c_2 h_9(k) - y_{9sp}$
$\hat{y}_{10}(k+1) = y_{10}(k) + c_3 h_9(k-4) + c_4 h_{10}(k) - y_{10sp}$
$\hat{h}_3(k+1) = h_{10}(k) + \Delta h_{10}(k)$

$\frac{\partial \mathbf{\hat{y}_9}}{\partial \mathbf{\Delta h_8}}$:

$\frac{\partial \hat{y}_9(1)}{\partial \Delta h_8(j)} = \frac{\partial y_9(0)}{\partial \Delta h_8(j)} + c_1 \frac{\partial h_8(-6)}{\partial \Delta h_8(j)} = 0$
$\vdots$
$\frac{\partial \hat{y}_9(6+j)}{\partial \Delta h_8(j)} = \frac{\partial y_9(5+j)}{\partial \Delta h_8(j)} + c_1 \frac{\partial h_8(j-1)}{\partial \Delta h_8(j)} = 0$

$\frac{\partial \hat{y}_9(7+j)}{\partial \Delta h_8(j)} = \frac{\partial y_9(6+j)}{\partial \Delta h_8(j)} + c_1 \frac{\partial h_8(j)}{\partial \Delta h_8(j)} = c_1$

$\vdots$

$\frac{\partial \hat{y}_9(k)}{\partial \Delta h_8(j)} = \frac{\partial y_9(k-1)}{\partial \Delta h_8(j)} + c_1 \frac{\partial h_8(k-7)}{\partial \Delta h_8(j)} = (k-(6+j))c_1$

$\vdots$

$\frac{\partial \hat{y}_9(n)}{\partial \Delta h_8(j)} = \frac{\partial y_9(n-1)}{\partial \Delta h_8(j)} + c_1 \frac{\partial h_8(n-7)}{\partial \Delta h_8(j)} = (n-(6+j))c_1$

for j=1 ... n

$\frac{\partial \mathbf{\hat{y}_9}}{\partial \mathbf{\Delta h_9}}$:

$\frac{\partial \hat{y}_9(1)}{\partial \Delta h_9(j)} = \frac{\partial y_9(0)}{\partial \Delta h_9(j)} + c_2 \frac{\partial h_9(0)}{\partial \Delta h_9(j)} = 0$

$\vdots$

$\frac{\partial \hat{y}_9(j)}{\partial \Delta h_9(j)} = \frac{\partial y_9(j-1)}{\partial \Delta h_9(j)} + c_2 \frac{\partial h_9(j-1)}{\partial \Delta h_9(j)} = 0$

$\frac{\partial \hat{y}_9(j+1)}{\partial \Delta h_9(j)} = \frac{\partial y_9(j)}{\partial \Delta h_9(j)} + c_2 \frac{\partial h_9(j)}{\partial \Delta h_9(j)} = c_2$

$\vdots$

$\frac{\partial \hat{y}_9(k)}{\partial \Delta h_9(j)} = \frac{\partial y_9(k-1)}{\partial \Delta h_9(j)} + c_2 \frac{\partial h_9(k-1)}{\partial \Delta h_9(j)} = (k-j)c_2$

$\vdots$

$\frac{\partial \hat{y}_9(n)}{\partial \Delta h_9(j)} = \frac{\partial y_9(n-1)}{\partial \Delta h_9(j)} + c_2 \frac{\partial h_9(n-1)}{\partial \Delta h_9(j)} = (n-j)c_2$

for j=1 ... n

$\frac{\partial \mathbf{\hat{y}_9}}{\partial \mathbf{\Delta h_{10}}}$:

$\hat{y}_9$ does not depend on Δh_9 so all derivatives are zero.

A.2.2 Derivatives of $\hat{y}_{10}$

$\frac{\partial \mathbf{\hat{y}_{10}}}{\partial \mathbf{\Delta h_8}}$:

$\hat{y}_{10}$ does not depend on Δh_8 so all derivatives are zero.

$\frac{\partial \mathbf{\hat{y}_{10}}}{\partial \mathbf{\Delta h_9}}$:

$\frac{\partial \hat{y}_{10}(1)}{\partial \Delta h_9(j)} = \frac{\partial y_{10}(0)}{\partial \Delta h_9(j)} + c_3 \frac{\partial h_9(-4)}{\partial \Delta h_9(j)} = 0$

$\vdots$

$\frac{\partial \hat{y}_{10}(j+4)}{\partial \Delta h_9(j)} = \frac{\partial y_{10}(j+3)}{\partial \Delta h_9(j)} + c_3 \frac{\partial h_9(j-1)}{\partial \Delta h_9(j)} = 0$

$\frac{\partial \hat{y}_{10}(j+5)}{\partial \Delta h_9(j)} = \frac{\partial y_{10}(j+4)}{\partial \Delta h_9(j)} + c_3 \frac{\partial h_9(j)}{\partial \Delta h_9(j)} = c_3$

$\vdots$

$\frac{\partial \hat{y}_{10}(k)}{\partial \Delta h_9(j)} = \frac{\partial y_{10}(k-1)}{\partial \Delta h_9(j)} + c_3 \frac{\partial h_9(k-5)}{\partial \Delta h_9(j)} = (k-(4+j))c_3$

$\vdots$

$\frac{\partial \hat{y}_{10}(n)}{\partial \Delta h_9(j)} = \frac{\partial y_{10}(n-1)}{\partial \Delta h_9(j)} + c_3 \frac{\partial h_9(n-5)}{\partial \Delta h_9(j)} = (n-(4+j))c_3$

for j=1 ... n

$\frac{\partial \mathbf{\hat{y}_{10}}}{\partial \mathbf{\Delta h_{10}}}$:

$\frac{\partial \hat{y}_{10}(1)}{\partial \Delta h_{10}(j)} = \frac{\partial y_{10}(0)}{\partial \Delta h_{10}(j)} + c_4 \frac{\partial h_{10}(0)}{\partial \Delta h_{10}(j)} = 0$

$\vdots$

$\frac{\partial \hat{y}_{10}(j)}{\partial \Delta h_{10}(j)} = \frac{\partial y_{10}(j-1)}{\partial \Delta h_{10}(j)} + c_4 \frac{\partial h_{10}(j-1)}{\partial \Delta h_{10}(j)} = 0$

$\frac{\partial \hat{y}_{10}(j+1)}{\partial \Delta h_{10}(j)} = \frac{\partial y_{10}(j)}{\partial \Delta h_{10}(j)} + c_4 \frac{\partial h_{10}(j)}{\partial \Delta h_{10}(j)} = c_4$

$\vdots$

$\frac{\partial \hat{y}_{10}(k)}{\partial \Delta h_{10}(j)} = \frac{\partial y_{10}(k-1)}{\partial \Delta h_{10}(j)} + c_4 \frac{\partial h_{10}(k-1)}{\partial \Delta h_{10}(j)} = (k-j)c_4$

$\vdots$

$\frac{\partial \hat{y}_{10}(n)}{\partial \Delta h_{10}(j)} = \frac{\partial y_{10}(n-1)}{\partial \Delta h_{10}(j)} + c_4 \frac{\partial h_{10}(n-1)}{\partial \Delta h_{10}(j)} = (n-j)c_4$

for j=1 . . . n

A.2.3 Derivatives of $\hat{h}_3$

$\frac{\partial \mathbf{\hat{h}_3}}{\partial \mathbf{\Delta h_8}}$:

$\hat{h}_3$ does not depend on Δh_8 so all derivatives are zero.

$\frac{\partial \mathbf{\hat{h}_3}}{\partial \mathbf{\Delta h_9}}$:

$\hat{h}_3$ does not depend on Δh_9 so all derivatives are zero.

$\frac{\partial \mathbf{\hat{h}_3}}{\partial \mathbf{\Delta h_{10}}}$:

$\frac{\partial \hat{h}_3(1)}{\partial \Delta h_{10}(j)} = \frac{\partial h_{10}(0)}{\partial \Delta h_{10}(j)} + \frac{\partial \Delta h_{10}(0)}{\partial \Delta h_{10}(j)} = 0$

$\vdots$

$\frac{\partial \hat{h}_3(j)}{\partial \Delta h_{10}(j)} = \frac{\partial h_{10}(j-1)}{\partial \Delta h_{10}(j)} + \frac{\partial \Delta h_{10}(j-1)}{\partial \Delta h_{10}(j)} = 0$

$\frac{\partial \hat{h}_3(j+1)}{\partial \Delta h_{10}(j)} = \frac{\partial h_{10}(j)}{\partial \Delta h_{10}(j)} + \frac{\partial \Delta h_{10}(j)}{\partial \Delta h_{10}(j)} = 1$

$\vdots$

$\frac{\partial \hat{h}_3(k)}{\partial \Delta h_{10}(j)} = \frac{\partial h_{10}(k-1)}{\partial \Delta h_{10}(j)} + \frac{\partial \Delta h_{10}(k-1)}{\partial \Delta h_{10}(j)} = 1$

$\vdots$

$\frac{\partial \hat{h}_3(n)}{\partial \Delta h_{10}(j)} = \frac{\partial h_{10}(n-1)}{\partial \Delta h_{10}(j)} + \frac{\partial \Delta h_{10}(n-1)}{\partial \Delta h_{10}(j)} = 1$

for j=1 . . . n

A.2.4 Derivatives of the constraints

The constraints are written on the form g(k) ≤ 0

$$
\begin{aligned}
g_1(k) &= -(h_8(k) - y_8 + 27.693) && (8)\\
g_2(k) &= h_8(k) - y_8 + 27.693 - 2.005 && (9)\\
g_3(k) &= -(h_9(k) - y_9(k) + 26.545) && (10)\\
g_4(k) &= h_9(k) - y_9(k) + 26.545 - 1.474 && (11)\\
g_5(k) &= -(h_{10}(k) - y_{10}(k) + 24.018) && (12)\\
g_6(k) &= h_{10}(k) - y_{10}(k) + 24.018) - 1.487 && (13)\\
g_7(k) &= -(h_8) && (14)\\
g_8(k) &= -(h_9) && (15)\\
g_9(k) &= -(h_{10}) && (16)\\
& && (17)
\end{aligned}
$$

Derivatives of constraint function 8

$\frac{\partial \mathbf{g_1}}{\partial \mathbf{\Delta h_8}}$:

$\frac{\partial g_1(1)}{\partial \Delta h_8(j)} = (-1) \cdot \frac{\partial h_8(1)}{\partial \Delta h_8(j)} = 0$

$\vdots$

$\frac{\partial g_1(j-1)}{\partial \Delta h_8(j)} = (-1) \cdot \frac{\partial h_8(j-1)}{\partial \Delta h_8(j)} = 0$

$\frac{\partial g_1(j)}{\partial \Delta h_8(j)} = (-1) \cdot \frac{\partial h_8(j)}{\partial \Delta h_8(j)} = -1$

$\vdots$

$\frac{\partial g_1(n)}{\partial \Delta h_8(j)} = (-1) \cdot \frac{\partial h_8(n)}{\partial \Delta h_8(j)} = -1$

for j=1 . . . n

$\frac{\partial \mathbf{g_1}}{\partial \mathbf{\Delta h_9}}$:

g_1 does not depend on Δh_9 so all derivatives are zero.

$\frac{\partial \mathbf{g_1}}{\partial \mathbf{\Delta h_{10}}}$:

g_1 does not depend on Δh_{10} so all derivatives are zero.

for j=1 . . . n

Derivatives of constraint function 9

$\frac{\partial \mathbf{g_2}}{\partial \mathbf{\Delta h_8}}$:

$\frac{\partial g_2(1)}{\partial \Delta h_8(j)} = \frac{\partial h_8(1)}{\partial \Delta h_8(j)} = 0$

$\vdots$

$\frac{\partial g_2(j-1)}{\partial \Delta h_8(j)} = \frac{\partial h_8(j-1)}{\partial \Delta h_8(j)} = 0$

$\frac{\partial g_2(j)}{\partial \Delta h_8(j)} = \frac{\partial h_8(j)}{\partial \Delta h_8(j)} = 1$

$\vdots$

$\frac{\partial g_2(n)}{\partial \Delta h_8(j)} = \frac{\partial h_8(n)}{\partial \Delta h_8(j)} = 1$

$\frac{\partial \mathbf{g_2}}{\partial \mathbf{\Delta h_9}}$:

g_2 does not depend on Δh_9 so all derivatives are zero.

$\frac{\partial \mathbf{g_2}}{\partial \mathbf{\Delta h_{10}}}$:

g_3 does not depend on Δh_{10} so all derivatives are zero.

for j=1 . . . n

Derivatives of constraint function 10

$\frac{\partial \mathbf{g_3}}{\partial \mathbf{\Delta h_8}}$:

$$\frac{\partial g_3(1)}{\partial \Delta h_8(j)} = \frac{\partial y_9(1)}{\partial \Delta h_8(j)} = 0$$
$$\vdots$$
$$\frac{\partial g_3(6+j)}{\partial \Delta h_8(j)} = \frac{\partial y_9(6+j)}{\partial \Delta h_8(j)} = 0$$
$$\frac{\partial g_3(7+j)}{\partial \Delta h_8(j)} = \frac{\partial y_9(7+j)}{\partial \Delta h_8(j)} = c_1$$
$$\vdots$$
$$\frac{\partial g_3(k)}{\partial \Delta h_8(j)} = \frac{\partial y_9(k)}{\partial \Delta h_8(j)} = (k-(6+j))c_1$$
$$\vdots$$
$$\frac{\partial g_3(n)}{\partial \Delta h_8(j)} = \frac{\partial y_9(n)}{\partial \Delta h_8(j)} = (n-(6+j))c_1$$

$\frac{\partial \mathbf{g_3}}{\partial \mathbf{\Delta h_9}}$:

$$\frac{\partial g_3(1)}{\partial \Delta h_9(j)} = (-1)\cdot(\frac{\partial h_9(1)}{\partial \Delta h_9(j)} - \frac{\partial y_9(1)}{\partial \Delta h_9(j)}) = 0$$
$$\vdots$$
$$\frac{\partial g_3(j-1)}{\partial \Delta h_9(j)} = (-1)\cdot(\frac{\partial h_9(j-1)}{\partial \Delta h_9(j)} - \frac{\partial y_9(j-1)}{\partial \Delta h_9(j)}) = 0$$
$$\frac{\partial g_3(j)}{\partial \Delta h_9(j)} = (-1)\cdot(\frac{\partial h_9(j)}{\partial \Delta h_9(j)} - \frac{\partial y_9(j)}{\partial \Delta h_9(j)}) = -1$$
$$\frac{\partial g_3(j+1)}{\partial \Delta h_9(j)} = (-1)\cdot(\frac{\partial h_9(j+1)}{\partial \Delta h_9(j)} - \frac{\partial y_9(j+1)}{\partial \Delta h_9(j)}) = (-1)\cdot(1-c_2)$$
$$\vdots$$
$$\frac{\partial g_3(k)}{\partial \Delta h_9(j)} = (-1)\cdot(\frac{\partial h_9(k)}{\partial \Delta h_9(j)} - \frac{\partial y_9(k)}{\partial \Delta h_9(j)}) = (-1)\cdot(1-(k-j)c_2)$$
$$\vdots$$
$$\frac{\partial g_3(n)}{\partial \Delta h_9(j)} = (-1)\cdot(\frac{\partial h_9(n)}{\partial \Delta h_9(j)} - \frac{\partial y_9(n)}{\partial \Delta h_9(j)}) = (-1)\cdot(1-(n-j)c_2)$$

$\frac{\partial \mathbf{g_3}}{\partial \mathbf{\Delta h_{10}}}$:

g_3 does not depend on Δh_{10} so all derivatives are zero.

for j=1 . . . n

Derivatives of constraint function 11

$\frac{\partial \mathbf{g_4}}{\partial \mathbf{\Delta h_8}}$:

$\frac{\partial g_4(1)}{\partial \Delta h_8(j)} = -\frac{\partial y_9(1)}{\partial \Delta h_8(j)} = 0$

$\vdots$

$\frac{\partial g_4(6+j)}{\partial \Delta h_8(j)} = -\frac{\partial y_9(6+j)}{\partial \Delta h_8(j)} = 0$
$\frac{\partial g_4(7+j)}{\partial \Delta h_8(j)} = -\frac{\partial y_9(7+j)}{\partial \Delta h_8(j)} = -c_1$

$\vdots$

$\frac{\partial g_4(k)}{\partial \Delta h_8(j)} = -\frac{\partial y_9(k)}{\partial \Delta h_8(j)} = -(k-(6+j))c_1$

$\vdots$

$\frac{\partial g_4(n)}{\partial \Delta h_8(j)} = -\frac{\partial y_9(n)}{\partial \Delta h_8(j)} = -(n-(6+j))c_1$

$\frac{\partial \mathbf{g_4}}{\partial \mathbf{\Delta h_9}}$:

$\frac{\partial g_4(1)}{\partial \Delta h_9(j)} = 1 \cdot (\frac{\partial h_9(1)}{\partial \Delta h_9(j)} - \frac{\partial y_9(1)}{\partial \Delta h_9(j)}) = 0$

$\vdots$

$\frac{\partial g_4(j-1)}{\partial \Delta h_9(j)} = 1 \cdot (\frac{\partial h_9(j)}{\partial \Delta h_9(j)} - \frac{\partial y_9(j-1)}{\partial \Delta h_9(j)}) = 0$
$\frac{\partial g_4(j)}{\partial \Delta h_9(j)} = 1 \cdot (\frac{\partial h_9(j)}{\partial \Delta h_9(j)} - \frac{\partial y_9(j)}{\partial \Delta h_9(j)}) = 1$
$\frac{\partial g_4(j+1)}{\partial \Delta h_9(j)} = 1 \cdot (\frac{\partial h_9(j+1)}{\partial \Delta h_9(j)} - \frac{\partial y_9(j+1)}{\partial \Delta h_9(j)}) = (1-c_2)$

$\vdots$

$\frac{\partial g_4(k)}{\partial \Delta h_9(j)} = 1 \cdot (\frac{\partial h_9(k)}{\partial \Delta h_9(j)} - \frac{\partial y_9(k)}{\partial \Delta h_9(j)}) = (1-(k-j)c_2)$

$\vdots$

$\frac{\partial g_4(n)}{\partial \Delta h_9(j)} = 1 \cdot (\frac{\partial h_9(n)}{\partial \Delta h_9(j)} - \frac{\partial y_9(n)}{\partial \Delta h_9(j)}) = (1-(n-j)c_2)$

$\frac{\partial \mathbf{g_4}}{\partial \mathbf{\Delta h_{10}}}$:

g_4 does not depend on Δh_{10} so all derivatives are zero.

for j=1 ... n

Derivatives of constraint function 12

$\frac{\partial \mathbf{g_5}}{\partial \mathbf{\Delta h_8}}$:

g_5 does not depend on Δh_{10} so all derivatives are zero.

$\frac{\partial \mathbf{g_5}}{\partial \mathbf{\Delta h_9}}$:

$\frac{\partial g_5(1)}{\partial \Delta h_9(j)} = \frac{\partial y_{10}(1)}{\partial \Delta h_9(j)} = 0$

$\vdots$

$\frac{\partial g_5(j+4)}{\partial \Delta h_9(j)} = \frac{\partial y_{10}(j+4)}{\partial \Delta h_9(j)} = 0$
$\frac{\partial g_5(j+5)}{\partial \Delta h_9(j)} = \frac{\partial y_{10}(j+5)}{\partial \Delta h_9(j)} = c_3$

$\vdots$

$\frac{\partial g_5(k)}{\partial \Delta h_9(j)} = \frac{\partial y_{10}(k)}{\partial \Delta h_9(j)} = (k-(4+j))c_3$

$\vdots$

$\frac{\partial g_5(n)}{\partial \Delta h_9(j)} = \frac{\partial y_{10}(n)}{\partial \Delta h_9(j)} = (n-(4+j))c_3$

$\frac{\partial \mathbf{g_5}}{\partial \mathbf{\Delta h_{10}}}$:

$\frac{\partial g_5(1)}{\partial \Delta h_{10}(j)} = (-1)\cdot(\frac{\partial h_{10}(1)}{\partial \Delta h_{10}(j)} - \frac{\partial y_{10}(1)}{\partial \Delta h_{10}(j)}) = 0$

$\vdots$

$\frac{\partial g_5(j-1)}{\partial \Delta h_{10}(j)} = (-1)\cdot(\frac{\partial h_{10}(j-1)}{\partial \Delta h_{10}(j)} - \frac{\partial y_{10}(j-1)}{\partial \Delta h_{10}(j)}) = 0$
$\frac{\partial g_5(j)}{\partial \Delta h_{10}(j)} = (-1)\cdot(\frac{\partial h_{10}(j)}{\partial \Delta h_{10}(j)} - \frac{\partial y_{10}(j)}{\partial \Delta h_{10}(j)}) = -1$
$\frac{\partial g_5(j+1)}{\partial \Delta h_{10}(j)} = (-1)\cdot(\frac{\partial h_{10}(j+1)}{\partial \Delta h_{10}(j)} - \frac{\partial y_{10}(j+1)}{\partial \Delta h_{10}(j)}) = (-1)\cdot(1-c_4)$

$\vdots$

$\frac{\partial g_5(k)}{\partial \Delta h_{10}(j)} = (-1)\cdot(\frac{\partial h_{10}(k)}{\partial \Delta h_{10}(j)} - \frac{\partial y_{10}(k)}{\partial \Delta h_{10}(j)}) = (-1)\cdot(1-(k-j)c_4)$

$\vdots$

$\frac{\partial g_5(n)}{\partial \Delta h_{10}(j)} = (-1)\cdot(\frac{\partial h_{10}(n)}{\partial \Delta h_{10}(j)} - \frac{\partial y_{10}(n)}{\partial \Delta h_{10}(j)}) = (-1)\cdot(1-(n-j)c_4)$

for j=1 . . . n

Derivatives of constraint function 13

$\frac{\partial \mathbf{g_6}}{\partial \mathbf{\Delta h_8}}$:

g_6 does not depend on Δh_{10} so all derivatives are zero.

$\frac{\partial \mathbf{g_6}}{\partial \mathbf{\Delta h_9}}$:

$\frac{\partial g_6(1)}{\partial \Delta h_9(j)} = -\frac{\partial y_{10}(1)}{\partial \Delta h_9(j)} = 0$

$\vdots$

$\frac{\partial g_6(j+4)}{\partial \Delta h_9(j)} = -\frac{\partial y_{10}(j+4)}{\partial \Delta h_9(j)} = 0$
$\frac{\partial g_6(j+5)}{\partial \Delta h_9(j)} = -\frac{\partial y_{10}(j+5)}{\partial \Delta h_9(j)} = -c_3$

$\vdots$

$\frac{\partial g_6(k)}{\partial \Delta h_9(j)} = -\frac{\partial y_{10}(k)}{\partial \Delta h_9(j)} = -(k-(4+j))c_3$

$\vdots$

$\frac{\partial g_6(n)}{\partial \Delta h_9(j)} = -\frac{\partial y_{10}(n)}{\partial \Delta h_9(j)} = -(n-(4+j))c_3$

$\frac{\partial \mathbf{g_6}}{\partial \mathbf{\Delta h_{10}}}$:

$\frac{\partial g_6(1)}{\partial \Delta h_{10}(j)} = 1\cdot(\frac{\partial h_{10}(1)}{\partial \Delta h_{10}(j)} - \frac{\partial y_{10}(1)}{\partial \Delta h_{10}(j)}) = 0$

$\vdots$

$\frac{\partial g_6(j-1)}{\partial \Delta h_{10}(j)} = 1\cdot(\frac{\partial h_{10}(j-1)}{\partial \Delta h_{10}(j)} - \frac{\partial y_{10}(j-1)}{\partial \Delta h_{10}(j)}) = 0$

$\frac{\partial g_6(j)}{\partial \Delta h_{10}(j)} = 1\cdot(\frac{\partial h_{10}(j)}{\partial \Delta h_{10}(j)} - \frac{\partial y_{10}(j)}{\partial \Delta h_{10}(j)}) = 1$

$\frac{\partial g_6(j+1)}{\partial \Delta h_{10}(j)} = 1\cdot(\frac{\partial h_{10}(j+1)}{\partial \Delta h_{10}(j)} - \frac{\partial y_{10}(j+1)}{\partial \Delta h_{10}(j)}) = 1\cdot(1-c_4)$

$\vdots$

$\frac{\partial g_6(k)}{\partial \Delta h_{10}(j)} = 1\cdot(\frac{\partial h_{10}(k)}{\partial \Delta h_{10}(j)} - \frac{\partial y_{10}(k)}{\partial \Delta h_{10}(j)}) = 1\cdot(1-(k-j)c_4)$

$\vdots$

$\frac{\partial g_6(n)}{\partial \Delta h_{10}(j)} = 1 \cdot (\frac{\partial h_{10}(n)}{\partial \Delta h_{10}(j)} - \frac{\partial y_{10}(n)}{\partial \Delta h_{10}(j)}) = 1 \cdot (1 - (n-j)c_4)$

for j=1 ... n

Derivatives of constraint function 14

$\frac{\partial \mathbf{g_7}}{\partial \mathbf{\Delta h_8}}$:

$\frac{\partial g_7(1)}{\partial \Delta h_8(j)} = -\frac{\partial h_8(1)}{\partial \Delta h_8(j)} = 0$

$\vdots$

$\frac{\partial g_7(j-1)}{\partial \Delta h_8(j)} = -\frac{\partial h_8(j-1)}{\partial \Delta h_8(j)} = 0$

$\frac{\partial g_7(j)}{\partial \Delta h_8(j)} = -\frac{\partial h_8(j)}{\partial \Delta h_8(j)} = -1$

$\vdots$

$\frac{\partial g_7(k)}{\partial \Delta h_8(j)} = -\frac{\partial h_8(k)}{\partial \Delta h_8(j)} = -1$

$\vdots$

$\frac{\partial g_7(n)}{\partial \Delta h_8(j)} = -\frac{\partial h_8(n)}{\partial \Delta h_8(j)} = -1$

$\frac{\partial \mathbf{g_7}}{\partial \mathbf{\Delta h_9}}$:

g_7 does not depend on Δh_9 so all derivatives are zero.

$\frac{\partial \mathbf{g_7}}{\partial \mathbf{\Delta h_{10}}}$:

g_7 does not depend on Δh_{10} so all derivatives are zero.

Derivatives of constraint function 15

$\frac{\partial \mathbf{g_8}}{\partial \mathbf{\Delta h_8}}$:

g_8 does not depend on Δh_8 so all derivatives are zero.

$\frac{\partial \mathbf{g_8}}{\partial \mathbf{\Delta h_9}}$:

$\frac{\partial g_8(1)}{\partial \Delta h_9(j)} = -\frac{\partial h_9(1)}{\partial \Delta h_9(j)} = 0$

$\vdots$

$\frac{\partial g_8(j-1)}{\partial \Delta h_9(j)} = -\frac{\partial h_9(j-1)}{\partial \Delta h_9(j)} = 0$

$\frac{\partial g_8(j)}{\partial \Delta h_9(j)} = -\frac{\partial h_9(j)}{\partial \Delta h_9(j)} = -1$

$\vdots$

$\frac{\partial g_8(k)}{\partial \Delta h_9(j)} = -\frac{\partial h_9(k)}{\partial \Delta h_9(j)} = -1$

$\vdots$

$\frac{\partial g_8(n)}{\partial \Delta h_9(j)} = -\frac{\partial h_9(n)}{\partial \Delta h_9(j)} = -1$

$\frac{\partial \mathbf{g_8}}{\partial \mathbf{\Delta h_{10}}}$:

g_8 does not depend on Δh_{10} so all derivatives are zero.

Derivatives of constraint function 16

$\frac{\partial \mathbf{g_9}}{\partial \mathbf{\Delta h_8}}$:

g_9 does not depend on Δh_8 so all derivatives are zero.

$\frac{\partial \mathbf{g_9}}{\partial \mathbf{\Delta h_9}}$:

g_9 does not depend on Δh_9 so all derivatives are zero.

$\frac{\partial \mathbf{g_9}}{\partial \mathbf{\Delta h_{10}}}$:

$\frac{\partial g_9(1)}{\partial \Delta h_{10}(j)} = -\frac{\partial h_{10}(1)}{\partial \Delta h_{10}(j)} = 0$

$\vdots$

$\frac{\partial g_9(j-1)}{\partial \Delta h_{10}(j)} = -\frac{\partial h_{10}(j-1)}{\partial \Delta h_{10}(j)} = 0$

$\frac{\partial g_9(j)}{\partial \Delta h_{10}(j)} = -\frac{\partial h_{10}(j)}{\partial \Delta h_{10}(j)} = -1$

$\vdots$

$\frac{\partial g_9(k)}{\partial \Delta h_{10}(j)} = -\frac{\partial h_{10}(k)}{\partial \Delta h_{10}(j)} = -1$

$\vdots$

$\frac{\partial g_9(n)}{\partial \Delta h_{10}(j)} = -\frac{\partial h_{10}(n)}{\partial \Delta h_{10}(j)} = -1$

B Derivatives in C-program, version one

Version one with linear model, nonlinear constraints and one minute sample. Only derivatives not equal to zero are shown.

B.1 Derivatives of the cost function

$$J = \frac{1}{2} \cdot (x_{13}^2 \cdot q_1 + x_{14}^2 \cdot q_2 + x_{12}^2 \cdot q_3 + u_0^2 \cdot r_1 + u_1^2 \cdot r_2 + u_2^2 \cdot r_3)$$

$\frac{\partial \mathbf{J}}{\partial \mathbf{u}}$:	$\frac{\partial J}{\partial u_0} = r_1 \cdot u_0$	$\frac{\partial J}{\partial u_1} = r_2 \cdot u_1$	$\frac{\partial J}{\partial u_2} = r_1 \cdot u_2$
$\frac{\partial \mathbf{J}}{\partial \mathbf{x}}$:	$\frac{\partial J}{\partial x_{12}} = q_3 \cdot x_{12}$	$\frac{\partial J}{\partial x_{13}} = q_1 \cdot x_{13}$	$\frac{\partial J}{\partial x_{14}} = q_2 \cdot x_{14}$
$\frac{\partial \mathbf{J^2}}{\partial \mathbf{u^2}}$:	$\frac{\partial J^2}{\partial u_0^2} = r_1$	$\frac{\partial J^2}{\partial u_1^2} = r_2$	$\frac{\partial J^2}{\partial u_1^2} = r_3$
$\frac{\partial \mathbf{J^2}}{\partial \mathbf{x^2}}$:	$\frac{\partial J^2}{\partial x_{12}^2} = q_3$	$\frac{\partial J^2}{\partial x_{13}^2} = q_1$	$\frac{\partial J^2}{\partial x_{14}^2} = q_2$

$\frac{\partial \mathbf{J^2}}{\partial \mathbf{x} \partial \mathbf{u}}$: All derivatives are zero.

B.2 Derivatives of the transitions

$t_0 = x_0 + u_0$
$t_1 = x_0$
$t_2 = x_1$
$t_3 = x_2$
$t_4 = x_3$
$t_5 = x_4$
$t_6 = x_5$
$t_7 = x_7 + u_1$
$t_8 = x_7$
$t_9 = x_8$
$t_{10} = x_9$
$t_{11} = x_{10}$
$t_{12} = x_{12} + u_2$
$t_{13} = c_1 \cdot x_6 + c_2 \cdot x_7 + x_{13}$
$t_{14} = c_3 \cdot x_{11} + c_4 \cdot x_{12} + x_{14}$

$\frac{\partial \mathbf{t}}{\partial \mathbf{x}}$:	$\frac{\partial t_0}{\partial x_0} = 1$	$\frac{\partial t_1}{\partial x_0} = 1$	$\frac{\partial t_2}{\partial x_1} = 1$	$\frac{\partial t_3}{\partial x_2} = 1$
	$\frac{\partial t_4}{\partial x_3} = 1$	$\frac{\partial t_5}{\partial x_4} = 1$	$\frac{\partial t_6}{\partial x_5} = 1$	$\frac{\partial t_7}{\partial x_7} = 1$
	$\frac{\partial t_8}{\partial x_7} = 1$	$\frac{\partial t_9}{\partial x_8} = 1$	$\frac{\partial t_{10}}{\partial x_9} = 1$	$\frac{\partial t_{11}}{\partial x_{10}} = 1$
	$\frac{\partial t_{12}}{\partial x_{12}} = 1$	$\frac{\partial t_{13}}{\partial x_6} = c_1$	$\frac{\partial t_{13}}{\partial x_7} = c_2$	$\frac{\partial t_{13}}{\partial x_{13}} = 1$
	$\frac{\partial t_{14}}{\partial x_{11}} = c_3$	$\frac{\partial t_{14}}{\partial x_{12}} = c_4$	$\frac{\partial t_{14}}{\partial x_{14}} = 1$	

$\frac{\partial \mathbf{t}}{\partial \mathbf{u}}$:	$\frac{\partial t_0}{\partial u_0} = 1$	$\frac{\partial t_7}{\partial u_1} = 1$	$\frac{\partial t_{12}}{\partial u_2} = 1$

$\frac{\partial \mathbf{t^2}}{\partial \mathbf{u^2}}$: All derivatives are zero.

$\frac{\partial \mathbf{t^2}}{\partial \mathbf{x^2}}$: All derivatives are zero.

$\frac{\partial \mathbf{t^2}}{\partial \mathbf{x} \partial \mathbf{u}}$:All derivatives are zero.

B.3 Derivatives of the constraints

The constraints are written on the form g ≥ 0

$$\begin{aligned}
g_0 &= x_0^{2/3} - y_8 + 27.693 && (18)\\
g_1 &= -(x_0^{2/3} - y_8 + 27.693 - 2.005) && (19)\\
g_2 &= x_7^{2/3} - x_{13} + 26.545 && (20)\\
g_3 &= -(x_7^{2/3} - x_{13} + 26.545 - 1.474) && (21)\\
g_4 &= x_{12}^{2/3} - x_{14} + 24.018 && (22)\\
g_5 &= -(x_{12}^{2/3} - x_{14} + 24.018 - 1.487) && (23)\\
g_6 &= x_0 && (24)\\
g_7 &= x_7 && (25)\\
g_8 &= x_{12} && (26)\\
& && (27)
\end{aligned}$$

$\frac{\partial \mathbf{g}}{\partial \mathbf{u}}$: All derivatives are zero.

$\frac{\partial \mathbf{g}}{\partial \mathbf{x}}$: $\frac{\partial g_0}{\partial x_0} = \frac{2}{3} \cdot x_0^{-1/3}$ $\frac{\partial g_1}{\partial x_0} = -\frac{2}{3} \cdot x_0^{-1/3}$ $\frac{\partial g_2}{\partial x_7} = \frac{2}{3} \cdot x_7^{-1/3}$ $\frac{\partial g_2}{\partial x_{13}} = -1$

$\frac{\partial g_3}{\partial x_7} = -\frac{2}{3} \cdot x_7^{-1/3}$ $\frac{\partial g_3}{\partial x_{13}} = 1$ $\frac{\partial g_4}{\partial x_{12}} = \frac{2}{3} \cdot x_{12}^{-1/3}$ $\frac{\partial g_4}{\partial x_{14}} = -1$

$\frac{\partial g_5}{\partial x_{12}} = -\frac{2}{3} \cdot x_{12}^{-1/3}$ $\frac{\partial g_5}{\partial x_{14}} = 1$ $\frac{\partial g_6}{\partial x_0} = 1$ $\frac{\partial g_7}{\partial x_7} = 1$

$\frac{\partial g_8}{\partial x_{12}} = 1$

$\frac{\partial \mathbf{g^2}}{\partial \mathbf{u^2}}$: All derivatives are zero.

$\frac{\partial \mathbf{g^2}}{\partial \mathbf{x^2}}$: $\frac{\partial g_0^2}{\partial x_0^2} = -\frac{2}{9} \cdot x_0^{-4/3}$ $\frac{\partial g_1^2}{\partial x_0^2} = \frac{2}{9} \cdot x_0^{-4/3}$ $\frac{\partial g_2^2}{\partial x_7^2} = -\frac{2}{9} \cdot x_7^{-4/3}$ $\frac{\partial g_3^2}{\partial x_7^2} = \frac{2}{9} \cdot x_7^{-4/3}$

$\frac{\partial g_4^2}{\partial x_{12}^2} = -\frac{2}{9} \cdot x_{12}^{-4/3}$ $\frac{\partial g_5^2}{\partial x_{12}^2} = \frac{2}{9} \cdot x_{12}^{-4/3}$

C Derivatives in C-program, version two

Version two with nonlinear model, linear constraints and one minute sample.
Only derivatives not equal to zero are shown.

C.1 Derivatives of the cost function

$$J = \frac{1}{2} \cdot (x_{13}^2 \cdot q_1 + x_{14}^2 \cdot q_2 + (x_{14} + x_{12} - g_3)^2 \cdot q_3 + u_0^2 \cdot r_1 + u_1^2 \cdot r_2 + u_2^2 \cdot r_3)$$

$\frac{\partial \mathbf{J}}{\partial \mathbf{u}}$:	$\frac{\partial J}{\partial u_0} = u_0 \cdot r_1$	$\frac{\partial J}{\partial u_1} = u_1 \cdot r_2$	$\frac{\partial J}{\partial u_2} = u_2 \cdot r_3$
$\frac{\partial \mathbf{J}}{\partial \mathbf{x}}$:	$\frac{\partial J}{\partial x_{12}} = (x_{14} + x_{12} - g_3) \cdot q_3$	$\frac{\partial J}{\partial x_{13}} = x_{13} \cdot q_1$	$\frac{\partial J}{\partial x_{14}} = x_{14} \cdot q_2 + (x_{14} + x_{12} - g_3) \cdot q_3$
$\frac{\partial \mathbf{J^2}}{\partial \mathbf{u^2}}$:	$\frac{\partial J^2}{\partial u_0^2} = r_1$	$\frac{\partial J^2}{\partial u_1^2} = r_2$	$\frac{\partial J^2}{\partial u_2^2} = r_3$
$\frac{\partial \mathbf{J^2}}{\partial \mathbf{x^2}}$:	$\frac{\partial J^2}{\partial x_{12}^2} = q_3$	$\frac{\partial J^2}{\partial x_{14} \partial x_{12}} = q_3$	$\frac{\partial J^2}{\partial x_{13}^2} = q_1$
	$\frac{\partial J^2}{\partial x_{12} \partial x_{14}} = q_3$	$\frac{\partial J^2}{\partial x_{14}^2} = q_2 + q_3$	

$\frac{\partial \mathbf{J^2}}{\partial \mathbf{u} \partial \mathbf{x}}$: All derivatives are zero.

C.2 Derivatives of the transitions

$t_0 = x_0 + u_0$
$t_1 = y_8 + x_0 - g_1$
$t_2 = x_1$
$t_3 = x_2$
$t_4 = x_3$
$t_5 = x_4$
$t_6 = x_5$
$t_7 = x_7 + u_1$
$t_8 = x_{13} + x_7 - g_2$
$t_9 = x_8$
$t_{10} = x_9$
$t_{11} = x_{10}$
$t_{12} = x_{12} + u_2$
$t_{13} = x_{13} + c_1 \cdot x_6^{3/2} + c_2 \cdot (x_{13} + x_7 - g_2)^{3/2}$
$t_{14} = x_{14} + c_3 \cdot x_{11}^{3/2} + c_4 \cdot (x_{14} + x_{12} - g_3)^{3/2}$

$\frac{\partial \mathbf{t}}{\partial \mathbf{u}}$: $\frac{\partial t_0}{u_0} = 1$ $\frac{\partial t_7}{u_1} = 1$ $\frac{\partial t_{12}}{u_2} = 1$

$\frac{\partial \mathbf{t}}{\partial \mathbf{x}}$: $\frac{\partial t_0}{x_0} = 1$ $\frac{\partial t_1}{x_0} = 1$ $\frac{\partial t_2}{x_0} = 1$

$\frac{\partial t_3}{x_2} = 1$ $\frac{\partial t_4}{x_3} = 1$ $\frac{\partial t_5}{x_4} = 1$

$\frac{\partial t_6}{x_5} = 1$ $\frac{\partial t_7}{x_7} = 1$ $\frac{\partial t_8}{x_7} = 1$

$\frac{\partial t_8}{x_{13}} = 1$ $\frac{\partial t_9}{x_8} = 1$ $\frac{\partial t_{10}}{x_9} = 1$

$\frac{\partial t_{11}}{x_{10}} = 1$ $\frac{\partial t_{12}}{x_{12}} = 1$ $\frac{\partial t_{13}}{x_6} = \frac{3}{2} \cdot c_1 \cdot x_6^{1/2}$

$\frac{\partial t_{13}}{\partial x_7} = \frac{3}{2} \cdot c_2 \cdot (x_{13} + x7 - g_2)^{1/2}$ $\frac{\partial t_{13}}{\partial x_{13}} = 1 + \frac{3}{2} \cdot c_2 \cdot (x_{13} + x7 - g_2)^{1/2}$ $\frac{\partial t_{14}}{x_{11}} = \frac{3}{2} \cdot c_3 \cdot x_{11}^{1/2}$

$\frac{\partial t_{14}}{\partial x_{12}} = \frac{3}{2} \cdot c_4 \cdot (x_{14} + x12 - g_3)^{1/2}$ $\frac{\partial t_{14}}{\partial x_{14}} = 1 + \frac{3}{2} \cdot c_4 \cdot (x_{14} + x12 - g_3)^{1/2}$

$\frac{\partial \mathbf{t^2}}{\partial \mathbf{u^2}}$: All derivatives are zero.

$\frac{\partial \mathbf{t^2}}{\partial \mathbf{u} \partial \mathbf{x}}$:All derivatives are zero.

$\frac{\partial \mathbf{t^2}}{\partial \mathbf{x^2}}$: $\frac{\partial t_{13}}{\partial x_6^2} = \frac{3}{4} \cdot x_6^{-1/2}$ $\frac{\partial t_{13}}{\partial x_7^2} = \frac{3}{4} \cdot c_2 \cdot (x_{13} + x_7 - g_2)^{-1/2}$

$\frac{\partial t_{13}}{\partial x_7 \partial x_{13}} = \frac{3}{4} \cdot c_2 \cdot (x_{13} + x_7 - g_2)^{-1/2}$ $\frac{\partial t_{13}}{\partial x_{13} \partial x_7} = \frac{3}{4} \cdot c_2 \cdot (x_{13} + x_7 - g_2)^{-1/2}$

$\frac{\partial t_{13}}{\partial x_{13}^2} = \frac{3}{4} \cdot c_2 \cdot (x_{13} + x_7 - g_2)^{-1/2}$ $\frac{\partial t_{14}}{\partial x_{11}^2} = \frac{3}{4} \cdot c_3 \cdot x_{11}^{-1/2}$

$\frac{\partial t_{14}}{\partial x_{12}^2} = \frac{3}{4} \cdot c_4 \cdot (x_{14} + x_{12} - g_3)^{-1/2}$ $\frac{\partial t_{14}}{\partial x_{14} \partial x_{12}} = \frac{3}{4} \cdot c_4 \cdot (x_{14} + x_{12} - g_3)^{-1/2}$

$\frac{\partial t_{14}}{\partial x_{12} \partial x_{14}} = \frac{3}{4} \cdot c_4 \cdot (x_{14} + x_{12} - g_3)^{-1/2}$ $\frac{\partial t_{14}}{\partial x_{14}^2} = \frac{3}{4} \cdot c_4 \cdot (x_{14} + x_{12} - g_3)^{-1/2}$

C.3 Derivatives of the constraints

The constraints are written on the form g ≥ 0

$$g_0 = x_0 \tag{28}$$
$$g_1 = 2.005 - x_0 \tag{29}$$
$$g_2 = x_7 \tag{30}$$
$$g_3 = 1.474 - x_7 \tag{31}$$
$$g_4 = x_{12} \tag{32}$$
$$g_5 = 1.487 - x_{12} \tag{33}$$
(34)

$\frac{\partial \mathbf{g}}{\partial \mathbf{u}}$: All derivatives are zero.

$\frac{\partial \mathbf{g}}{\partial \mathbf{x}}$: $\frac{\partial g_0}{\partial x_0} = 1$ $\frac{\partial g_1}{\partial x_0} = -1$ $\frac{\partial g_2}{\partial x_7} = 1$

$\frac{\partial g_3}{\partial x_7} = -1$ $\frac{\partial g_4}{\partial x_{12}} = 1$ $\frac{\partial g_5}{\partial x_{12}} = -1$

$\frac{\partial \mathbf{g^2}}{\partial \mathbf{u^2}}$: All derivatives are zero.

$\frac{\partial \mathbf{g^2}}{\partial \mathbf{x^2}}$: All derivatives are zero.

$\frac{\partial \mathbf{g^2}}{\partial \mathbf{u} \partial \mathbf{x}}$:All derivatives are zero.

D Derivatives in C-program, version three

Version three with nonlinear model, linear constraints and two minutes sample
Only derivatives not equal to zero are shown.

D.1 Derivatives of the cost function

$$J = \frac{1}{2} \cdot (x_{10}^2 \cdot q_1 + x_{11}^2 \cdot q_2 + (x_{11} + x_9 - g_3)^2 \cdot q_3 + u_0^2 \cdot r_1 + u_1^2 \cdot r_2 + u_2^2 \cdot r_3)$$

$\frac{\partial \mathbf{J}}{\partial \mathbf{u}}$: $\frac{\partial J}{\partial u_0} = u_0 \cdot r_1$ $\frac{\partial J}{\partial u_1} = u_1 \cdot r_2$ $\frac{\partial J}{\partial u_2} = u_2 \cdot r_3$

$\frac{\partial \mathbf{J}}{\partial \mathbf{x}}$: $\frac{\partial J}{\partial x_9} = (x_{11} + x_9 - g_3) \cdot q_3$ $\frac{\partial J}{\partial x_{10}} = x_{10} \cdot q_1$ $\frac{\partial J}{\partial x_{11}} = x_{11} \cdot q_2 + (x_{11} + x_9 - g_3) \cdot q_3$

$\frac{\partial \mathbf{J^2}}{\partial \mathbf{u^2}}$: $\frac{\partial J^2}{\partial u_0^2} = r_1$ $\frac{\partial J^2}{\partial u_1^2} = r_2$ $\frac{\partial J^2}{\partial u_2^2} = r_3$

$\frac{\partial \mathbf{J^2}}{\partial \mathbf{x^2}}$: $\frac{\partial J^2}{\partial x_9^2} = q_3$ $\frac{\partial J^2}{\partial x_{12} \partial x_9} = q_3$ $\frac{\partial J^2}{\partial x_{10}^2} = q_1$

$\frac{\partial J^2}{\partial x_9 \partial x_{11}} = q_3$ $\frac{\partial J^2}{\partial x_{11}^2} = q_2 + q_3$

$\frac{\partial \mathbf{J^2}}{\partial \mathbf{u} \partial \mathbf{x}}$: All derivatives are zero.

D.2 Derivatives of the transitions

$t_0 = x_0 + u_0$
$t_1 = y_8 + x_0 - g_1$
$t_2 = x_1$
$t_3 = x_2$
$t_4 = x_3$
$t_5 = x_5 + u_1$
$t_6 = x_{10} + x_5 - g_9$
$t_7 = x_6$
$t_8 = x_7$
$t_9 = x_9 + u_2$
$t_{10} = x_{10} + c_1 \cdot x_4^{3/2} + c_2 \cdot (x_{10} + x_5 - g_2)^{3/2}$
$t_{11} = x_{11} + c_3 \cdot x_8^{3/2} + c_4 \cdot (x_{11} + x_9 - g_3)^{3/2}$

$\frac{\partial \mathbf{t}}{\partial \mathbf{u}}$: $\frac{\partial t_0}{u_0} = 1$ $\frac{\partial t_5}{u_1} = 1$ $\frac{\partial t_9}{u_2} = 1$

$\frac{\partial \mathbf{t}}{\partial \mathbf{x}}$: $\frac{\partial t_0}{x_0} = 1$ $\frac{\partial t_1}{x_0} = 1$ $\frac{\partial t_2}{x_1} = 1$

$\frac{\partial t_3}{x_2} = 1$ $\frac{\partial t_4}{x_3} = 1$ $\frac{\partial t_5}{x_5} = 1$

$\frac{\partial t_6}{x_5} = 1$ $\frac{\partial t_6}{x_{10}} = 1$ $\frac{\partial t_7}{x_6} = 1$

$\frac{\partial t_8}{x_7} = 1$ $\frac{\partial t_9}{x_9} = 1$ $\frac{\partial t_{10}}{x_4} = \frac{3}{2} \cdot c_1 \cdot x_4^{1/2}$

$\frac{\partial t_{10}}{\partial x_5} = \frac{3}{2} \cdot c_2 \cdot (x_{10} + x5 - g_2)^{1/2}$ $\frac{\partial t_{10}}{\partial x_{10}} = 1 + \frac{3}{2} \cdot c_2 \cdot (x_{10} + x5 - g_2)^{1/2}$ $\frac{\partial t_{11}}{x_8} = \frac{3}{2} \cdot c_3 \cdot x_8^{1/2}$

$\frac{\partial t_{11}}{\partial x_9} = \frac{3}{2} \cdot c_4 \cdot (x_{11} + x9 - g_3)^{1/2}$ $\frac{\partial t_{11}}{\partial x_{11}} = 1 + \frac{3}{2} \cdot c_4 \cdot (x_{11} + x9 - g_3)^{1/2}$

$\frac{\partial \mathbf{t^2}}{\partial \mathbf{u^2}}$: All derivatives are zero.

$\frac{\partial \mathbf{t^2}}{\partial \mathbf{u} \partial \mathbf{x}}$:All derivatives are zero.

$\frac{\partial \mathbf{t^2}}{\partial \mathbf{x^2}}$: $\frac{\partial t_{10}}{\partial x_4^2} = \frac{3}{4} \cdot x_4^{-1/2}$ $\frac{\partial t_{10}}{\partial x_5^2} = \frac{3}{4} \cdot c_2 \cdot (x_{10} + x_5 - g_2)^{-1/2}$

$\frac{\partial t_{10}}{\partial x_5 \partial x_{10}} = \frac{3}{4} \cdot c_2 \cdot (x_{10} + x_5 - g_2)^{-1/2}$ $\frac{\partial t_{10}}{\partial x_{10} \partial x_5} = \frac{3}{4} \cdot c_2 \cdot (x_{10} + x_5 - g_2)^{-1/2}$

$\frac{\partial t_{10}}{\partial x_{10}^2} = \frac{3}{4} \cdot c_2 \cdot (x_{10} + x_5 - g_2)^{-1/2}$ $\frac{\partial t_{11}}{\partial x_8^2} = \frac{3}{4} \cdot c_3 \cdot x_8^{-1/2}$

$\frac{\partial t_{11}}{\partial x_9^2} = \frac{3}{4} \cdot c_4 \cdot (x_{11} + x_9 - g_3)^{-1/2}$ $\frac{\partial t_{11}}{\partial x_{11} \partial x_{12}} = \frac{3}{4} \cdot c_4 \cdot (x_{11} + x_9 - g_3)^{-1/2}$

$\frac{\partial t_{11}}{\partial x_9 \partial x_{11}} = \frac{3}{4} \cdot c_4 \cdot (x_{11} + x_9 - g_3)^{-1/2}$ $\frac{\partial t_{11}}{\partial x_{11}^2} = \frac{3}{4} \cdot c_4 \cdot (x_{11} + x_9 - g_3)^{-1/2}$

D.3 Derivatives of the constraints

The constraints are written on the form g ≥ 0

$$
\begin{aligned}
g_0 &= x_0 && (35)\\
g_1 &= 2.005 - x_0 && (36)\\
g_2 &= x_5 && (37)\\
g_3 &= 1.474 - x_5 && (38)\\
g_4 &= x_9 && (39)\\
g_5 &= 1.487 - x_9 && (40)\\
& && (41)
\end{aligned}
$$

$\frac{\partial \mathbf{g}}{\partial \mathbf{u}}$: All derivatives are zero.

$$\frac{\partial \mathbf{g}}{\partial \mathbf{x}} : \quad \frac{\partial g_0}{\partial x_0} = 1 \quad \frac{\partial g_1}{\partial x_0} = -1 \quad \frac{\partial g_2}{\partial x_5} = 1$$

$$\frac{\partial g_3}{\partial x_5} = -1 \quad \frac{\partial g_4}{\partial x_9} = 1 \quad \frac{\partial g_5}{\partial x_9} = -1$$

$\frac{\partial \mathbf{g^2}}{\partial \mathbf{u^2}}$: All derivatives are zero.

$\frac{\partial \mathbf{g^2}}{\partial \mathbf{x^2}}$: All derivatives are zero.

$\frac{\partial \mathbf{g^2}}{\partial \mathbf{u} \partial \mathbf{x}}$:All derivatives are zero.

E Costder

```
function [f,grad] = costder(x)

%This function gives the cost-function and the derivatives with respect to x

n=length(x)/3;  % x should be a multiple of 3

c1=0.0208; c2=-0.0278; c3=0.0650; c4=-0.0660;
y8=27.60; y9sp=26.50; y10sp=23.85;

y09=zeros(1,n); y010=zeros(1,n); h010=zeros(1,n);


load resultat1

h080(1)=h08(7);
h081(1)=h08(6);
h082(1)=h08(5);
h083(1)=h08(4);
h084(1)=h08(3);
h085(1)=h08(2);
h086(1)=h08(1);
h090(1)=h09(7);
h091(1)=h09(6);
h092(1)=h09(5);
h093(1)=h09(4);
h094(1)=h09(3);
h010(1)=h10(7);
y09(1)=y9(7);
y010(1)=y10(7);


for k=1:n-1;
  h080(k+1)=h080(k)+x(k);
  h081(k+1)=h080(k);
  h082(k+1)=h081(k);
  h083(k+1)=h082(k);
  h084(k+1)=h083(k);
  h085(k+1)=h084(k);
  h086(k+1)=h085(k);
  h090(k+1)=h090(k)+x(n+k);
  h091(k+1)=h090(k);
  h092(k+1)=h091(k);
  h093(k+1)=h092(k);
  h094(k+1)=h093(k);
  h010(k+1)=h010(k)+x(2*n+k);
  y09(k+1)=y09(k)+c1*h086(k)+c2*h090(k);
  y010(k+1)=y010(k)+c3*h094(k)+c4*h010(k);
end;
```

```
Q=diag([ones(1,3)]);
R=diag([ones(1,3)]);
Flagy9=zeros([1,n]);
Flagy10=zeros([1,n]);

for k=1:n;
y9hat(k)=y09(k)-y9sp;
y10hat(k)=y010(k)-y10sp;
h10hat(k)=h010(k);

%-----------------------------------------------------------------------------
%Penalty function
%-----------------------------------------------------------------------------

if y9hat(k) > 0.05
   y9prim(k)=(y9hat(k)-0.05);
   Flagy9(k)=1;
elseif y9hat(k) < -0.05
   y9prim(k)=(-0.05-y9hat(k));
   Flagy9(k)=-1;
else
   y9prim(k)=0;
   Flagy9(k)=0;
end

if y10hat(k) > 0.05
   y10prim(k)=(y10hat(k)-0.05);
   Flagy10(k)=1;
elseif y10hat(k) < -0.05
   y10prim(k)=(-0.05-y10hat(k));
   Flagy10(k)=-1;
else
   y10prim(k)=0;
   Flagy10(k)=0;
end


nbr1(k)=[y9prim(k); y10prim(k); h10hat(k)]'*Q*[y9prim(k); y10prim(k);
h10hat(k)];
nbr2(k)=[x(k); x(n+k); x(2*n+k)]'*R*[x(k); x(n+k); x(2*n+k)];

end;

f=0.5*sum(nbr1)+0.5*sum(nbr2);



if nargout>1
```

```
%=================================================================================
%The gradient
%=================================================================================

dery9dh1=zeros(n);
dery9dh2=zeros(n);
dery9dh3=zeros(n);
dery10dh1=zeros(n);
dery10dh2=zeros(n);
dery10dh3=zeros(n);
derh10dh1=zeros(n);
derh10dh2=zeros(n);
derh10dh3=zeros(n);


%_________________________________________________________________________________
% Part.der y9dh1
%---------------------------------------------------------------------------------
b=1;
while b<(n-8)
 for k=9:n;
   for j=1:b;
     dery9dh1(k,j)=Flagy9*(k-(7+j))*c1;
   end;
   b=b+1;
 end;
end;

%_________________________________________________________________________________
% Part.der y9dh2
%---------------------------------------------------------------------------------
b=1;
while b<(n-2)
 for k=3:n;
   for j=1:b;
     dery9dh2(k,j)=Flagy9*(k-j-1)*c2;
   end;
   b=b+1;
 end;
end;

%_________________________________________________________________________________
% Part.der y10dh2
%---------------------------------------------------------------------------------
b=1;
while b<(n-6)
 for k=7:n;
   for j=1:b;
     dery10dh2(k,j)=Flagy10*(k-(5+j))*c3;
   end;
   b=b+1;
 end;
end;

%_______________________________________________________________________________
% Part.der y10dh3
%-------------------------------------------------------------------------------
b=1;
while b<(n-2)
 for k=3:n;
   for j=1:b;
     dery10dh3(k,j)=Flagy10*(k-j-1)*c4;
   end;
   b=b+1;
 end;
end;

%_______________________________________________________________________________
% Par.der h010dh3
%-------------------------------------------------------------------------------
b=1;
while b<(n-1)
 for k=2:n;
   for j=1:b;
    derh10dh3(k,j)=1;
   end;
   b=b+1;
 end;
end;

%-------------------------------------------------------------------------------

derh10=diag(ones([1,n]),0);


for j=1:n
  for k=1:n
    p1(j,k)=y9prim(k)*Q(1,1)*dery9dh1(k,j);     %Der of y9 wrt dh1(j)
    p2(j,k)=y10prim(k)*Q(2,2)*dery10dh1(k,j);   %Der of y10 wrt dh1(j)
    p3(j,k)=h10hat(k)*Q(3,3)*derh10dh1(k,j);    %Der of h10 wrt dh1(j)

    p4(j,k)=y9prim(k)*Q(1,1)*dery9dh2(k,j);     %Der of y9 wry dh2(j)
    p5(j,k)=y10prim(k)*Q(2,2)*dery10dh2(k,j);   %Der of y10 wrt dh2(j)
    p6(j,k)=h10hat(k)*Q(3,3)*derh10dh2(k,j);    %Der of h10 wrt dh2(j)

    p7(j,k)=y9prim(k)*Q(1,1)*dery9dh3(k,j);     %Der of y9 wrt dh3(j)
    p8(j,k)=y10prim(k)*Q(2,2)*dery10dh3(k,j);   %Der of y10 wrt dh3(j)
    p9(j,k)=h10hat(k)*Q(3,3)*derh10dh3(k,j);    %Der of h10 wrt dh3(j)
    p10(j,k)=x(k)*R(1,1)*derh10(k,j);            %Der of dh1 wrt dh1(j)
    p11(j,k)=x(n+k)*R(2,2)*derh10(k,j);          %Der of dh2 wrt dh2(j)
    p12(j,k)=x(2*n+k)*R(3,3)*derh10(k,j);        %Der of dh3 wrt dh3(j)
  end;
end;



s1=sum(p1+p2+p3+p10,2);
s2=sum(p4+p5+p6+p11,2);
s3=sum(p7+p8+p9+p12,2);


grad=[s1' s2' s3'];

end;
```

F Nonlcon

```
function [c,ceq,GC,GCeq]=nonlcon(x)

n=length(x)/3;  % x should be a multiple of 3

c1=0.0208; c2=-0.0278; c3=0.0650; c4=-0.0660;

y8=27.60; y9sp=26.50; y10sp=23.85;

y9=zeros(1,n); y10=zeros(1,n); h10=zeros(1,n);

load resultat1

h080(1)=h08(7);
h081(1)=h08(6);
h082(1)=h08(5);
h083(1)=h08(4);
h084(1)=h08(3);
h085(1)=h08(2);
h086(1)=h08(1);
h090(1)=h09(7);
h091(1)=h09(6);
h092(1)=h09(5);
h093(1)=h09(4);
h094(1)=h09(3);
h010(1)=h10(7);
y9(1)=y9(7);
y10(1)=y10(7);

for k=1:n-1;
  h080(k+1)=h080(k)+x(k);
  h081(k+1)=h080(k);
  h082(k+1)=h081(k);
  h083(k+1)=h082(k);
  h084(k+1)=h083(k);
  h085(k+1)=h084(k);
  h086(k+1)=h085(k);
  h090(k+1)=h090(k)+x(n+k);
  h091(k+1)=h090(k);
  h092(k+1)=h091(k);
  h093(k+1)=h092(k);
  h094(k+1)=h093(k);
  h010(k+1)=h010(k)+x(2*n+k);
  y9(k+1)=y9(k)+c1*h086(k)+c2*h090(k);
  y10(k+1)=y10(k)+c3*h094(k)+c4*h010(k);
end;
```

```
%compute nonlinear inequalities
for k=1:n;
co1(k)=-(h080(k)-y8+27.693);                       %Constraint on p08
co2(k)=((h080(k)-y8+27.693)-2.005);
co3(k)=-(h090(k)-y9(k)+26.545);                    %Constraint on p09
co4(k)=((h090(k)-y9(k)+26.545)-1.474);
co5(k)=-(h010(k)-y10(k)+24.018);                   %Constraint on p10
co6(k)=((h010(k)-y10(k)+24.018)-1.487);
co7(k)=-h080(k);                                   %Constraint on h080
co8(k)=-h090(k);                                   %Constraint on h090
co9(k)=-h010(k);                                   %Constraint on h10
end


k=1:n;
c=[co1(k) co2(k) co3(k) co4(k) co5(k) co6(k) co7(k) co8(k) co9(k)];

ceq=[];

if nargout > 2

%compute derivatives of nonlinear inequalities

GC11=zeros([n,n]); GC12=zeros([n,n]); GC13=zeros([n,n]); GC21=zeros([n,n]);
GC22=zeros([n,n]); GC23=zeros([n,n]); GC31=zeros([n,n]); GC32=zeros([n,n]);
GC33=zeros([n,n]); GC41=zeros([n,n]); GC42=zeros([n,n]); GC43=zeros([n,n]);
GC51=zeros([n,n]); GC52=zeros([n,n]); GC53=zeros([n,n]); GC61=zeros([n,n]);
GC62=zeros([n,n]); GC63=zeros([n,n]); GC71=zeros([n,n]); GC72=zeros([n,n]);
GC73=zeros([n,n]); GC81=zeros([n,n]); GC82=zeros([n,n]); GC83=zeros([n,n]);
GC91=zeros([n,n]); GC92=zeros([n,n]); GC93=zeros([n,n]);

%compute derivatives of co1(k) wrt dh1

b=1;
while b<(n-1)
 for k=2:n;
   for j=1:b;
     GC11(k,j)=-1;
   end;
   b=b+1;
 end;
end;


%compute derivatives of co1(k) wrt dh2
GC12=zeros([n,n]);

%compute derivatives of co1(k) wrt dh3
GC13=zeros([n,n]);
```

```
GC1=[GC11 GC12 GC13];


%compute derivatives of co2(k) wrt dh1

b=1;
while b<(n-1)
 for k=2:n;
   for j=1:b;
     GC21(k,j)=1;
   end;
   b=b+1;
 end;
end;


%compute derivatives of co2(k) wrt dh2
GC22=zeros([n,n]);

%compute derivatives of co2(k) wrt dh3
GC23=zeros([n,n]);


GC2=[GC21 GC22 GC23];


%compute derivatives of co3(k) wrt dh1

b=1;
while b<(n-8)
 for k=9:n;
   for j=1:b;
      GC31(k,j)=(k-(7+j))*c1;
   end;
   b=b+1;
 end;
end;

%compute derivatives of co3(k) wrt dh2

b=1;
while b<(n-1)
 for k=2:n;
   for j=1:b;
     GC32(k,j)=-(1-(k-j-1)*c2);
   end;
   b=b+1;
 end;
end;
```

```
%compute derivative of co3(k) wrt dh3
GC33=zeros([n,n]);


GC3=[GC31 GC32 GC33];


%compute derivative of co4(k) wrt dh1
GC41=-GC31;

%compute derivative of co4(k) wrt dh2
GC42=-GC32;

%compute derivative of co4(k) wrt dh3
GC43=-GC33;


GC4=[GC41 GC42 GC43];


%compute derivative of co5(k) wrt dh1
GC51=zeros([n,n]);


%compute derivative of co5(k) wrt dh2
b=1;
while b<(n-6)
 for k=7:n;
   for j=1:b;
      GC52(k,j)=(k-(5+j))*c3;
   end;
   b=b+1;
 end;
end;

%compute derivative of co5(k) wrt dh3
b=1;
while b<(n-1)
 for k=2:n;
   for j=1:b;
     GC53(k,j)=-(1-(k-j-1)*c4);
   end;
   b=b+1;
 end;
end;


GC5=[GC51 GC52 GC53];
```

```
%compute derivative of co6(k) wrt dh1
GC61=-GC51;

%compute derivative of co6(k) wrt dh2
GC62=-GC52;

%compute derivative of co6(k) wrt dh3
GC63=-GC53;


GC6=[GC61 GC62 GC63];


%compute derivative of co7(k) wrt dh1

b=1;
while b<(n-1)
for k=2:n;
   for j=1:b;
     GC71(k,j)=-1;
   end;
   b=b+1;
 end;
end;

%compute derivative of co7(k) wrt dh2
GC72=zeros([n,n]);

%compute derivative of co7(k) wrt dh3
GC73=zeros([n,n]);


GC7=[GC71 GC72 GC73];


%compute derivative of co8(k) wrt dh1
GC81=GC72;

%compute derivative of co8(k) wrt dh2
GC82=GC71;

%compute derivative of co8(k) wrt dh3
GC83=GC73;


GC8=[GC81 GC82 GC83];


%compute derivative of co9(k) wrt dh1
```

```
GC91=GC72;

%compute derivative of co9(k) wrt dh2
GC92=GC72;

%compute derivative of co9(k) wrt dh3
GC93=GC71;


GC9=[GC91 GC92 GC93];


GC=[GC1' GC2' GC3' GC4' GC5' GC6' GC7' GC8' GC9'];


GCeq=[];

end;
```

G LQR-test

```
function [u,x0,K] = lqrtest(n)


y8=27.60; y9sp=26.50; y10sp=23.85;

load resultat1

x=[h08(7) h08(6) h08(5) h08(4) h08(3) h08(2) h08(1) h09(7) h09(6) h09(5)
h09(4) h09(3) h10(7) y9(7)-y9sp y10(7)-y10sp]';

Q=diag([zeros(1,12) ones(1,3)]);
R=diag([ones(1,3)]);



c1=0.0208; c2=-0.0278; c3=0.0650; c4=-0.0660;

A=[1 0 0 0 0 0 0 0 0 0 0 0 0 0 0;
   1 0 0 0 0 0 0 0 0 0 0 0 0 0 0;
   0 1 0 0 0 0 0 0 0 0 0 0 0 0 0;
   0 0 1 0 0 0 0 0 0 0 0 0 0 0 0;
   0 0 0 1 0 0 0 0 0 0 0 0 0 0 0;
   0 0 0 0 1 0 0 0 0 0 0 0 0 0 0;
   0 0 0 0 0 1 0 0 0 0 0 0 0 0 0;
   0 0 0 0 0 0 0 1 0 0 0 0 0 0 0;
   0 0 0 0 0 0 0 1 0 0 0 0 0 0 0;
   0 0 0 0 0 0 0 0 1 0 0 0 0 0 0;
   0 0 0 0 0 0 0 0 0 1 0 0 0 0 0;
   0 0 0 0 0 0 0 0 0 0 1 0 0 0 0;
   0 0 0 0 0 0 0 0 0 0 0 0 1 0 0;
   0 0 0 0 0 0 c1 c2 0 0 0 0 0 1 0;
   0 0 0 0 0 0 0 0 0 0 0 c3 c4 0 1];

B=[1 0 0;
   0 0 0;
   0 0 0;
   0 0 0;
   0 0 0;
   0 0 0;
   0 0 0;
   0 1 0;
   0 0 0;
   0 0 0;
   0 0 0;
   0 0 0;
   0 0 1;
   0 0 0;
   0 0 0];
```

```
N=zeros([15,3]);

[K,S,E] = dlqr(A,B,Q,R,N);

u=-K*x;

x0=[u(1)*ones(1,n) u(2)*ones(1,n) u(3)*ones(1,n)];
```

References

[1] Leonard M. Cantor *A World Geography of Irrigation* Oliver and Boyd, 1967

[2] Herbert Addison *Land, Water and Food* Chapman & Hall LTD, 1955

[3] Murray River *www.schools.ash.org.au/elanorah/snriver.htm*

[4] Sandra Postel and Linda Starke. *Last Oasis: Facing Water Scarcity* Worldwatch Institute, June 1997

[5] Rubicon Systems Australia. *www.rubicon.com.au*

[6] Erik Weyer. *System Identification of an Open Water Channel* Presented at IFAC Symposium on System Identification, Santa Barbara, California, June 2000

[7] Su Ki Ooi and Erik Weyer *Closed Loop Identification of An Irrigation Channel* August 31, 2000

[8] Erik Weyer. *Analysis of September data from the Haughton Main Channel* March 24, 2000

[9] Erik Weyer, I. Mareels, J. Fenton, D. Aughton. *Advanced Management and Control of Water Networks*

[10] Carlos E. Garcia, David M. Prett, Manfred Morari. *Model Predictive Control: Theory and Practice-a Survey* Automatica, Vol 25, No. 3, 1989

[11] Karl J. Åström, Björn Wittenmark. *Computer Controlled Systems* Prentice Hall 1997

[12] James Wettenhall. *Computational Algorithms for Constrained Nonlinear Optimal Control* 1999

[13] Thomas Coleman, Mary Ann Branch, Andrew Grace. *Optimization Toolbox, MATLAB* The MathsWorks, Inc. 1999

[14] The MathsWork, Inc. *Simulink-Dynamic System Simulation for MATLAB* The Mathswork, Inc. 1999

[15] The MathsWork, Inc. *MATLAB The Language of Technical Computing* The Mathswork, Inc. 1998

[16] Boris J. Lurie, Paul J. Enright *Classical Feedback Control with MATLAB* Marcel Dekker, Inc. 2000

[17] Edoardo Mosca *Optimal, predictive, and adaptive control* Prentice Hall, 1995

[18] Counge, J.A., F.M. Holly, and A. Verwey *Practical aspects of Computational River Hydraulics* 1980