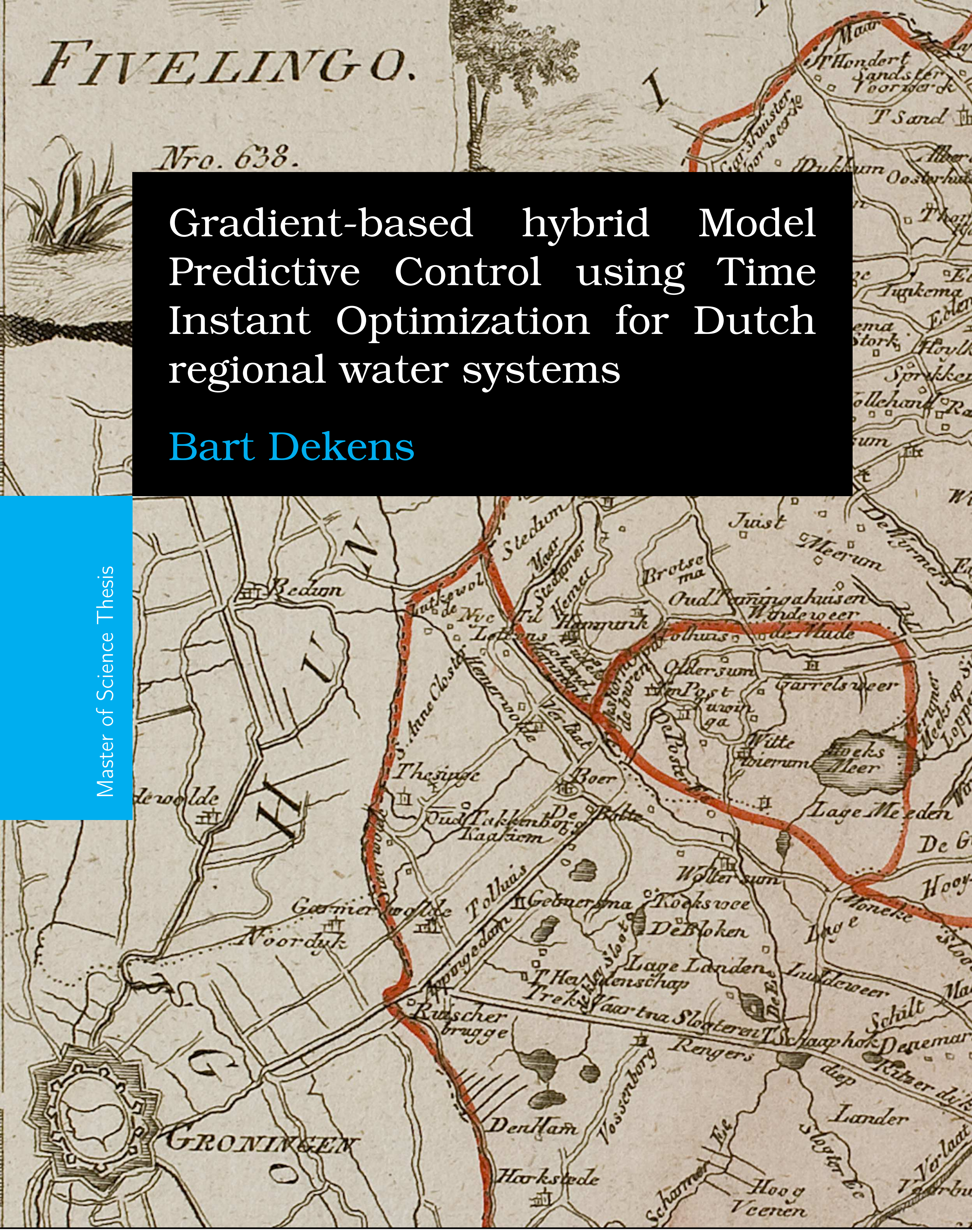


Gradient-based hybrid Model Predictive Control using Time Instant Optimization for Dutch regional water systems

Bart Dekens

Master of Science Thesis

TU Delft Delft University of Technology

Section of Water Resources Management

Gradient-based hybrid Model Predictive Control using Time Instant Optimization for Dutch regional water systems

Master of Science Thesis

For the degree of Master of Science in Water Resources Management at Delft University of Technology

Bart Dekens

June 18, 2013

Faculty of Civil Engineering and Geosciences (CEG) · Delft University of Technology

The work in this thesis was supported by Deltares. Their cooperation is hereby gratefully acknowledged.

This thesis was written using LaTeX.

Cover picture: excerpt from old map of Fivelingo, dated around 1791. Source: F.J.J. von Reilly, Schauplatz der fünf Theile der Welt. 3. Bd. Wien s.a. http://irs.ub.rug.nl/ppn/120296985

Delft University of Technology
Section of
Water Resources Management

The undersigned hereby certify that they have read and recommend to the Faculty of Civil Engineering and Geosciences (CEG) for acceptance a thesis entitled

Gradient-based hybrid Model Predictive Control using Time Instant Optimization for Dutch regional water systems

by

Bart Dekens

in partial fulfillment of the requirements for the degree of

Master of Science Water Resources Management

Dated: June 18, 2013

Supervisor(s):

prof.dr.ir. N.C. van de Giesen

dr.ir. A.D. Sadowska

Reader(s):

dr.ir. P.J.A.T.M. van Overloop

dr.ing. D. Schwanenberg

Abstract

Many areas in the Netherlands can be characterized as low-lying polder systems. In order to keep our feet dry, a lot of effort is put into ensuring the safety of the physical structures that protect us from flooding, such as dams and dikes. To control the water quantity and quality within the polders, hydraulic structures such as pumps and gates are in place. These can be operated to meet different requirements. Some of these structures are operated manually, but often the control has been automated. The operation of such structures is in many cases done by rule-based (if-then) operators, which base their control actions on a comparison of the current state (e.g. water levels) with the desired state. The field of operational water management aims at optimizing the control of these automated structures.

Model Predictive Control (MPC) is an anticipatory control technique that originated in the process industries, and found its way into water management over the last one or two decades. This methodology uses a mathematical model of the controlled process and forecasts of future process states and external disturbances to determine the optimal sequence of control actions over a finite prediction horizon. The trade-off between different, often conflicting objectives of a controller is described mathematically in an objective function. At every time step, an optimization problem has to be solved by minimizing the objective function subject to constraints. Only the first control step is implemented, and the optimization starts again with updated measurements and predictions at the new time step.

This thesis focusses on Dutch regional water systems. These systems are often low-lying polder-belt canal systems, where many pumps are needed to meet different requirements regarding water quantity and quality. The control of a complex water system, consisting of continuous dynamics (evolution of water flow and levels) and discrete elements (e.g. barriers and pumps that are operated on or off) can be optimized using a so-called hybrid Model Predictive Controller. However, particularly the optimization of the combination of discrete and continuous elements requires extensive computational effort. Even with the ongoing increase in computational power, computational time remains an issue for the optimization of large hybrid systems in real-time control applications.

Time Instant Optimization MPC has been proposed in literature as an alternative to the computationally more demanding Mixed Logical Dynamical models. TIO-MPC involves the

optimization of a (a priori determined) number of time instants, which are the moments that a discrete variable changes its state. The rationale behind this approach is that in many cases, it is undesirable to have too many switching of controllers. This significantly reduces the amount of optimization variables, as the controller does not have to decide at every time step whether or not to switch the state of discrete variables. The latter would normally lead to a large combinatorial optimization problem.

The contribution of this thesis is the extension of the current TIO-MPC in such a way that the time instants become continuous. This way, the gradient of the objective function can be derived. The gradient allows the use of efficient gradient-based solvers, which require a gradient vector of the objective function for finding an optimum. Furthermore, hybrid schemes including the optimization of time instants and standard MPC can easily be integrated.

The analytical gradient of the objective function is derived by applying algorithmic differentiation in reverse mode. This way, the gradient of the objective function can be derived to machine precision at the computational costs of a single function evaluation. Another benefit of the use of algorithmic differentiation over finite differencing is the absence of a truncation error.

Multiple TIO-MPC algorithms have been designed, and their performance has been tested using two test cases. Different performance indicators are defined and used to compare the results. The first test case involves the closed-loop simulation of a fictitious linear reservoir with one discrete control variable (pump) using two time instants. The second test case involves the pump scheduling of two continuous pumps, two discrete pumps and one a gate on a model of the Fivelingo boezem, an existing water system in the province of Groningen in the north of the Netherlands. For the latter, only open-loop simulations have been carried out using algorithms with a different number of time instants. For comparison, one continuous optimization was done.

The gradient-based TIO-MPC algorithms are perfectly capable of optimizing both continuous and discrete elements, well within the allowed control time step. The computational time is correlated with the amount of iterations that is needed to converge to a solution. Therefore it can not be said in advance which scheme is the quickest.

A comparison of the different TIO-MPC algorithms used for the Fivelingo test case demonstrated that the use of more time instants generally leads to lower objective function values, indicating better control performance. The experiments also showed that the solution is likely to get stuck in (suboptimal) local minima if a user-supplied initial guess of the time instants is not given. Supplying a good initial guess, or using a multi-start optimization procedure will potentially overcome this problem. The latter option is more realistic if one cannot come up with a good initial guess.

Contents

List of Figures

List of Tables

Acknowledgements

The report presented here is the result of my MSc thesis, which started in September 2012 at Delft University of Technology and Deltares. It were nine very interesting months, in which I learned a lot. I would like to thank a few people for their help and support.

First of all, I would like to thank my daily supervisor Anna Sadowska for her patient guidance, enthusiastic encouragement, comments, sound advice and 'gezelligheid' throughout the process. Thanks for helping me to get familiar with the subject and being picky on my writing. I learned a lot about control theory, modelling and writing in LaTeX. Thanks to professor Nick van de Giesen for allowing me to work on a project outside the Civil Engineering faculty. Thanks to Peter-Jules van Overloop for his personal approach and guidance. Your enthusiasm and trust convinced me to pursue a thesis in operational water management while lying in a hammock in southern France (they have Wi-Fi everywhere these days...). You were right that modelling can be darn frustrating at times, but also very satisfying and even fun when things start to work out. Thanks to Dirk Schwanenberg for introducing me into both the subject and into the Deltares community. Your advice and assistance has been of great help. Thanks to the guys at the *control room*, you made me feel welcome at Deltares.

Finally, I wish to thank my girlfriend Marlou and my family for their unconditional support and encouragement throughout my study.

I would like to dedicate this thesis to my mother who didn't live to see me finish.

Bart Dekens
June 18, 2013

"I don't think there's a punch-line scheduled, is there?"
— *Monty Python*

Chapter 1

Introduction

1.1 Background

Water is an essential resource for life. People use water for consumption, sanitation, navigation, agriculture, hydropower and leisure. Throughout history, people have always tended to live close to water resources. However, water can also be experienced as a burden. Especially in low-lying polder systems, floods can occur. In order to manage the water around them, societies have installed infrastructure, such as gates and pumps that can be adjusted according to their objectives, e.g. deliver water for agriculture or protect the land against flooding. This can be done by operating the structures that influence the water levels and flows. These structures are commonly operated manually by operators, or automatically with rule based controllers. These controllers determine whether a control action should be taken based on the current state of the system.

The field of operational water management focuses on optimizing control actions of these structures. It not only deals with the mitigation of extreme events, but also with complex, interdependent and often conflicting water quality and quantity objectives. Another important goal in e.g. polder systems, where surplus water has to be discharged by pumps, is increasing the energy efficiency [1]. With rising energy prices, water authorities are interested in saving energy by operating the existing infrastructure in a more intelligent way.

A common rule based control algorithm is feedback control, which compares the current state (e.g. water level) with the desired output or *setpoint* of the system, and decides what control action should be taken to get the state back to the desired output. Feedback control algorithms react on external disturbances when they are measured, and thus the algorithm is not able to anticipate on predicted or measured disturbances.

With feedforward control, the control action is based on a predicted or measured disturbance. The control algorithm is able to anticipate on disturbances, for instance by controlling a

[1] In the Netherlands alone, the combined energy consumption of all water authorities is equal to that of a city with 200 thousands inhabitants (Rivierenland, 2013)

water level by pre-releasing water if a certain incoming discharge is measured. Because the controller only reacts to known disturbances, uncertainty of predictions and actual system behaviour influences the performance. To compensate for this, feedforward is often combined with feedback control. If the disturbance e.g. due to rainfall is higher than the control capacity of the actuator (e.g. a pump), the output will deviate from setpoint.

A control method that applies the concepts of feedback and feedforward control while also taking constraints into account, is called Model Predictive Control (MPC), an anticipatory control methodology that originated in the process industries. This methodology uses a model of the controlled process and forecasts of future process states and external disturbances to determine the optimal sequence of control actions by using an optimization algorithm, taking into account physical and operational constraints. The constraints can either be hard constraints (e.g. maximum pump capacities) or soft constraints (e.g. water levels). The trade-off between different, often conflicting objectives of a controller, is described mathematically in an objective function. At every time step, an optimization problem has to be solved by minimizing the objective function subject to constraints. The control of a complex water system, consisting of continuous dynamics (evolution of water flow and levels) and discrete elements (e.g. barriers and pumps that are operated on or off) can be optimized using a MPC controller.

Model Predictive Control has gained increasing attention in the field of hydrology and water management over the last years. Some controllers have successfully been implemented in water systems (Van Overloop, 2006; Blanco et al., 2010). However there are some drawbacks to the methodology when discrete actuators such as barriers are present, especially concerning the computational effort. Continuous and discrete elements are not easily combined in MPC.

To reduce the computational effort, TIO-MPC has been proposed (Van Ekeren, 2010), where TIO stands for Time Instant Optimization. This methodology consists of an algorithm that optimizes discrete time instants, e.g. when a pump should be switched on and off. The advantage of this method is that the computational effort is reduced because instead of deciding at each time step whether to switch the pump on or off, the decision making process involves optimizing the discrete moments at which control actions should take place. Therefore, the number of optimization variables can be significantly reduced.

1.2 Motivation

The method described in (Van Ekeren, 2010) is able to find a (sub)optimal solution, but it needs many evaluations of solutions and a, a priori unknown, large set of initial solutions to do this. The method uses a multi-start pattern search optimization algorithm to converge to a solution of this combinatorial optimization problem, without the use of a gradient search. This makes the optimization slow.

For this thesis, the idea is to extend TIO-MPC in the way that the time instances become continuous. This way, the gradient of the objective function can be calculated and efficient gradient-based optimizers can be used. This will decrease the computational effort of the TIO-MPC controller. Furthermore, hybrid schemes including the optimization of time instants and standard MPC can be integrated.

Gradient-based solvers using sequential quadratic programming (SQP) or integer programming (IP) algorithms require a gradient vector of the objective function for efficient performance. These gradients can be calculated by means of finite differences, but this requires many function evaluations. The method becomes computationally inefficient for problems with hundreds of dimensions, and hence disqualifies itself for being used within an operational setting. An efficient method to derive the derivative of the objective function at computational costs in the order of a single model execution is the setup of an adjoint model for each component by applying algorithmic differentiation.

1.3 Research objectives

The aim of this research is to develop gradient based TIO-MPC schemes for the control of gates and pumps in Dutch regional water systems. The hybrid TIO-MPC schemes should be able to optimize both continuous and discrete control input, using efficient gradient-based solvers. The research contains the following aspects:

- Theoretical assessment of novel TIO-MPC approach.
- The extension of the existing TIO-MPC controller in the way that the time instants become continuous.
- The design and application of this controller on a simple academic test case, in order to test the theory and gain knowledge on the functioning of TIO-MPC.
- The setup of an adjoint model for the academic test case by applying algorithmic differentiation in reverse mode, which yields the gradient of the objective function.
- Application and performance testing of various hybrid TIO-MPC algorithms on a relevant test case. The test case focusses on pump scheduling in the Fivelingo Boezem, a typical Dutch polder-belt canal water system that is managed by the regional water authority Noorderzijlvest.

Chapter 2

Real-time control of hydraulic structures

This chapter will deal with the modelling of open channel flow, and flow through hydraulic structures. Further, the discretization of these model is discussed and different types of MPC are introduced.

2.1 Modelling of open channel flow

Many places in the Netherlands are polders, which are low-lying grounds surrounded by dikes. Brooks and ditches collect water, which is then transported through a network of canals. From these canals, surplus water needs to be pumped out into what is called a *boezem canal*, which conveys water to e.g. the sea. Figure 2-1 shows the typical layout of a polder. Many of the channels are man-made, with a prismatic shape.

The flow-governing equations describing one-dimensional gradually varying non-steady flow in prismatic channels are the dynamic wave equations, also referred to as the De Saint-Venant (SV) equations or shallow-water equations. The SV equations are coupled nonlinear hyperbolic partial differential equations which are derived from equations of conservation of mass (continuity) and momentum, respectively. The equations of De Saint-Venant are (Chow et al., 1988):

Continuity equation:

$$\frac{\partial Q}{\partial x} + \frac{\partial A}{\partial t} = q_{\text{lat}}. \tag{2.1a}$$

Momentum equation[1]*:*

$$\underbrace{\frac{\partial v}{\partial t}}_{\substack{\text{Local}\\ \text{acceleration}}} + \underbrace{v\frac{\partial v}{\partial x}}_{\substack{\text{Convective}\\ \text{acceleration}}} + \underbrace{g\frac{\partial y}{\partial x}}_{\substack{\text{Pressure}\\ \text{force}}} + \underbrace{gS_{\text{f}}}_{\substack{\text{Friction}\\ \text{force}}} - \underbrace{gS_0}_{\substack{\text{Gravity}\\ \text{force}}} = 0, \tag{2.1b}$$

[1] written in the non-conservation form, see (Chow et al., 1988)

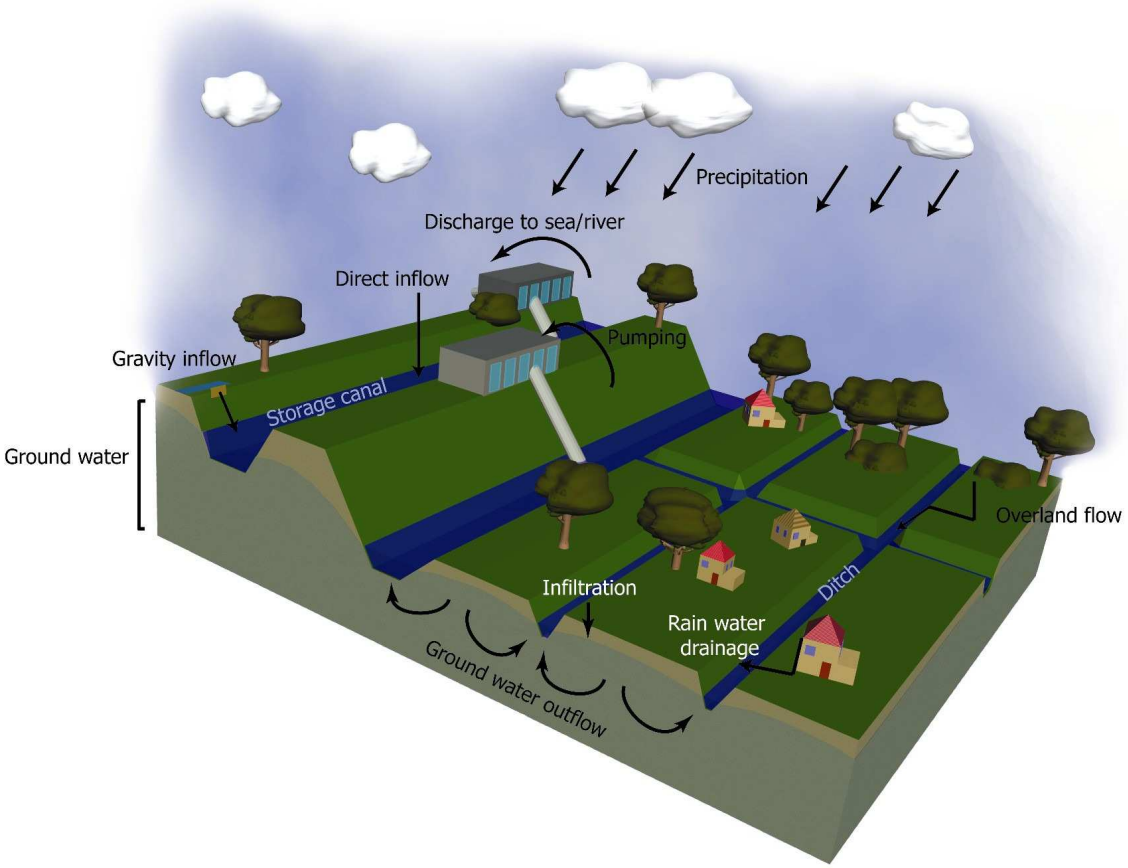


Figure 2-1: Schematization of a typical polder system (Van Overloop et al., 2010a)

.

where Q is the flow [$\mathrm{m^3\,s^{-1}}$], A is the average cross-sectional area of flow [$\mathrm{m^2}$], q_{lat} is the lateral inflow per unit length [$\mathrm{m^2\,s^{-1}}$], y is the water depth [m], v is the mean water velocity [$\mathrm{m\,s^{-1}}$], g is the acceleration due to gravity [$\mathrm{m\,s^{-2}}$], S_{f} is the friction slope (energy gradient) [$\mathrm{m\,m^{-1}}$], S_0 is the bed slope [$\mathrm{m\,m^{-1}}$], t is the time [s] and x is the longitudinal distance [m].

The longitudinal profile and cross-section of a canal reach are schematized in Figure 2-2, in which z_{b} is the elevation of the channel bed above reference level [m] and $S_0 = -\partial z_{\mathrm{b}}/\partial x$ is the bed slope [$\mathrm{m\,m^{-1}}$].

We can also write the momentum equation in terms of the water level $h = y + z_{\mathrm{b}}$, where h is the water level above reference level [m]. This is done by substituting $y = h - z_{\mathrm{b}}$ into (2.1b) and using $S_0 = -\partial z_{\mathrm{b}}/\partial x$. This way we can obtain:

$$\frac{\partial v}{\partial t} + v\,\frac{\partial v}{\partial x} + g\,\frac{\partial h}{\partial x} + gS_{\mathrm{f}} = 0. \tag{2.2}$$

The velocity of the flow in a prismatic canal with an arbitrary cross-section, subject to resistance due to channel bed friction, is often represented by the Chézy equation. In the Chézy equation, the friction resistance is proportional to the square of the mean velocity. Under

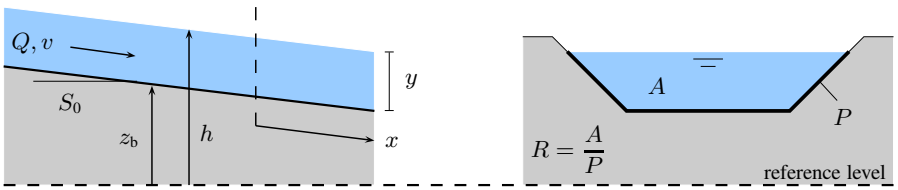


Figure 2-2: Schematization of a canal reach

uniform flow conditions, the Chézy equation reads:

$$v = C\sqrt{RS_\mathrm{f}}, \tag{2.3}$$

where C is the Chézy roughness coefficient [$\mathrm{m}^{1/2}\,\mathrm{s}^{-1}$], which depends on the roughness of the channel and the water depth and is assumed to be valid for non-steady flow (Shaw, 1994) and R [m] is the hydraulic radius, which is given by the wetted area A [m^2] divided by the wetted perimeter P [m]. Substitution of (2.3) into (2.2) yields:

$$\frac{\partial v}{\partial t} + v\,\frac{\partial v}{\partial x} + g\,\frac{\partial h}{\partial x} + g\frac{v|v|}{C^2 R} = 0. \tag{2.4}$$

Unfortunately there is no known analytical solution of the De Saint-Venant equations in real geometry (Malaterre and Baume, 1998) so the system of equations has to be solved numerically. Depending on the characteristics of the flow and the required accuracy, different one-dimensional distributed flood routing equations can be derived by using the full continuity equations (sometimes neglecting lateral inflow) while neglecting some terms of the momentum equation:

- The *dynamic wave model* considers all terms of the momentum equation. It is the most detailed SV-based flood routing model.
- The *kinematic wave model* neglects local acceleration, convective acceleration and the pressure term of the momentum equation.
- The *diffusive wave model* neglects the local and convective acceleration terms of the momentum equation.

The dynamic wave model is computationally heavy due to its high complexity and therefore not well suited for real-time control purposes. The kinematic wave model is the simplest distributed model, which can be used for gradually varied unsteady flows when the inertial and pressure forces are not important to the movement of the wave (MacArthur and DeVries, 1993). The terms for local acceleration, convective acceleration and the pressure term in the momentum equation (2.1b) can be neglected and the weight component (acceleration due to gravity) is then balanced by the resistance due to bed friction. Hence, the flow does not accelerate appreciably and the bed slope is approximately equal to the friction slope (Chow et al., 1988) and the momentum equation (2.1b) simplifies to:

$$S_0 = S_\mathrm{f}. \tag{2.5}$$

The kinematic wave model is a stage-discharge relationship that is applicable when there are no appreciable backwater effects, which makes the model useful for channels with steep gradients.

The diffusive wave model neglects the local and convective acceleration terms of the momentum equation. Unlike the kinematic wave model, it does include the pressure term of the momentum equation in (2.4). By substitution of $v = Q/A$ it follows that:

$$g\,\frac{\partial h}{\partial x} = -\frac{gQ|Q|}{C^2A^2R}, \tag{2.6}$$

and this can be rewritten to obtain:

$$Q = -\text{sign}\left(\frac{\partial h}{\partial x}\right) CA\sqrt{\left|\frac{\partial h}{\partial x}\right| R}. \tag{2.7}$$

Compared to the kinematic wave model, the diffusive wave model is able to capture more of the dynamics that occur in the water system. The dynamic wave model is preferred over the kinematic wave model for very flat canal reaches, where backwater effects might play a role.

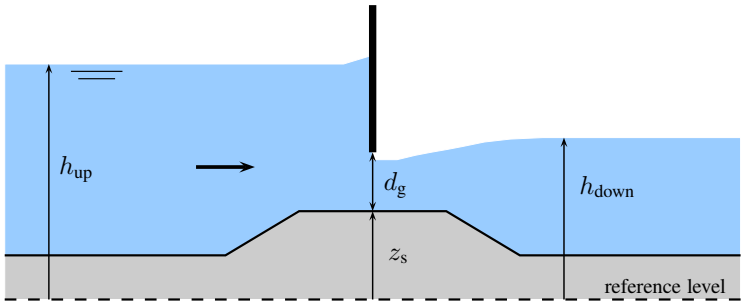
2.2 Modelling of hydraulic structures

The general equation describing the flow through a vertical gate or *orifice* can be derived from the Bernoulli equation (Shaw, 1994), which is based on the principle of the conservation of energy. Depending on the dimensions and settings of the gate and the flow conditions we can distinguish between free orifice flow, submerged orifice flow, free weir flow and submerged weir flow, which are all described by different equations. In case of free flow, the flow is (unlike during submerged flow) not dependent of the downstream water level (Lewin, 2001, p. 157). Weir flow occurs when the water level does not touch the vertical gate. See Figure 2-3 for a schematization of the flow through a vertical gate with free flow and submerged flow.

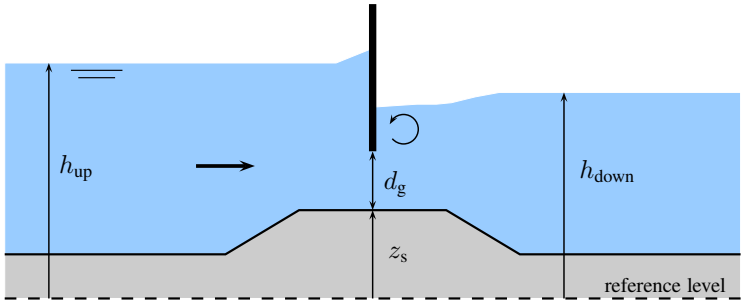
When assuming positive flow (upstream water level is higher than downstream water level), the flow equations are, for free flow and submerged flow respectively (Ankum, 2002; Schwanenberg and Becker, 2012):

$$\text{Orifice flow:}\quad Q = \begin{cases} c_\text{w} w_\text{s} \mu d_\text{g} \sqrt{2g(h_\text{up} - z_\text{s} - \mu d_\text{g})}, & \text{if} \quad h_\text{up} - z_\text{s} \geq \dfrac{3}{2} d_\text{g} \\ & \text{and} \quad h_\text{down} \leq z_\text{s} + d_\text{g}; \\ c_\text{w} w_\text{s} \mu d_\text{g} \sqrt{2g(h_\text{up} - h_\text{down})}, & \text{if} \quad h_\text{up} - z_\text{s} \geq \dfrac{3}{2} d_\text{g} \\ & \text{and} \quad h_\text{down} > z_\text{s} + d_\text{g}. \end{cases} \tag{2.8}$$

$$\text{Weir flow:}\quad Q = \begin{cases} c_\text{w} w_\text{s} \dfrac{2}{3}\sqrt{\dfrac{2}{3}g}(h_\text{up} - z_\text{s})^{\frac{3}{2}}, & \text{if} \quad h_\text{up} - z_\text{s} < \dfrac{3}{2} d_\text{g} \\ & \text{and} \quad h_\text{up} - z_\text{s} > \dfrac{3}{2}(h_\text{down} - z_\text{s}); \\ c_\text{e} c_\text{w} w_\text{s}(h_\text{down} - z_\text{s} - \dfrac{u_\text{s}^2}{2g}) & \text{if} \quad h_\text{up} - z_\text{s} < \dfrac{3}{2} d_\text{g} \\ \quad \times \sqrt{2g(h_\text{up} - h_\text{down})}, & \text{and} \quad h_\text{up} - z_\text{s} \leq \dfrac{3}{2}(h_\text{down} - z_\text{s}), \end{cases} \tag{2.9}$$



(a) Free flow



(b) Submerged flow

Figure 2-3: Schematization of a vertical gate with orifice flow

where c_w is the lateral contraction coefficient [−], c_e is the discharge coefficient [−], w_s is the crest width [m], μ is the contraction coefficient [−], g is the gravitational acceleration [$m\,s^{-2}$], u_s is the mean flow velocity on top of the crest [$m\,s^{-1}$], z_s is the crest level [m above datum], d_g is the opening height of the gate [m] and h_{up} and h_{down} are the upstream and downstream water level [m above datum], respectively.

The orifice equations can be simplified by assuming a low flow velocity upstream of the structure. This assumption holds when the wetted area A of the upstream canal is large and/or the flow is low. When the flow velocity is low, the term for the velocity head ($\frac{u_s^2}{2g}$) in (2.9) can be neglected. In that case, the water level is equal to the hydraulic head (the sum of the elevation head and the pressure head) and the discharge is a function of the gate settings and the upstream and downstream water levels:

$$Q^{k+1} = f(h_{up}^k, h_{down}^k, d_g^{k+1}). \tag{2.10}$$

The gate setting d_g cannot be written as an explicit function of the discharge and has to be solved iteratively.

2.3 Discretization of the hydraulic model

Since there is no analytical solution of the De Saint-Venant (SV) equations in real geometry (Malaterre and Baume, 1998), the system of equations has to be solved numerically. This is done by discretizing the partial differential equations of the SV equations in time (Δt) and

Table 2-1: Some general properties of the θ-method

	Scheme	Truncation error	Stability
$\theta = 0$	Forward (Explicit) Euler	$\mathcal{O}(\Delta t, \Delta x^2)$	conditionally stable
$\theta = \frac{1}{2}$	Crank-Nicolson	$\mathcal{O}(\Delta t^2, \Delta x^2)$	unconditionally stable
$\theta = 1$	Backward (Implicit) Euler	$\mathcal{O}(\Delta t, \Delta x^2)$	unconditionally stable

space (Δx). There are numerous numerical 'recipes' which all have different properties in terms of accuracy, numerical stability and computational efficiency. Staggered grids are often being used for the spatial discretization of the SV equations (Stelling and Duinmeijer, 2003; Van Overloop, 2006).

The diffusive wave model can also be spatially discretized on a staggered grid, where the flow is schematized in branches between upstream and downstream storage nodes (Schwanenberg et al., 2011). The water levels in the storage nodes are calculated from the continuity equation, while the flow in the branches that connect the storage nodes is described by the diffusive wave model (2.7). If we define the distance between the storage nodes as Δx, we can rewrite (2.7) into a function of the upstream and downstream water levels h_{up} and h_{down} by applying central differences:

$$\begin{aligned} Q^{k+1} &= f(h^k_{\text{up}}, h^k_{\text{down}}) \\ &= -\text{sign}(\frac{h^k_{\text{up}} - h^k_{\text{down}}}{\Delta x}) C(\overline{h}^{\,k}) A(\overline{h}^{\,k}) \sqrt{\left| \frac{h^k_{\text{up}} - h^k_{\text{down}}}{\Delta x} \right| R(\overline{h}^{\,k})}, \end{aligned} \tag{2.11}$$

where the variables C, A and R are functions of the mean water level $\overline{h}^{\,k} = \frac{h^k_{\text{up}} + h^k_{\text{down}}}{2}$ [m] in a representative cross-section between storage nodes.

A numerical approximation of the continuity equation (2.1a) can be obtained by applying the well known θ-method (Zijlema, 2011; Stelling and Duinmeijer, 2003) on a staggered grid. The θ-method is a finite difference discretization of the time-derivative, where Δt is defined as the time step and $\theta \in [0, 1]$ is a weighting parameter indicating the "implicitness" of the scheme.

$$\frac{y^{k+1} - y^k}{\Delta t} = \theta f(y^{k+1}) + (1 - \theta) f(y^k). \tag{2.12}$$

The resulting scheme is fully explicit when $\theta = 0$, semi-implicit when $0 < \theta < 1$ and fully implicit when $\theta = 1$. Table 2-1 summarizes the truncation errors of the different schemes of the θ-method (Zijlema, 2011). When applied to the continuity equation, it reads:

$$\frac{A(h^{k+1}) - A(h^k)}{\Delta t} + \theta \frac{Q^{k+1}_{\text{down}} - Q^{k+1}_{\text{up}}}{\Delta x} + (1 - \theta) \frac{Q^k_{\text{down}} - Q^k_{\text{up}}}{\Delta x} = \theta q^{k+1}_{\text{lat}} + (1 - \theta) q^k_{\text{lat}}, \tag{2.13}$$

where Q_{up} and Q_{down} are the discharges [m^3s^{-1}] of the upstream and downstream nodes connected to the branch, Δt is the time between two discrete time steps [s] and $\theta \in [0, 1]$ is a weighting parameter indicating the implicitness of the solution.

A necessary condition for the convergence of an explicit finite-difference scheme to a set of (hyperbolic) partial differential equations, is the Courant-Friedrichs-Lewy (CFL) stability cri-

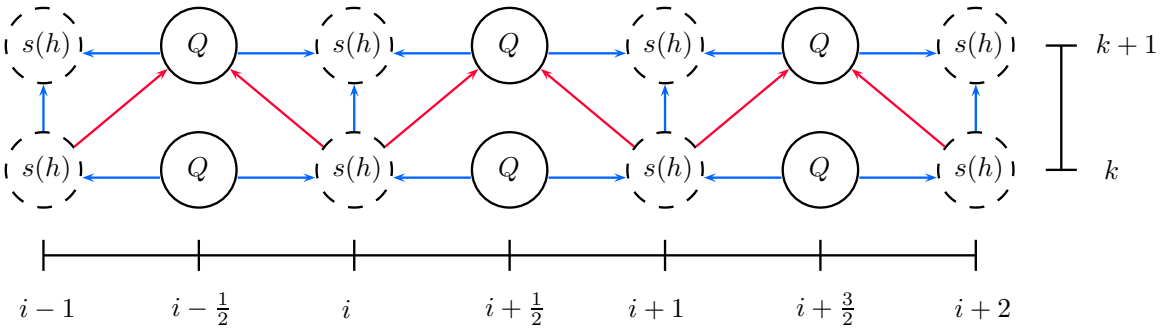


Figure 2-4: Schematization of the internal MPC model on a staggered grid. The vertical and horizontal axes represent the discretization in time and space, respectively. The arrows indicate the flow of information, where the blue arrows reprents the continuity equation and the red arrows the diffusive wave model.

terion, which for a one-dimensional case reads (Press et al., 2007):

$$\sigma \equiv \frac{|v|\Delta t}{\Delta x} \leq C, \tag{2.14}$$

where σ is the Courant number [$-$], v is the flow velocity [$\mathrm{m^3\,s^{-1}}$], equal to Q/A, Δt is the temporal step size [s], Δx is the spatial step size [m] and C is a dimensionless constant which depends on the numerical scheme that is being considered. A disadvantage of the Euler Forward Scheme is that the CFL-condition restricts the size of the time step Δt, and as a consequence, more time steps are needed. This causes an increase in computation time.

The combination of Explicit Euler with central differencing (in space) is inappropriate as this combination is known to form a unconditionally unstable scheme, unless there is some damping involved (Zijlema, 2011). By using the (implicit) Euler Backward scheme ($\theta = 1$), equations (2.10) and (2.11), multiplying by Δx and substitution of $s(h) = A(h)\Delta x$, the continuity equation (2.13) can be written as a water balance in the domain of a node (Schwanenberg and Becker, 2012):

$$s^{k+1} = s^k + \Delta t(Q_{\mathrm{up}}^{k+1} - Q_{\mathrm{down}}^{k+1} + Q_{\mathrm{lat}}^{k+1}), \tag{2.15}$$

where s^k is the storage [$\mathrm{m^3}$] at the node and Q_{lat}^{k+1} is the aggregated lateral inflow [$\mathrm{m^3\,s^{-1}}$] flowing into the domain of the node, and Q_{up}^{k+1} and Q_{down}^{k+1} are the upstream and downstream discharge at the intermediate reaches connected to the node and k is a time step counter. All nodes require a level-storage relationship to adequately link the storage to the water level. Figure 2-4 shows the schematization of the model on a staggered grid for two subsequent time steps, where the storage and discharge are evaluated at alternating storage points and discharge points. The storage points have whole indices and the discharge points have half indices.

2.4 Model Predictive Control (MPC)

Model Predictive Control (MPC) (Maciejowski, 2002; Wang, 2009) is an advanced deterministic control technique that was developed in the process industry in the seventies. This

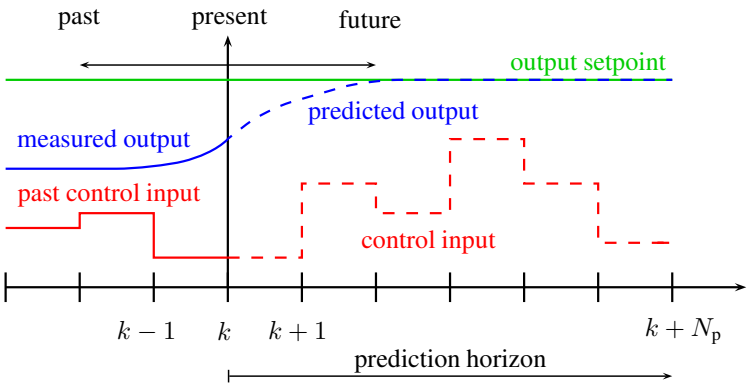


Figure 2-5: Moving horizon concept of Model Predictive Control

methodology uses a mathematical model of the controlled process and forecasts of future process states and external disturbances to determine the optimal trajectory of control variables and states by using an optimization algorithm, while taking physical and operational constraints into account. In general, the goals of Model Predictive Controllers used in the process industry are:

- Reducing the controller effort and/or variability in the (production) process by optimizing the dynamic control.
- Moving the output towards a setpoint, while operating close to the constraints.

MPC was originally developed and used for the petrochemical industry and power plants. Nowadays it is successfully being used in different fields of application including chemicals, food processing, automotive, aerospace and metallurgy (Qin and Badgwell, 1997), power networks and even in the field of biomedical sciences, see e.g. (Plank et al., 2006). Over the last decade, MPC has been attracting increasing attention in the field of operational water management, with applications focussing on efficient delivery of water in irrigation systems (Van Overloop, 2006; Malaterre and Baume, 1998; Wahlin, 2004; Negenborn et al., 2009), drainage and flood prevention (Blanco et al., 2010; Van Overloop, 2006; Van Overloop et al., 2010b), reservoir network management for multiple operational objectives (Goedbloed et al., 2011; Anand et al., 2013), hydroelectricity generation (Glanzmann et al., 2005; Şahin and Morari, 2010) and on the combined control of water quantity and quality in open channels (Xu, 2013; Xu et al., 2010).

There is a wide variety of MPC algorithms, however they all consist of the following main components:

1. An *internal model* that describes the dynamics of the system under consideration. This model is used to predict, at time step k, the future process output of the system $y^{k+i|k}$ for $i = 1, \ldots, N_p$ over a finite *prediction horizon* N_p. The optimal future output depends on the current (measured) value of y^k and the predicted disturbances and future control input.
2. An MPC algorithm is able to take physical and operational *constraints* of the process into account. The constraints are formulated as equality or inequality constraints, and can be applied to states and control variables.

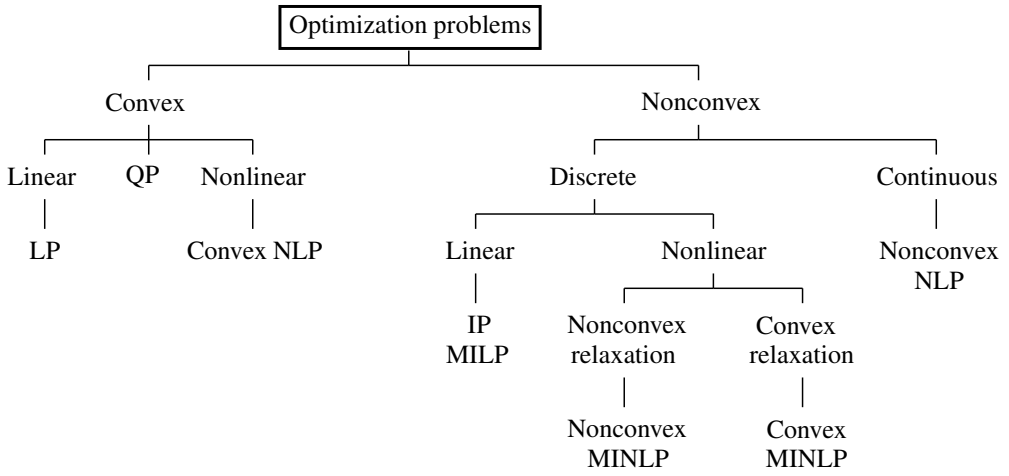


Figure 2-6: Overview of different optimization problem types, using: linear programming, (LP), quadratic programming (QP), nonlinear programming (NLP), integer programming (IP), mixed-integer linear programming (MILP), mixed-integer nonlinear programming (MINLP). Convex relaxation methods aim at finding a global minimum by approximating the original nonconvex problem with a convex problem, see (Lin et al., 2012) and references therein.

3. An *objective function* (or cost function) is used to quantify the trade-off between (often conflicting) objectives that the controller needs to achieve, for instance keeping the output close to a desired reference trajectory while minimizing (changes in) controller effort. The differences between desired and actual responses for the different (sub)goals are penalized and added up. In order to make the 'best' decision, the objective function needs to be minimized using an optimization algorithm. The result is a sequence of optimal control inputs over the prediction horizon, that satisfies the constraints. The relative values of the different penalties can be used to prioritize certain sub-objectives. The values of the penalties are typically used to tune the controller.

4. *Receding horizon control*: The optimal sequence of control steps $u^{k+i|k}$ for $i = 0, \ldots, N_\mathrm{p}-1$ is calculated at every time step k, but only the first value is applied, while neglecting the rest of the trajectory. At the new time step $k+1$, new information about current states and/or measurements are available and a new optimization problem is solved for the next N_p steps.

Figure 2-5 illustrates the basic idea of MPC.

The properties of the internal process model, objective function and constraints determine the resulting type of optimization problem, and hence the MPC technique that can to be used. (Lin et al., 2012) made a division of different optimization problems, based on characteristics such as (non)linearity and (non)convexity, see Figure 2-6.

2.4.1 Internal model

As the name already reveals, a Model Predictive Controller requires a mathematical model of the system that is being optimized. This model should be able to adequately represent the

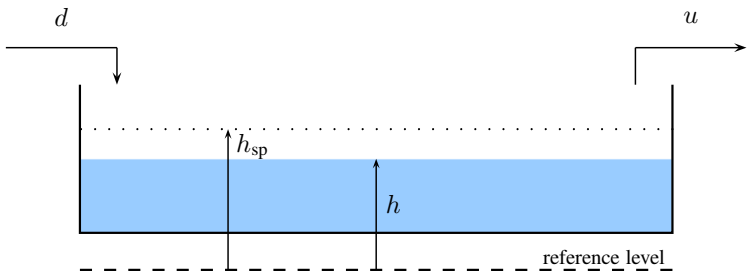


Figure 2-7: Schematization of a linear reservoir

Table 2-2: Parameters of fictitious reservoir model

Parameter	Value	Unit
T	300	s
A_s	10000	m^2
N_p	15	-
h_{sp}	3	m
u_{min}	0	$m^3\,s^{-1}$
u_{max}	3	$m^3\,s^{-1}$

dynamics of the system under consideration. Often a simplified model is being used, because a very detailed model could make the optimization routine computationally heavy.

Throughout this research, some principles are illustrated using a simplified model of a water system. This fictitious reservoir model is also being used to test different MPC algorithms. By using a storage area approach, the in- and outflows are linked to the water level inside the reservoir. The model consists of a single linear reservoir with an uncontrolled inflow (e.g. due to precipitation) and a controlled outflow that is being used to keep the water level close to a desired setpoint h_{sp} by pumping. It is assumed that the water level in the reservoir is horizontal, which means that any changes in inflow and outflow cause an instantaneous change in water level over the storage area. The basic equation for reservoir routing is based on the conservation of mass. It reads:

$$d - u = \frac{\partial S(h)}{\partial t}, \tag{2.16}$$

where $S(h)$ is the storage [m^3] in the reservoir as a function of the water level h [m], A_s is the storage area of the reservoir [m^2], d is the uncontrolled inflow or *disturbance* [$m^3\,s^{-1}$] and u is a controlled outflow [$m^3\,s^{-1}$]. A schematization of the reservoir is given in Figure 2-7. The reservoir has a linear level-storage relation, which means that A_s is constant over the vertical and $\partial S/\partial t = A_s \partial h/\partial t$ (hence the name linear reservoir). By applying a forward-difference approximation for the time derivative, we obtain the following state-space model:

$$h^{k+1} = h^k + \frac{T}{A_s}(d^k - u^k), \tag{2.17}$$

where k is the index of the time step [$-$] and T is the control length step [s]. The parameters of the reservoir model are summarized in Table 2-2.

2.4.2 Linear MPC

The fictitious model described in Section 2.4.1 is a linear system. The (non)linearity of a process model is an important characteristic, that influences the type of numerical solvers that can be used for MPC. Linear systems allow the application of standard linear algebra theory and software tools. The linear model of the controlled process is often described as a linear discrete-time system, represented by a state-space model (Van Overloop, 2006; Blanco et al., 2008; Kothare et al., 1996):

$$x^{k+1} = A^k x^k + B_{\mathrm{u}}^k u^k + B_{\mathrm{d}}^k d^k \tag{2.18}$$

$$y^k = C x^k, \tag{2.19}$$

where $x \in \mathbb{R}^{n_x}$ is the state, $u \in \mathbb{R}^{n_u}$ the control input, $d \in \mathbb{R}^{n_d}$ the (a priori known or predicted) disturbance, $y \in \mathbb{R}^{n_y}$ the output of the system, at time step k, A the system matrix, B_{u} the control input matrix, B_{d} the disturbance matrix and C the output matrix. The state-space model (2.18) can be extended over a finite prediction horizon N_{p}, from $k+1$ to $k+N_{\mathrm{p}}$. When the initial state x^k and all (predicted) disturbances and control inputs are known, all future output variables $x^{k+i|k}$ for $i = 1, \ldots, N_{\mathrm{p}}$ can be computed. The superscript $k+i|k$ denotes the sequence of $k+i$ future values that are evaluated at the current time step k.

The process model and predictions can be used to evaluate an objective function, which quantifies the trade-off between different (sometimes conflicting) objectives. Often, a quadratic objective function is used, which penalizes the squares of deviations from setpoint x_{sp} of the simulated states $\tilde{x}^k(u^k, d^k)$ and (changes in) control input. The sequential approach as described in (Xu and Schwanenberg, 2012) and (Diehl et al., 2009) is being used here. With the sequential (or Single Shooting) approach, the internal model is integrated in the objective function and the states are eliminated as controlled variables. Within each optimization step, the simulation and optimization are being performed sequentially, one after the other. The objective function to be minimized is at each time step k is:

$$\min_{\Delta u} \quad J = \sum_{i=0}^{N_{\mathrm{p}}} W_{\mathrm{e}} (e^{k+i|k})^2 + \sum_{i=0}^{N_{\mathrm{p}}-1} W_{\Delta \mathrm{u}} (\Delta u^{k+i|k})^2, \tag{2.20}$$

where:

$$e^k = \tilde{x}^k - x_{\mathrm{sp}} \tag{2.21}$$

$$\Delta u^k = u^k - u^{k-1} \tag{2.22}$$

and W_{e} and $W_{\Delta \mathrm{u}}$ are penalties on water level deviation from setpoint and change in control input, respectively. The goal of the Model Predictive Controller is to find at each time step k the optimal sequence of control actions $u^{k+i|k}$ for $i = 0, \ldots, N_{\mathrm{p}} - 1$ by minimizing the objective function (Camacho and Bordons, 1999). With a linear process model and a quadratic objective function, the optimization problem can be written as a convex QP problem (Qin and Badgwell, 2000), that has to be solved at every discrete time step k.

There are dedicated solvers for QP problems, e.g. the function `quadprog` in MATLAB's Optimization Toolbox. In Appendix A, the reservoir model of Section 2.4.1 is used to demonstrate

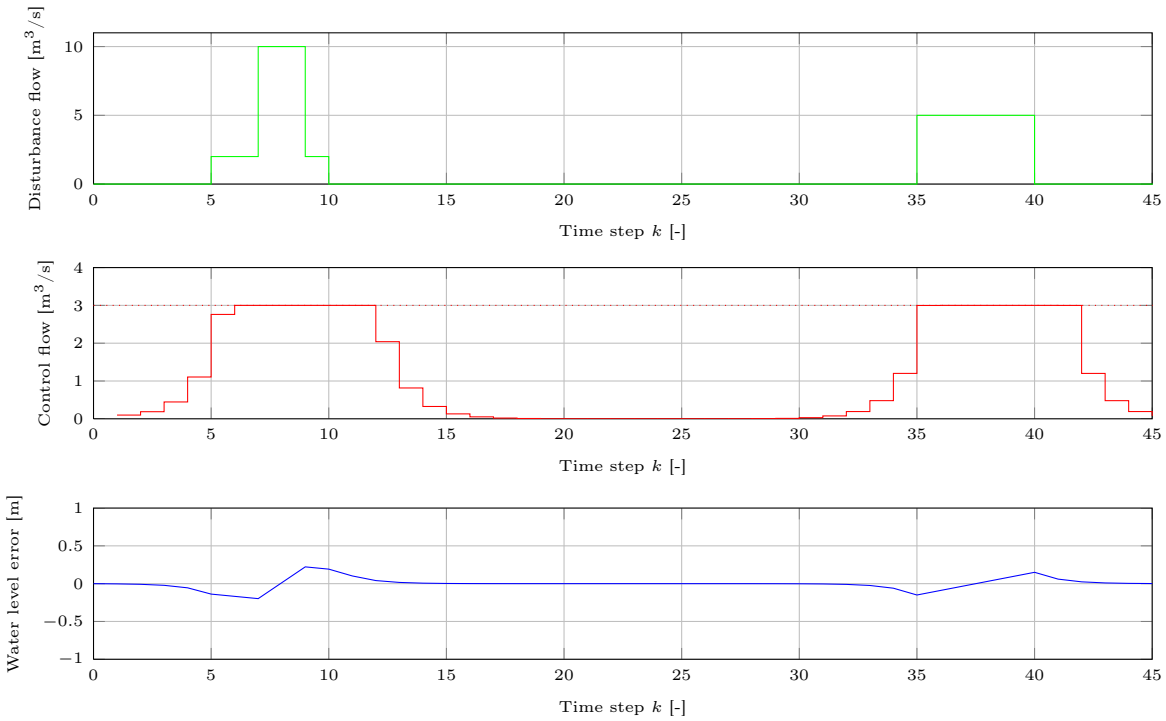


Figure 2-8: Results of closed-loop test of linear MPC applied to the fictitious reservoir model

how the linear process model and quadratic objective function result in a quadratic programming problem. Some results of a closed-loop example are shown in Figure 2-8. It can be seen that the controller anticipates on the disturbances by pre-releasing water from the reservoir, hence creating extra storage.

2.4.3 Nonlinear MPC

The reservoir model from Section 2.4.1 is a linear system, for which linear Model Predictive Control (LMPC) could be applied. The optimization problem could be cast into a quadratic programming problem, for which dedicated solvers exist. Because the quadratic programming problem is convex, the optimization algorithm is always able to find the global minimum of the objective function subject to constraints, provided that a solution exists (Van Overloop, 2006).

In general, the dynamics of a water system can be considered as nonlinear. Examples of nonlinearities in a water system are nonlinear level-storage relationships in reservoirs and flows through hydraulic structures such as weirs and orifices. In case the internal model is nonlinear, it might no longer be sufficient to use linear MPC (Qin and Badgwell, 2000; Blanco et al., 2010). In such cases, nonlinear Model Predictive Control (NMPC) can be applied. A nonlinear solver can be used to find the minimum of the objective function. An solver such as `fmincon` from the MATLAB Optimization Toolbox is able deal with constrained nonlinear multivariable optimization.

Assume that the process to be controlled can be described by the following discrete-time,

nonlinear, state-space model:

$$x^{k+1} = f(x^k, u^k, d^k), \tag{2.23}$$

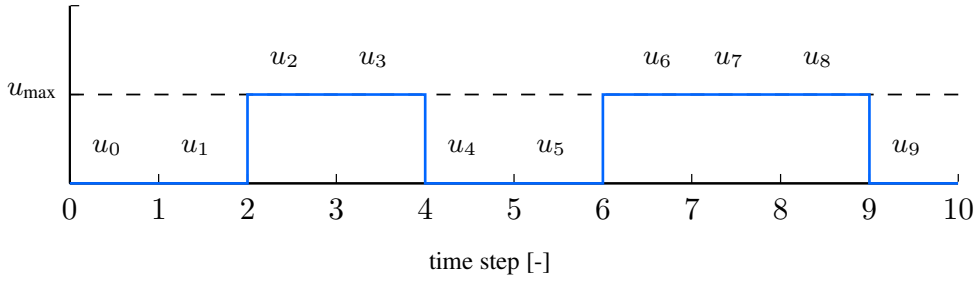
where $x \in \mathbb{R}^{n_x}$, $u \in \mathbb{R}^{n_u}$ and $d \in \mathbb{R}^{n_d}$ are the state, control and the (a priori known) disturbance. $f(\ )$ is the (nonlinear) function that represents the water resources model. The state-space model (2.23) is used to predict the optimal future trajectory of the state vector $x^{k+i|k}$ for $i = 1, \ldots, N_\mathrm{p}$ in order to determine the optimal set of control variables $u^{k+i|k}$ for $i = 0, \ldots, N_\mathrm{p}-1$ by minimizing the objective function subject to constraints, using a nonlinear optimization algorithm. When the state-space model (2.17) of the reservoir model in Section 2.4.1 is extended with nonlinear elements, such as a weir or gate, the nonlinear optimizer can be applied to find the optimal sequence of control inputs $u^{k+i|k}$ for $i = 0, \ldots, N_\mathrm{p} - 1$ such that the objective function (2.20) is minimized. In this research, we use MATLAB's solver `fmincon` (MATLAB, 2012), which is a dedicated solver for constrained nonlinear multivariable optimization problems.

The receding horizon control principle is being used i.e. only the first step of the trajectory $u^{k|k}$ will be carried out, and the optimization starts again at the next discrete time step $k+1$, resulting in the trajectory $u^{k+1+i|k+1}$ for $i = 0, \ldots, N_\mathrm{p} - 1$.

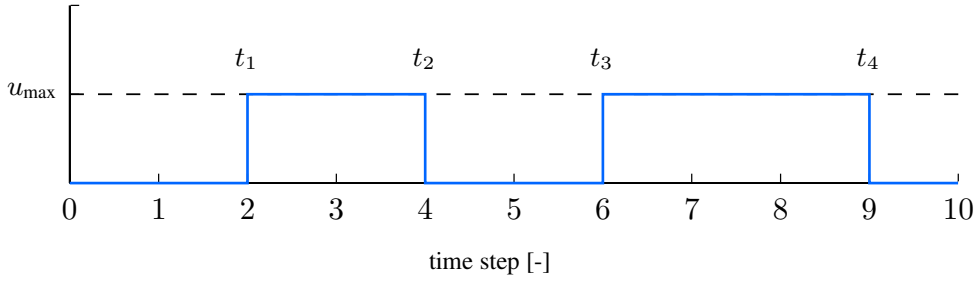
2.4.4 Hybrid MPC methods

The 'classic' MPC methods described in Sections 2.4.2 and 2.4.3 perform an optimization at each discrete time step k, where the optimization algorithm determines the optimal sequence of control actions $u^{k+i|k}$ for $i = 0, \ldots, N_\mathrm{p} - 1$, see Figure 2-5. The control inputs $u^{k+i|k}$ are continuous variables, that can have any value between u_min and u_max in accordance with the physical or operational constraints of the actuator. In the fictitious reservoir test case, this corresponds with a variable-frequency drive (VFD) pump. In reality, many polder systems also contain pumping stations that are not operated with VFD pumps, but with pumps that are operated either on or off. Other elements in water systems that can be seen as discrete elements are storm surge barriers, which are either open or closed. For these variables, an MPC algorithm would have to decide whether or not to change the state of the binary variable for every time step of the prediction horizon. For one binary input variable and a prediction horizon of N_p steps, this leads to a combinatorial optimization problem with dimensions 2^{N_p}. When there are more binary input variables and the prediction horizon is long, the complexity of the optimization problem increases rapidly and this may disqualify the method from being used in an operational setting.

Systems that consist of discrete and continuous components that interact with each other are called *hybrid* systems, and they require a hybrid MPC controller. An example of such a system is MPC with a mixed logical dynamical (MLD) prediction model, that consists of linear dynamic equations that are subject to mixed-integer (continuous and integer) inequalities (Morari and Barić, 2006; Zabiri and Samyudia, 2006). Mixed integer programming methods are less efficient in terms of computational effort compared to continuous optimization problems.



(a) 'Classic' MPC input variables



(b) TIO-MPC input variables, using four time instants

Figure 2-9: Example of different MPC approaches with binary control input

2.4.5 Time Instant Optimization MPC (TIO-MPC)

Another hybrid MPC approach is Time Instant Optimization Model Predictive Control (TIO-MPC), a method that has been first been used for traffic control (De Schutter, 1999). TIO-MPC has been applied for the control of hydraulic structures by (Van Ekeren, 2010), with a focus on storm surge barriers in the Rhine-Meuse delta. The TIO-MPC algorithm optimizes n time instants $t_1, \ldots, t_n$ over the prediction horizon N_p, where the time instants are the discrete time steps that the discrete elements change their state, see Figure 2-9b. The rationale behind this approach is that from a practical point of view, it is often undesired to have too many on/off switches of actuators and therefore it makes sense to define a priori how many switches are allowed within the prediction horizon. The result of this approach is that the amount of control variables is reduced. A TIO-MPC prediction model can be described as (Van Ekeren et al., 2011):

$$\tilde{x}^k = f(\tilde{t}^k, \tilde{u}^k, x^k), \tag{2.24}$$

with:

$$\tilde{x}^k = [(x^{k+1|k})^\top \quad (x^{k+2|k})^\top \quad \ldots \quad (x^{k+N_\mathrm{p}|k})^\top, \tag{2.25}$$

$$\tilde{u}^k = [(u^{k|k})^\top \quad (u^{k+1|k})^\top \quad \ldots \quad (u^{k+N_\mathrm{p}-1|k})^\top]^\top, \tag{2.26}$$

$$\tilde{t}^k = [t_1^k \quad t_2^k \quad t_3^k \quad t_4^k]^\top, \tag{2.27}$$

where x^k is the actual state at time step k, $\tilde{u}^k$ and $\tilde{x}^k$ are the input variables and state variables, respectively, and $\tilde{t}^k$ is a vector that contains the time instants that need to be optimized within the prediction horizon N_p. In this case, four time instants are defined. The

difference with the continuous MPC algorithms from Sections 2.4.2 and 2.4.3 is that $\tilde{u}^k$ is being optimized, where $\tilde{u}^k$ is defined as the control input vector which results from the optimized discrete-time equivalents of the time instants, taking into account that:

$$u^{k+i|k} = \begin{cases} u_{\max} & \text{if a pump is on at time } k+i \\ 0 & \text{otherwise,} \end{cases} \tag{2.28}$$

for $i = 0, \ldots, N_\mathrm{p} - 1$. To be able to have time instants as input of the optimization problem, a transformation from continuous time instants into their discrete-time equivalents is needed. This is done be defining for (2.26):

$$u^{k+i|k} = \begin{cases} u_{\max} & \text{if } i \leq k_1 \\ & \quad \text{or } k_2 \leq i \leq k_3 \\ & \quad \text{or } i \geq k_4 \\ 0 & \text{otherwise} \end{cases} \tag{2.29}$$

for $i = 0, \ldots, N_\mathrm{p} - 1$ and where k_1, k_2, k_3 and k_4 are discrete-time rounded equivalents of the continuous time instants.

Just like the continuous MPC schemes discussed in Sections 2.4.2 and 2.4.3, the TIO-MPC algorithm requires an objective function that quantifies the trade-off between the different objectives of the controller. The objective that needs to be minimized at every time step k is, in a generalized formulation:

$$J = f(\tilde{t}^k, \tilde{u}^k, \tilde{x}^k), \tag{2.30}$$

subject to the following constraints:

$$0 \leq t_1^k, \tag{2.31}$$
$$t_1^k - t_{\min} \leq t_2^k, \tag{2.32}$$
$$t_2^k - t_{\min} \leq t_3^k, \tag{2.33}$$
$$t_3^k - t_{\min} \leq t_4^k, \tag{2.34}$$
$$t_4^k \leq t_{\max}, \tag{2.35}$$
$$0 \leq u^{k+i|k} \leq u_{\max}, \tag{2.36}$$

for $i = 0, \ldots, N_\mathrm{p} - 1$. $t_{\min}$ is the minimum time between two state changes and $t_{\max}$ is the maximum value of t_4. By choosing $t_{\max} = N_\mathrm{p}$, a certain amount of state witches is forced into the prediction horizon. If $t_{\max} > N_\mathrm{p}$, the optimization can results into the decision not to switch states at all within the prediction horizon.

Chapter 3

Gradient-based Time Instant Optimization MPC

3.1 Introduction

Open water systems can be characterised by the presence of both continuous (evolution of flows and water levels) and discrete dynamics (opening and closing of barriers). The evolution of flows and water levels are continuous, while the opening and closing of barriers are discrete events. In polder systems, the water levels are maintained by gates and pumps. Some of these pumps are frequency driven, i.e. the pump rate can be chosen within a certain range. Other pumps have a fixed pump rate, which means they are either on or off. (Van Ekeren, 2010) proposed a MPC scheme that can take into account this hybrid nature of the system. This MPC technique is referred to as Time Instant Optimization Model Predictive Control (TIO-MPC).

3.2 TIO-MPC with continuous time instants

A so-called *brute force* optimization method would calculate all possible solutions of the combinatorial optimization problem, and afterwards decide which one is the best. If the optimization problem consists of many states and variables, a brute force optimization method may require considerable computation power since it requires many function evaluations. For the setup of the TIO-MPC controller from Section 2.4.5, the cost function can be minimized using a derivative-free optimizer such as the multi-start `pattern search` algorithm (MATLAB, 2012). This algorithm does not evaluate all possible solutions, but uses an iterative search algorithm that explores potential good solutions by following a search pattern, starting from different points. A gradient-based optimizer cannot be used efficiently because the time instants are discrete values, causing a stepwise objective function.

If we could rewrite the optimization problem in such a way that the time instants are continuous, this would enable us to calculate the gradient of the objective function $\mathrm{d}J/\mathrm{d}t$ and

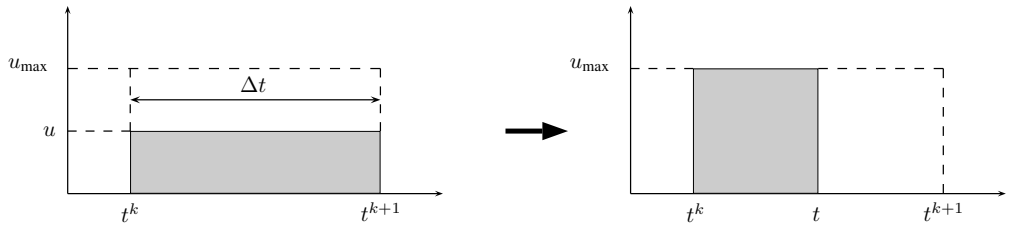


Figure 3-1: Towards continuous Time Instant Optimization

to use a gradient-based optimizer. In Figure 3-1, the translation from discrete to continuous time instants is shown. On the left, the pump discharge can be freely chosen between 0 and $u_{\max}$ on the discrete interval $[t^k, t^{k+1}]$. On the right, the pump is either off or operating with discharge $u_{\max}$ on the continuous interval $[t^k, t]$. The water balance on the interval $[t^k, t^{k+1}]$ is equal for both cases, so we can write:

$$u \cdot (t^{k+1} - t^k) = u_{\max} \cdot (t - t^k) \tag{3.1}$$

or:

$$u = \frac{u_{\max}}{t^{k+1} - t^k} \cdot (t - t_1). \tag{3.2}$$

By applying the chain rule we can obtain:

$$\frac{\mathrm{d}J}{\mathrm{d}t} = \frac{\mathrm{d}J}{\mathrm{d}u}\frac{\mathrm{d}u}{\mathrm{d}t} \tag{3.3}$$

$$= \frac{\mathrm{d}J}{\mathrm{d}u}\frac{u_{\max}}{t^{k+1} - t^k} \tag{3.4}$$

and since $\Delta t = t^{k+1} - t^k$, this results in:

$$\frac{\mathrm{d}J}{\mathrm{d}t} = \frac{\mathrm{d}J}{\mathrm{d}u}\frac{u_{\max}}{\Delta t}. \tag{3.5}$$

If we recall the process model (2.17) we can observe that the model calculations are only performed at the discrete time steps k. To be able to optimize continuous time instants, we need to perform a transformation of the control vector u similar to the procedure in Figure 3-1. This is done by defining the following: if t_1 and t_2 are the continuous time instants that the pumps are switched on and off respectively, where $t_1 < t_2$, $t_1 \in [a, b]$ and $t_2 \in [c, d]$ and a, b, c and d are positive integers, then the value of the control within the corresponding interval is:

$$u_{\mathrm{t1}} = u_{\mathrm{ab}} = \frac{b - t_1}{b - a}\, u_{\max} \tag{3.6}$$

for t_1, and

$$u_{\mathrm{t2}} = u_{\mathrm{cd}} = \frac{t_2 - c}{d - c}\, u_{\max} \tag{3.7}$$

for t_2 and $u_{\mathrm{bc}} = u_{\max}$. When $t_1, t_2 \in [a, b]$, the control corresponds to:

$$u_{\mathrm{ab}} = \frac{t_2 - t_1}{b - a}\, u_{\max}. \tag{3.8}$$

The transformed control vector is equal to the situation where $u(t) = u_{\max}$ for $t_1 \leq t \leq t_2$, and 0 otherwise. By using this control vector, the solver is able to optimize continuous time instants.

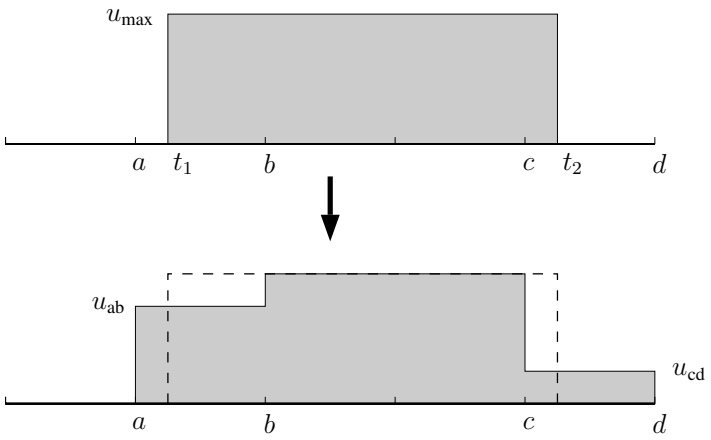


Figure 3-2: Time instants and corresponding control input

3.3 Algorithmic Differentiation

Solutions of numerical optimization problems and sensitivity analysis often require derivative information in the form of gradients, Jacobian and Hessian matrices. An accurate approximation of the derivative is therefore essential to many optimization problems. Efficient gradient-based solvers such as MATLAB's `fmincon` (MATLAB, 2012) are able to approximate the gradient of the objective function itself by means of finite difference methods. The procedure for doing this is quite straightforward, however for an optimization problem with many dimensions and parameters the method can be inefficient (Errico, 1997). Supplying the gradient with respect to each control variable to the solver will speed up the optimization routine. Using the analytical gradient is preferable, but it might be a tedious and error-prone process to derive this for complex models with many variables (Fournier et al., 2012). Another way to obtain the gradient is to use *automatic differentiation* (AD), sometimes also referred to as *algorithmic differentiation* (Griewank and Walther, 2008). This technique makes use of the fact that even the most complex computer models are in fact (potentially long) composites of elementary arithmetic operations such as multiplication, division, addition, subtraction and elementary functions such as sin, cos, tan, log and exp.

The idea behind AD is that basic derivative rules from calculus such as the multivariable chain rule can be systematically implemented in a numerical environment (Neidinger, 2010). An advantage of the AD method is that it, unlike finite difference methods, does not incur truncation errors, and that the derivatives obtained are accurate to machine precision (Griewank and Walther, 2008).

There are different commercially available AD software packages (for some examples see `http://www.autodiff.org` (Autodiff, 2013); Bischof et al. (1996); Verma (1999)) which are able to automatically transform source code into adjoint code for obtaining derivatives. See (Bischof et al., 2008) for a comparison of different AD tools.

3.3.1 Forward and Reverse Mode Algorithmic Differentiation

The AD method takes advantage of the fact that a computer needs to take a certain sequence of basic operations in order to calculate an output y out of the input variables $x_1, x_2, \ldots, x_n$. This run of intermediate steps can be evaluated in an *evaluation trace*, which shows the sequence of intermediate variables (defined as v_i, with $i > 0$) that a computer calculates in the so-called *forward sweep*. The derivative of the output with respect to a certain input variable can be obtained by systematically applying to chain rule to all the intermediate variables of the evaluation trace. The associativity of the chain rule allows multiple "modes" of accumulation of partial derivatives (Bischof et al., 2008). This leads to different approaches to AD e.g. the *forward* mode, *reverse* or *adjoint* mode and the *hybrid* or *cross-country* mode. The former two methods are widely used and will be explained below. Both examples are based on a function $f : x \in \mathbb{R}^n \rightarrow y \in \mathbb{R}^m$ where x is the independent variable and y is the dependent variable.

Forward Mode

Suppose we have the differentiable function $y = f(x_1, x_2, x_3)$, where the output y depends on a number of intermediate operations v_i, and we want to differentiate the dependent output y with respect to the independent input x_1. In the forward mode, a variable $\dot{v}_i = \partial v_i / \partial x_1$ (not to be mistaken with the first order time derivative) is associated with each intermediate variable v_i of the evaluation trace. The derivative of interest needs to be initialized or *seeded*, which means that $\dot{x}_1 = \partial x_1 / \partial x_1 = 1$, $\dot{x}_2 = 0$ and $\dot{x}_3 = 0$. The derivative $\partial y / \partial x_1$ is obtained by applying the chain rule to each line in the evaluation trace, in the same order as the evaluation trace itself. This method is called "forward" because the derivative values are accumulated in the same order as the intermediate values v_i in the evaluation trace. The total amount of floating point operations needed to evaluate $\partial y / \partial x_1$ is a small multiple of that for the underlying code to evaluate y (Griewank and Walther, 2008). In the forward mode, the chain rule is traversed from right to left.

Reverse mode

In the reverse mode, the (partial) derivatives are accumulated starting from the output variables y and propagating backwards towards the input variables. This equals traversing the chain rule from left to right. The difference with the forward mode is that an output variable is chosen, and that the sensitivity of this variable with respect to each of the intermediate variables v_i is calculated. A variable $\overline{v}_i = \partial y / \partial v_i$, called the *adjoint variable* is associated to each of the intermediate variables v_i. There is only one variable to be seeded, which is (by definition) $\overline{y} = 1$ (Griewank and Walther, 2008). The derivative is obtained by applying the chain rule to each intermediate value v_i, working backwards through the evaluation trace.

Even with many inputs x_i, the amount of floating points operations needed to evaluate all $\nabla_x y$ is between one and four times that of the evaluation of y. For the computation of the gradient of a function $f : x \in \mathbb{R}^n \rightarrow y \in \mathbb{R}^m$ where $n \gg m$, the reverse mode is advantageous over the forward mode (Griewank and Walther, 2008; Giering and Kaminski, 1998). This property, together with the absence of a truncation error makes the method extremely attractive for

complex computational models where derivatives are needed. A drawback of the method is the need to temporarily store the computation trace on a so-called *tape* (see Bennett, 1973), because the propagation of the chain rule is done backwards through the evaluation trace (Pearlmutter and Siskind, 2008).

3.3.2 AD in reverse mode applied to fictitious reservoir model

To demonstrate algorithmic differentiation in reverse mode, we will derive the gradient of an objective function of a TIO-MPC model. The fictitious reservoir model from Section 2.4.1 is used as a process model. In this case, two time instants need to be optimized within the prediction horizon N_p: t_1 and t_2. Recall the linear reservoir model from Section 2.4.1:

$$h^{k+1} = h^k + \frac{T}{A_\mathrm{s}}(d^k - u^k). \tag{3.9}$$

For the TIO-MPC controller considered here, the following objective function $J = f(\tilde{t}^k, \tilde{u}^k, \tilde{h}^k)$ can be defined, where $\tilde{u}$ is a function of the time instants:

$$\min_{\tilde{u}} \quad J = W_\mathrm{h}\Big(\tilde{h}^k(\tilde{u}^k, d^k) - h_\mathrm{sp}\Big)^2 + W_\mathrm{t}(t_2 - t_1) \tag{3.10}$$

$$\begin{aligned} \text{subject to} \quad & 0 \leq t_1^k && (3.11)\\ & t_1^k \leq t_2^k && (3.12)\\ & t_2^k \leq N_\mathrm{p} && (3.13)\\ & 0 \leq \tilde{u}^k \leq u_\mathrm{max} && (3.14) \end{aligned}$$

where:

$$\begin{aligned} \tilde{h}^k &= [(h^{k+1|k})^\top \quad (h^{k+2|k})^\top \quad \dots \quad (h^{k+N_\mathrm{p}|k})^\top && (3.15)\\ \tilde{u}^k &= [(u^{k|k})^\top \quad (u^{k+1|k})^\top \quad \dots \quad (u^{k+N_\mathrm{p}-1|k})^\top]^\top && (3.16)\\ \tilde{t}^k &= [t_1^k \quad t_2^k]^\top, && (3.17) \end{aligned}$$

and h_sp is the setpoint [m], t_min is the minimum time between two state changes [s] and u_max is the maximum pump capacity [$\mathrm{m^3\,s^{-1}}$]. There is a quadratic penalty W_h on the water level deviation from setpoint, and a linear penalty W_t on the pump volume (or: the time that the pump is switched on). For this reservoir test case, only two time instants are being considered.

Some rewriting results in the following objective function:

$$\min_{\tilde{u}} \quad J = W_\mathrm{h} \sum_{k=1}^{N_\mathrm{p}} J^i + W_\mathrm{t}(t_2 - t_1) \tag{3.18}$$

where:

$$J^i = (h^{k+i|k} - h_\mathrm{sp})^2, \quad i \in [1, \dots, N_\mathrm{p}]. \tag{3.19}$$

We would like to obtain the gradient (or *Jacobian*):

$$\nabla J(t_1, t_2) = \Big(\frac{\partial J}{\partial t_1}, \frac{\partial J}{\partial t_2}\Big). \tag{3.20}$$

Table 3-1: Reverse-derived evaluation trace for J^i

v_{-5}	$= u^k$	
v_{-4}	$= d^k$	
v_{-3}	$= T$	
v_{-2}	$= A$	
v_{-1}	$= h^k$	
v_0	$= h_{\text{sp}}$	
v_1	$= v_{-4} - v_{-5}$	
v_2	$= v_{-3} * v_1$	
v_3	$= v_2 / v_{-2}$	
v_4	$= v_{-1} + v_3$	$(= h^{k+1})$
v_5	$= v_4 - v_0$	
v_6	$= {v_5}^2$	
J_k	$= v_6$	
$\overline{v}_6$	$= \overline{J}^k = 1$	
$\overline{v}_5$	$= \overline{v}_6 \frac{\partial v_6}{\partial v_5}$	$= \overline{v}_6 * 2 * v_5$
$\overline{v}_4$	$= \overline{v}_5 \frac{\partial v_5}{\partial v_4}$	$= \overline{v}_5 * 1$
$\overline{v}_3$	$= \overline{v}_4 \frac{\partial v_4}{\partial v_3}$	$= \overline{v}_4 * 1$
$\overline{v}_2$	$= \overline{v}_3 \frac{\partial v_3}{\partial v_2}$	$= \overline{v}_3 / v_{-2}$
$\overline{v}_1$	$= \overline{v}_2 \frac{\partial v_2}{\partial v_1}$	$= \overline{v}_2 * v_{-3}$
$\overline{v}_{-5}$	$= \overline{v}_1 \frac{\partial v_1}{\partial v_{-5}}$	$= \overline{v}_1 * -1$

Because the predicted sequence of states $h^{k+i|k}$ for $i = 1, \ldots, N_{\text{p}}$ of the reservoir model depends on the current state h^k, one can write an explicit formulation of the objective function. For example, writing out the first two terms of J^i results in:

$$J^1 = (h^{k+1|k} - h_{\text{sp}})^2 = \left(h^k + \frac{T}{A}(d^k - u^{k|k}) - h_{\text{sp}}\right)^2 \tag{3.21}$$

$$J^2 = (h^{k+2|k} - h_{\text{sp}})^2 = \left(h^k + \frac{T}{A}(d^k - u^{k|k} + d^{k+1} - u^{k+1|k}) - h_{\text{sp}}\right)^2. \tag{3.22}$$

Writing out the complete objective function will result in a very long expression, for which it is a tedious task to derive the analytical derivatives with respect to t_1 and t_2. Therefore we apply algorithmic differentiation for the first term on the right-hand side of (3.18). As an example, the reverse-derived evaluation trace of J^1 is shown in Table 3-1. The result is the derivative $\partial J^1 / \partial u^{k|k}$. Writing out the reverse-derived evaluation trace yields:

$$\frac{\partial J^1}{\partial u^{k|k}} = \frac{\partial J^1}{\partial h^{k+1|k}} \frac{\partial h^{k+1|k}}{\partial u^{k|k}} = -2(h^{k+1|k} - h_{\text{sp}}) \frac{T}{A}. \tag{3.23}$$

The derivative $\frac{\partial J^1}{\partial t_1}$ can now be calculated, since:

$$\frac{\partial J^1}{\partial t_1} = \frac{\partial J^1}{\partial h^{k+1|k}} \frac{\partial h^{k+1|k}}{\partial u^{k|k}} \frac{\partial u^{k|k}}{\partial t_1} \tag{3.24}$$

where:

$$\frac{\partial u^{k|k}}{\partial t_1} = -\frac{u_{\max}}{\Delta t} \tag{3.25}$$

and hence, with $\Delta t = t^{k+1} - t^k = 1$:

$$\frac{\partial J^1}{\partial t_1} = 2(h^{k+1|k} - h_{\text{sp}})\frac{T}{A}u_{\max}. \tag{3.26}$$

The term $\partial u^{k|k}/\partial t_1$ is equal to zero for all intervals smaller than the one containing t_1. This is obvious because the gradient of $u^{k|k}$ with respect to t_1 is not defined for $t < t_1$. After all, the time instant t_1 determines when the pump starts pumping. The gradient of the objective function J with respect to t_1 is:

$$\frac{\partial J}{\partial t_1} = W_{\text{h}} \sum_{i=1}^{N_{\text{p}}} \frac{\partial J^i}{\partial t_1} - W_{\text{t}}. \tag{3.27}$$

The gradient of the objective function J is:

$$\nabla J = \left(W_{\text{h}} \sum_{i=1}^{N_{\text{p}}} \frac{\partial J^i}{\partial t_1} - W_{\text{t}}, -W_{\text{h}} \sum_{i=1}^{N_{\text{p}}} \frac{\partial J^i}{\partial t_2} + W_{\text{t}} \right) \tag{3.28}$$

3.4 (Non-)convexity of the objective function

If the objective function J is convex, every local minimum must be a global minimum. If the objective function is non-convex, a gradient-based optimizer is not guaranteed to find the global minimum. Depending on the starting point used by the optimization routine, the solver might find a local minimum and hence obtain a suboptimal solution. To serve as a global solver, the algorithm can tackle this issue by using multi-start optimization.

When multi-start optimization is being used, the model is run from n different (random or user-defined) starting points for every time step k. In each time step k, the algorithm keeps track of all the obtained local minima and then uses only the 'best' value. By using a multi-start optimization, the dimensions of the optimization problem get considerably larger since many function evaluations are required. The length of the control time step gives an upper bound for the time that is available for running a multi-start procedure. Figure 3-3 shows the objective function J as a function of two time instants t_1 and t_2, for one simulation step. The yellow star indicates the location of the minimum. From the figure it can be seen that the objective function seems to be convex. The convexity of a twice continuously differentiable function of multiple variables can be proven by determining whether the Hessian matrix of that function is positive semidefinite. The function is convex if and only if the Hessian is positive semidefinite. If the Hessian is always positive definite, the function is strictly convex (Boyd and Vandenberghe, 2004). A strictly convex function $f(t_1, t_2)$ is convex for any $t_1 \neq t_2$ and has at most one optimal solution (Rockafellar and Wets, 1998). By definition, a matrix A is positive semidefinite if all eigenvalues of A are nonnegative and positive definite if all

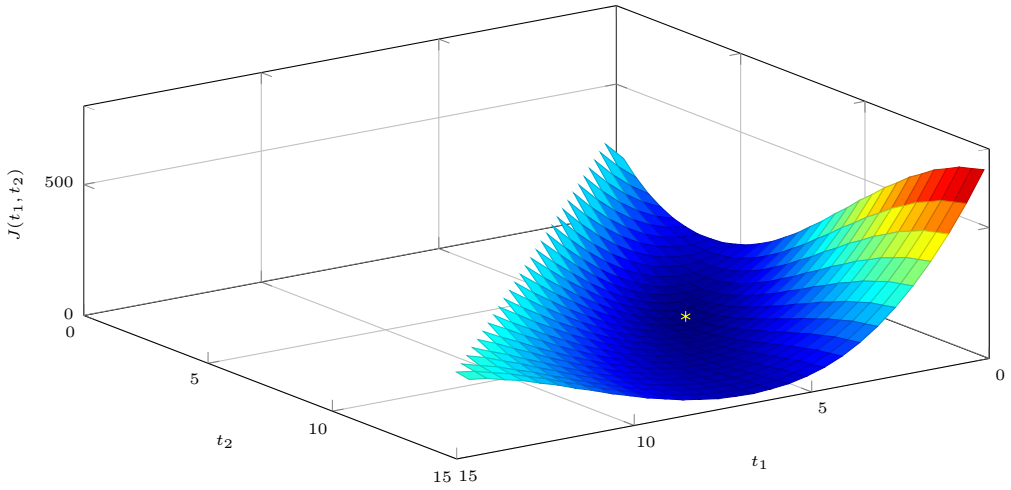


Figure 3-3: Objective function J as a function of two time instants ($W_\mathrm{h} = 200$ and $W_\mathrm{t} = 1$). Time instant t_1 is the moment the pump is switched on and t_2 is the moment when the pump is stopped. The prediction horizon N_p is 15 steps.

eigenvalues of A are positive. The Hessian of the objective function J is defined as:

$$H(J) = \nabla^2 J(t_1, t_2) = \begin{bmatrix} \dfrac{\partial^2 J}{\partial t_1^2} & \dfrac{\partial^2 J}{\partial t_1 \partial t_2} \\ \dfrac{\partial^2 J}{\partial t_2 \partial t_1} & \dfrac{\partial^2 J}{\partial t_2^2} \end{bmatrix}. \tag{3.29}$$

The objective function is a composition of quadratic terms, of which the Jacobian is dependent on the location of the discrete sampling intervals of the continuous time instants t_1 and t_2 on the domain of $J \in [0, N_\mathrm{p}]$, as was explained in Section 3.3.2. The same applies to the Hessian, which consists of second order partial derivatives and thus is different for different combinations of t_1 and t_2 (in short notation: $t_{1,2}$). For this reason, derivation of the Hessian is not straightforward. Convexity can be proven by checking the eigenvalues of all different 'modes' of the Hessian belonging to the objective function. We distinguish between two situations:

1. Time instants t_1 and t_2 are element of different sampling intervals $[a, b]$ and $[c, d]$.
2. Time instants t_1 and t_2 are element of the same sampling interval $[a, b]$.

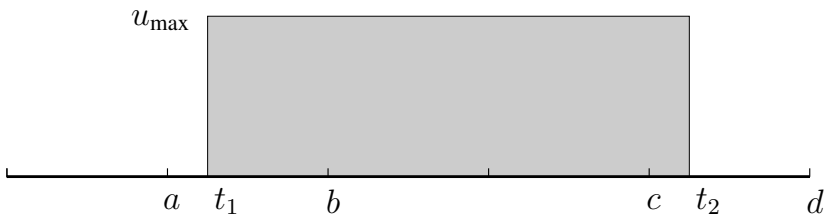
The different cases are illustrated in Figure 3-4.

For the situation where $t_1 < t_2$, $t_1 \in [a, b]$, $t_2 \in [c, d]$, $t_{1,2} \in \mathbf{dom}(J) = [0, N_\mathrm{p}]$ with a, b, c and d positive integers, some elaboration yields:

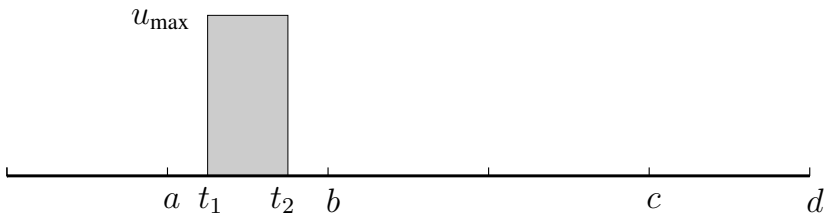
$$H(J) = \zeta \begin{bmatrix} \alpha & -\beta \\ -\beta & \beta \end{bmatrix} \tag{3.30}$$

where:

$$\zeta = W_h \frac{T^2}{A^2} \frac{u_{\max}^2}{\Delta t^2} \tag{3.31}$$



(a) Case 1: time instants are element of different sampling intervals



(b) Case 2: time instants are element of the same sampling interval

Figure 3-4: Different configurations of time instants

$$\alpha = 2(N_{\mathrm{p}} - b + 1) \tag{3.32}$$

$$\beta = 2(N_{\mathrm{p}} - c) \tag{3.33}$$

where ζ is a constant, and α and β are integers that depend on the locations of t_1 and t_2 on the domain of J. The eigenvalues λ_1 and λ_2 (in short notation: $\lambda_{1,2}$) of the Hessian can be determined by solving the characteristic equation, defined as:

$$\det(H - \lambda I) = \det \begin{bmatrix} \zeta\alpha - \lambda & -\zeta\beta \\ -\zeta\beta & \zeta\beta - \lambda \end{bmatrix} = 0. \tag{3.34}$$

of which the eigenvalues $\lambda_{1,2}$ are:

$$\lambda_{1,2} = \frac{\zeta\alpha}{2} + \frac{\zeta\beta}{2} \pm \frac{1}{2}\sqrt{\zeta(\alpha^2 - 2\alpha\beta + 5\beta^2)}. \tag{3.35}$$

It follows that $\alpha, \beta \geq 2$, $\alpha > \beta$ (since $t_1 < t_2$) and hence $\lambda_{1,2} > 0$. This implies that the eigenvalues are positive definite and hence, the objective function is strictly convex as long as t_1 and t_2 are not located in the same discrete interval.

Similarly, it can be derived that in case $t_{1,2} \in [a, b]$, $t_1 < t_2$, $t_{1,2} \in \mathbf{dom}(J) = [0, N_{\mathrm{p}}]$ with a and b positive integers we have (see Figure 3-4b):

$$H(J) = \zeta \begin{bmatrix} \alpha & -\alpha \\ -\alpha & \alpha \end{bmatrix}. \tag{3.36}$$

Since $b \in [1, N_{\mathrm{p}}]$, $\alpha > 0$ and it follows that $\lambda_1 = 0$ and $\lambda_2 = 2\zeta\alpha$. This means that the Hessian matrix is positive semidefinite and the objective function J is convex.

Table 3-2: Parameters of reservoir test case using TIO

Parameter	Value	Unit
T	300	s
A	10000	m^2
N_p	15	-
h_{sp}	3	m
u_{min}	0	$m^3 s^{-1}$
u_{max}	3	$m^3 s^{-1}$

3.5 Gradient-based TIO-MPC applied to fictitious reservoir model

The TIO Model Predictive Controller applied to the linear reservoir from Section 3.3.2 was modelled in closed-loop using Matlab. Two time instants have been used. For obtaining the gradient of the objective function, algorithmic differentiation in reverse mode has been applied as described in Section 3.3.2. Since the objective function is convex, a multi-start optimization procedure was not necessary. The used model parameters are given in Table 3-2. It should be mentioned again that the pump has discrete settings, i.e. the pump discharge is either 0 or u_{max}. Figures 3-5 and 3-6 show the results for two model runs, where different combinations of penalties have been used in the objective function. It can be observed from Figure 3-5 and 3-6 that in both cases the controller tries to keep the water level at setpoint by pre-releasing water from the reservoir. With a relatively higher penalty on the deviation from setpoint, the controller applies a tighter control on the water level, as is the case in Figure 3-5. The latter two types of system behaviour are common to Model Predictive Controllers in general, and fortunately it also applies to TIO-MPC. In Figure 3-5 there are two periods in which the pumps are pumping i.e. the pumps change their states four times. In Figure 3-6, we can observe that there are three periods in which the pumps are pumping i.e. the pumps change their states six times. This difference is possible because the tests are performed in closed-loop. It can be observed that the pumps are indeed changing their state at continuous time instants.

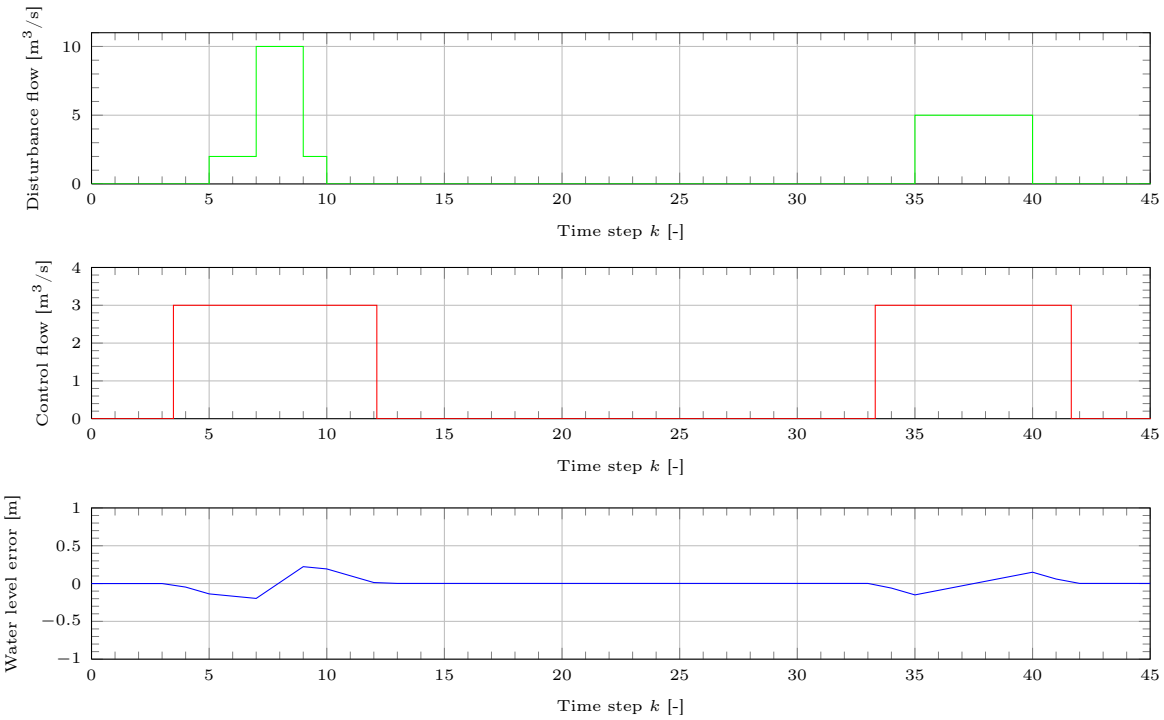


Figure 3-5: Results of TIO-MPC controller, using $W_{\mathsf{h}} = 200$ and $W_{\mathsf{t}} = 1$

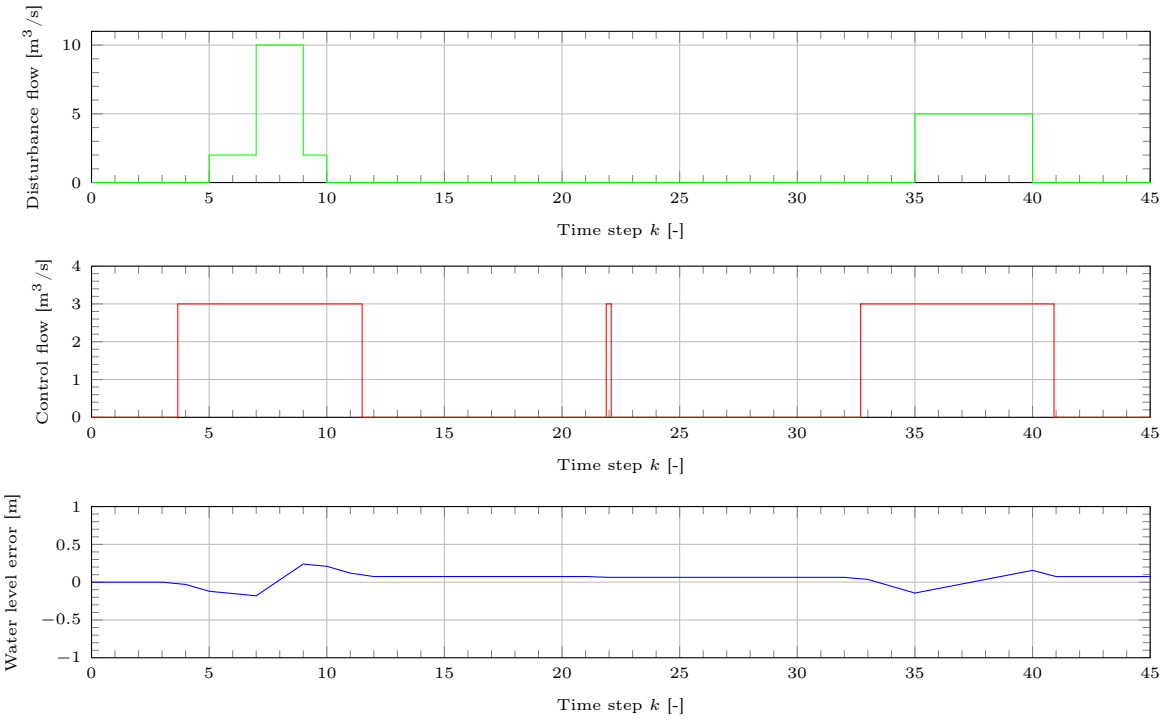


Figure 3-6: Results of TIO-MPC controller, using $W_{\mathsf{h}} = 5$ and $W_{\mathsf{t}} = 1$

Chapter 4

Application of TIO-MPC to the Fivelingo water system

This chapter describes the tasks of the Regional Water Authority Noorderzijlvest and the Fivelingo water system. Furthermore, the Model Predictive Controller used for this system is discussed. Finally, the experiments are introduced.

4.1 Water system of Regional Water Authority Noorderzijlvest

Regional Water Authority Noorderzijlvest is responsible for the water management in the northern part of the provinces Groningen and Drenthe. Their task is to protect the area against inundation from the regional water system, to ensure safe water quality, to maintain the ecological status and to manage the water supply during dry periods. Also, they are responsible for collecting and treating waste water. Regional Water Board Noorderzijlvest manages an area of 144.000 hectares, see Figure 4-1

4.1.1 Fivelingo boezem

The Fivelingo water system is located in the eastern corner of the Noorderzijlvest service area, see Figure 4-1. It can be considered as an isolated system, with many canals discharging into a *boezem canal*, which is called the Damsterdiep. In the map, it looks like there is a connection to the northern part of the system but in reality there is a ship lock that separates the Fivelingo system from the rest of the service area.

The Damsterdiep canal connects the city of Groningen with the sea at Delfzijl. The man-made canal is about 27 kilometres long and can be divided into two distinctive parts. The eastern part has a meandering course due to the former open connection to the sea. The western part was dug out later and it follows a straighter course. There is a pumping station in the harbour of Delfzijl called 'De Drie Delfzijlen'. At low tide, water can flow out of the system through a gate.

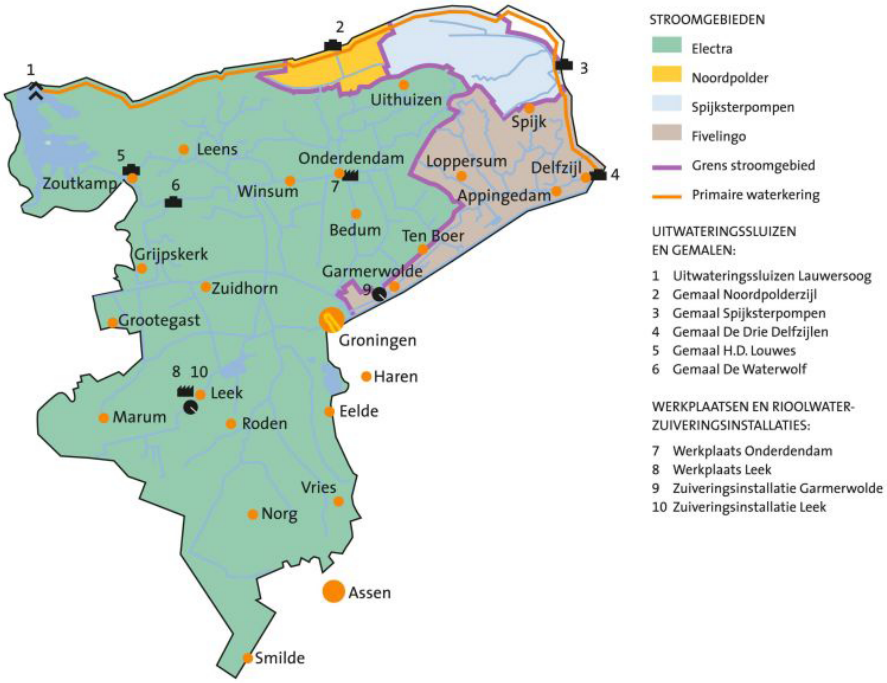


Figure 4-1: Service area of Regional water authority Noorderzijlvest (Noorderzijlvest, 2013)

.

The Damsterdiep is not very important for commercial shipping, as it is quite shallow. There is however some recreational shipping. The canal has mainly a drainage function for the surrounding polders. The polders pump their surplus water into the boezem using pumps that are operated either on or off.

4.1.2 Schematization of Fivelingo water system

A coarse MPC model of the Fivelingo water system has previously been derived in a research project carried out by the research institute Deltares (Schwanenberg and van Heeringen, 2012). This goal of this study was twofold: improvement of the management of the water resources system of Regional Water Authority Noorderzijlvest (NZV) during floods by applying real-time control (RTC), and the application of RTC-Tools (Schwanenberg and Becker, 2012), a novel software package developed by Deltares for RTC purposes. In the research, the internal MPC models were created for all primary canals of the subsystems of the NZV water system. In (Schwanenberg and van Heeringen, 2012) it was shown that the performance of the internal models is sufficiently good for RTC-applications.

The MPC models in RTC-Tools consist of storage nodes and branches on a staggered grid, as described in Section 2.3. The Fivelingo water system consists of three storage nodes, in order to take into account water level gradients between the central node (representing the average water level in the Damsterdiep canal) and the storage node upstream of the outlet to the sea at the harbour of Delfzijl. The level-storage relations of the storage nodes was derived from a detailed SOBEK hydrodynamic model (Deltares, 2013) by aggregating the available storage

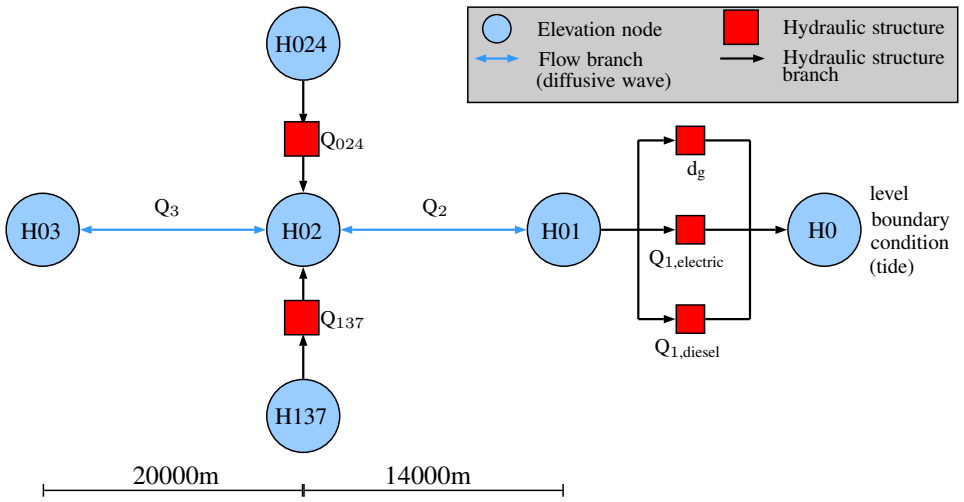


Figure 4-2: Schematic overview of the Fivelingo water system. The two blue arrows together represent the Damsterdiep canal

from the primary canals.

For the current research, the MPC model is extended with two polder canals, which discharge water into the Damsterdiep canal using pumps that are operated either on or off. In the MPC model, this was modelled by adding two storage nodes and two pumps. Time Instant Optimization is applied for these two pumps. A schematization of the extended model is given in Figure 4-2. The extended model consists of six storage nodes, two flow branches and five hydraulic structures, which are described in Table 4-1.

4.2 Optimization problem

4.2.1 Objective function

The goals of the controller are expressed in the objective function. Since the water system is a typical polder-belt canal system, the main goal of the controller is to minimize the water level deviations from setpoint in the belt- and polder canals, while avoiding potential flooding events by anticipating on expected disturbances. A second goal, that applies mainly to day-to-day operations, is to achieve energy and cost savings on pumping by:

1. Making efficient use of the available storage in the system
2. Creating storage in the system by pre-releasing water prior to an expected disturbance
3. Releasing as much of the water surplus as possible through the gate at low tide

A third goal is to minimize wear and tear of hydraulic structures by limiting the changes in pump flow and gate height. These goals can be described mathematically in the following

Table 4-1: Description of model components

Component	Type	Description
H0	Storage node	Downstream boundary condition, representing the water level at sea near the outlet in the harbour of Delfzijl
H01	Storage node	Represents the water level just upstream of the outlet structures
H02	Storage node	Intermediate node that represents the mean water level in the Damsterdiep canal
H03	Storage node	Upstream boundary of the Damsterdiep canal
H024	Storage node	Represents the water level in polder #024
H137	Storage node	Represents the water level in polder #137
Q_2	Flow branch	Flow based on diffusive wave model on a staggered grid
Q_3	Flow branch	Flow based on diffusive wave model on a staggered grid
Q_{024}	Pump flow	Flow from polder #024 (discrete)
Q_{137}	Pump flow	Flow from polder #137 (discrete)
$Q_{1,\text{electric}}$	Pump flow	Flow from electric pump at Delfzijl (continuous)
$Q_{1,\text{diesel}}$	Pump flow	Flow from diesel pump at Delfzijl (continuous)
d_g	Gate height	Flow (as a function of the gate height) through the gate at Delfzijl

objective function, that needs to be minimized:

$$\begin{aligned}\min \quad J^k = &\sum_{i=1}^{N_\mathrm{p}} \sum_{j=1}^{m} W_{\mathrm{e},j} \ (e_j^{k+i|k})^2 \\ &+ \sum_{j=1}^{m} W_{\mathrm{e,Np},j} \ (e_j^{k+N_\mathrm{p}|k})^2 \\ &+ \sum_{i=1}^{N_\mathrm{p}} \sum_{j=1}^{l} W_{\Delta\mathrm{Q},j} \ (\Delta Q_j^{k+i|k})^2 \\ &+ \sum_{i=1}^{N_\mathrm{p}} \sum_{j=1}^{l} W_{\mathrm{Q},j} \ Q_j^{k+i|k} \\ &+ \sum_{i=1}^{N_\mathrm{p}} \sum_{j=1}^{p} W_{\Delta\mathrm{dg},j} \ (\Delta d_{\mathrm{g},j}^{k+i|k})^2,\end{aligned} \tag{4.1}$$

where Q_j^k is the pump flow [$\mathrm{m^3\,s^{-1}}$] that follows from the time instants, e_j^k is the water level deviation from setpoint m, ΔQ_j^k is the change in pump flow [$\mathrm{m^3\,s^{-1}}$] and $\Delta d_{\mathrm{g},j}^k$ is the change in gate setting [m], at time step k:

$$e_j^k = h_j^k - h_{\mathrm{sp},j} \tag{4.2}$$

$$\Delta Q_j^k = Q_j^k - Q_j^{k-1} \tag{4.3}$$

$$\Delta d_{\mathrm{g},j}^k = d_{\mathrm{g},j}^k - d_{\mathrm{g},j}^{k-1}. \tag{4.4}$$

J^k is the objective function that needs to be minimized, in which k is the step counter of the prediction horizon N_p [−], m the number of flow branches between storage nodes [−], l is the number of pumps [−], p is the number of gates [−], $W_{\mathrm{e},j}$ is the penalty on the water level deviation from setpoint for a storage node, $e_j^{k+N_\mathrm{p}}$ is the water level error [m] at the end of the prediction horizon, $W_{\mathrm{e,Np},j}$ is the penalty on this error [−], $W_{\mathrm{Q},j}$ is the penalty on the change in pump flow [−], $W_{\mathrm{Q},j}$ is a penalty on pump flow and $W_{\Delta\mathrm{dg},j}$ is a penalty on the change of the gate height setting [−].

The setpoints and penalties used in (4.1) are summarized in Tables 4-2 and 4-3. All penalties except the ones for the polder pumps that were added, were adopted from the existing model, assuming that this model has been tuned thoroughly [1].

The academic test case from Section 2.4.1 consisted of a linear process model and a quadratic objective function, which results in a highly structured convex quadratic programming (QP) problem. For the Fivelingo water system, a nonlinear process model and a quadratic objective function are being used. The use of a nonlinear model leads to a loss of convexity (Qin and Badgwell, 2000). For this reason it may be harder to find a (sufficiently good) solution, and if a solution is found it is not guaranteed to be the global optimum: there may be multiple local minima.

[1] It is important to note that the tuning of the penalties is an important part of the configuration of a Model Predictive Controller, because the penalties determine the behaviour of the controller.

Table 4-2: Setpoints used in the objective function

Term	Setpoint value [m]
H01	-1.55
H02	-1.33
H024	-1.40
H137	-2.55

Table 4-3: Penalties used in the objective function

	Penalty				
	W_e	$W_{e,Np}$	$W_{\Delta Q}$	W_Q	$W_{\Delta dg}$
H01	200	-	-	-	-
H02	50	500	-	-	-
H024	200	-	-	-	-
H137	200	-	-	-	-
Q_{024}	-	-	-	0.30	-
Q_{137}	-	-	-	0.30	-
$Q_{1,electric}$	-	-	1	0.10	-
$Q_{1,diesel}$	-	-	1	0.30	-
d_g	-	-	-	-	0.01

4.2.2 Constraints

The objective function is subject to physical and operational constraints, which limit the (change in) pump flow and gate settings according to:

$$Q_{\min,j} \leq Q_j^k \leq Q_{\max,j} \tag{4.5}$$

$$d_{g,\min,j} \leq d_{g,j}^k \leq d_{g,\max,j} \tag{4.6}$$

$$\tag{4.7}$$

The constraints that apply to the Fivelingo water system are summarized in Table 4-4. The values represent minimum and maximum pump flows and gate height settings.

Table 4-4: Constraints

Term	Min	Max	Unit
$Q_{1,electric}$	0	3.33	$m^3\,s^{-1}$
$Q_{1,diesel}$	0	20	$m^3\,s^{-1}$
Q_{024}	0	3.25	$m^3\,s^{-1}$
Q_{137}	0	3	$m^3\,s^{-1}$
d_g	0	5	m

Table 4-5: Specifications of computer and software used for simulations

Processor	Intel Core i3 350M @ 2.27 GHz
Memory	4.00 GB DDR3
Operating system	Windows 7 Home Premium SP1 64-bit
Matlab version	R2013a 32-bit
Solver	`fmincon`, using Interior-Point algorithm

4.2.3 Performance indicators

Because the objective function J describes mathematically the different goals of the controller, the value of the objective function at each optimization step is an indicator for the overall performance of the model. The value of the objective function does not give information about how well specific goals are achieved. Since the used objective function is the same in all experiments, the only difference in the experiments is the amount of time instants used for optimizing the control input Q_{024} and Q_{137} of the discrete pumps. We can check how these discrete controllers are performing by looking at the simulated water levels in the nodes that are being controlled by these pumps. A performance indicator that can be used to compare the controllers of the discrete elements in the experiments is the root mean square error (RMSE). The RMSE can be used to measure the average deviation from setpoint for the simulated water levels, for the different experiments. A low value of the RMSE corresponds with tighter control. The RMSE of simulated water levels reads:

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{N_\text{p}} (h^{k+i|k} - h_\text{sp})^2}{N_\text{p}}} \tag{4.8}$$

where $h^{k+i|k}$ for $i = 1, \ldots, N_\text{p}$ are the simulated water levels over the prediction horizon N_p, evaluated at time step k.

A different, but also relative performance indicator is the computational time, since this determines whether or not a scheme can be used in real-time control applications. The upper bound for the computational time is equal to the used control time step of the model, which is 5 minutes for the Fivelingo model. The computational time depends on the specific computer that is being used for the optimization, see Table 4-5. Since the computational time is depending on the amount of iterations that the solver needs to converge to a solution, these are also listed in the results.

4.3 Test bed

The model has been implemented in Real-Time Control Tools (RTC-Tools) (Schwanenberg and Becker, 2012), an open source software package developed by the research institute Deltares for modelling routing processes for real-time control applications. The used model is based on a diffusive wave model, and consists of storage nodes and branches on a staggered grid. RTC-Tools is also used to configure and evaluate the objective function. RTC-Tools has an internal solver, and is also equipped with an adjoint model to evaluate the gradient with respect to continuous input variables. However, this adjoint model is not configured such that it can

Table 4-6: Summary of experiments

Experiment	# of time instants	Initial guess for time instants
1	2	no initial guess (default: zeros)
2	4	no initial guess (default: zeros)
3	8	no initial guess (default: zeros)
4	16	no initial guess (default: zeros)
5	2	equidistant over prediction horizon
6	4	equidistant over prediction horizon
7	8	equidistant over prediction horizon
8	16	equidistant over prediction horizon

deal with time instants input and the reduced amount of variables (compared to continuous optimization) that results from Time Instant Optimization. Therefore the optimization is done using an external solver in MATLAB. The correct gradient vector, that represents the gradient with respect to both continuous variables and time instants is used in MATLAB's solver `fmincon`.

A converter calculates the control input of the discrete variables, based on the optimized pump schedule derived from continuous time instant optimization, see Figure 3-2. This control input can then be used RTC-Tools to evaluate the objective function.

4.4 Experiments

Nine open-loop experiments will be used to test various TIO-MPC controllers on the simplified MPC model of a real-world system. The first eight experiments use a TIO-MPC controller; experiments 1-4 do not use an initial guess for the time instants. Instead they use the default value, which is a vector of zeros. Since the large nonlinear optimization problem can have many local minima, these experiments are used to test whether the solver `fmincon` gets stuck in local minima near $t = 0$.

In experiments 5-8, an initial guess of the time instants is supplied to the solver. Pairs of time instants are divided equidistant over the prediction horizon N_p. These experiments will be used to test whether supplying a (rough estimate of an) initial guess leads to a better solution i.e. lower objective function value and tighter control on water levels.

A ninth experiment is an open-loop test using a continuous optimization, in which all pumps can have continuous setting between their lower and upper bound. This does not represent the actual control, since the polder pumps can only be operated on or off and hence have much less degrees of freedom. However it is interesting to compare the performance in terms of control accuracy and computational time with the TIO-MPC controllers.

An overview of the experiments is listed in Table 4-6. The input data for the RTC-Tools model related to disturbances and boundary conditions is available in the form of time series, which have been extracted from a Delft-Flood Early Warning System (FEWS) flow forecasting system (FEWS Noorderzijlvest) prior to this study. Delft-FEWS is an operational forecasting

platform, that can be used to link data and models in real time, producing forecasts (Werner et al., 2013). The following time series are available:

- Observed sea water levels at Delfzijl (period 2012-02-17 - 2012-02-27)
- Lateral (aggregated) inflows for the nodes, based on a SOBEK-RR (rainfall-runoff) model. This model used precipitation data from rainfall gauges and radar data as input in the pre-processing. (period 2012-02-17 - 2012-02-27)

The prediction model uses hindcast data as predictions, with 48-hour time series of rainfall and inflows. The control time step is 5 minutes, resulting in a prediction horizon $N_{\mathrm{p}} = 576$. If we would decide for every time step whether or not to switch the pumps on or off, this would lead to a very large combinatorial optimization problem. For a single discrete variable, there are 2^{576} different combinations.

There are five control variables (four pumping stations and one gate) resulting in an optimization problem of 2880 dimensions if we apply a continuous optimization. However, a continuous optimization would not give a proper solution as this is not the correct physical representation of the discrete dynamics. We will use Time Instant Optimization for the two discrete input variables, which will reduce the number of dimensions of the optimization problem. Because the time instants are continuous, we can use an efficient gradient-based solver which makes the use of a derivative-free pattern search algorithm unnecessary. Also, the discrete and continuous control input variables can easily be combined in the optimization.

In order to test the performance of the hybrid TIO-MPC controllers, experiments 1-8 were carried out using various amounts of time instants per discrete variable: 2, 4, 8 and 16. It is expected that a larger number of time instants will yield better results regarding the objective function. All tests are performed for one time step, with equal stopping criteria for the solver.

4.5 Results and discussion

This chapter summarizes the various experiments from Section 4.4 that were performed to test the hybrid TIO-MPC algorithms on a model of an existing water system, in order to answer the research questions. The experiment with continuous input variables is used to show the performance if all pumps could have been operated continuously. Then, the pumps can change their settings at every time step, with a pump flow ranging between Q_{min} and Q_{max}. For this reason, there are many more degrees of freedom for the solver and the controller is able to exert a much tighter control compared to their hybrid counterparts. However, this does not represent the actual physical behaviour of the controller, since in reality some pumps are operated only on or off. Also, the TIO-MPC algorithm limits the amount of allowed state changes. In general, it can be observed that the hybrid TIO-MPC algorithms is able to adequately optimize both continuous and discrete elements, and that using more time instants generally leads to better performance in terms of the objective function value. This is because the controller can use more periods to pump, and these pumping intervals can be timed strategically e.g. discharging as much as possible through the gate at low tide.

The simulated water levels and control inputs of the continuous optimization are given in Figure 4-3. The results of the eight TIO-MPC experiments are given in Figures 4-3 through

4-11. The results and performance indicators of all experiments are summarized in Table 4-7. The percentages between brackets indicate the relative volume of water that is discharged at the outlet by either the gate or one of the two pumps.

It can be observed in Figures 4-3 through 4-11 that for all experiments, node H137 has a larger deviation in the water levels compared to node H024. This can be explained by the difference in level-storage relationship, where node H137 has less relatively less storage and therefore the water levels rises faster. It is interesting to note that for this system, the use of more time instants in experiments 1-4 and 5-8 leads to a tighter control of the polder nodes H024 and H137, but it also causes a rise in the percentage of pumped volume at the outlet, see Table 4-7. Using even more time instants would in theory give the controller more degrees of freedom, which would allow the controller to use a more tactical planning with respect to the water level at sea. A different tuning of the controller could also be used to prioritize gate flow even more. However, extensive tuning of this water system is out of the scope of this research.

4.5.1 Optimization using a user-defined initial point

Supplying the solver with an educated guess of an initial point for the optimization may prevent the solver from getting "stuck" in the first (suboptimal) local minimum it encounters. Experiments 1-4 and 5-8 can be compared to test the importance of a user-supplied initial point.

It can be observed that in both experiments with 2 and 4 time instants, a comparable value of the objective function and RMSE is found, while the time instants are different, see Table 4-7. In experiments 3 and 4 (using 8 and 16 time instants, respectively) the pump intervals of the discrete pumps are scheduled at the beginning of the prediction horizon, which might indicate that the solver got stuck in a local minimum, see Figures 4-6 and 4-7. Especially experiment 4 results in poor performance, with a high objective function value due to the large deviation from setpoint of the water levels in storage nodes H024 and H137. If we compare experiments 3-4 with 7-8, we can observe that a better solution in terms of the objective function J, RMSE and CPU time is found in the experiments where an initial solution has been supplied, which indicates the added value of a user-supplied initial point.

The experiments show that the use of equidistantly divided time instants as an initial guess seems to yield better results than an optimization without a user-supplied initial guess. In practice it will be hard, if not impossible to come up with a reasonable initial guess. If the computational time allows for it, one could use multi-start optimization to increase the chance of finding a better feasible solution.

4.5.2 Computational time

One of the benefits of using TIO-MPC instead of MLD-MPC is that the amount of optimization variables is reduced significantly. Recall that for MLD-MPC, the solver has to decide at every time step within the prediction horizon whether or not to change its state, which leads to a large combinatorial optimization problem. The reduced amount of variables, and the fact that an efficient gradient-based solver can be used for the hybrid optimization, pays

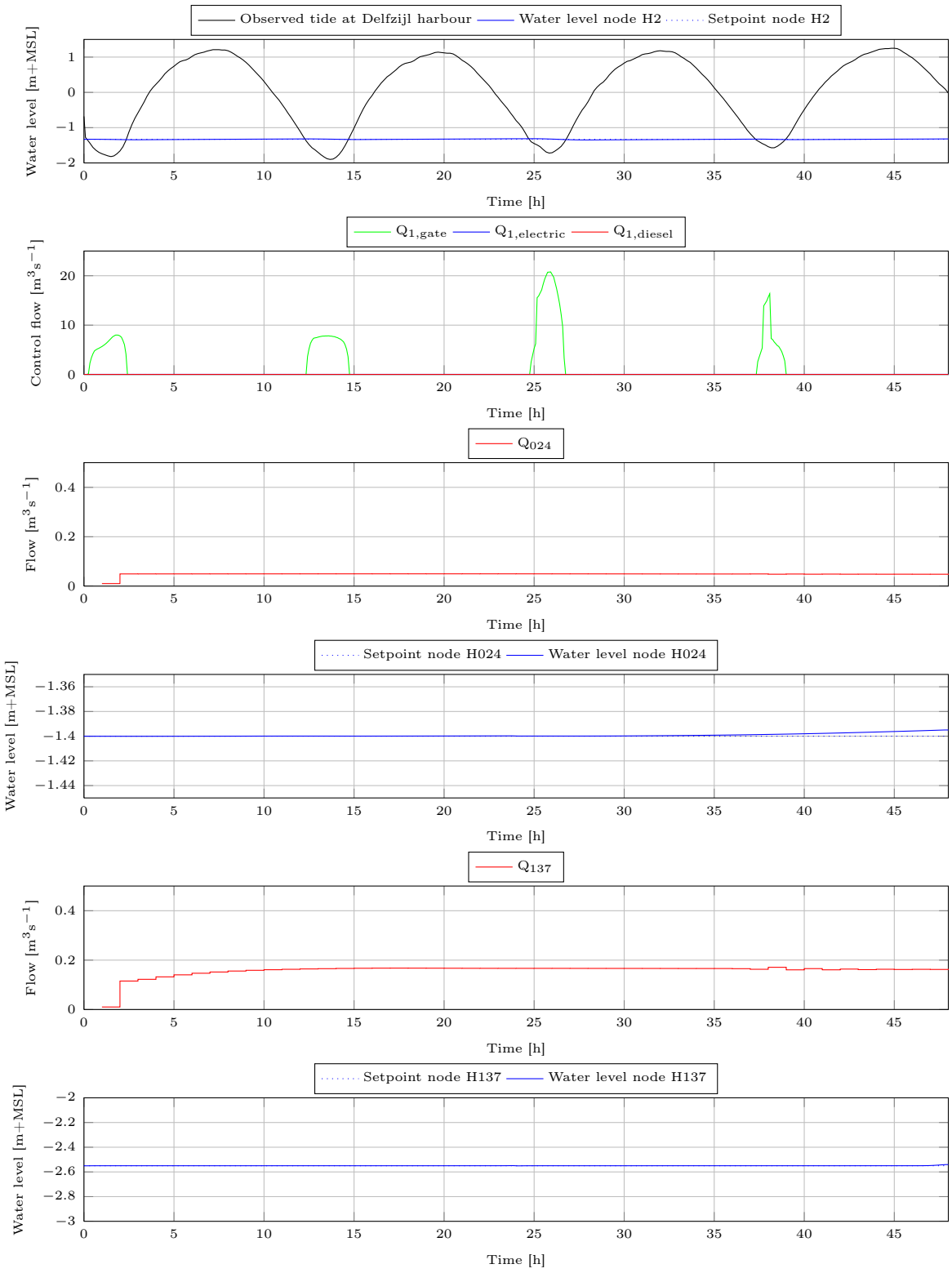


Figure 4-3: Results of continuous optimization

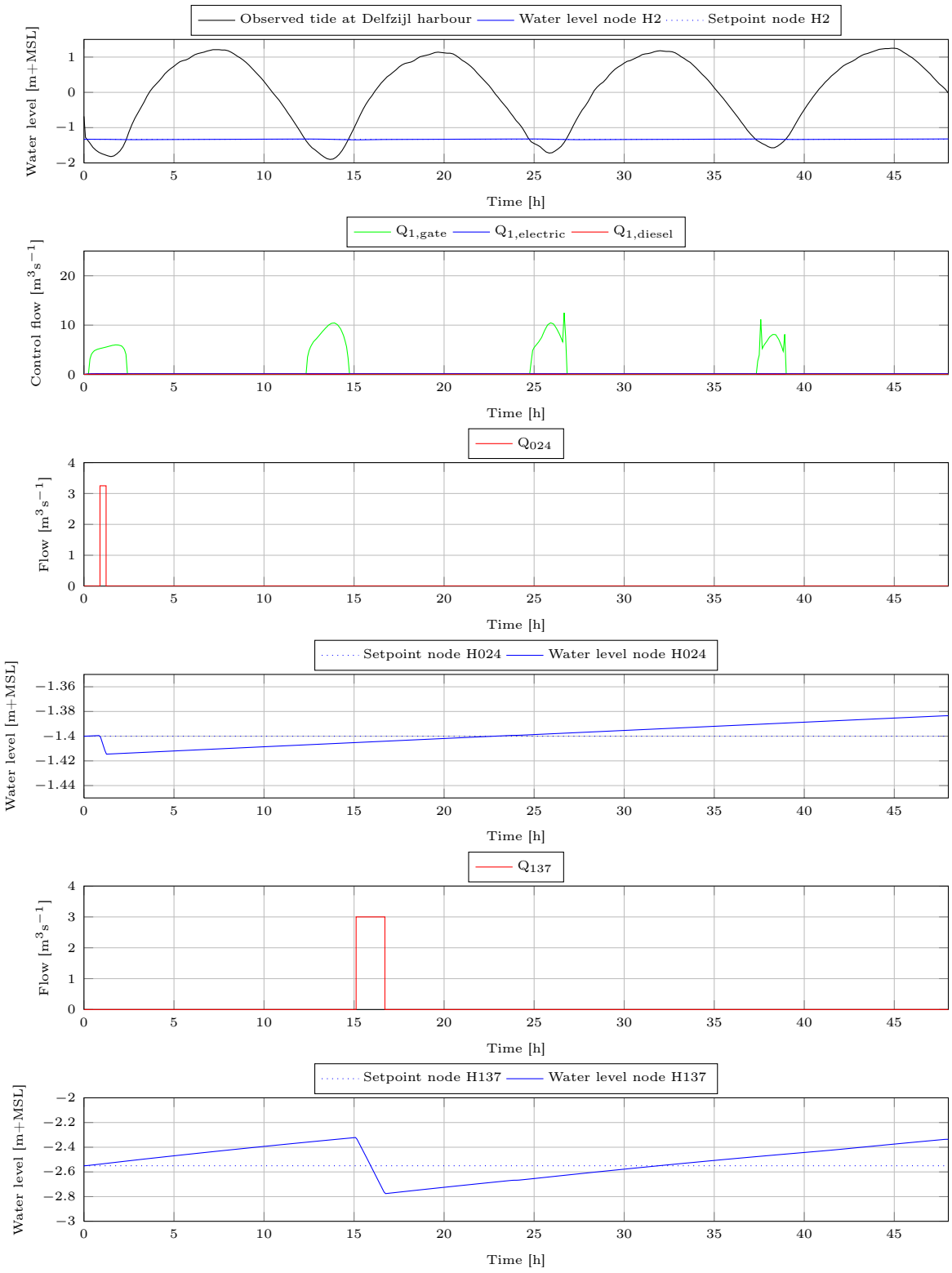


Figure 4-4: Results of experiment 1

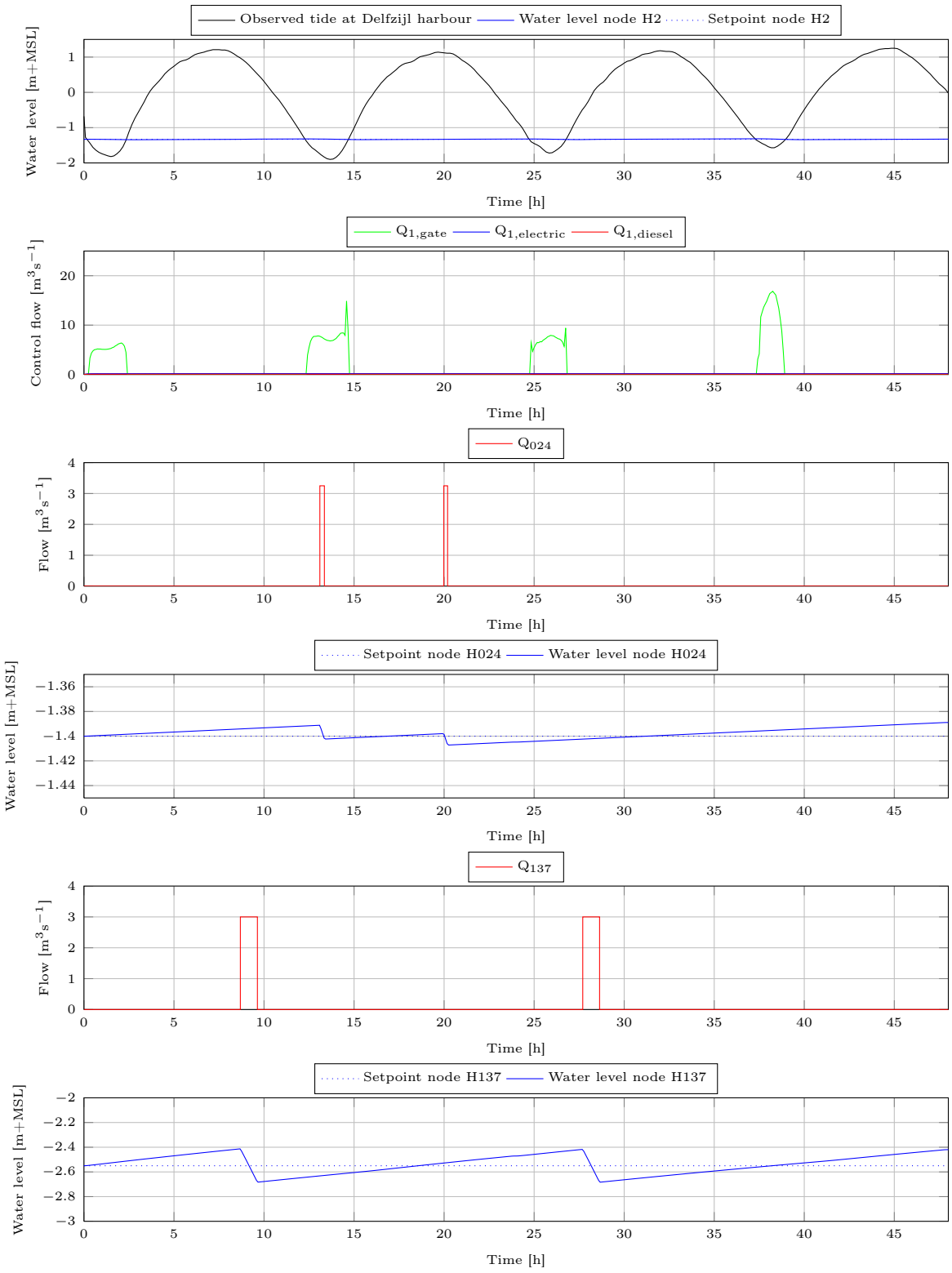


Figure 4-5: Results of experiment 2

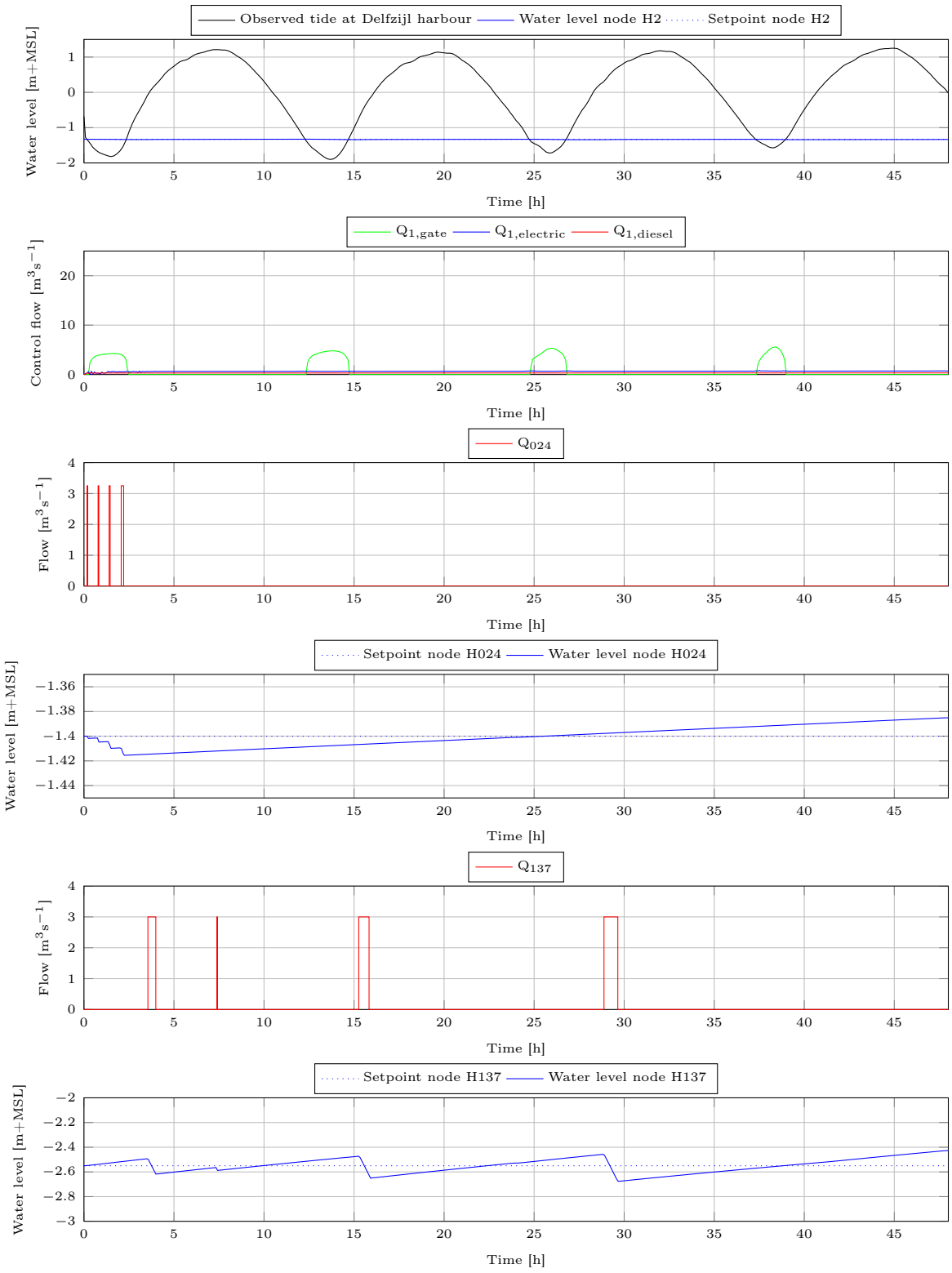


Figure 4-6: Results of experiment 3

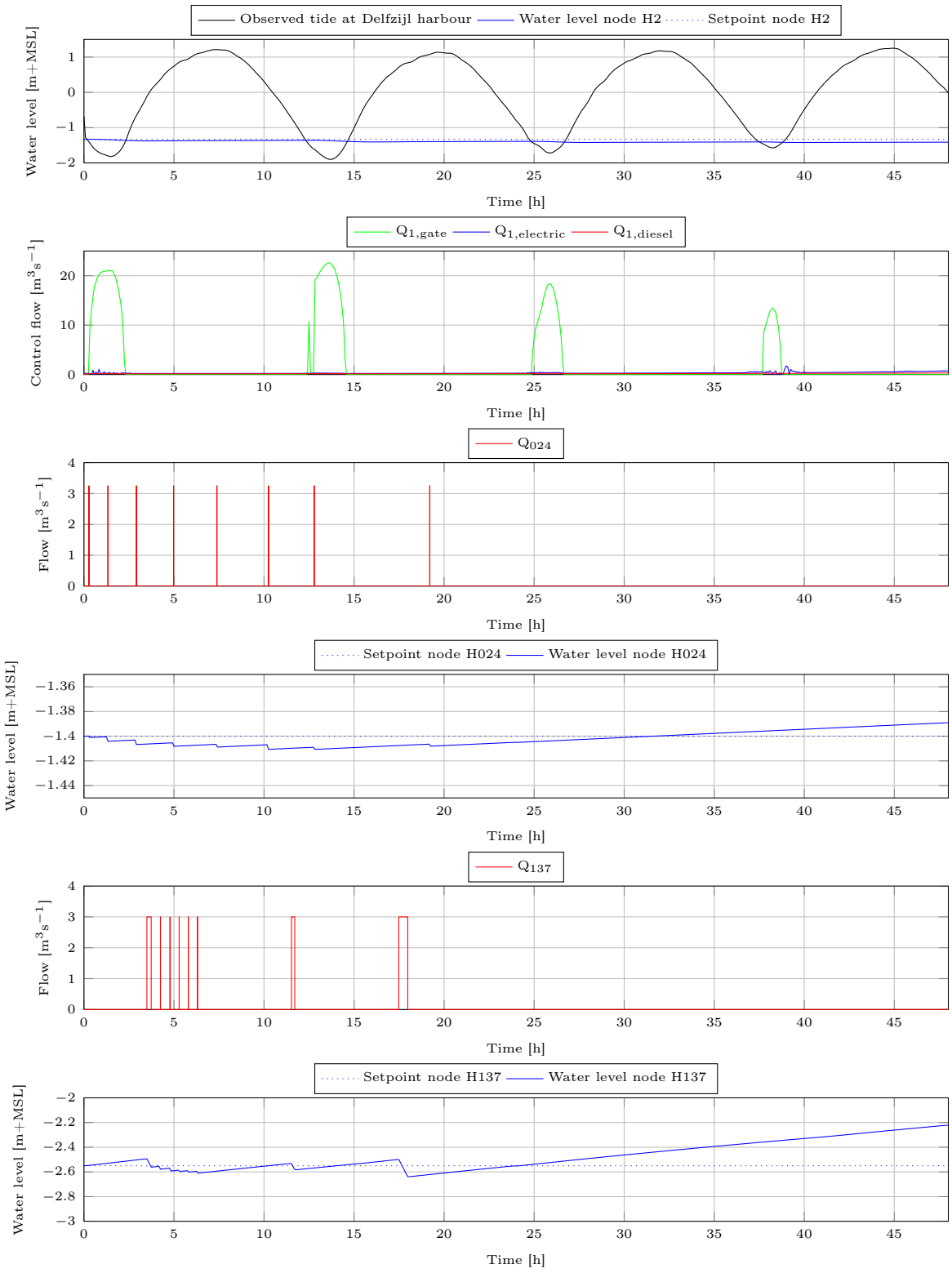


Figure 4-7: Results of experiment 4

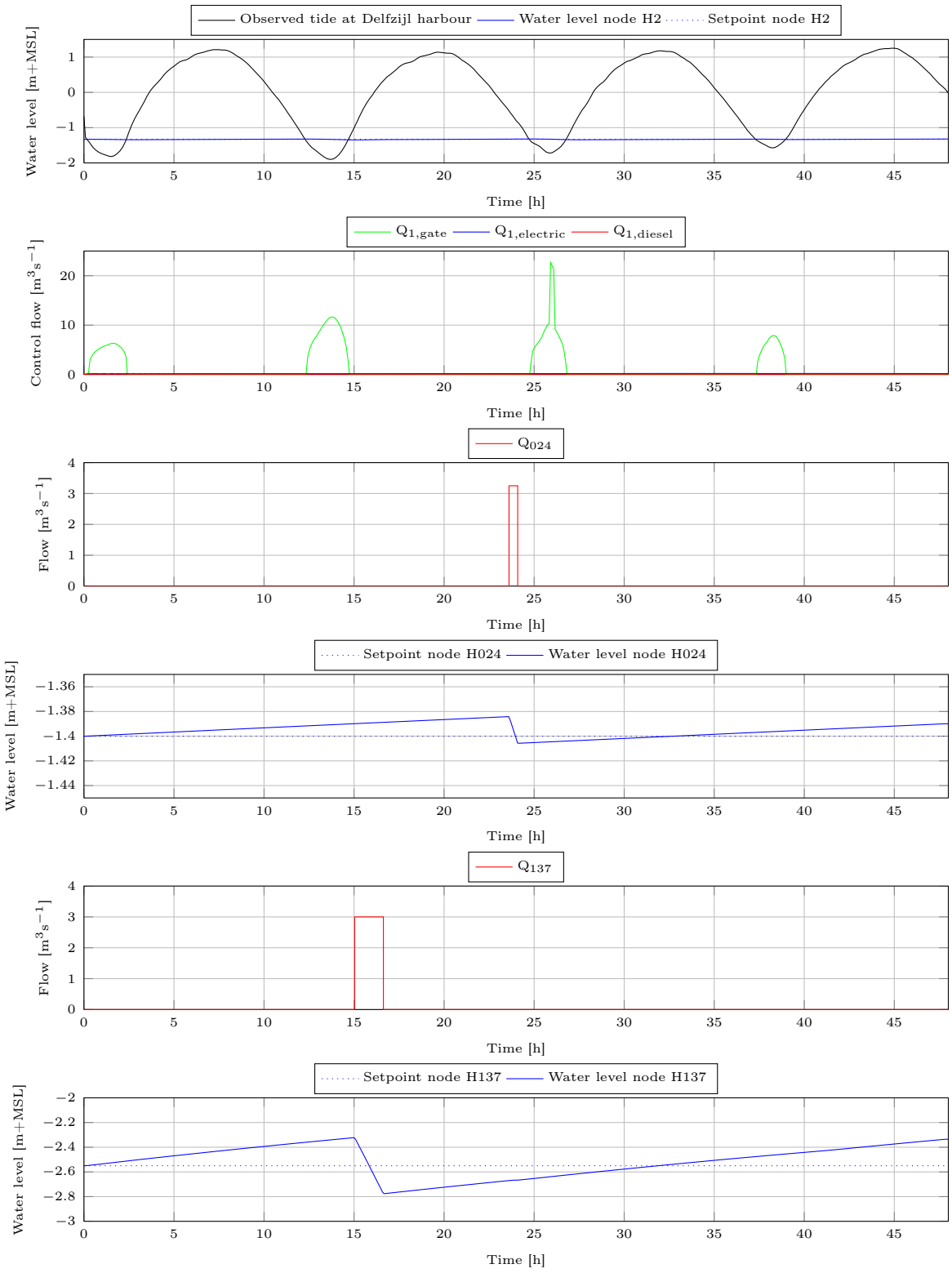


Figure 4-8: Results of experiment 5

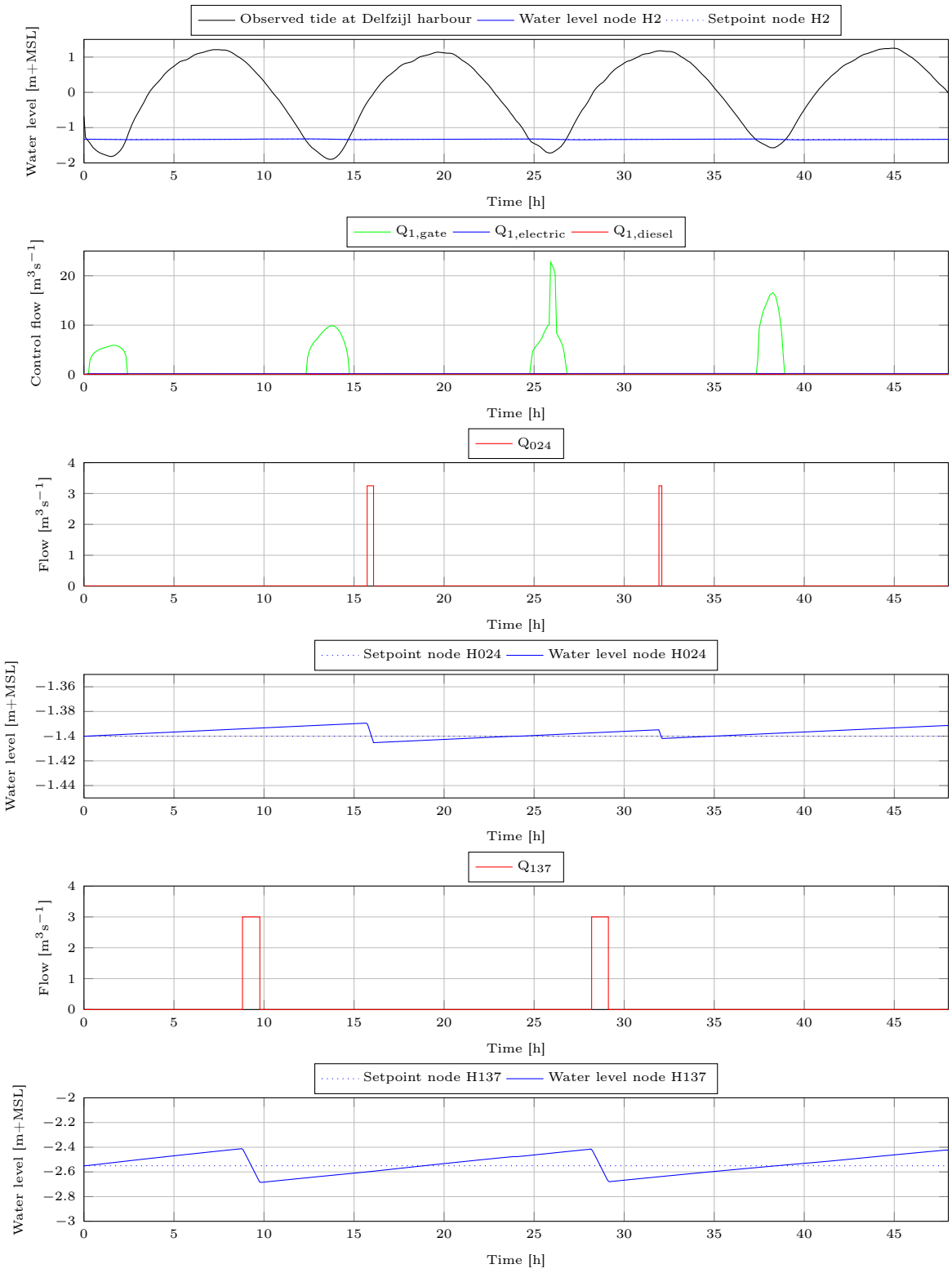


Figure 4-9: Results of experiment 6

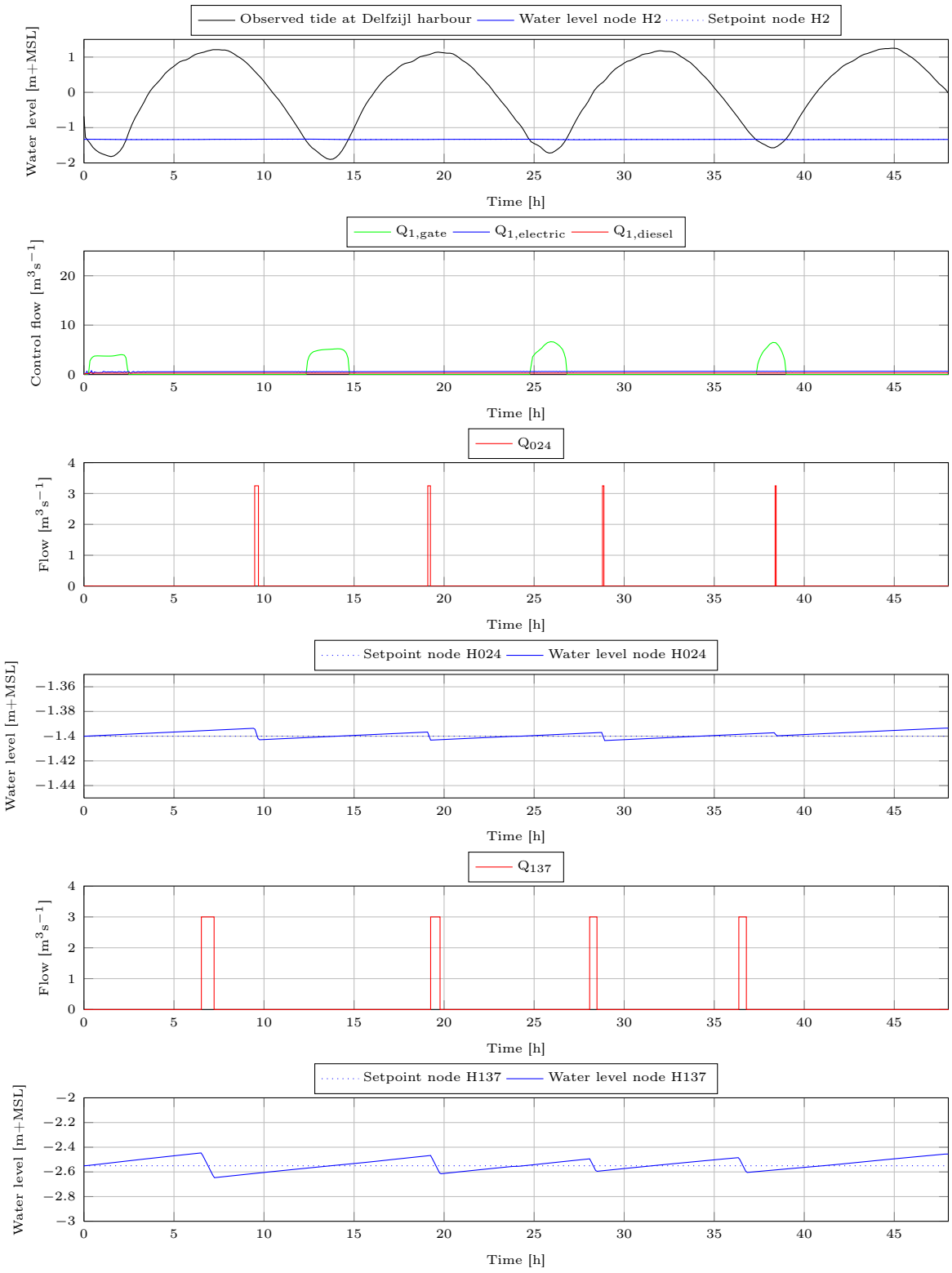


Figure 4-10: Results of experiment 7

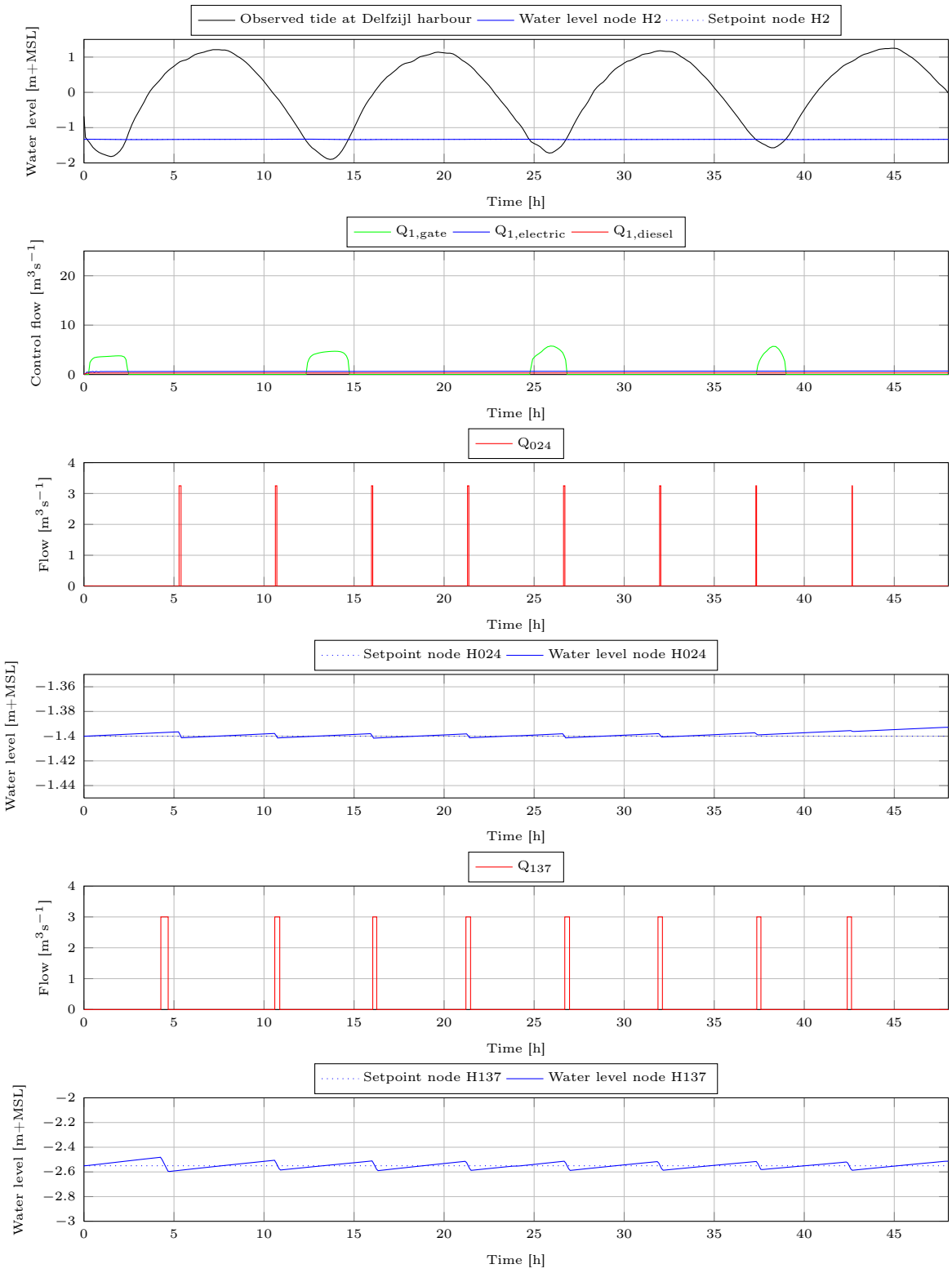


Figure 4-11: Results of experiment 8

off in the computational time. All TIO-MPC experiments have computational times under 300 seconds, which is the control time step of the used controller and thus the upper limit for RTC applications. The only difference between the eight TIO-MPC experiments was the initial guess of the time instants. From Table 4-7 we can conclude that for these experiments, the runs with a user-supplied initial guess for the time instants are generally faster. This may indicate that the initial guess was quite good, and less iterations were needed to converge to a (local) minimum. These iterations take time, since MATLAB and RTC-Tools have to send data back and forth via the MEX-file.

One would expect a higher computational time when more time instants are being used. In the experiments, using more time instants does not automatically lead to more CPU time. If we analyse the results of experiments 5-8, we can see that there is no obvious relationship between the amount of used time instants and the computational time. The computational time is correlated with the number of needed iterations, see Table 4-7. The amount of needed iterations depends on how fast the solver converges to a solution, which again emphasizes the dependency on a good initial guess.

Table 4-7: Results of hybrid TIO-MPC controller versus continuous optimization for the Fivelingo water system, without supplying an initial guess for time instants to the solver

Experiment	# variables	J [−]	# iter	CPU time [s]	RMSE of water level [m]			Discharged volume [m^3]				
					H02	H024	H137	$Q_{1,gate}$	$Q_{1,electric}$	$Q_{1,diesel}$	Q_{024}	Q_{137}
Continuous	*2880*	*38.5*	110	*636.6*	*0.0080*	*0.0015*	*0.0010*	*828.2 (96.4%)*	*22.9 (2.7%)*	*7.7 (0.9%)*	*23.1*	*82.3*
1	1732	1995.4	69	97.9	0.0056	0.0090	0.1297	680.0 (82.0%)	110.5 (13.3%)	38.7 (4.7%)	13.2	57.5
2	1736	740.7	140	204.0	0.0066	0.0051	0.0772	738.4 (83.3%)	109.5 (12.4%)	38.4 (4.3%)	17.9	67.7
3	1744	519.5	67	101.4	0.0054	0.0087	0.0582	402.0 (41.8%)	379.2 (39.4%)	180.1 (18.7%)	14.6	68.7
4	1760	2432.4	56	87.6	0.0677	0.0066	0.1383	1235.0 (80.3%)	188.5 (12.3%)	114.6 (7.4%)	18.1	43.8
5	1732	1994.5	54	76.7	0.0063	0.0074	0.1297	699.3 (82.3%)	110.8 (13.0%)	39.2 (4.6%)	18.8	57.5
6	1736	742.2	73	106.1	0.0072	0.0047	0.0773	784.9 (83.9%)	111.6 (11.9%)	38.8 (4.1%)	20.1	68.2
7	1744	367.2	23	35.1	0.0060	0.0028	0.0471	448.7 (46.5%)	347.2 (35.9%)	170.0 (17.6%)	21.8	72.0
8	1760	190.0	37	57.6	0.0050	0.0024	0.0241	402.6 (42.0%)	376.3 (39.3%)	179.0 (18.7%)	21.3	79.1

Chapter 5

Conclusions and recommendations

This chapter gives an overview of the conclusions drawn in this thesis, and recommendation for possible future work.

5.1 Conclusions

In this thesis, a TIO-MPC controller with continuous time instants was introduced in Chapter 3. The continuous time instants allowed the use of efficient gradient-based solvers. These solvers require a gradient vector of the objective function. To obtain this gradient, reverse-mode algorithmic differentiation was applied to the reservoir test case of Section 2.4.1 in Section 3.3.2. The controller was first tested in closed-loop on the simple test case, using a linear reservoir model and a pump that is operated either on or off. The test cases showed that the TIO-MPC controller is, similar to any other MPC algorithm, capable of controlling a system with conflicting control objectives.

In Chapter 4, various TIO-MPC schemes were introduced and tested on a MPC model of an existing water system. This hydraulic model was readily available in RTC-Tools, a novel software package developed by Deltares for modelling routing processes for real-time control applications. RTC-Tools has an internal solver, and is also equipped with an adjoint model to evaluate the gradient with respect to continuous input variables. However this adjoint model is not configured such that it can deal with time instants input and the reduced amount of variables (compared to continuous optimization) that results from Time Instant Optimization.

To allow flexibility, the optimization is done using an external solver from MATLAB's Optimization Toolbox. The correct gradient vector, that represents the gradient with respect to both continuous variables and time instants was used in MATLAB's solver `fmincon`. A converter calculates the control input of the discrete variables, based on the optimized pump schedule derived from continuous time instant optimization. This control input is can then be used RTC-Tools to evaluate the objective function.

Various TIO-MPC schemes, using different amounts of time instants, were introduced and tested in eight open-loop experiments. From these experiments, the following conclusions can be drawn:

5.1.1 Optimization of hybrid systems using gradient-based TIO-MPC

Both the academic test case and the Fivelingo test case demonstrated that the gradient-based TIO-MPC controller is perfectly capable of optimizing a hybrid system that involves both continuous and discontinuous structure operations. Since some pumps in the test case can only be operated on or off, and TIO-MPC prescribes the number of allowed on/off switches, it goes without saying that the degrees of freedom for applying a tight control are limited compared to continuous pumps. When optimizing more time instant over the prediction horizon:

- The controller is able to exert a tighter control on water levels, since it is allowed to pump more often. It can anticipate on water level deviations by pumping in short intervals. The average deviation from setpoint will be lower compared to a case where one is only allowed to pump for one interval. This will result in lower objective function values.
- More time instants allow for a more strategical pump scheduling over the prediction horizon, anticipating better on overall system dynamics e.g. water levels at sea (exploiting 'free' gate flow).

5.1.2 Local optima

Whenever optimizing large nonlinear optimization problem, there is a risk that an obtained optimum is a local minimum. In the ideal case the solver finds the global minimum, provided that there exists a solution. The experiments of the Fivelingo water system were used to test, amongst other goals, the influence of supplying different initial solutions to the solver. The experiments showed that lower objective function values are obtained when an initial guess is supplied to the solver. When no initial guess was supplied, the solver seemed to get stuck in (suboptimal) local minima near the origin, resulting in high objective function values which indicate poor control performance. This behaviour was especially encountered for the experiments with a larger amount of time instants.

There is no golden rule for this initial guess, but an initial guess that gave promising results was when pairs of time instants were divided equidistant over the prediction horizon. This way the solver is forced to look for a solution, looking a bit further away than local minima near the origin.

One could also try to synchronize the initial guess with predicted peaks of the disturbances and with periods with low tide. This might point the solver in the right direction, because these are generally the moments that a pump will be activated.

An additional strategy is the use multi-start optimization. By running several optimizations from different initial starting points, the chance of finding a better feasible solution increases. This will increase the computational time since many more function evaluations are needed. The control time step gives an upper bound for the available computational time.

5.1.3 Computational time

From the experiments, there seems to be no obvious relationship between the amount of time instants and the computational time. This might be due to the fact that the total amount of input variables is not that different for the eight experiments (see Table 4-7). The computational time is quite dependent on the initial guess since this determines the amount of, a priori unknown, iterations. The solver converges faster to a solution if (by chance) a good initial solution has been supplied.

In some cases, having more time instants seems to be even decreasing the computational time. This can be caused by a (potentially good) initial guess, or by the constraints and pre-described order of time instants that decreases the feasible region of a possible solution.

In the current setup with RTC-Tools and an external solver in MATLAB, the two programs needs to send data back and forth via a MEX-file. Each iteration takes time because of the bookkeeping and data transfer. A potential option to further decrease the computational time is to put the whole optimization in RTC-Tools or any other low-level programming language using another, potentially more efficient internal solver than MATLAB's `fmincon`.

5.1.4 Number of time instants

The amount of time instants that are used in the optimization determines the performance of the model. The experiments showed that using more time instants generally leads to a tighter control. The choice for a certain amount of time instants will have to be made based on both physical and computational considerations. In the end, the choice for a certain amount of time instants is a trade-off between computational effort and control accuracy. The maximum number of time instants that *can* be used depends on the setup of the optimization problem. Constraints that prescribe e.g. the domain of the time instants and/or a minimum time between state switching limit the feasible amount of time instants that can be used within the prediction horizon. If such constraints are not given, the solver could use many more time instants to increase control accuracy. The open-loop tests from Section 4.4

An interesting case is when a TIO-MPC algorithm optimizes a sequence of control input variables $\tilde{u}^{k+i|k}$ for $i = 0, \ldots, N_\mathrm{p} - 1$, where every i-th interval contains two continuous time instants. With two time instants per interval as in Figure 3-4b, and the converter from Section 4.3, the solver finds a continuous control input value for every i-th step. Hence, this TIO-MPC algorithm is in theory equivalent to a continuous MPC algorithm (but with twice as much variables to optimize).

It is hard to say in advance how many time instants *should* be used. For the Fivelingo water system, only open-loop tests were performed. Closed-loop tests using different amounts of time instants may give more insights into the system behaviour. In addition, it might be interesting to extend the model, using parallel computations with different amount of time instants where the model switches between different TIO-MPC schemes.

5.2 Recommendations

Time Intant Optimization MPC is a promising technique for the control of systems with both continuous and discrete elements. It allows the algorithm to step away from computationally heavy mixed-integer schemes. The work presented here made a step forward by implementing the use of continuous time instants. This allows the use of efficient gradient-based solvers, which are widely available in both commercial and open source software packages. The recommendations for future work are as follows:

- The test case of the Fivelingo model focussed only on computational issues and the performance of an open loop model. Since the receding horizon principle is used with MPC, only the first step of the optimized sequence of steps is carried out, before another optimization starts at the new time step. Closed-loop testing might give more feeling for the number of time instants to be used over a given prediction horizon. It will be interesting to see how these algorithms translate into the actual pump scheduling of a system such as the Fivelingo water system. Further research into the closed-loop performance of a TIO-MPC model is recommended.
- The use of gradient-based solvers for the optimization of the hybrid model speeds up the optimization. Further research could focus on extending the current model by using a parallel optimization, using different amounts of time instants. The scheme could than switch between different TIO-MPC schemes, depending on which scheme gives the lowest objective function value.

It is recommended to extend RTC-Tools with some features that would enable Time Instant Optimization. Using the internal solver in RTC-Tools will be beneficial for the computational time, as it does not require the RTC-Tools/MATLAB model coupling that slows down the optimization. Some additions that were modelled in MATLAB in this work, will have to be programmed into RTC-Tools:

- The converter that translates continuous time instants input into regular control input. This input can then be used for the evaluation of the objective function.
- The adjoint has to be adjusted such that it gives the gradient with respect to time instants input.

Using a low-level programming language instead of MATLAB will probably speed up the optimization. However, this work focussed on MATLAB to allow flexibility in modelling.

The use of equidistantly divided pairs of time instants gave promising results in this work. Future research of a similar test case could try to link the initial guess of the time instants to predicted disturbance peaks and periods of low tides. From a practical point of view, it could be reasoned that these are the moments that the polder pumps are operating. Using these points in time might be a well educated guess that points the solver into the right direction, however the solution will depend on the quality of the predictions.

A different strategy is to let the algorithm do more number crunching by running a multi-start optimization procedure. If a solution exists, the solver is not guaranteed to find the

global minimum. If one has no clue about a good initial guess of the time instants and the computational time and the length of the control time step allow it, a multi-start procedure will increase the chance of finding a better (or at least acceptable) solution.

Good candidates for other fields of application of gradient-based TIO-MPC would be processes with continuous and discrete elements that include storage and accumulation of matter, whether it is water, oil or traffic. If a system can somehow be described with mass balances, it might be possible to use continuous time instants. The control of large sewer systems would make an interesting test case: these systems contain many pumps, spatially distributed storage, and many constraints regarding storage capacity and allowed overflow frequencies.

Appendix A

Linear MPC

A.1 State-Space Model

The linear system under consideration can be written in a state-space representation, which reads:

$$x^{k+1} = A^k x^k + B_{\mathrm{u}}^k u^k + B_{\mathrm{d}}^k d^k, \tag{A.1}$$

$$y^k = C x^k, \tag{A.2}$$

where x^k is the state vector, u^k the input vector, d^k the disturbance vector, y^k the output of the system, at time k, A the system matrix, B_{u} the control input matrix, B_{d} the disturbance matrix and C the output matrix.

When the state-space model is extended over the prediction horizon N_{p} and the initial state x^k and inputs are known, the future output can be calculated. Extending the model yields:

$$\begin{aligned}
x^{k|k} =& x^k, \\
x^{k+1|k} =& A^{k|k} x^k + B_{\mathrm{u}}^{k|k} u^{k|k} + B_{\mathrm{d}}^{k|k} d^{k|k}, \\
x^{k+2|k} =& A^{k+1|k} x^{k+1|k} + B_{\mathrm{u}}^{k+1|k} u^{k+1|k} + B_{\mathrm{d}}^{k+1|k} d^{k+1|k}, \\
=& A^{k+1|k} A^{k|k} x^k + A^{k+1|k} B_{\mathrm{u}}^{k|k} u^{k|k} + A^{k+1|k} B_{\mathrm{d}}^{k+1|k} d^{k|k} + B_{\mathrm{u}}^{k+1|k} u^{k+1|k}, \\
& + B_{\mathrm{d}}^{k+1|k} d^{k+1|k}, \\
& \vdots
\end{aligned} \tag{A.3}$$

In compact notation:

$$\mathbf{X} = \mathbf{A} x^k + \mathbf{B}_{\mathrm{u}} \mathbf{U} + \mathbf{B}_{\mathrm{d}} \mathbf{D}, \tag{A.4}$$

$$\mathbf{Y} = \mathbf{C} \mathbf{X}, \tag{A.5}$$

where:

$$\mathbf{X} = \begin{bmatrix} x^{k|k} \\ x^{k+1|k} \\ \vdots \\ x^{k+N_\mathrm{p}|k} \end{bmatrix}, \ \mathbf{U} = \begin{bmatrix} u^{k|k} \\ u^{k+1|k} \\ \vdots \\ u^{k+N_\mathrm{p}-1|k} \end{bmatrix}, \ \mathbf{D} = \begin{bmatrix} d^{k|k} \\ d^{k+1|k} \\ \vdots \\ d^{k+N_\mathrm{p}-1|k} \end{bmatrix},$$

$$\mathbf{A} = \begin{bmatrix} I \\ A^{k|k} \\ A^{k+1|k}A^{k|k} \\ \vdots \\ A^{k+N_\mathrm{p}-1|k}A^{k+N_\mathrm{p}-1|k}\dots A^{k|k} \end{bmatrix},$$

$$\mathbf{B}_\mathrm{u} = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ B_\mathrm{u}^{k|k} & 0 & & 0 \\ A^{k+1|k}B_\mathrm{u}^{k|k} & B_\mathrm{u}^{k+1|k} & & 0 \\ \vdots & & \ddots & \vdots \\ A^{k+N_\mathrm{p}-1|k}A^{k+N_\mathrm{p}-2|k}\dots A^{k+1|k}B_\mathrm{u}^{k|k} & & & B_\mathrm{u}^{k+N_\mathrm{p}-1|k} \end{bmatrix},$$

$$\mathbf{B}_\mathrm{d} = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ B_\mathrm{d}^{k|k} & 0 & & 0 \\ A^{k+1|k}B_\mathrm{d}^{k|k} & B_\mathrm{d}^{k+1|k} & & 0 \\ \vdots & & \ddots & \vdots \\ A^{k+N_\mathrm{p}-1|k}A^{k+N_\mathrm{p}-2|k}\dots A^{k+1|k}B_\mathrm{d}^{k|k} & & & B_\mathrm{d}^{k+N_\mathrm{p}-1|k} \end{bmatrix},$$

$$\text{and } \mathbf{C} = \begin{bmatrix} C & 0 & \cdots & 0 \\ 0 & C & & \vdots \\ \vdots & & C & 0 \\ 0 & \cdots & 0 & C \end{bmatrix}.$$

A.2 MPC formulated as a quadratic programming problem

A general quadratic objective function, where a penalty is imposed on states $\mathbf{X}$ and input $\mathbf{U}$ is:

$$J = \mathbf{X}^\top \mathbf{Q}\mathbf{X} + \mathbf{U}^\top \mathbf{R}\mathbf{U}, \tag{A.6}$$

in which $\mathbf{Q}$ and $\mathbf{R}$ are cost weight matrices:

$$\mathbf{Q} = \begin{bmatrix} q & 0 & \dots & 0 \\ 0 & q & & \vdots \\ \vdots & & q & 0 \\ 0 & \dots & 0 & q \end{bmatrix}; \mathbf{R} = \begin{bmatrix} r & 0 & \dots & 0 \\ 0 & r & & \vdots \\ \vdots & & r & 0 \\ 0 & \dots & 0 & r \end{bmatrix}.$$

With the output $\mathbf{Y}$ equal to the state $\mathbf{X}$ and taking into account a desired set point x_{sp}, the objective function becomes:

$$J = (\mathbf{X} - x_{\text{sp}})^\top \mathbf{Q}(\mathbf{X} - x_{\text{sp}}) + \mathbf{U}^\top \mathbf{R}\mathbf{U}. \tag{A.7}$$

For linear systems, the state x^k is linearly dependent of u^k, so the quadratic cost function can be written as:

$$J = \mathbf{U}^\top \mathbf{H}\mathbf{U} + \mathbf{f}^\top \mathbf{U}, \tag{A.8}$$

where:

$$\mathbf{H} = \mathbf{B}_{\text{u}}^\top \mathbf{Q}\mathbf{B}_{\text{u}} + \mathbf{R}, \tag{A.9}$$

$$\mathbf{f} = 2(x^\top \mathbf{A}^\top + \mathbf{D}^\top \mathbf{B}_{\text{d}}^\top - x_{\text{sp}}^\top)\mathbf{Q}\mathbf{B}_{\text{u}}. \tag{A.10}$$

$\mathbf{H}$ is a symmetric matrix (the Hessian) and $\mathbf{f}$ is a vector. Linear input and state constraints imply (inequality) constraints on the control input u^k, which can be expressed in the following formulation:

$$\mathbf{A}_c\mathbf{U} \leq \mathbf{b}_c. \tag{A.11}$$

The optimization problem now becomes:

$$\begin{aligned} \min_{\text{U}} \quad & \mathbf{U}^\top \mathbf{H}\mathbf{U} + \mathbf{f}^\top \mathbf{U} \\ \text{subject to} \quad & \mathbf{A}_c\mathbf{U} \leq \mathbf{b}_c. \end{aligned} \tag{A.12}$$

(A.12) is a quadratic programming problem, for which dedicated solvers exist. In MATLAB, this optimization can be solved numerically using the `quadprog` function from the Optimization Toolbox, which solves the quadratic programming problem:

$$\min_{U} \quad \frac{1}{2}U^\top \overline{H}U + f^\top U \tag{A.13}$$

$$\text{subject to} \quad AU \leq b \tag{A.14}$$

$$A_{\text{eq}}U = b_{\text{eq}} \tag{A.15}$$

$$lb \leq U \leq ub \tag{A.16}$$

By using $\overline{H} = 2\mathbf{H}$, optimization problem (A.12) can be solved.

A.3 Application to linear reservoir model

A simple explicit linear reservoir model can be described by a mass balance consisting of an inflow or *disturbance* and a controlled outflow (see also Section 2.4.1). The state-space model for this reservoir model is given by:

$$h^{k+1} = h^k + \frac{T}{A_{\text{s}}}(d^k - u^k), \tag{A.17}$$

where h is the water level [m], T is the control length step [s], A_{s} is the storage area of the reservoir [m^2], d is the uncontrolled inflow or *disturbance* [$\text{m}^3\,\text{s}^{-1}$], u is a controlled outflow [$\text{m}^3\,\text{s}^{-1}$] and k is the step counter [$-$].

Table A-1: Parameters of linear MPC model

Parameter	Value	Unit
A^k	1	-
T	300	s
A_s	10000	m^2
B_u	-T/Area	$\mathrm{s\,m}^{-2}$
B_d	T/Area	$\mathrm{s\,m}^{-2}$
N_p	15	-
h_sp	3	m
h_min	2.6	m
h_max	3.4	m
U_min	0	$\mathrm{m}^3\,\mathrm{s}^{-1}$
U_max	3	$\mathrm{m}^3\,\mathrm{s}^{-1}$

Rewriting the above in the shape of equations A.1 and A.2, with $x^k = h^k$, $A^k = 1$, $B_\mathrm{u} = T/A_\mathrm{s}$ and $B_\mathrm{d} = -T/A_\mathrm{s}$ yields:

$$x^{k+1} = A^k x^k + B_\mathrm{u}^k u^k + B_\mathrm{d}^k d^k, \tag{A.18}$$

$$y^k = x^k. \tag{A.19}$$

When the water level is bounded between h_min and h_max, this can be expressed in the inequality constraint of the quadratic programming problem.

$$h_\mathrm{min} \leq Ax(k) + B_\mathrm{u}U + B_\mathrm{d}D \leq h_\mathrm{max} \tag{A.20}$$

$$\begin{bmatrix} B_\mathrm{u} \\ -B_\mathrm{u} \end{bmatrix} U \leq \begin{bmatrix} h_{max} - Ax(k) - B_\mathrm{d}D \\ -h_{min} + Ax(k) + B_\mathrm{d}D \end{bmatrix} \tag{A.21}$$

The following parameters have been chosen for the reservoir model, see Table A-1:

A.4 Simulation results

The reservoir model was modelled using MATLAB. The model behaviour for different cost weight matrices can be observed by looking at the output. It can be seen from Figures A-1 and A-2 that the pump starts pumping before the disturbance takes place, as expected with any Model Predictive Controller. Since the penalty on water level deviation is very low in Figure A-2 the controller does not keep a very tight control on the water level. When the penalty on water level deviation from setpoint is set to a higher value, it can be seen from Figure A-1 that the controller exerts a much tighter control on the water level deviation from setpoint. The controller operates at the constraints: the pump is pumping at maximum capacity before the disturbance takes place. The water level deviation from set point is smaller compared to the second case. In both cases, the constraints on the water level are respected. If we define the deviation from set point as the error $e = h - h_{setpoint}$, the maximum error is 0.22 m for the first case, and 0.40 m for the latter.

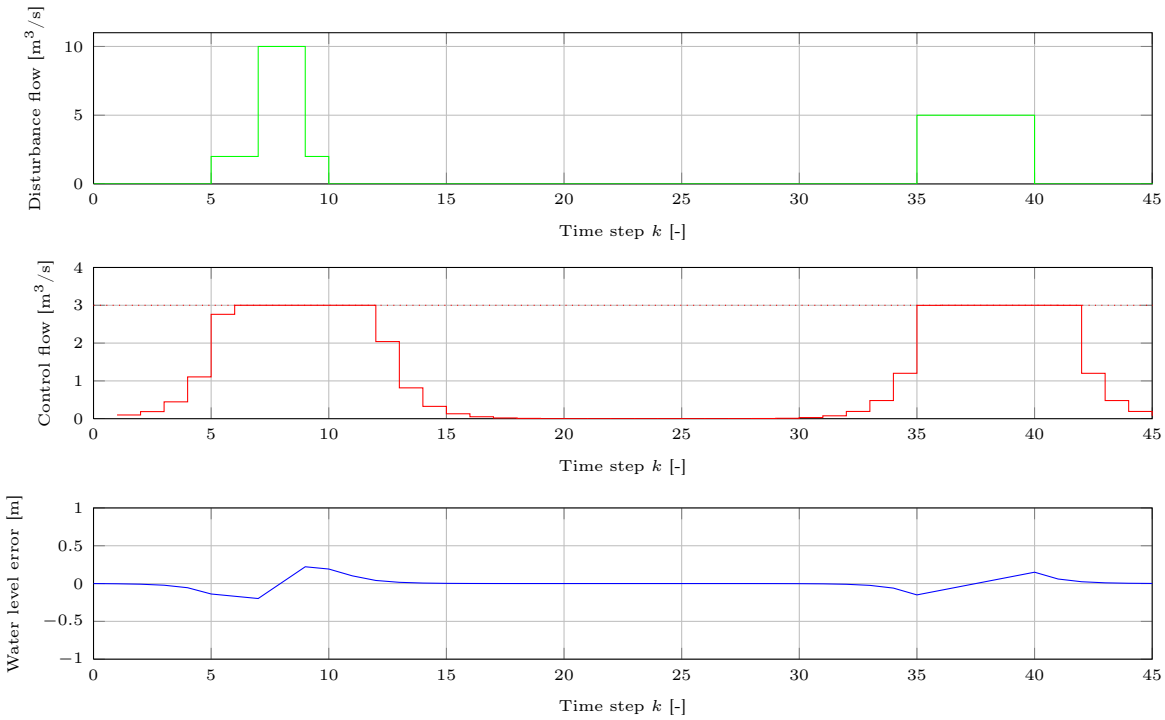


Figure A-1: Simulation results for $q = 10$ and $r = 0.01$

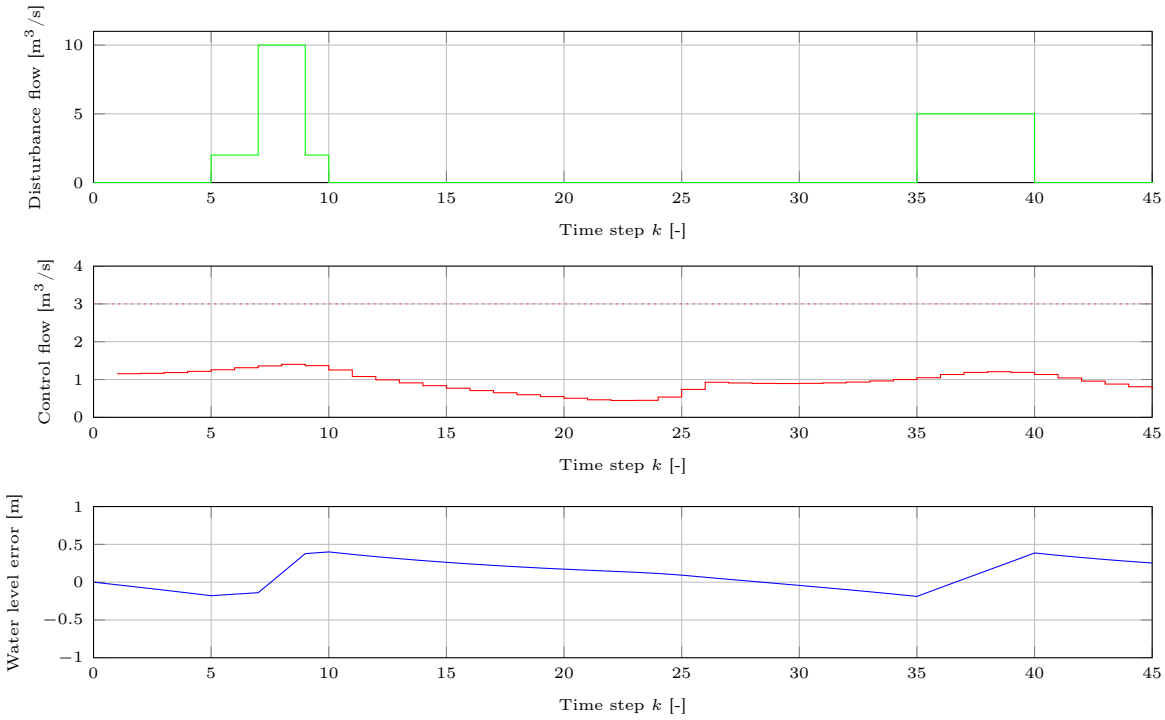


Figure A-2: Simulation results for $q = 0.1$ and $r = 0.01$

References

Anand, A., Galelli, S., Samavedham, L., and Sundaramoorthy, S. (2013). Coordinating multiple model predictive controllers for the management of large-scale water systems. *Journal of Hydroinformatics*, 15(2):293–305.

Ankum, P. (2002). *Design of open-channels and hydraulic structures*. Lecture notes. Delft University of Technology, Delft, the Netherlands.

Autodiff (2013). Community portal for automatic differentiation. http://www.autodiff.org. (Accessed February 20, 2013).

Bennett, C. H. (1973). Logical reversibility of computation. *IBM journal of Research and Development*, 17(6):525–532.

Bischof, C., Khademi, P., Mauer, A., and Carle, A. (1996). ADIFOR 2.0: Automatic differentiation of Fortran 77 programs. *Computational Science & Engineering, IEEE*, 3(3):18–32.

Bischof, C. H., Hovland, P. D., and Norris, B. (2008). On the implementation of automatic differentiation tools. *Higher-Order and Symbolic Computation*, 21(3):311–331.

Blanco, T. B., Willems, P., Chiang, P.-K., Haverbeke, N., Berlamont, J., and Moor, B. D. (2010). Flood regulation using nonlinear model predictive control. *Control Engineering Practice*, 18(10):1147 – 1157.

Blanco, T. B., Willems, P., De Moor, B., and Berlamont, J. (2008). Flooding prevention of the Demer river using model predictive control. In *Proceedings of the 17th World Congress The International Federation of Automatic Control*.

Boyd, S. and Vandenberghe, L. (2004). *Convex Optimization*. Cambridge University Press.

Camacho, E. F. and Bordons, C. (1999). *Model Predictive Control*. Springer London.

Chow, V. T., Maidment, D. R., and Mays, L. W. (1988). *Applied hydrology*. McGraw-Hill series in water resources and environmental engineering. McGraw-Hill Book Company.

De Schutter, B. (1999). Optimal traffic light control for a single intersection. In *American Control Conference, 1999. Proceedings of the 1999*, volume 3, pages 2195–2199. IEEE.

Deltares (2013). *SOBEK Technical Reference Manual*. Delft, the Netherlands.

Diehl, M., Ferreau, H. J., and Haverbeke, N. (2009). Efficient numerical methods for nonlinear MPC and moving horizon estimation. *Lecture Notes in Control and Information Sciences*, 384:391–417.

Errico, R. M. (1997). What is an adjoint model? *Bulletin of the American Meteorological Society*, 78(11):2577–2591.

Fournier, D. A., Skaug, H. J., Ancheta, J., Ianelli, J., Magnusson, A., Maunder, M. N., Nielsen, A., and Sibert, J. (2012). AD model builder: Using automatic differentiation for statistical inference of highly parameterized complex nonlinear models. *Optimization Methods and Software*, 27(2):233–249.

Giering, R. and Kaminski, T. (1998). Recipes for adjoint code construction. *ACM Transactions on Mathematical Software (TOMS)*, 24(4):437–474.

Glanzmann, G., von Siebenthal, M., Geyer, T., Papafotiou, G., and Morari, M. (2005). Supervisory water level control for cascaded river power plants. In *Proceedings of the Hydropower Conference, Stavanger, Norway*.

Goedbloed, A., Galelli, S., and Schwanenberg, D. (2011). Assessing the effectiveness of a real-time control method for Marina Reservoir management. In *19th International Congress on Modelling and Simulation. Modelling and Simulation Society of Australia and New Zealand*, pages 4036–4042.

Griewank, A. and Walther, A. (2008). *Evaluating derivatives: principles and techniques of algorithmic differentiation*. Society for Industrial and Applied Mathematics (SIAM).

Kothare, M. V., Balakrishnan, V., and Morari, M. (1996). Robust constrained model predictive control using linear matrix inequalities. *Automatica*, 32(10):1361–1379.

Lewin, J. (2001). *Hydraulic gates and valves: in free surface flow and submerged outlets*. Thomas Telford Publishing, 2nd edition.

Lin, M.-H., Tsai, J.-F., and Yu, C.-S. (2012). A review of deterministic optimization methods in engineering and management. *Mathematical Problems in Engineering*, 2012.

MacArthur, R. and DeVries, J. J. (1993). Introduction and application of kinematic wave routing techniques using HEC-1. Training Document TD-10, US Army Corps of Engineers Institute for Water Resources Hydrologic Engineering Center (HEC).

Maciejowski, J. M. (2002). *Predictive control with constraints*. Pearson education.

Malaterre, P.-O. and Baume, J.-P. (1998). Modeling and regulation of irrigation canals: existing applications and ongoing researches. In *Systems, Man, and Cybernetics, 1998. 1998 IEEE International Conference on*, volume 4, pages 3850–3855. IEEE.

MATLAB (2012). *version 7.14.0.739 (R2012a)*. The MathWorks Inc., Natick, Massachusetts.

Morari, M. and Barić, M. (2006). Recent developments in the control of constrained hybrid systems. *Computers & chemical engineering*, 30(10):1619–1631.

Negenborn, R. R., van Overloop, P.-J., Keviczky, T., and De Schutter, B. (2009). Distributed model predictive control for irrigation canals. *Networks and Heterogeneous Media*, 4(2):359–380.

Neidinger, R. D. (2010). Introduction to automatic differentiation and MATLAB object-oriented programming. *SIAM Review*, 52(3):545–563.

Noorderzijlvest, W. (2013). Waterschap noorderzijlvest. http://www.noorderzijlvest.nl. (Accessed June 10, 2013).

Pearlmutter, B. A. and Siskind, J. M. (2008). Reverse-mode AD in a functional framework: Lambda the ultimate backpropagator. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 30(2):7.

Plank, J., Blaha, J., Cordingley, J., Wilinska, M. E., Chassin, L. J., Morgan, C., Squire, S., Haluzik, M., Kremen, J., Svacina, S., et al. (2006). Multicentric, randomized, controlled trial to evaluate blood glucose control by the model predictive control algorithm versus routine glucose management protocols in intensive care unit patients. *Diabetes care*, 29(2):271–276.

Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P. (2007). *Numerical recipes: The art of scientific computing*. Cambridge University Press, 3rd edition.

Qin, S. J. and Badgwell, T. A. (1997). An overview of industrial model predictive control technology. In *AIChE Symposium Series*, volume 93, pages 232–256. New York, NY: American Institute of Chemical Engineers, 1971-c2002.

Qin, S. J. and Badgwell, T. A. (2000). An overview of nonlinear model predictive control applications. In *Nonlinear model predictive control*, pages 369–392. Springer.

Rivierenland, W. (2013). Werken aan duurzaamheid: Klimaatactieprogramma (KAP). http://www.klimaatactieprogramma.nl/energiebesparing-en-beperking-uitstoot-broeikasgassen. (Accessed April 27, 2013).

Rockafellar, R. T. and Wets, R. J.-B. (1998). *Variational Analysis: Grundlehren der Mathematischen Wissenschaften*, volume 317. Springer Verlag.

Şahin, A. and Morari, M. (2010). Decentralized model predictive control for a cascade of river power plants. In *Intelligent Infrastructures*, pages 463–485. Springer.

Schwanenberg, D. and Becker, B. (2012). *RTC-Tools: Software Tools for Modeling Real-Time Control. Reference Manual*. Deltares, Delft, the Netherlands.

Schwanenberg, D., Galelli, S., and Sheret, I. (2011). Nonlinear model predictive control for heterogeneous process models in water resources. In *World Congress*, volume 18, pages 10565–10570.

Schwanenberg, D. and van Heeringen, K.-J. (2012). 2011.06.08 Pilot RTC Noorderzijlvest. Technical report, Flood Control 2015.

Shaw, E. M. (1994). *Hydrology in Practice*. Chapman & Hall, 3rd edition. Reprinted in 2002 by Nelson Thornes Ltd.

Stelling, G. S. and Duinmeijer, S. P. A. (2003). A staggered conservative scheme for every Froude number in rapidly varied shallow water flows. *International Journal for Numerical Methods in Fluids*, 43(12):1329–1354.

van Ekeren, H. (2010). Hybrid model predictive control of the Rhine-Meuse delta. Master's thesis, Delft University of Technology, Delft, the Netherlands.

van Ekeren, H., Negenborn, R. R., van Overloop, P. J., and De Schutter, B. (2011). Hybrid model predictive control using time-instant optimization for the Rhine-Meuse delta. In *2011 IEEE International Conference on Networking, Sensing and Control (ICNSC)*, pages 215 –220.

van Overloop, P., Negenborn, R., Weijs, S., Malda, W., Bruggers, M., and De Schutter, B. (2010a). Linking water and energy objectives in lowland areas through the application of model predictive control. In *Control Applications (CCA), 2010 IEEE International Conference on*, pages 1887–1891. IEEE.

van Overloop, P. J. (2006). *Model Predictive Control On Open Water Systems*. PhD thesis, Delft University of Technology, Delft, the Netherlands.

van Overloop, P. J., Negenborn, R. R., De Schutter, B., and van de Giesen, N. C. (2010b). Predictive control for national water flow optimization in the Netherlands. In *Intelligent Infrastructures*, pages 439–461. Springer.

Verma, A. (1999). ADMAT: Automatic differentiation in MATLAB using object oriented methods. In *SIAM Interdiscplinary Workshop on Object Oriented Methods for Interoperability*, pages 174–183.

Wahlin, B. T. (2004). Performance of model predictive control on ASCE test canal 1. *Journal of irrigation and drainage engineering*, 130(3):227–238.

Wang, L. (2009). *Model Predictive Control System Design and Implementation Using MATLAB®*. Advances in Industrial Control. Springer-Verlag London Limited.

Werner, M., Schellekens, J., Gijsbers, P., van Dijk, M., van den Akker, O., and Heynert, K. (2013). The Delft-FEWS flow forecasting system. *Environmental Modelling & Software*, 40(0):65–77.

Xu, M. (2013). *Real-time control of combined water quantity and quality in open channels*. PhD thesis, Delft University of Technology, Delft, the Netherlands.

Xu, M. and Schwanenberg, D. (2012). Comparison of sequential and simultaneous model predictive control of reservoir systems. In *Proceedings of the International Conference on Hydroinformatics, (HIC 2012), Hamburg, Germany, 14-18 July 2012*.

Xu, M., Van Overloop, P. J., Van De Giesen, N. C., and Stelling, G. S. (2010). Real-time control of combined surface water quantity and quality: polder flushing. *Water science and technology*, 61(4):869–878.

Zabiri, H. and Samyudia, Y. (2006). A hybrid formulation and design of model predictive control for systems under actuator saturation and backlash. *Journal of Process Control*, 16(7):693–709.

Zijlema, M. (2011). *Computational modelling of flow and transport*. Lecture notes. Delft University of Technology, Delft, the Netherlands.

Glossary

List of Acronyms

LMPC	linear Model Predictive Control
NMPC	nonlinear Model Predictive Control
MPC	Model Predictive Control
TIO-MPC	Time Instant Optimization Model Predictive Control
CFL	Courant-Friedrichs-Lewy
SV	De Saint-Venant
AD	automatic differentiation
QP	quadratic programming
LP	linear programming,
NLP	nonlinear programming
IP	integer programming
MILP	mixed-integer linear programming
MINLP	mixed-integer nonlinear programming
NZV	Noorderzijlvest
RTC	real-time control
MLD	mixed logical dynamical
FEWS	Flood Early Warning System
RMSE	root mean square error
VFD	variable-frequency drive

RTC-Tools Real-Time Control Tools

SQP sequential quadratic programming