Mixed integer predictive control

Mixed integer predictive control

DCT report: 2001-49

Student: M.B.G.Cloosterman

September 2001

Supervisor: dr. ir. H.A. van Essen

Contents

Chapter	Title	Page
	Introduction	3
1.	Model predictive control	4
1.1	Concepts	4
1.2	Relative optimisation scheme	6
1.2.1	Constraints	7
1.3	Absolute optimisation scheme	8
1.3.1.	Constraints	8
1.4	Final remarks	9
2.	Mixed logical dynamical systems	10
2.1	Introduction	10
2.2	Logical variables	11
2.3	Examples	12
2.3.1	Logical constraints on the state	12
2.3.2	Piecewise linear systems	13
2.3.3	Discrete inputs	15
3.	Model with three water tanks	16
3.1	Overall model	16
3.2	Simulation models	17
4.	Mixed integer predictive control	19
4.1	Model with non-continuous inputs	19
4.1.1	Logical statements and system equations	19
4.1.2	Constraints for relative optimisation	20
4.1.3	Results	22
4.2	Model with a discrete input	23
4.2.1	Logical statements and system equations	23
4.2.2	Constraints for absolute optimisation	25
4.2.3	Results	27
5.1	Conclusion	29
5.2	Recommendations	29
	Literature	30

Appendices

1	List of symbols
2	Constraints for piecewise linear system
3	Determination of the parameters
4	Reliability of the linearised and the discrete linearised systems
5	Results of the model with non-continuous inputs
6	Φ and Γ matrices of the model with non continuous inputs
7	Φ and Γ matrices of the model with a discrete input
	The next appendices are given a separate report with appendices
8	M-files: linearisation and discretisation
9	Model with non-continuous inputs.
9a	Mipwaterbak
9b	Function file to compute discrete model, weighing matrices and prediction matrix Y
9c	Function file to compute optimisation variables and matrices
9d	Function file which gives the model
10	Model with a discrete input
10a	Mipwaterbak
10b	Function file to compute discrete model, weighing matrices and prediction matrix Y
10c	Function file to compute optimisation variables and matrices
11	Function file which computes the setpoints

Introduction

When a certain output of a system is desired, a controller can be implemented to reach these goals. For many applications it is useful the controller can stop the machine or a part of the machine during some time instead of running machines at an inefficient level. This feature is for instance useful in assembly lines, with different parallel machines. The controller will turn off a machine when the throughput descends, other machines will take over a part of the work of the closed machine and their operating point will be more efficient. A model predictive controller can deal with this problem, because it computes the most optimal operating point of all machines.
Hardly any machine will turn out softly or continuously but turn out abrupt, when a minimal value is reached. This behaviour can be modelled with logical constraints. These determine when a machine is on or off and can be written in mathematical (inequality) equations. MPC controls the system with use of an optimisation problem, which uses these constraints.
To investigate if the demands written above are possible a model predictive controller, which can handle with logical constraints will be designed and simulated.

First MPC and logical constraints were studied. For logical constraints this means to rewrite them into inequalities and check if the new equation gives the same results as the logical constraint. Then the rewriting of logical constraints and the theory of MPC are combined in mixed integer predictive control (MIPC). A matlab-file according to this theory is written. This file uses mixed integer quadratic programming (miqp), an existing Matlab-function. To test the Matlab-file a simulation is done on a model with logical constraints, which are expressed in inequalities. This implemented model is controlled by MIPC. The results of the simulation are interpreted. The results show if the Matlab-file and the constraints are implemented correct for MIPC.
This report follows the different steps that were made during this traineeship. In chapter 1 the model predictive controller will be introduced. Chapter 2 handles about mixed integer control. Therefore logical constraints will be used, explained and rewritten in a useful notation for MIPC. Chapter 3 describes two models, with logical constraints, which are used in simulations. To make these simulations possible the logical constraints have to be rewritten into inequalities and implemented in a Matlab-file, which is described in chapter 4. In this chapter the results of the models are also interpreted. This report will end with a conclusion and recommendations.

Chapter 1: Model predictive control

1.1 Concepts

Model Predictive Control (MPC) is a control strategy that uses a model of the process that will be controlled to obtain the control inputs by minimising an objective function.

Equation 1.1 gives a linear, discrete time model. In figure 1 a schematic representation of this model is given. This scheme is based on a system with single input and single output.

$$x(k+1) = \mathbf{\Phi} x(k) + \mathbf{\Gamma} u(k)$$
$$y(k+1) = \mathbf{C} x(k+1) + \mathbf{D} u(k) \qquad [1.1]$$

$x(k)$: current state vector
$u(k)$: inputvector

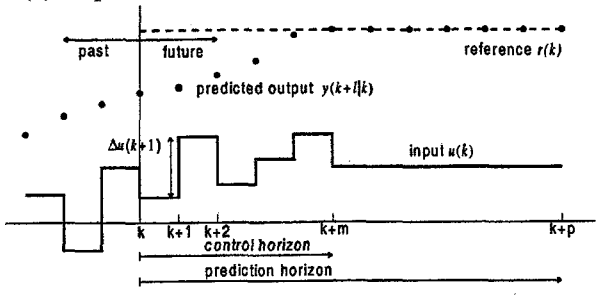


Figure 1: Concept of Model Predictive Control

At the present time k, the response of the output y is predicted over the prediction horizon with a length of p samples. This prediction is calculated with past inputs, current model states, latest process measurements, proposed future inputs and possibly predicted disturbances. The variables are allowed to vary over the control horizon with a length of m samples. The inputs (absolute or relative) are computed such that the future deviations between the predicted and desired output are minimised. This will be done with an optimisation routine, i.e. Quadratic Programming. Of the computed optimal control moves, only the first value is implemented and the algorithm repeats the same procedure for the next sample.
To minimise the future deviations the following quadratic objective function is used:

$$\min_{\Delta u(k+1)\ldots\Delta u(k+m)} \sum_{l=2}^{p} \left[Q\left(y(k+l \mid k) - r(k+l)\right)\right]^2 + \sum_{l=1}^{m} \left[R\left(\Delta u(k+l)\right)\right]^2 \qquad [1.2]$$

m: control horizon
p: prediction horizon
$y(k)$: process output at sample k
$y(k+l)$: process output at sample k predicted l steps ahead..
$y(k+l|k)$: estimation of the process output y:$(k+l)$.
$r(k)$: reference signal at sample k
$r(k+l)$: reference signal computed for sample k and l steps predicted ahead
Δu: vector with new inputs (change of the input from equation 1.1)
Q: weights for the relevance of the output deviations
R: weights for the actions of the manipulated variables

The weighing matrices are chosen diagonal. Both matrices depend on the number of repetitions of the algorithm. Further, **R** depends on the control horizon; **Q** depends on the prediction horizon.

An advantage of MPC in comparison with standard feedback is the explicit use of a finite prediction horizon in the control problem. This allows MPC controller to take control action at the current time step in response to the future behaviour of the system, even if the current error is zero. Another advantage is the capability to handle with constraints. Physical, safety and performance demands or laws can be taken into account.

The most important disadvantages of the MPC are the computational demand and the performance of MPC strongly depends on tuning parameters, like weighting factors, length of horizons and sample intervals and the accuracy of the model.
The computational demanding makes it only useful for relatively slow processes, e.g. valves in water circuits or burning furnaces.

Two optimisation schemes of MPC can be distinguished, a relative and an absolute. The relative optimisation makes use of input changes to compute the optimal value. The absolute optimisation uses the input itself. Both optimisation schemes will be explained in the next paragraphs. In general the relative optimisation can schematically be presented, as shown in figure 2.

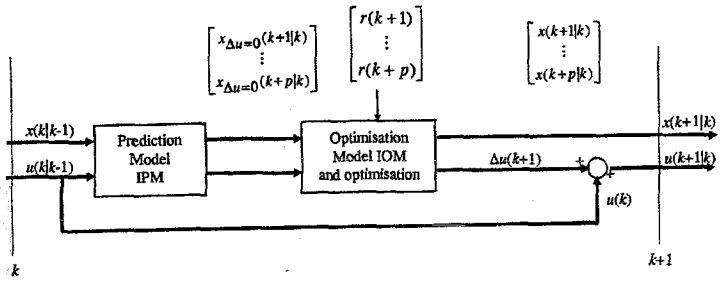


Figure 2: Schematic view of a relative MPC controller

In this figure the new estimated state $x(k+1|k)$ and input u$(k+1|k)$are calculated. For this calculation the old estimation of the current sample of the state $x(k|k-1)$ and the input u$(k|k-1)$ are used. Further the estimated states with Δu=0 and the reference signal for the current state are used in the optimisation model. The estimated state with Δu=0 computes the behaviour of the system when nothing is done.
The new input is calculated as the computed change of the input together with the estimated input of the sample k.
The absolute optimisation is schematically given by:

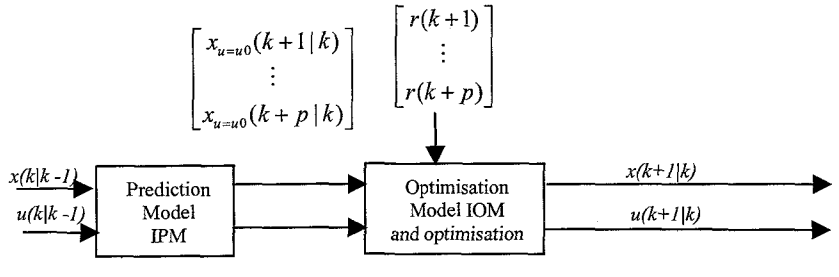


Figure 3: Schematic view of an absolute MPC controller

For this optimisation the same inputs are used at the left side as in the relative scheme. A difference is that the estimated state is computed with u=$u0$, which means again that the input is not changed. The last difference with the relative scheme is that this optimisation gives the new estimated input as an output, so no extra computations have to be done.

1.2 Relative optimisation scheme

The relative optimisation uses the input changes as optimisation parameters. The minimal and maximal values of the inputs are given as linear constraints. A Quadratic Programming Problem will be used to compute the optimal values.

$$x_s^p(k+1) = x_{s,\Delta u=0}^p(k+1) + x_{s,\Delta u}^p(k+1) \quad [1.3]$$

$$y_o^p(k+1) = y_{o,\Delta u=0}^p(k+1) + y_{o,\Delta u}^p(k+1) \quad [1.4]$$

In these equations x_s^p is a (s*p) column. The different states are given by s and p correspondents to the p samples over the prediction horizon. The same holds for the value o in the output, it gives the different measured outputs. The output is therefore a (o*p) column. The sample is given by (k+1).

According to equation 1.1 and with known initial values, the MPC problem can be written as:

$$\begin{bmatrix} x_s(k+2\,|\,k) \\ x_s(k+3\,|\,k) \\ \vdots \\ x_s(k+m+1\,|\,k) \\ x_s(k+m+2\,|\,k) \\ \vdots \\ x_s(k+p+1\,|\,k) \end{bmatrix} = \underbrace{\begin{bmatrix} \Phi^2 \\ \Phi^3 \\ \vdots \\ \Phi^{m+1} \\ \Phi^{m+2} \\ \vdots \\ \Phi^{p+1} \end{bmatrix} x(k) + \begin{bmatrix} \Phi\Gamma+\Gamma \\ \sum_{i=0}^{2}\Phi^i\Gamma \\ \vdots \\ \sum_{i=0}^{m}\Phi^i\Gamma \\ \sum_{i=0}^{m+1}\Phi^i\Gamma \\ \vdots \\ \sum_{i=0}^{p}\Phi^i\Gamma \end{bmatrix} u(k)}_{past} + \underbrace{\begin{bmatrix} \Gamma & 0 & \cdots & 0 \\ \Phi\Gamma+\Gamma & \Gamma & \cdots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ \sum_{i=0}^{m-1}\Phi^i\Gamma & \cdots & \Phi\Gamma+\Gamma & \Gamma \\ \sum_{i=0}^{m}\Phi^i\Gamma & \cdots & \ddots & \Phi\Gamma+\Gamma \\ \vdots & \vdots & \vdots & \vdots \\ \sum_{i=0}^{p-2}\Phi^i\Gamma & \cdots & \cdots & \sum_{i=0}^{p-m-1}\Phi^i\Gamma \end{bmatrix} \begin{bmatrix} \Delta u(k+1\,|\,k) \\ \vdots \\ \Delta u(k+m\,|\,k) \end{bmatrix}}_{future} \quad [1.5]$$

In this equation contains x the states and u the inputs of the model at sample k.

The measured output, when the inputs do not influence the output, so **D=0** is defined as:

$$y_o^p = \mathbf{C}x_s^p(k+1) \quad [1.6]$$

The new output can be written as:

$$y_0^p(k+1) = y_{0,\Delta u=0}^p(k+1) + \mathbf{Y}\Delta u_i^m(k+1) \quad [1.7]$$

Y is called the prediction matrix. **Y** can be calculated with the part of equation 1.5 which is called future. The past is given by: $y_{0,\Delta u=0}^p(k+1)$

The tracking error e is then defined as:

$$e_o^p(k) = y_o^p(k) - r_o^p(k) \quad [1.8]$$

In this equation r is a known trajectory. $r_o^p(k)$ gives the trajectory for all controlled outputs over the whole prediction horizon. Analogue the predicted error, when nothing is done, is defined as:

$$e^{p}_{o,\Delta u=0}(k) = y^{p}_{o,\Delta u=0}(k) - r^{p}_{o}(k) \qquad [1.9]$$

The predicted error is not influenced by Δu, and when the error becomes zero the change of the input Δu should be chosen zero. The quadratic performance criterion is now:

$$\min_{\Delta u_i^m} = \left(e^{p}_{o,\Delta u=0} + \mathbf{Y}\Delta u_i^m\right)^T \mathbf{Q}\left(e^{p}_{o,\Delta u=0} + \mathbf{Y}\Delta u_i^m\right) + \left(\Delta u_i^m\right)^T \mathbf{R}\left(\Delta u_i^m\right) \qquad [1.10]$$

This can be written as:

$$\min_{\Delta u_i^m} = {e^{p}_{o,\Delta u=0}}^T \mathbf{Q} e^{p}_{o,\Delta u=0} + {e^{p}_{o,\Delta u=0}}^T \mathbf{Q}\mathbf{Y}\Delta u_i^m + \left(\mathbf{Y}\Delta u_i^m\right)^T \mathbf{Q} e^{p}_{o,\Delta u=0} + \left(\mathbf{Y}\Delta u_i^m\right)^T \mathbf{Q}\mathbf{Y}\Delta u_i^m + \left(\Delta u_i^m\right)^T \mathbf{R}\left(\Delta u_i^m\right) \qquad [1.11]$$

and rewritten to a standard QP problem:

$$\min_{\Delta u_i^m} = \frac{1}{2}\left(\Delta u_i^m\right)^T \left[\mathbf{Y}^T\mathbf{Q}\mathbf{Y} + \mathbf{R}\right]\left(\Delta u_i^m\right) + \left(\mathbf{Y}^T \mathbf{Q} e^{p}_{o,\Delta u=0}\right)\left(\Delta u_i^m\right) \qquad [1.12]$$

${e^{p}_{o,\Delta u=0}}^T \mathbf{Q} e^{p}_{o,\Delta u=0}$ is not used in the equation above, because it has a constant value and therefore does not influence the optimisation. The standard QP formulation is:

$$\min_{\Delta u_i^m(k+1)} f(\Delta u) = \left\{\tfrac{1}{2}\Delta u_i^m(k+1)^T \mathbf{H}\Delta u_i^m(k+1) - \mathbf{G}\Delta u_i^m(k+1)\right\} \qquad [1.13]$$

H is defined as the Hessian and contains second derivatives; **G** is the gradient with first derivatives.

$$\mathbf{H} = \mathbf{Y}^T\mathbf{Q}\mathbf{Y} + \mathbf{R}$$
$$\mathbf{G} = \mathbf{Y}^T\mathbf{Q} e^{p}_{o,\Delta u=0} \qquad [1.14]$$

The solution of the unconstrained problem can be solved:

$$\frac{\partial f(\Delta u)}{\partial(\Delta u)} = \mathbf{H}\Delta u + \mathbf{G} = 0 \qquad [1.15]$$
$$\Delta u = -\mathbf{H}^{-1}\mathbf{G} = \left[\mathbf{Y}^T\mathbf{Q}\mathbf{Y} + \mathbf{R}\right]^{-1}\mathbf{Y}^T\mathbf{Q} e^{p}_{o,\Delta u=0}$$

From equation 1.12 can be seen that no final state error occurs, because when the input is the desired input, the change of input is zero so the optimisation can reach its minimal value zero.

1.2.1 Constraints

Constraints will always be present at some time. Three types of constraints can be distinguished in a continuous model.

- Constraints on the output of the process, for instance a minimal and maximal water height in a water tank
- Constraints on the input signals, for instance a minimal and maximal value of the valves
- Constraints on the changes in input signals, or move constraints. This is the maximal move rate of the input signal per sample.

The optimisation problem can be written as:

$$\min_{\Delta u_i^m(k+1)} \left\{\tfrac{1}{2}\Delta u_i^m(k+1)^T \mathbf{H}\Delta u_i^m(k+1) - \mathbf{G}\Delta u_i^m(k+1)\right\} \qquad [1.16]$$

subject to $\quad \mathbf{C}\Delta u_i^m(k+1) \geq \mathbf{c}$

H and **G** are the Hessian matrix and the Gradient vector of the QP formulation respectively. **C** and **c** formulate the inequality constraints. The Hessian contains the second derivatives of the problem. The Gradient vector contains the first derivatives of the optimisation problem.

The constraints from equation 1.16 can be written in an upper and lower bounded form. This simplifies the implementation in Matlab.

$$b_l \le \begin{bmatrix} \Delta u \\ C\Delta u \end{bmatrix} \le b_u \qquad [1.17]$$

In equation 1.17 b_u and b_l are the upper and lowerbounds, b_l is equal to c if there are only lowerbounds in the equation. In this case b_u contains only the values infinite.

1.3 Absolute optimisation scheme

The input (u) is used as optimisation parameter for the absolute optimisation. The maximum and minimum step sizes are given as linear constraints. For this case Quadratic Programming is again useful.

$$\begin{bmatrix} x_s(k+2\,|\,k) \\ x_s(k+3\,|\,k) \\ \vdots \\ x_s(k+m+1\,|\,k) \\ x_s(k+m+2\,|\,k) \\ \vdots \\ x_s(k+p+1\,|\,k) \end{bmatrix} = \underbrace{\begin{bmatrix} \Phi \\ \Phi^2 \\ \vdots \\ \Phi^m \\ \Phi^{m+1} \\ \vdots \\ \Phi^p \end{bmatrix}}_{past} x(k+1\,|\,k) + \underbrace{\begin{bmatrix} \Gamma & 0 & \cdots & \cdots & 0 \\ \Phi\Gamma & \Gamma & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ \Phi^{m-1}\Gamma & \Phi^{m-2}\Gamma & \cdots & \Phi\Gamma & \Gamma \\ \Phi^m\Gamma & \Phi^{m-1}\Gamma & \cdots & \Phi^2\Gamma & \Phi\Gamma+\kappa\Gamma \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \Phi^{p-1}\Gamma & \Phi^{p-2}\Gamma & \cdots & \Phi^{p+1-m}\Gamma & \Phi^{p-m}\Gamma+\kappa\sum_{i=0}^{p-m-1}\Phi^i\Gamma \end{bmatrix}}_{future} \begin{bmatrix} u(k+1\,|\,k) \\ \vdots \\ u(k+m\,|\,k) \end{bmatrix} \qquad [1.18]$$

The output y is defined as:

$$y_0^p = \mathbf{C}x_s^p(k+1) \qquad [1.19]$$

This equation is equal to the relative problem. The optimisation criterion can be defined as:

$$\min_{u_i^m} = \frac{1}{2}\left(u_i^m\right)^T\left[\mathbf{Y}^T\mathbf{Q}\mathbf{Y}+\mathbf{R}\right]\left(u_i^m\right)+\left(\mathbf{Y}^T\mathbf{Q}e_{o,u=u0}^p\right) \qquad [1.20]$$

$e_{o,u=u0}^p$ is the predicted error and is defined as:

$$e_{o,u=u0}^p(k) = y_{o,u=u0}^p(k) - r_o^p(k) \qquad [1.20b]$$

Therefore the optimisation routine will stay the same as described in paragraph 1.2.

From equation 1.20 can be seen that a small final state error occurs. When the desired value is nearly reached there will be still a value for the input u, the optimisation will therefore never reach its minimal value zero. When the final error should be zero the absolute optimisation gives less results than the relative optimisation.

1.3.1 Constraints

The constraints for absolute optimisation can be derived from paragraph 1.2.1. The formulation differs, because the input is used. The optimisation problem will be:

$$\min_{u_i^m(k+1)} \left\{\tfrac{1}{2}u_i^m(k+1)^T\mathbf{H}u_i^m(k+1) - \mathbf{G}u_i^m(k+1)\right\}$$

$$\text{subject to} \quad \mathbf{C}u_i^m(k+1) \ge \mathbf{c} \qquad [1.21]$$

All constraints are written in Matlab according to the next formulation, on the same reasons as equation 1.17:

$$b_l \leq \begin{bmatrix} u \\ \mathbf{C}u \end{bmatrix} \leq b_u \qquad [1.22]$$

1.4 Final remarks

The optimisation schemes derived above make use of continuous (real) variables. When integer variables are used, the equations are the same, but the optimisation routine Quadratic Programming is not able to solve discrete integer problems. The optimisation will be executed with Mixed Integer Quadratic Programming. The input and state vector can contain both real and integer variables. The constraints are in this case the ones mentioned in paragraph 1.2.1 and additional constraints for the relations between different integer variables or integer and real variables.

Chapter 2: Mixed logical dynamical systems

This chapter describes a optimisation scheme to model and control systems described by physical laws, logical rules and operating constraints, denoted as a mixed logical dynamical (MLD) system.

2.1 Introduction

A model is traditionally written as a differential or difference equation, derived from the physical laws governing the dynamics of the system. In many applications the system to be controlled is also constituted by parts described by logic variables. Examples are on/ off switches or valves, gears or speed selectors, whose evolutions depend on if-then-else rules. Those systems will be written as a mixed logical dynamical (MLD) system, described by linear dynamic equations subject to linear mixed integer inequalities, i.e. equations containing both continuous and binary (or 0-1, or logical) variables.

2.2 Logical variables

Statements are implemented to model logical variables. Capital letters (X_i) represent statements, e.g. "$x \geq 0$", or "valve is open". Boolean algebra enables statements to be combined in compound statements by means of the connectives (table 2.1).

Connective	Description
$\wedge$	And
$\vee$	Or
$\sim$	Not
$\rightarrow$	Implies
$\leftrightarrow$	If and only if
$\oplus$	Exclusive or

Table 2.1: Boolean connectives and their descriptions

The connectives are defined in truth tables (table 2.2).

X_1	X_2	$\sim X_1$	$X_1 \vee X_2$	$X_1 \wedge X_2$	$X_1 \rightarrow X_2$	$X_1 \leftrightarrow X_2$	$X_1 \oplus X_2$
F	F	T	F	F	T	T	F
F	T	T	T	F	T	F	T
T	F	F	T	F	F	F	T
T	T	F	T	T	T	T	F

Table 2.2: Truth table

With the literal X_i a logical variable $\delta_i \{0,1\}$, which can have a value of either 1 (if X_i is true), or 0 (if X_i is false), can be associated. The statement X_i can be written as an inequality involving logical variables δ_i.

$X_1 \vee X_2$ is equivalent to $\delta_1+\delta_2 \geq 1$
$X_1 \wedge X_2$ is equivalent to $\delta_1=1, \delta_2=1$
$\sim X_1$ is equivalent to $\delta_1=0$
$X_1 \rightarrow X_2$ is equivalent to $\delta_1-\delta_2 \leq 0$
$X_1 \leftrightarrow X_2$ is equivalent to $\delta_1-\delta_2=0$
$X_1 \oplus X_2$ is equivalent to $\delta_1+\delta_2=1$ [2.1]

This technique will be used to model logical parts of processes.

Consider the statement X is true if $f(x) \leq 0$, where f: $\mathbb{R}^n \rightarrow \mathbb{R}$ is linear. Assume $x \in \mathcal{L}$, where $\mathcal{L}$ is a given bounded set and define

$M \triangleq \max f(x)$ [2.1b]

$m \triangleq \min f(x)$

These logical statements can be written in a Boolean formula, which can be rewritten as inequalities, with help of the truth table. In these equations ε is a small value, often machine precision, beyond which the constraint is regarded as violated.

- $[f(x) \leq 0] \wedge [\delta = 1]$ is true if and only if: [2.2a]
  $f(x) - \delta \leq -1 + m(1-\delta)$
  Prove: if $f(x) \leq 0$, δ should be equal to 1 to give true. The equation gives $f(x) \leq 0$, which is correct. If δ would be zero, the equations give $f(x) \leq -1+m$, this gives a false answer, because m is defined as the minimum of $f(x)$.

- $[f(x) \leq 0] \vee [\delta = 1]$ is true if and only if: [2.2b]
  $f(x) \leq M\delta$
  Prove: if $f(x) \leq 0$, δ can be 1 or 0 to give true. The equations give: $f(x) \leq 0$ or $f(x) \leq M$, both equations are correct
  If $\delta = 1$ $f(x)$ can have all possible values to give true. The equations give: $f(x) \leq M$, which is correct.

- $[f(x) \leq 0] \rightarrow [\delta = 1]$ is true if and only if: [2.2c]
  $f(x) \geq \varepsilon + (m-\varepsilon)\delta$
  Prove: if $f(x) \leq 0$ δ should be 1. The inequality gives: $f(x) \geq m$, which is true.
  if $f(x)$ is larger than zero, δ can be 0 and 1, this gives:
  $f(x) \geq \varepsilon$, which is also correct in this case, or
  $f(x) \geq m$, which is also true.

- $[f(x) \leq 0] \leftrightarrow [\delta = 1]$ is true if and only if: [2.2d]
  $f(x) \geq \varepsilon + (m-\varepsilon)\delta$
  $f(x) \leq M(1-\delta)$
  Prove: if $f(x) \leq 0$, δ should be 1 to give true. The equations give:
  $f(x) \geq m$ and $f(x) \leq 0$ which is true
  if $f(x) \leq 0$ and δ is 0, it has to give false. The equations give:
  $f(x) \geq \varepsilon$ and $f(x) \leq M$ which is false because $f(x) \leq 0$.
  if $f(x) > 0$, δ should be equal to 0. In the equations this gives:
  $f(x) \geq \varepsilon$ and $f(x) \leq M$. When ε is chosen to be machine precision these equations give $f(x) > 0$
  if $f(x) > 0$ and δ is 1, the equations must be incorrect. They give:
  $f(x) \geq m$ and $f(x) \leq 0$. This gives incorrect because $f(x)$ was said to be larger than zero. So equation 2.2d is correct.

These logical variables can be used in a mixed logical dynamical system. Equation 2.3 gives the general form. The logical constraints, which are rewritten as inequalities, can be implemented in the third equation.

$$\begin{aligned} x(t+1) &= A_t x(t) + B_{1t} u(t) + B_{2t} \delta(t) + B_{3t} z(t) \\ y(t) &= C_t x(t) + D_{1t} u(t) + D_{2t} \delta(t) + D_{3t} z(t) \\ E_{2t} \delta(t) + E_{3t} z(t) &\le E_{1t} u(t) + E_{4t} x(t) + E_{5t} \delta(t) \end{aligned} \quad [2.3]$$

Where x is the state of the system, y(t) is the output vector and u(t) is the input vector. The first two equations are the system equations. They can be in a continuous and a discrete form. The last one gives the constraints. Both δ(t) and z(t) represent auxiliary variables. It is possible to augment the input u(t) with both δ(t) and z(t). The different B and D matrices will form one B and D matrix which sizes are the same as the new input vector. The same holds for the different E-matrices. All vectors can contain integer and real variables. The first and the second equation give the dynamics of the system. The third equation gives the constraints. It contains the relations between integer and real variables, or between integers and input, output and move constraints, as mentioned in chapter 1. The next examples will all be written in the formulation of equation 2.3.

2.3 Examples

The next examples will explain the usefulness of these equations. In the examples an application of this optimisation scheme for logical constraints on the state, piecewise linear systems and discrete inputs can be seen.

2.3.1 Logical constraints on the state

A logical constraint on the state is implemented for a single input single output system:

$$x(t+1) = \begin{cases} 0.8x(t) + u(t) & \text{if } x(t) \ge 0 \\ -0.8x(t) + u(t) & \text{if } x(t) < 0 \end{cases} \quad [2.4]$$

where

$x(t) \in [-10,10]$

$u(t) \in [-1,1]$

The condition $x(t) \ge 0$ can be associated with a binary variable $\delta(t)$ such that

$$[\delta(t) = 1] \leftrightarrow [x(t) \ge 0] \quad [2.5]$$

By using the transformation of equation 2.2 this can be expressed by the inequalities

$$\begin{aligned} m\delta(t) &\le x(t) \\ -(M+\varepsilon)\delta &\le -x(t) - \varepsilon \end{aligned} \quad [2.6]$$

Where $M = -m = 10$

Equation 2.4 can be rewritten as

$$x(t+1) = 1.6\delta(t)x(t) - 0.8x(t) + u(t) \quad [2.7]$$

When $\delta(t)x(t)$ is expressed by a new variable $z(t)$ the evolution of the system is ruled by

$$x(t+1) = 1.6z(t) - 0.8x(t) + u(t) \quad [2.8]$$

To guarantee z(t) is the same as δ(t)x(t), some constraints are implemented.

$$\begin{aligned} z(t) &\le M\delta(t) \\ z(t) &\ge m\delta(t) \\ z(t) &\le x(t) - m(1-\delta(t)) \\ z(t) &\ge x(t) - M(1-\delta(t)) \end{aligned} \qquad [2.9]$$

The first constraint gives that $z(t)$ is always smaller than the maximum of $\delta(t)x(t)$. The second constraint gives that $z(t)$ is always larger than the minimum of $\delta(t)x(t)$. The last two constraints are derived from equation 2.6.

At this moment equation 2.4 is written in one equation and four constraints. In the standard formulation of equation 2.3 it gives:

$$\begin{aligned} x(t+1) &= -0.8x(t) + u(t) + 1.6z(t) \\ y(t) &= x(t) \end{aligned} \qquad [2.10a]$$

And the constraints of equation 2.3 give:

$$\begin{bmatrix} -m \\ -(M+\varepsilon) \\ -M \\ -m \\ -m \\ M \end{bmatrix} \delta(t) + \begin{bmatrix} 0 \\ 0 \\ 1 \\ -1 \\ 1 \\ -1 \end{bmatrix} z(t) \le \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} u(t) + \begin{bmatrix} 1 \\ -1 \\ 0 \\ 0 \\ 1 \\ -1 \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ -\varepsilon \\ 0 \\ 0 \\ -m \\ M \end{bmatrix} \qquad [2.10b]$$

2.3.2 Piecewise linear system

Piecewise linear systems contain different linear equation. These equation are useful for given values of the state, therefore they succeed each other. When the maximal value (or a specified value) of the first system is reached the system alternates to the next system equation, which is again linear and holds for the new values of the state (x_i). An example of a piecewise linear system, with a single inputs and single output, is given in equation 2.9.

$$\begin{aligned} x(t+1) &= A_1x(t) + B_1u(t) \quad \text{if } \delta_1(t) = 1 \\ x(t+1) &= A_2x(t) + B_2u(t) \quad \text{if } \delta_2(t) = 1 \\ x(t+1) &= A_3x(t) + B_3u(t) \quad \text{if } \delta_3(t) = 1 \end{aligned} \qquad [2.11]$$

Where $\delta_i(t)$ is a 0-1 variable, satisfying the exclusive or condition;

$$[\delta_1(t)=1] \oplus [\delta_2(t)=1] \oplus [\delta_3(t)=1] \qquad [2.12]$$

In equation 2.11 is not defined when $\delta_i(t)$ is zero and when it is one. The next equation is implemented to prescribe these rules.

$$[\delta_i = 1] \leftrightarrow [A_ix(t) + B_iu(t) \le T_i] \qquad [2.13]$$

The value of T_i can be chosen. It is in this example defined as the maximal value for which the equation is linear. For a larger value the system has to alternate to the next equation. The value T_i can therefore also be the minimum of the next system equation.

$$T_i = \max\{A_i x(t) + B_i u(t)\} \quad [2.14]$$

With these assumptions it is possible to rewrite equation 2.11. The A_i and B_i matrices are only used when $\delta_i(t)$ is one, so it can be written as:

$$x(t+1) = \sum_{i=1}^{3} (A_i x(t) + B_i u(t))\delta_i(t) \quad [2.15]$$

This equation is non-linear since it involves products between logical variables and states or inputs. The equation can be written into an equivalent mixed integer linear inequality.

$$x(t+1) = \sum_{i=1}^{3} (z_i(t)) \quad [2.16]$$

The next definitions are needed to derive the constraints. Every system equation needs is own minimum and maximum value for which the system has to alternate to another system equation.

$$\begin{aligned} M_i &= \max(A_i x(t) + B_i u(t)) = T_i \\ m_i &= \min(A_i x(t) + B_i u(t)) \end{aligned} \quad [2.17]$$

The exclusive or condition of equation 2.12 can be rewritten as:

$$\begin{bmatrix} 1 & 1 & 1 \\ -1 & -1 & -1 \end{bmatrix} \begin{bmatrix} \delta_1(t) \\ \delta_2(t) \\ \delta_3(t) \end{bmatrix} \geq \begin{bmatrix} 1 \\ -1 \end{bmatrix} \quad [2.18]$$

Equation 2.13 can be rewritten as:

$$\begin{aligned} A_i x(t) + B_i u(t) &\geq (m - M)\delta + M \\ A_i x(t) + B_i u(t) &\leq -M\delta + 2M \end{aligned} \quad [2.19]$$

For $z(t)$ the constraints are, just as in the previous example:

$$\begin{aligned} z_i(t) &\leq M\delta_i(t) \\ z_i(t) &\geq m\delta_i(t) \\ z_i(t) &\leq A_i x(t) + B_i u - m(1 - \delta_i(t)) \\ z_i(t) &\geq A_i x(t) + B_i u - M(1 - \delta_i(t)) \end{aligned} \quad [2.20]$$

Equation 2.16 gives the new system equation, in equation 2.21 it is written in the standard formulation. The constraints are given by equation 2.19 and 2.20. Their standard formulation is given in appendix 2, for the three different equations in 2.12.

$$x(t) = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} z_1(t) \\ z_2(t) \\ z_3(t) \end{bmatrix} \quad [2.21]$$

2.3.3 Discrete inputs

Discrete inputs are for example useful for application with on/off switches, gears and speed selectors. Such systems can be easily modelled by logical variables, for example the system in equation 2.22.

$$x(t+1) = Ax(t) + Bu(t)$$
$$u(t) \in \{u_1, u_2, u_3, u_4\} \qquad [2.22]$$

To simplify the equations a single input, single output system is chosen, so the dimension of every u_i is one. The dimension of the state $x(t)$ also one.
When four discrete values of the input are used, equation 2.22 can be written as:

$$x(t+1) = Ax(t) + B\begin{bmatrix} u_1 & u_2 & u_3 & u_4 \end{bmatrix} u_k(t) \qquad [2.23a]$$

A new B-matrix can be defined as:

$$B_{new} = B\begin{bmatrix} u_1 & u_2 & u_3 & u_4 \end{bmatrix} \qquad [2.23b]$$

To guarantee that never two discrete inputs are implemented together on the system constraints are used. These give that exactly one $u_k(t)$ is one and the others are zero. This gives the following constraints:

$$\begin{bmatrix} -1 & -1 & -1 & -1 \\ 1 & 1 & 1 & 1 \end{bmatrix} u_k(t) \le \begin{bmatrix} -1 \\ 1 \end{bmatrix} \qquad [2.24]$$

with

$$u_k(t) \equiv \begin{bmatrix} u_{k1}(t) & u_{k2}(t) & u_{k3}(t) & u_{k4}(t) \end{bmatrix}$$
$$u_{ki}(t) \in \{0,1\}$$

The first equation in 2.24 guarantees that at least one $u_k(t)$ is one, the second equation guarantees that at most one $u_k(t)$ is one.
In the standard formulation it can be rewritten as:

$$x(t+1) = Ax(t) + B_{new} u_k(t) \qquad [2.25a]$$

$$\begin{bmatrix} 0 \\ 0 \end{bmatrix} \le \begin{bmatrix} 1 & 1 & 1 & 1 \\ -1 & -1 & -1 & -1 \end{bmatrix} u_k(t) + \begin{bmatrix} -1 \\ 1 \end{bmatrix} \qquad [2.25b]$$

Chapter 3: Model with three water tanks

In this chapter the model will be derived and it will be rewritten into the formulation of chapter two. This will be used to test this optimisation scheme and formulations.

3.1 Overall model

The schematic model of the system with three water tanks is given in figure 3.1.

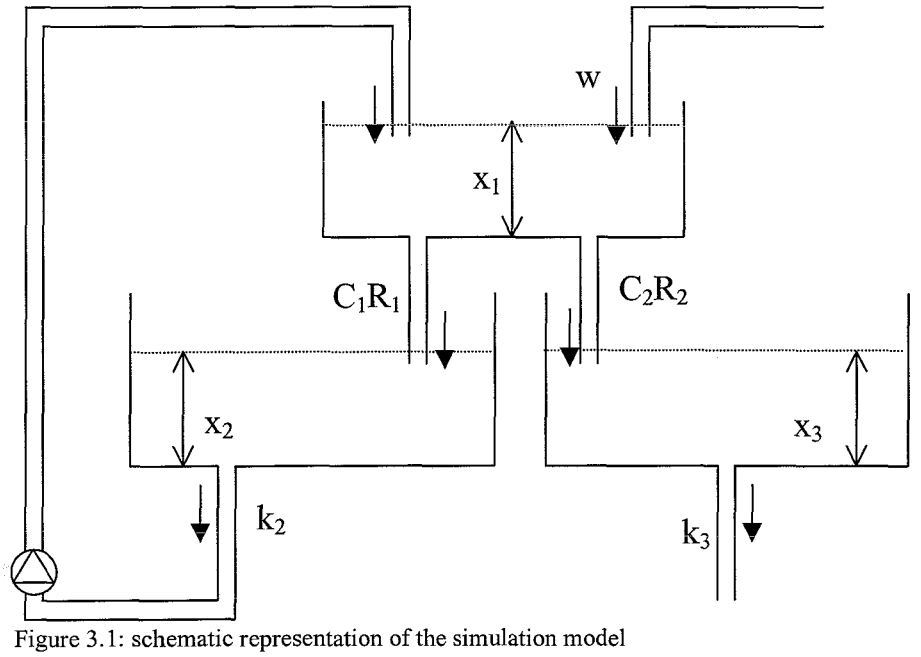


Figure 3.1: schematic representation of the simulation model

The controller should stabilise the water heights in tank 1, 2 and 3. To achieve these demands the valve values (R_1, R_2) and the water input (w) can be controlled.
At first the equations for the different water tanks are derived:

$$\begin{cases} \dot{x}_1 = \dfrac{1}{\rho \cdot A_1}\left(w(t) + k_2\sqrt{\rho g x_2} - c_1 R_1\sqrt{\rho g x_1} - c_2 R_2\sqrt{\rho g x_1}\right) \\ \dot{x}_2 = \dfrac{1}{\rho \cdot A_2}\left(c_1 R_1\sqrt{\rho g x_1} - k_2\sqrt{\rho g x_2}\right) \\ \dot{x}_3 = \dfrac{1}{\rho \cdot A_3}\left(c_2 R_2\sqrt{\rho g x_1} - k_3\sqrt{\rho g x_3}\right) \end{cases} \quad [3.1]$$

The constant parameters in this equation are:
$A_1 = 0.1\ m^2$
$A_2 = 0.1\ m^2$
$A_3 = 0.1\ m^2$
$k_2 = 0.055\ (kg \cdot m)^{1/2}$
$k_3 = 0.055\ (kg \cdot m)^{1/2}$
$c_1 = 0.06\ (kg \cdot m)^{1/2}$
$c_2 = 0.06\ (kg \cdot m)^{1/2}$
$g = 9.81\ m/s^2$
$\rho = 1000\ kg/m^3$
The pump efficiency is 100%.

The determination of these constants is given in appendix 3

In chapter 1 the equations are written for a linear system. The system given in equation 3.1 is non-linear and has to be linearised. The behaviour of the non-linear model and its linearisation is given in appendix 4. The linearisation is satisfying and therefore useful in the optimisation problem.

3.2 Simulation models

The equations in 3.1 assume that the valves *(R)* have values between 0 and 1.In practice a valve will not close softly, but turn off when a minimal value is reached. This behaviour will be modelled in two different ways.

- The first model uses two non-continuous inputs. When the valves *(R)* reach a value below 0.2 the valve will be closed completely, so R is set to zero. This assumption gives several equations for the system according to the value of R. The equations are given in formula 3.2.

$$
\dot{x} = \begin{cases}
\begin{cases}
\dot{x}_1 = \frac{1}{\rho \cdot A_1}\left(w(t) + k_2\sqrt{\rho g x_2} - c_1 R_1\sqrt{\rho g x_1} - c_2 R_2\sqrt{\rho g x_1}\right) \\
\dot{x}_2 = \frac{1}{\rho \cdot A_2}\left(c_1 R_1\sqrt{\rho g x_1} - k_2\sqrt{\rho g x_2}\right) \\
\dot{x}_3 = \frac{1}{\rho \cdot A_3}\left(c_2 R_2\sqrt{\rho g x_1} - k_3\sqrt{\rho g x_3}\right)
\end{cases} & \text{if } R_i \geq 0.2 \\
\begin{cases}
\dot{x}_1 = \frac{1}{\rho \cdot A_1}\left(w(t) + k_2\sqrt{\rho g x_2} - c_2 R_2\sqrt{\rho g x_1}\right) \\
\dot{x}_2 = \frac{1}{\rho \cdot A_2}\left(-k_2\sqrt{\rho g x_2}\right) \\
\dot{x}_3 = \frac{1}{\rho \cdot A_3}\left(c_2 R_2\sqrt{\rho g x_1} - k_3\sqrt{\rho g x_3}\right)
\end{cases} & \text{if } R_1 < 0.2 \wedge R_2 \geq 0.2 \\
\begin{cases}
\dot{x}_1 = \frac{1}{\rho \cdot A_1}\left(w(t) + k_2\sqrt{\rho g x_2} - c_1 R_1\sqrt{\rho g x_1} - c_2 R_2\sqrt{\rho g x_1}\right) \\
\dot{x}_2 = \frac{1}{\rho \cdot A_2}\left(c_1 R_1\sqrt{\rho g x_1} - k_2\sqrt{\rho g x_2}\right) \\
\dot{x}_3 = \frac{1}{\rho \cdot A_3}\left(c_2 R_2\sqrt{\rho g x_1} - k_3\sqrt{\rho g x_3}\right)
\end{cases} & \text{if } R_1 \geq 0.2 \wedge R_2 < 0.2 \\
\begin{cases}
\dot{x}_1 = \frac{1}{\rho \cdot A_1}\left(w(t) + k_2\sqrt{\rho g x_2}\right) \\
\dot{x}_2 = \frac{1}{\rho \cdot A_2}\left(-k_2\sqrt{\rho g x_2}\right) \\
\dot{x}_3 = \frac{1}{\rho \cdot A_3}\left(-k_3\sqrt{\rho g x_3}\right)
\end{cases} & \text{if } R_i \leq 0.2
\end{cases}
\qquad [3.2]
$$

- The second model uses a discrete R_1 and R_2 is continuous, to simplify the equations. Valve R_1 can have values {0, 0.2, 0.4, 0.6, 0.8, 1.0}. Equation 3.1 holds, with a discrete instead of a continuous R_1.

4. Mixed integer predictive control

The two different models of chapter 3 will be used to simulate the behaviour of the water tanks and their valves. They will be described after each other. Paragraph 4.1 gives the model with a non-continuous input. It describes the use of logical statements, the system, the constraints and the results. Paragraph 4.2 gives these subjects for the model with a discrete input

4.1 Model with non-continuous inputs

4.1.1 Logical statements and system equations

At first the equations will be derived with R (valve 1 and 2), which can vary continuous between 0.2 and 1 and will be closed (R=0) when R becomes smaller then 0.2.

The logical statement can replace the conditions for R_i in equation 3.2.

$$[d_i(t) = 1] \leftrightarrow [R_i \geq 0.2] \quad [4.1a]$$

$$[d_i(t) = 0] \leftrightarrow [R_i = 0] \quad [4.1b]$$

With:
$R \in \mathbb{R}$, $\mathbb{R}$: Real numbers
$d \in \mathbb{Z}$, $\mathbb{Z}$: Integer numbers

This logical statement can be written in inequality equations as, just as equation 2.6:

$$md \leq R(t)$$
$$-(M + \varepsilon)d \leq -R - \varepsilon \quad [4.2]$$

With:

$m = 0.2$

$M = 1$

The inequality equations become:

$$R(t) \geq 0.2d$$
$$R(t) \leq (1 - \varepsilon)d + \varepsilon \quad [4.3]$$

These inequality constraints satisfy equation 4.1:

- Equation 4.2a gives that if d=1, it means R should be larger or equal to 0.2 to be correct. The inequality constraints of equation 4.3 give:

  $$R(t) \geq 0.2$$
  $$R(t) \leq 1 \quad [4.4]$$

- The second logical statement (4.1b) says if d is zero R should be zero to be correct. Equation 4.3 gives when d is zero:

  $$R(t) \geq 0$$
  $$R(t) \leq \varepsilon \quad [4.5]$$

  So R will be zero.

For both situations the inequality constraints are correct. So these inequality constraints will substitute the logical variables. When these variables are substituted in equation 3.1, the system equation can be written as:

$$\begin{cases} x_1(k+1) = \dfrac{1}{\rho \cdot A_1}\left(w(k) + k_2\sqrt{\rho g x_2(k)} - c_1 R_1 d_1\sqrt{\rho g x_1(k)} - c_2 R_2 d_2\sqrt{\rho g x_1(k)}\right) \\ x_2(k+1) = \dfrac{1}{\rho \cdot A_2}\left(c_1 R_1 d_1\sqrt{\rho g x_1(k)} - k_2\sqrt{\rho g x_2(k)}\right) \\ x_3(k+1) = \dfrac{1}{\rho \cdot A_3}\left(c_2 R_2 d_2\sqrt{\rho g x_1(k)} - k_3\sqrt{\rho g x_3(k)}\right) \end{cases} \quad [4.6]$$

This non-linear equation will be used to simulate the system. For the MIP-controller a linear approximation is needed.

The linearised discrete equation is:

$$x(k+1) = \Phi x(k) + \Gamma u(k) \quad [4.7]$$

The Φ and Γ matrices are given in appendix 6. Appendix 4 shows that the linearised discrete system is a satisfying approximation for the non-linear model. It is remarkable that the first column of matrix Φ contains products of inputs. A relative optimisation will be used to solve the problem. The constraints will be derived with respect to Δu.

The change of the inputs Δu is defined as:

$$\Delta u = \begin{bmatrix} \Delta R_1 & \Delta R_2 & \Delta w & \Delta d_1 & \Delta d_2 \end{bmatrix}^T \quad [4.8a]$$

The state is defined as:

$$x = \begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix}^T \quad [4.8b]$$

The use of the change of inputs makes equation 1.5 applicable. It will be used in the optimisation problem. Note that it is derived for the standard MPC problem, but still useful is because only the constraints are extended. The input vector contains in this case also integer variables.

4.1.2 Constraints for relative optimisation

The different constraints for this optimisation problem will be described in this paragraph. The possible constraints are already mentioned in chapter 1. Below the move, input, output and logical constraints will be derived.

- As first the constraints for Δu, known as the move constraints:

$$\begin{bmatrix} -1 \\ -1 \\ -10 \\ -1 \\ -1 \end{bmatrix} \leq \begin{bmatrix} \Delta R_1 \\ \Delta R_2 \\ \Delta w \\ \Delta d_1 \\ \Delta d_2 \end{bmatrix} \leq \begin{bmatrix} 1 \\ 1 \\ 10 \\ 1 \\ 1 \end{bmatrix} \quad [4.9]$$

These constraints can be implemented in this way. For every prediction in the control horizon this equation holds. This equation repeats therefore m times per sample in the optimisation problem.

- The input constraints are given by their minimal and maximal values:

$$\begin{bmatrix}0\\0\\0\\0\\0\end{bmatrix} \le \begin{bmatrix}R_1\\R_2\\w\\d_1\\d_2\end{bmatrix} \le \begin{bmatrix}1\\1\\\inf\\1\\1\end{bmatrix} \qquad [4.10]$$

The absolute inputs are unknown in the relative optimisation problem; therefore 4.10 is rewritten as a function of Δu, with:

$$\begin{aligned}\Delta u &= u(k+1) - u(k)\\ u(k+1) &= \Delta u + u(k)\end{aligned} \qquad [4.11]$$

The minimal and maximal values will be implemented with the next equation:

$$\begin{bmatrix}0-u(k)\\0-u(k)\\0-u(k)\\0-u(k)\\0-u(k)\end{bmatrix} \le \begin{bmatrix}\Delta R_1\\\Delta R_2\\\Delta w\\\Delta d_1\\\Delta d_2\end{bmatrix} \le \begin{bmatrix}1-u(k)\\1-u(k)\\\inf-u(k)\\1-u(k)\\1-u(k)\end{bmatrix} \qquad [4.12]$$

- Next the constraints on the water height, which is the output, will be determined. The output must have values between the upper and lower bounds. For the implementation the output must be rewritten as a function of the input and the state, because it is not a part of the optimisation problem. Therefore the outputs of the system will be computed with equation 1.5 which gives the new state. The new outputs are equal to the new states, because all outputs are measured.
When the outputs are rewritten as a function of both the input and the state and must be between the upper and lower bounds it results in the next equation:

$$lb - \begin{bmatrix}\Phi^2\\\Phi^3\\\vdots\\\Phi^{m+1}\\\Phi^{m+2}\\\vdots\\\Phi^{p+1}\end{bmatrix} x(k) - \begin{bmatrix}\Phi\Gamma+\Gamma\\\sum_{i=0}^{2}\Phi^i\Gamma\\\vdots\\\sum_{i=0}^{m}\Phi^i\Gamma\\\sum_{i=0}^{m+1}\Phi^i\Gamma\\\vdots\\\sum_{i=0}^{p}\Phi^i\Gamma\end{bmatrix} u(k) \le \begin{bmatrix}\Gamma & 0 & \cdots & 0\\ \Phi\Gamma+\Gamma & \Gamma & \cdots & \vdots\\ \vdots & \ddots & \ddots & 0\\ \sum_{i=0}^{m-1}\Phi^i\Gamma & \cdots & \Phi\Gamma+\Gamma & \Gamma\\ \sum_{i=0}^{m}\Phi^i\Gamma & \cdots & \ddots & \Phi\Gamma+\Gamma\\ \vdots & \vdots & \vdots & \vdots\\ \sum_{i=0}^{p-2}\Phi^i\Gamma & \cdots & \cdots & \sum_{i=0}^{p-m-1}\Phi^i\Gamma\end{bmatrix} \begin{bmatrix}\Delta u(k+1|k)\\\vdots\\\Delta u(k+m|k)\end{bmatrix} \le ub - \begin{bmatrix}\Phi^2\\\Phi^3\\\vdots\\\Phi^{m+1}\\\Phi^{m+2}\\\vdots\\\Phi^{p+1}\end{bmatrix} x(k) - \begin{bmatrix}\Phi\Gamma+\Gamma\\\sum_{i=0}^{2}\Phi^i\Gamma\\\vdots\\\sum_{i=0}^{m}\Phi^i\Gamma\\\sum_{i=0}^{m+1}\Phi^i\Gamma\\\vdots\\\sum_{i=0}^{p}\Phi^i\Gamma\end{bmatrix} u(k) \qquad [4.13]$$

The columns *lb* and *ub* are *p* times the upper and lower bounds of the output.

$$lowerbound = \begin{bmatrix}0\\0\\0\end{bmatrix} \qquad upperbound = \begin{bmatrix}1\\1\\1\end{bmatrix} \qquad [4.14]$$

•• Equation 4.3 gives the inequalities of the logical constraints. These constraints hold for both R_1 and R_2. In matrix formulation this gives the next result:

$$\begin{bmatrix} 0 \\ 0 \\ -\varepsilon \\ -\varepsilon \end{bmatrix} \leq \begin{bmatrix} 1 & 0 & 0 & -0.2 & 0 \\ 0 & 1 & 0 & 0 & -0.2 \\ -1 & 0 & 0 & -\varepsilon+1 & 0 \\ 0 & -1 & 0 & 0 & -\varepsilon+1 \end{bmatrix} \begin{bmatrix} R_1 \\ R_1 \\ w \\ d_1 \\ d_2 \end{bmatrix} \leq \begin{bmatrix} \inf \\ \inf \\ \inf \\ \inf \end{bmatrix} \quad [4.15]$$

This formulation uses the input, but in the optimisation only the change of input (Δu) is given. So they are rewritten with Δu as input:

$$\begin{bmatrix} 0-u(k) \\ 0-u(k) \\ -\varepsilon-u(k) \\ -\varepsilon-u(k) \end{bmatrix} \leq \begin{bmatrix} 1 & 0 & 0 & -0.2 & 0 \\ 0 & 1 & 0 & 0 & -0.2 \\ -1 & 0 & 0 & -\varepsilon+1 & 0 \\ 0 & -1 & 0 & 0 & -\varepsilon+1 \end{bmatrix} \begin{bmatrix} \Delta R_1 \\ \Delta R_1 \\ \Delta w \\ \Delta d_1 \\ \Delta d_2 \end{bmatrix} \leq \begin{bmatrix} \inf \\ \inf \\ \inf \\ \inf \end{bmatrix} \quad [4.16]$$

The value ε in equation 4.15 and 4.16 is used to avoid numerical problems.

4.1.3. Results

The result of the given problem is shown in figure 4.2:

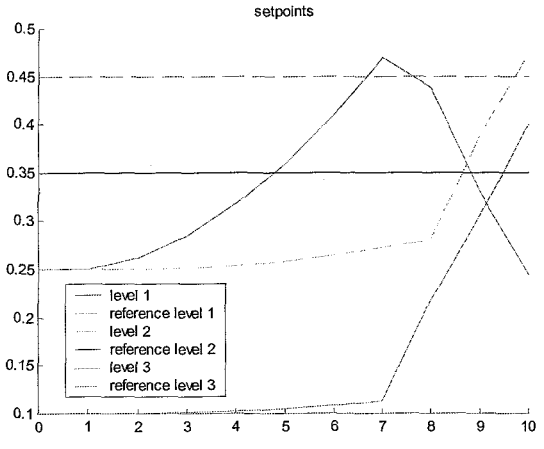


Figure 4.1: Setpoints and behaviour of the water tanks for the model with non-continuous inputs

The other results are shown in appendix 5. From this figure can be seen that the optimisation does not give a satisfying result. The optimisation gives after some samples an infeasible result if all constraints are used. Below the problems are described. Both the logical constraints and the move constraints on the integers must be excluded to give a feasible result. At first the move constraints on the integers will be described. The logical constraints were not used in this case. When the solution is computed with an optimisation to Δu the logical variable d_i is also computed in the form of Δd_i. The minimal and maximal values of Δd_i are between −1 and 1. When these constraints are implemented in the Matlab model the simulation gives an infeasible result. The change of the integer input has to be between 0 and 1 to obtain a feasible result. This input leads to a non-changing variable d, and it can therefore

not be optimised. This problem is probably due to linearisation. The values of the constraints are computed for the non-linear system. The move constraints are in all cases the same, but the input constraints need to be changed. This can lead to overlying constraints and then a value of -1 for the move constraints gives an infeasible result for the input constraints. Another solution is to exclude all move constraints, but this means that the changes of the input are not weighted anymore.
The optimisation only terminates successful when also the logical constraints are excluded. This problem is due to the dependence of R and d in the input vector. The optimisation routine needs independent optimisation parameters to compute an optimisation, because it cannot see the dependence between the different input variables. The optimisation routine computes the optimal input with respect to all input variables. The constraints contain some dependence of R and d, in equation 4.14, but do not tell the optimisation the product of R and d is the real optimal value. This can lead to sub-optimal or infeasible solutions. To avoid these problems the model should be written without dependence of the inputs. The problem is in this case written in such a way it cannot be written without this dependence. Another formulation of the problem is needed to solve this problem. The use of the auxiliary variable $z(t)$ does not give a good result, because all inputs have to be defined in one input and not in the same way as in equation 2.3. Another Matlab function, which optimises to the variable z, but uses the other inputs only to compute the constraints, can give a solution.

4.2 Model with a discrete input

4.2.1. Logical statements and system equations

To obtain independent variables a new model of valve 1 is chosen. Valve R_1 has one of the next values: {0, 0.2, 0.4, 0.6, 0.8, 1}. Valve R_2 can have values between 0 and 1.

This assumption leads to the next constraint for R_1.

$$
\begin{aligned}
&[d_1 = 0] \leftrightarrow [R_1 = 0] \\
&[d_1 = 1] \leftrightarrow [R_1 = 0.2] \\
&[d_2 = 1] \leftrightarrow [R_1 = 0.4] \\
&[d_3 = 1] \leftrightarrow [R_1 = 0.6] \\
&[d_4 = 1] \leftrightarrow [R_1 = 0.8] \\
&[d_5 = 1] \leftrightarrow [R_1 = 1] \\
&[d_5 = 1] \rightarrow [d_4 = 1] \wedge [d_3 = 1] \wedge [d_2 = 1] \wedge [d_1 = 1] \\
&[d_4 = 1] \rightarrow [d_3 = 1] \wedge [d_2 = 1] \wedge [d_1 = 1] \\
&[d_3 = 1] \rightarrow [d_2 = 1] \wedge [d_1 = 1] \\
&[d_2 = 1] \rightarrow [d_1 = 1] \\
&[d_1 = 0] \rightarrow [d_2 = 0] \wedge [d_3 = 0] \wedge [d_4 = 0] \wedge [d_5 = 0] \\
&[d_2 = 0] \rightarrow [d_3 = 0] \wedge [d_4 = 0] \wedge [d_5 = 0] \\
&[d_3 = 0] \rightarrow [d_4 = 0] \wedge [d_5 = 0] \\
&[d_4 = 0] \rightarrow [d_5 = 0]
\end{aligned}
\qquad [4.17]
$$

In this model R_1 is modelled as a parallel circuit of different valves (see figure 4.2), with logical constraints. These constraints determine the order of the valve parts to be opened. They are given in equation 4.18.

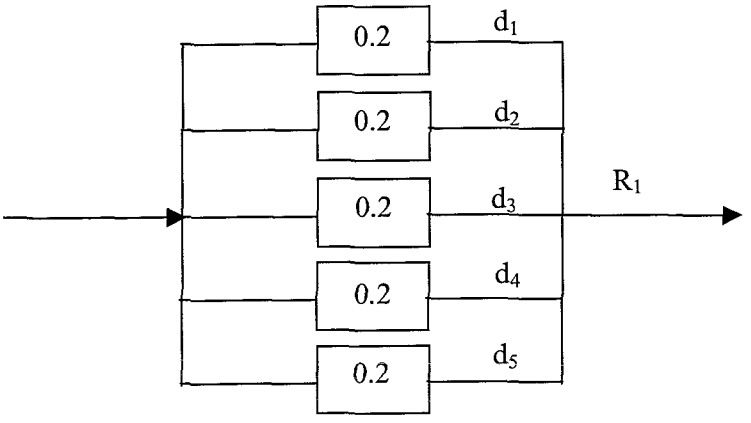


Figure 4.2: Schematic representation of R_1.

When the constraints of equation 4.17 are written in inequalities it gives:

- for the last 8 equations:

$$\begin{aligned} &0 \le d = d_1 + d_2 + d_3 + d_4 + d_5 \le 5 \\ &d_5 - d_4 \le 0 \\ &d_4 - d_3 \le 0 \\ &d_3 - d_2 \le 0 \\ &d_2 - d_1 \le 0 \end{aligned} \qquad [4.18]$$

- and according to the other equations $R(t)$ can be written as:

$$R(t) = 0.2(d_5 + d_4 + d_3 + d_2 + d_1) \qquad [4.19]$$

To prove these equations a possible solution is given:

$$\left.\begin{aligned} d_4 &= 0 \\ d_3 = d_2 = d_1 &= 1 \end{aligned}\right\} \Rightarrow \begin{aligned} R(t) &= 0.2d_3 + 0.2d_2 + 0.4d_1 = 0.8 \\ d_5 &= 0 \end{aligned} \qquad [4.20]$$

This is a correct result and can be done for all different solutions. Therefore the equations give a correct description of the logical constraints. These equations will be implemented in a Matlab file and are the linear constraints for this problem.

For the optimisation problem also the move constraints have to be implemented. The move constraints will only be implemented for valve two and the integer variables. To give R_1 move constraints would mean the value of R_1 is not completely computed by the values of the integer variables d.

The model is given by equation:

$$\begin{cases} \dot{x}_1 = \dfrac{1}{\rho \cdot A_1}\left(w(t) + k_2\sqrt{\rho g x_2} - 0.2c_1(d_1 + d_2 + d_3 + d_4 + d_5)\sqrt{\rho g x_1} - c_2 R_2\sqrt{\rho g x_1}\right) \\ \dot{x}_2 = \dfrac{1}{\rho \cdot A_2}\left(0.2c_1(d_1 + d_2 + d_3 + d_4 + d_5)\sqrt{\rho g x_1} - k_2\sqrt{\rho g x_2}\right) \\ \dot{x}_3 = \dfrac{1}{\rho \cdot A_3}\left(c_2 R_2\sqrt{\rho g x_1} - k_3\sqrt{\rho g x_3}\right) \end{cases} \qquad [4.21]$$

The Φ and Γ matrices of the linearised model are given in appendix 7.
The input is defined as:

$$u(k) = \begin{bmatrix} R_2 & w & d_1 & d_2 & d_3 & d_4 & d_5 \end{bmatrix}^T \quad [4.22]$$

The state is defined as:

$$x(k) = \begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix}^T \quad [4.23]$$

In comparison with the previous model, the Φ-matrix does not contain any multiplication between the input variables. The different variables in the input are now independent, which simplifies the optimisation.
The different variables in the input vector are independent, which is a great advantage in comparison with the previous optimisation scheme. The different constraints, as used in the relative optimisation problem, have changed. They will be described below.

4.2.2 Constraints for absolute optimisation

- The move constraints are defined as:

$$\begin{aligned} -mc \le u(k+1) - u(k) \le mc \\ -mc \le u(k+2) - u(k+1) \le mc \end{aligned} \quad [4.24]$$

  In this equation *mc* is a column of the same size a *u*, which gives the values for the move constraints. The move constraints are for a control and prediction horizon of three defined in equation 4.25. The move constraints are in this equation only implemented for the continuous variables as ΔR_2 and Δw. The discrete variables do not contain move constraints.

$$\begin{bmatrix} -\Delta R_2 + R_2(k) \\ -\Delta w + w(k) \\ -\Delta R_2 \\ -\Delta w \\ -\Delta R_2 \\ -\Delta w \end{bmatrix} \le \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 \end{bmatrix} \begin{bmatrix} R_2(k+1) \\ w(k+1) \\ R_2(k+2) \\ w(k+2) \\ R_2(k+3) \\ w(k+3) \end{bmatrix} \le \begin{bmatrix} \Delta R_2 + R_2(k) \\ \Delta w + w(k) \\ \Delta R_2 \\ \Delta w \\ \Delta R_2 \\ \Delta w \end{bmatrix} \quad [4.25]$$

- The constraints on the inputs are:

$$\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} R_2 \\ w \\ d_1 \\ d_2 \\ d_3 \\ d_4 \\ d_5 \end{bmatrix} \le \begin{bmatrix} 1 \\ \text{inf} \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \quad [4.26]$$

- The constraints on the outputs, the water heights in the three tanks can be derived from equation 1.16. The lower bounds and upper bounds are the same as in the first simulation, equation 4.14. This results in:

$$lb - \begin{bmatrix} \Phi \\ \Phi^2 \\ \vdots \\ \Phi^m \\ \Phi^{m+1} \\ \vdots \\ \Phi^p \end{bmatrix} x(k+1|k) \leq \begin{bmatrix} \Gamma & 0 & \cdots & \cdots & 0 \\ \Phi\Gamma & \Gamma & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ \Phi^{m-1}\Gamma & \Phi^{m-2}\Gamma & \cdots & \Phi\Gamma & \Gamma \\ \Phi^m\Gamma & \Phi^{m-1}\Gamma & \cdots & \Phi^2\Gamma & \Phi\Gamma + \kappa\Gamma \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \Phi^{p-1}\Gamma & \Phi^{p-2}\Gamma & \cdots & \Phi^{p+1-m}\Gamma & \Phi^{p-m}\Gamma + \kappa \sum_{i=0}^{p-m-1} \Phi^i\Gamma \end{bmatrix} \begin{bmatrix} u(k+1|k) \\ \vdots \\ u(k+m|k) \end{bmatrix} \leq ub - \begin{bmatrix} \Phi \\ \Phi^2 \\ \vdots \\ \Phi^m \\ \Phi^{m+1} \\ \vdots \\ \Phi^p \end{bmatrix} x(k+1|k)$$

[4.27]

- Finally the constraints between the integer inputs are given by equation 4.28. They are derived from equation 4.18. The first equation (of 4.18) is not needed, because all integer inputs have a maximal value of 1, the sum is therefore always smaller or equal to five.

$$\begin{bmatrix} -1 \\ -1 \\ -1 \\ -1 \end{bmatrix} \leq \begin{bmatrix} 0 & 0 & 0 & -1 & 1 \\ 0 & 0 & -1 & 1 & 0 \\ 0 & -1 & 1 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ d_4 \\ d_5 \end{bmatrix} \leq \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \qquad [4.28]$$

The discrete non-linear model is linearised in the Matlab program. Therefore these constraints need to be written as a function of the linearised variables. The input is:

$$\begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ d_4 \\ d_5 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 0 \end{bmatrix} \text{ and linearised } \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ d_4 \\ d_5 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \qquad [4.29]$$

The only difference in the equation is for d_4- d_3, because d_3 can have values 0 (open) and –1 (closed) and d_4 can have values 1 (open) and 0 (closed). So the values for open and closed are not corresponding. The other equations do not give this problem, because their values for open and closed are corresponding when the difference between to following d_i's is computed.

The implementation is therefore:

$$\begin{bmatrix} -1 \\ 0 \\ -1 \\ -1 \end{bmatrix} \leq \begin{bmatrix} 0 & 0 & 0 & -1 & 1 \\ 0 & 0 & -1 & 1 & 0 \\ 0 & -1 & 1 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ d_4 \\ d_5 \end{bmatrix} \leq \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \qquad [4.30]$$

4.2.3 Results

Different simulations are done on this system. Two of them will be presented below.

The setpoint (x0+0.1, y0+0.1, z0) gives the next results.

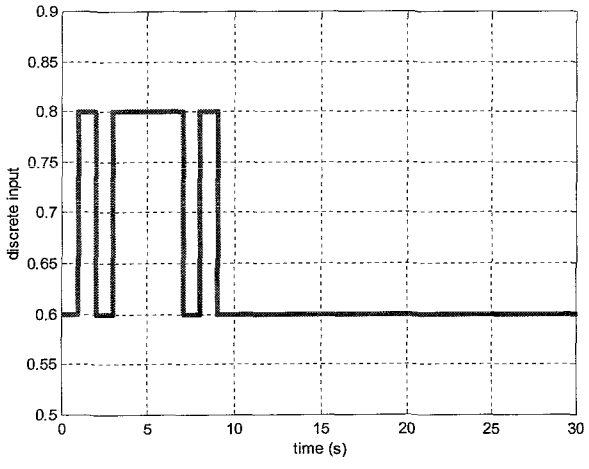


Figure 4.2: discrete input (first simulation)

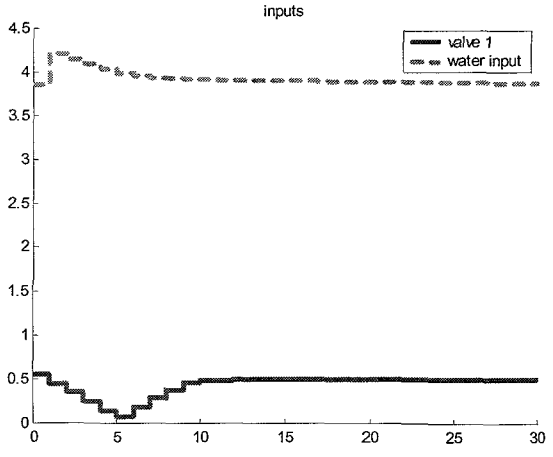


Figure 4.3: continuous inputs (first simulation)

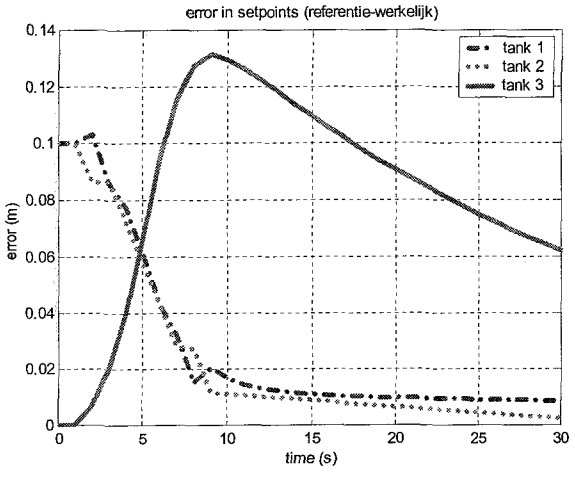


Figure 4.4: error in setpoints (first simulation)

As can be seen the discrete and the continuous inputs become constant, the error of water tank 1 and 2 tends to zero. Water tank three shows a descending error, the water height in this tank is only controlled by the output of the continuous valve in water tank 1. This output is less weighted than the others, so it is acceptable this error is relative large. The discrete input in figure 4.2 shows a correct behaviour, it uses only the values specified with the constraints and in equation 4.17.Figure 4.3 shows the continuous inputs, for valve R2 the only constraints was that it has values between 0 and 1, the result is therefore satisfying. For the water input no constraints were used. These results are satisfying. To check whether the optimisation scheme gives satisfying answers for different setpoints another problem is solved.

The setpoint (x0-0.1, y0+0.1, z0) gives the next results.

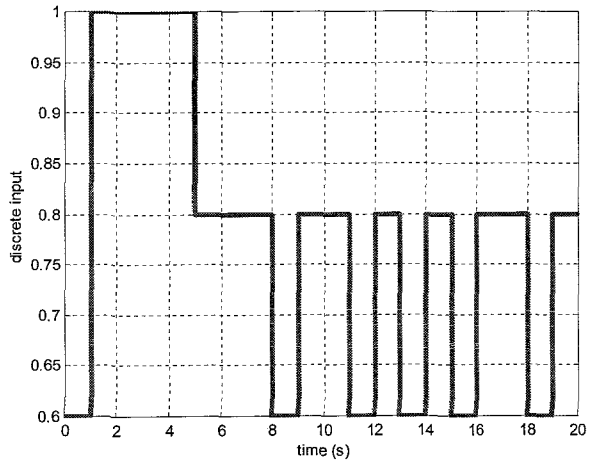


Figure 4.5: Discrete inputs (second simulation)

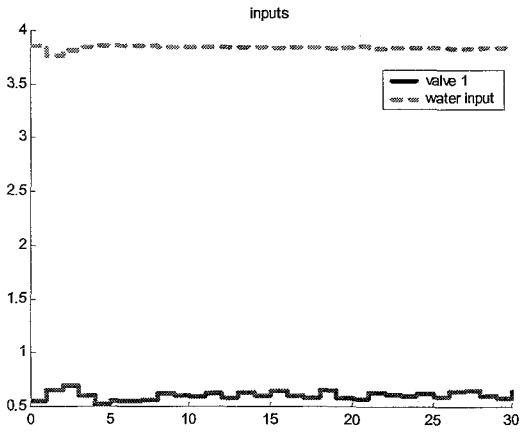


Figure 4.6: other inputs (second simulation)

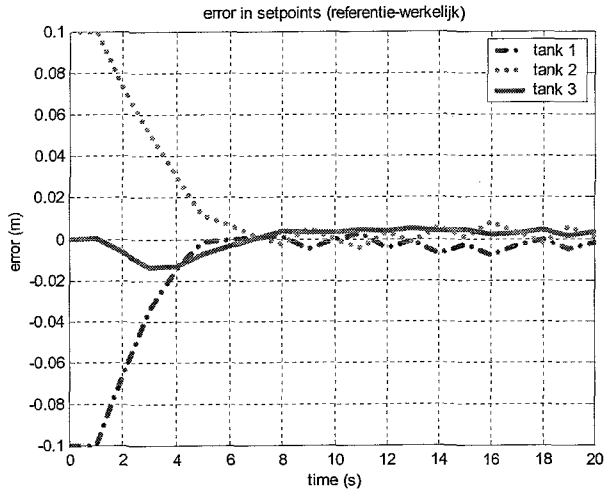


Figure 4.7: Error in the setpoints (second simulation)

As can been seen from figure 4.5 the discrete input shows a chattering behaviour, it cannot find an optimum. The optimum is somewhere between 0.6 end 0.8. Valve R_2 does not reach a constant value, because it shows also some chattering. The water input end up with a more constant value, but still less than in the previous example. It was expected the water input and the continuous valve would influence with their choices of values the capability to reach a constant value for the discrete valve, but this does not happen. Using weighing filters and implement a penalty on the last inputs can probably solve this.
The optimisation scheme can compute satisfying results, but the quality is dependent of the chosen setpoint. The setpoints that are used do not ask much changes in the value. Larger problems occur when the difference between the setpoints and the initial values increases.

Conclusions

Logical constraints can be rewritten into inequalities and used in an optimisation problem, as was shown in paragraph 4.2. Mixed integer predictive control can use them in its optimisation. With this routine MIPC is able to control discrete systems, but there are still problems. Chattering occurs for many setpoints and large differences between initial values and setpoints cannot be controlled successful. The use of weighing filters can probably solve this.
The absolute optimisation scheme of MPC is tested for the discrete problem and gives rather satisfying results, except the problem of chattering. A final error occurs as was mentioned in chapter one.
The model with non-continuous inputs could not be controlled with the relative scheme of MIPC. When the model is written without independence and the constraints are not overlying this should give a correct result.

Recommendations

- To make the implementation of the discrete inputs easier it is useful to test if the integer values can also become larger than 1. If this is possible the discrete input can be given by just one integer variable (d in the model) instead of five different integer variables.
- The optimisation scheme is only successful tested for a model with discrete inputs. It must be useful for all different MLD systems; therefore it should be tested on a non-continuous model.
- The relative optimisation scheme is mostly used in standard MPC control. It is easier, when linearised models are used, to construct the matrices and vectors for the constraints. Further on the error in the last sample is smaller, as was mentioned in chapter 1. It is therefore useful to test the relative optimisation scheme again, but with independent inputs.
- The use of weighing filters for the error and the input-changes should be tested. These weighing matrices are given in chapter one as Q and R. Their values are not changed in the current optimisation. Their values determine the importance of the error and the new inputs in the optimisation. A larger weight of the input-changes gives smaller input-changes between the different samples, but the error is less important and can therefore decrease.
- Filters can be used when the prediction differs from the measurements. This error is due to model mismatches, unmodelled phenomena and linearisation of the non-linear model. A filter (i.e. model-based) should adapt the states of the model in such a way that they are equal to the measured states.
- The non-continuous model has to be rewritten without independence or in the optimisation should be given that the product of R and d is the parameter that has to be optimised. This is possible when an auxiliary variable z is implemented as in chapter 2. But the optimisation problem has to optimise z in this case, with constraints for R and d. This should be given in the optimisation scheme.

Literature

1. A. Bemporad, M. Morari, *Control of systems integrating logic, dynamics and constraints*, Automatica 35 (1999), 407-427
2. dr.ir. H.A. van Essen, Model Predictive Control

Appendix 1: List of symbols

This appendix gives all symbols used in this report. It contains the numbers, vectors and matrices.

Table A1.1 gives the numbers that are used in this report. The values of constant parameters used in this report are given in the last column.

Symbol	Explanation	Dimension	Value
Numbers			
k	Sample	-	
l	Counting number	-	
p	Length of the prediction horizon	-	
m	Length of the control horizon	-	$m \leq p$
X_i	Statement	-	e.g. true/false
ε	Small number, often machine precision	-	e.g. 1e-16
d	Integer value	-	0 or 1
δ	Integer value	-	0 or 1
ρ	Density	kg/m^3	1e3
A	Surface	m^2	0.1
k	Constant valve value	$(kg \cdot m)^{1/2}$	0.055
c	Scaling value of the valves	$(kg \cdot m)^{1/2}$	0.06
R	Normalised valve-value	-	Between 0 and 1
w	Water input	kg/s	
g	Constant of gravity	m/s^2	9.81

Table A1.1: Numbers

The vectors and their explanation are given in table A1.2. For the vectors no dimensions and values are given, because the size, dimension and values are dependent of the problem.

Vectors	**Explanation**
x	State
y	Measured output
u	Input
r	Reference
e	Tracking error
G	Gradient-column
c	Column with constraints together with matrix C
lb	Lowerbound
ub	Upperbound
mc	Column with values for the move-constraints
s_s	Vector with all different states
s^p	Vector over the whole prediction horizon
$s_{s,\Delta u=0}$	Vector with all different states for $\Delta u=0$
$s(k+l)$	Vector at sample k predicted l steps ahead.
$s(k+l\|k)$	Estimation of the vector $s(k+l)$.

Table A1.2: Vectors

Matrices	**Explanation**
Φ	Matrix which gives the relation between the new state and the current state
Γ	Matrix which gives the relation between the new state and the input
C	Matrix which gives the relation between the measured output and the state
D	Matrix which gives the relation between the measured output and the input
Q	Matrix with weights for the relevance of the output deviations
R	Matrix with weights for the actions of the manipulated variables
Y	Prediction matrix
H	Hessian-matrix
C	Matrix with constraints together with column *c*

Table A1.3 Matrices

There are no dimensions and values given for these matrices. The size, the dimension and the values are dependent of the problem.

Appendix 2: Constraints for piecewise linear system

$$
\begin{bmatrix}
-1 & -1 & -1 \\
1 & 1 & 1 \\
m_1 - M_1 & 0 & 0 \\
0 & m_2 - M_2 & 0 \\
0 & 0 & m_3 - M_3 \\
M_1 & 0 & 0 \\
0 & M_2 & 0 \\
0 & 0 & M_3 \\
-M_1 & 0 & 0 \\
0 & -M_2 & 0 \\
0 & 0 & -M_3 \\
m_1 & 0 & 0 \\
0 & m_2 & 0 \\
0 & 0 & m_3 \\
-m_1 & 0 & 0 \\
0 & -m_2 & 0 \\
0 & 0 & -m_3 \\
M_1 & 0 & 0 \\
0 & M_2 & 0 \\
0 & 0 & M_3
\end{bmatrix}
\begin{bmatrix} \delta_1(t) \\ \delta_2(t) \\ \delta_3(t) \end{bmatrix} +
\begin{bmatrix}
0 & 0 & 0 \\
0 & 0 & 0 \\
0 & 0 & 0 \\
0 & 0 & 0 \\
0 & 0 & 0 \\
0 & 0 & 0 \\
0 & 0 & 0 \\
0 & 0 & 0 \\
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1 \\
-1 & 0 & 0 \\
0 & -1 & 0 \\
0 & 0 & -1 \\
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1 \\
-1 & 0 & 0 \\
0 & -1 & 0 \\
0 & 0 & -1
\end{bmatrix}
\begin{bmatrix} z_1(t) \\ z_2(t) \\ z_3(t) \end{bmatrix} \leq
\begin{bmatrix}
0 \\ 0 \\ B_1 \\ B_2 \\ B_3 \\ -B_1 \\ -B_2 \\ -B_3 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ B_1 \\ B_2 \\ B_3 \\ -B_1 \\ -B_2 \\ -B_3
\end{bmatrix} u(t) +
\begin{bmatrix}
0 \\ 0 \\ A_1 \\ A_2 \\ A_3 \\ -A_1 \\ -A_2 \\ -A_3 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ A_1 \\ A_2 \\ A_3 \\ -A_1 \\ -A_2 \\ -A_3
\end{bmatrix} x(t) +
\begin{bmatrix}
-1 \\ 1 \\ -M_1 \\ -M_2 \\ -M_3 \\ 2M_1 \\ 2M_2 \\ 2M_3 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ -m_1 \\ -m_2 \\ -m_3 \\ M_1 \\ M_2 \\ M_3
\end{bmatrix}
$$

Appendix 3: Determination of the parameters

Time constants

The parameters of the system have to be determined, because they influence the time constant of the water tanks. A time constant of the tanks between 5 and 30 seconds is desired. To compute this some simulations are done.
In these simulations the water tank, whose time constant is determined, is completely filled and the inputs and the other water heights are set to zero. These simulations give the time constant for a completely filled water tank without external disturbances.
The figures show the behaviour of the different water tanks. The time constant is equal to the gradient of the water height-time curve.

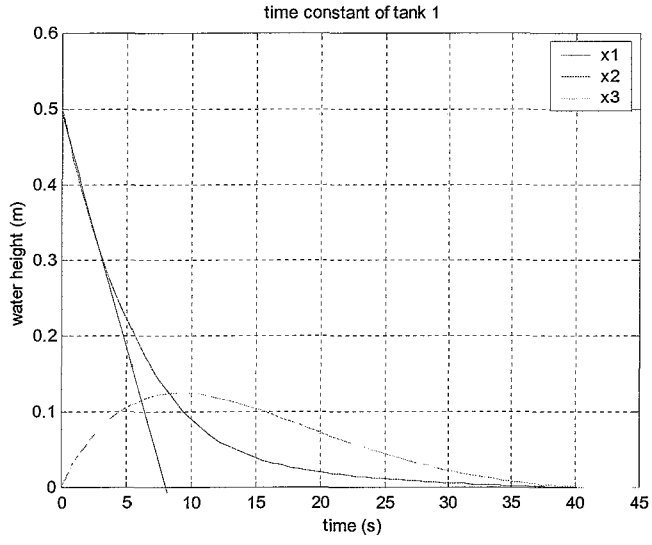


Figure 1:Time constant of water tank one.

The time constant of water tank one is 8 seconds. Water tanks two and three show the same behaviour. This is correct, because both water tanks are filled with the same amount of water form tank one. The valves are namely both completely opened.

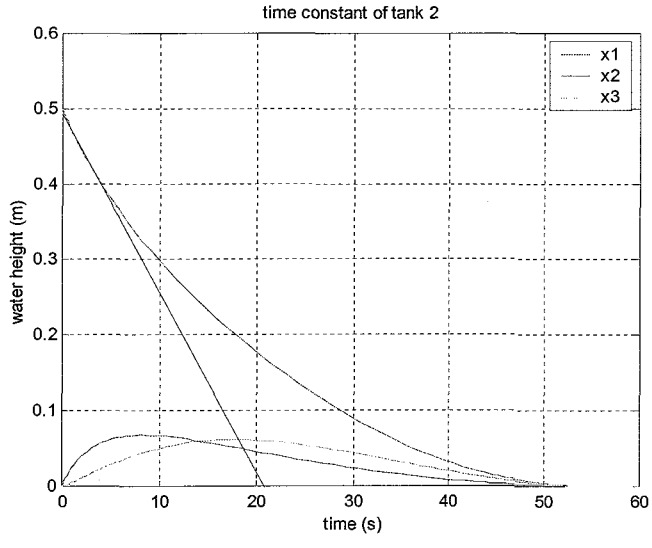


Figure 2:Time constant of water tank two.

When water tank two is filled, this water tank gives also water to tank one and therefore the water height in water tank one and three increases. Water tank one is in comparison with tank three sooner empty, because it can loose more water by the two controllable valves. Water tank one returns some water to tank two, which results in a larger time constant for tank 2 in comparison with the first water tank. This difference in time is mostly due to the different values of the valves, so less water can leave tank two. The time constant is 21 seconds.

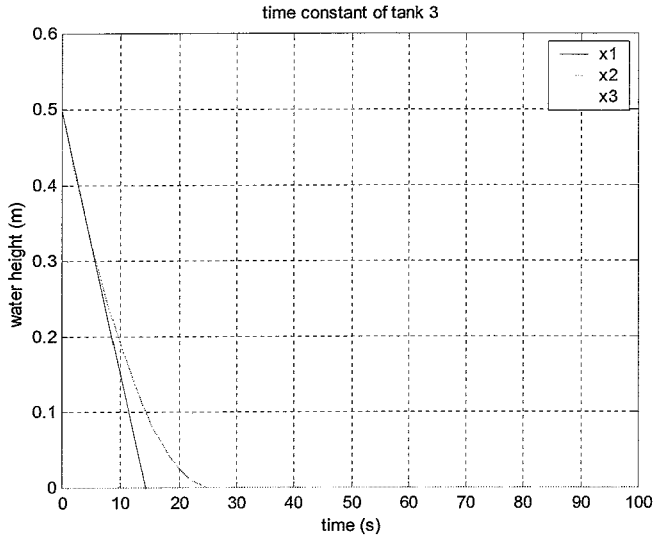


Figure 3:Time constant of water tank three.

Water tank three does not fill any of the other tanks, so they will stay empty. The time constant of tank three is 14 seconds. It is faster than tank two, because it is not filled anymore.

To achieve these results the next parameters were used:

Parameter	Value
k_2	0.055 $(kg \cdot m)^{1/2}$
k_3	0.055 $(kg \cdot m)^{1/2}$
c_1	0.06 $(kg \cdot m)^{1/2}$
c_2	0.06 $(kg \cdot m)^{1/2}$
A_1	0.1 m^2
A_2	0.1 m^2
A_3	0.1 m^2
w	0 m^3/s

Table 1: used parameters for calculating the time constants

Initial values

The initial value has to be a stable point. To achieve this, all differential equations should be equal to zero. There are more variables than equations, so some parameters can be chosen. The surfaces do not influence the derivative of the water height. They are all chosen to 0.1 m^2. The other necessary values are the water heights and the values of the inputs (both valves and water input). The constant parameters and the initial values are given in table 2 and 3. These values will be used in the optimisation problem.

Parameter	Symbol	Value
Constant valve values	k_2	0.055 $(kg \cdot m)^{1/2}$
	k_3	0.055 $(kg \cdot m)^{1/2}$
Scaling value controllable valves	c_1	0.06 $(kg \cdot m)^{1/2}$
	c_2	0.06 $(kg \cdot m)^{1/2}$
Surface	A_1	0.1 m^2
	A_2	0.1 m^2
	A_3	0.1 m^2

Table 2: The constant parameters of the system.

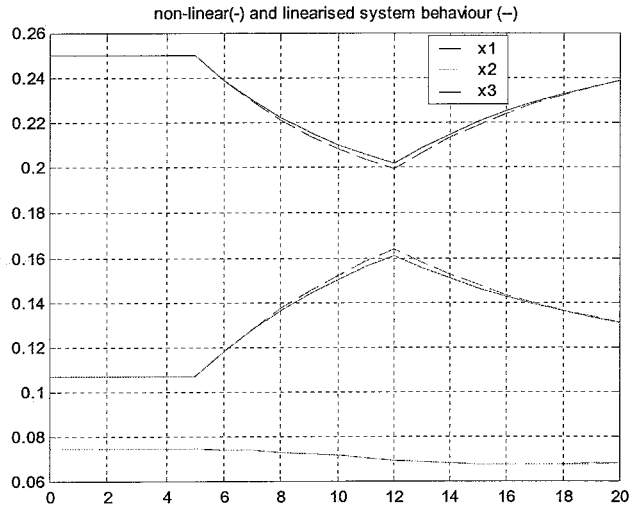
Parameter	Symbol	Value
Controllable valve values	R_{10}	0.6
	R_{20}	0.55
Water input	w_0	3.852 kg/s
Water height	x_{10}	0.5 m
	x_{20}	0.59 m
	x_{30}	0.5 m

Table 3: The initial values of the system.

Appendix 4: Reliability of the linearised and discrete linearised model

Linearised model

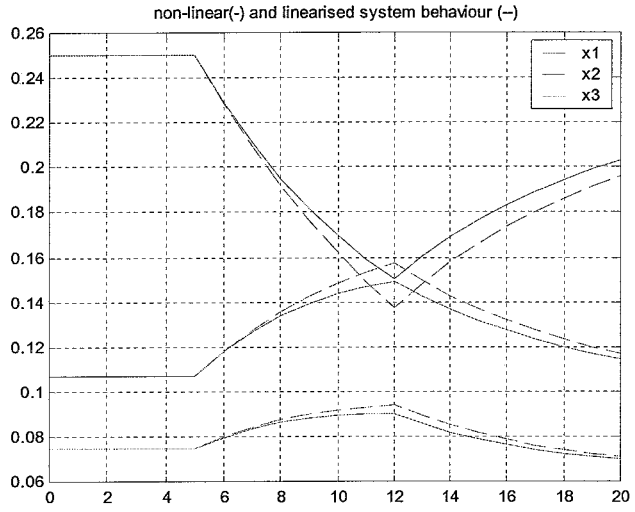
To check whether the linearisation gives a correct behaviour the non-linear model and the linearised model are simulated in Matlab. For a given input u the behaviour of the system is given in figure 1. As can be seen the linearised system gives a good approximation for the non-linear model.



The input was defined as:
n=[0.5; 1.4857; 1; 1; 1; 0;0]';
y=[0.5; 1.4857; 1; 1; 1; 1;1]';
upk=[n;n;n;n;n;y;y;y;y;y;y;n;n;n;
n;n;n;n;n]; xpk0=[0.25; 0.1071;
0.0744];

Figure 1: Comparison of the non-linear and linearised system

In this case only the value of the discrete valve was changed, when the continuous, discrete valves and the water input are changed the approximation of the linearised model is less exact, as can be seen in figure 2.

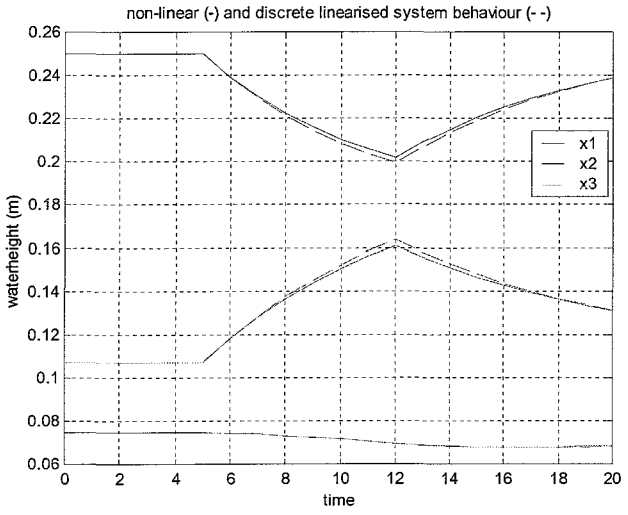


The inputs were in this case:
n=[0.5; 1.4857; 1; 1; 1; 0;0]';
y=[0.7; 1; 1; 1; 1; 1;1]';
upk=[n;n;n;n;n;y;y;y;y;y;y;n;n;
n;n;n;n;n];
xpk0=[0.25; 0.1071; 0.0744];

Figure 2: Comparison of the non-linear and linearised system

Discrete linearised model

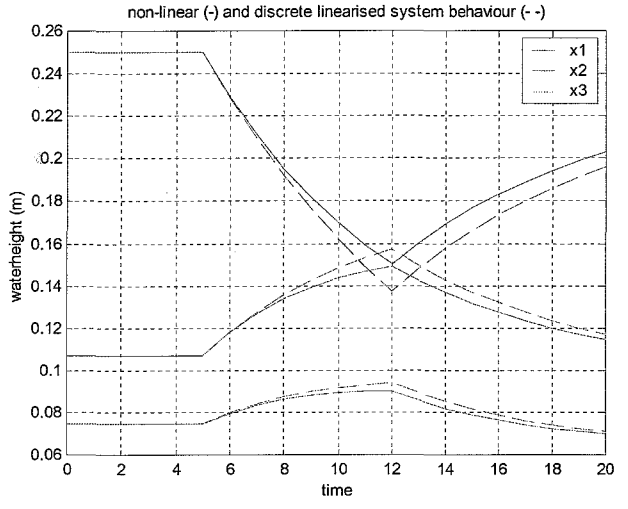
To check whether the discretisation gives a correct behaviour the non-linear model and the discrete linearised model are simulated in Matlab. The behaviour of the system and is discretisation is given in figure 3. As can be seen the discrete linearised system gives a good approximation for the non-linear model.



The input was defined as:
n=[0.5; 1.4857; 1; 1; 1; 0;0]';
y=[0.5; 1.4857; 1; 1; 1; 1;1]';
upk=[n;n;n;n;n;y;y;y;y;y;y;y;n;n;n;n;
n;n;n;n];
xpk0=[0.25; 0.1071; 0.0744];

Figure 3: Comparison of the non-linear and discrete linearised system

In this case only the value of the discrete valve was changed, when the continuous, discrete valves and the water input are changed the approximation of the discrete model is less exact, just as in the case of the linearised model.



The inputs are in this case:
n=[0.5; 1.4857; 1; 1; 1; 0;0]';
y=[0.7; 1; 1; 1; 1; 1;1]';
upk=[n;n;n;n;n;y;y;y;y;y;y;y;n;n;n;n;
n;n;n;n];
xpk0=[0.25; 0.1071; 0.0744];

Figure 4: Comparison of the non-linear and discrete linearised system

The approximation is still not worse, but is less than the previous one. The discrete linear system and the linearised system only give a good approximation if the values are near the linearisation point. These changes are larger and therefore the error is larger than in the previous example. The first five seconds of the setpoint are stable; this can be seen in both the non-linear and the discrete and linearised curves. The error of the discrete and linearised curves increases when the setpoint uses an instable point. The error decreases if the stable point is reached again.

Appendix 5: Results of the model with non-continuous inputs

The optimisation of this model is not successful, as can be seen in the next figures. From each simulation the error, the inputs and the setpoints, in comparison with the behaviour of the different water tanks are given.

At first an optimisation is done with all constraints. This gives the next results.

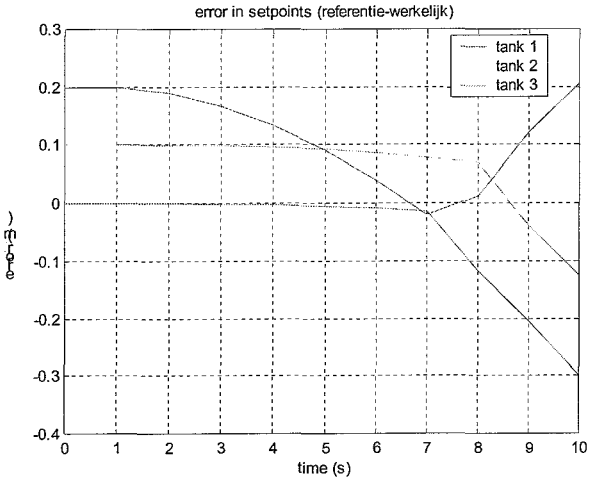


Figure 1: The error in the setpoints

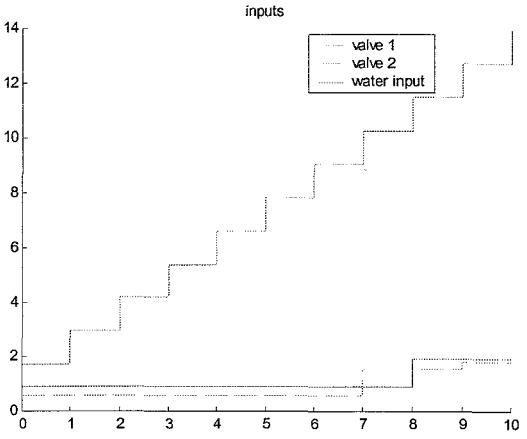


Figure 2: The inputs

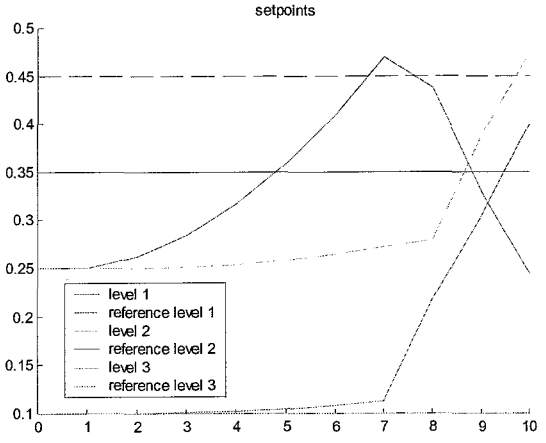


Figure 3: The setpoints for this optimisation

The optimisation gives infeasible from sample 7. This can be seen in the figures. The optimisation was not able to solve the problem, without becoming infeasible. The integer values are not seen as integers anymore and can get every value. In figure two it can be seen that the values of the valves are no longer made in steps of 0.2. Especially the end steps show more deviations. The logical constraints give therefore problems. This can be due to the move constraints on the integer values that are used in the optimisation or to the logical constraints itself. Exclusion of one of these constraints gives the same result, therefore the optimisation is also computed without logical constraints and without move constraints on the integer values and it gives:

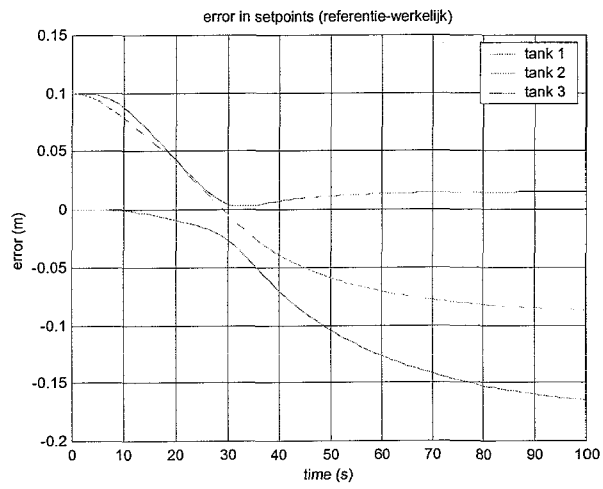


Figure 4: Error (no logical constraints)

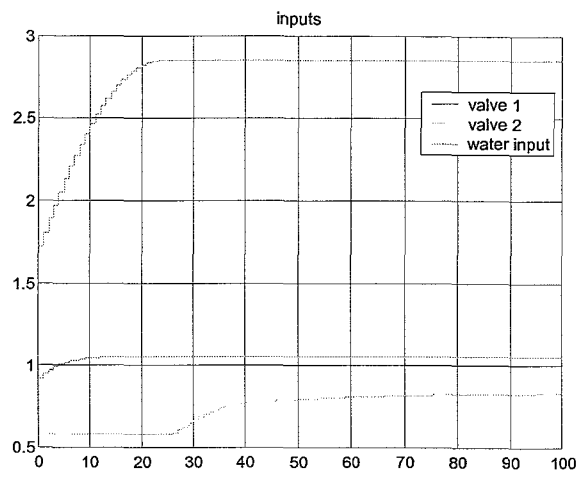


Figure 5: Inputs (no logical constraints)

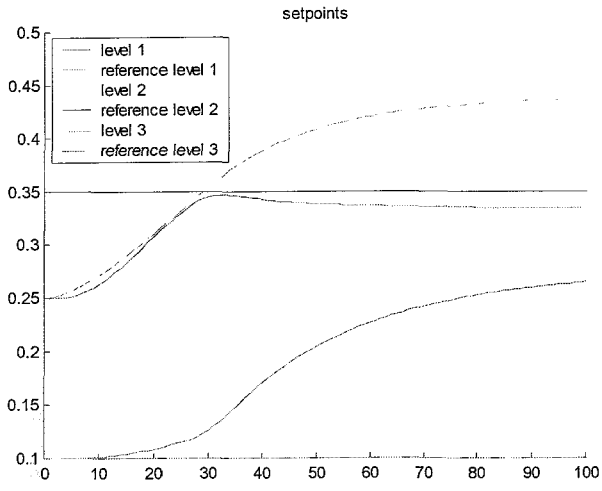


Figure 6: Setpoints and behaviour (no logical constraints)

This optimisation terminates successful. The error tends for the first water tank to zero; the second one shows a larger error. The valves are used in the optimisation problem, and show some satisfying result, although the maximal value is not correct. This error is due to the exclusion of the constraints on the valves. From these results it can be concluded that there are problems with the implementation of the logical constraints.

Appendix 6: The Φ and Γ matrices of the model with non-continuous inputs

$$\Phi = \begin{bmatrix} -\frac{1}{2A_1}\frac{g}{\sqrt{\rho g x_{10}}}\left(c_1 R_{10} d_{10} + c_2 R_{20} d_{20}\right) & \frac{1}{2A_1}\frac{g}{\sqrt{\rho g x_{20}}}k_2 & 0 \\ \frac{1}{2A_2}\frac{g}{\sqrt{\rho g x_{10}}}c_1 R_{10} d_{10} & \frac{1}{2A_2}\frac{g}{\sqrt{\rho g x_{20}}}k_2 & 0 \\ 0 & \frac{1}{2A_1}\frac{g}{\sqrt{\rho g x_1}}\left(c_2 R_{20} d_{20}\right) & \frac{1}{2A_3}\frac{g}{\sqrt{\rho g x_{30}}}k_3 \end{bmatrix}$$

$$\Gamma = \begin{bmatrix} -\frac{1}{\rho A_1}c_1 d_{10}\sqrt{\rho g x_{10}} & -\frac{1}{\rho A_1}c_2 d_{20}\sqrt{\rho g x_{10}} & \frac{1}{\rho A_1} & -\frac{1}{\rho A_1}c_1 R_{10}\sqrt{\rho g x_{10}} & -\frac{1}{\rho A_1}c_2 R_{20}\sqrt{\rho g x_{10}} \\ \frac{1}{\rho A_2}c_1 d_{10}\sqrt{\rho g x_{10}} & 0 & 0 & \frac{1}{\rho A_2}c_1 R_{10}\sqrt{\rho g x_{10}} & 0 \\ 0 & \frac{1}{\rho A_3}c_2 d_{20}\sqrt{\rho g x_{10}} & 0 & 0 & \frac{1}{\rho A_3}c_2 R_{20}\sqrt{\rho g x_{10}} \end{bmatrix}$$

Appendix 7: The Φ and Γ matrices of the model with a discrete input

$$\Phi = \begin{bmatrix} -\frac{1}{2A_1}\frac{g}{\sqrt{\rho g x_{10}}}\left(0.2c_1\left(d_{10}+d_{20}+d_{30}+d_{40}+d_{50}\right)+c_2R_{20}\right) & \frac{1}{2A_1}\frac{g}{\sqrt{\rho g x_{20}}}k_2 & 0 \\ \frac{1}{2A_2}\frac{g}{\sqrt{\rho g x_{10}}}0.2c_1\left(d_{10}+d_{20}+d_{30}+d_{40}+d_{50}\right) & -\frac{1}{2A_2}\frac{g}{\sqrt{\rho g x_{20}}}k_2 & 0 \\ \frac{1}{2A_3}\frac{g}{\sqrt{\rho g x_1}}c_2R_{20} & 0 & -\frac{1}{2A_3}\frac{g}{\sqrt{\rho g x_{30}}}k_3 \end{bmatrix}$$

$$\Gamma = \begin{bmatrix} -\frac{1}{\rho A_1}c_2\sqrt{\rho g x_{10}} & \frac{1}{\rho A_1} & -\frac{0.2}{\rho A_1}c_1\sqrt{\rho g x_{10}} & -\frac{0.2}{\rho A_1}c_1\sqrt{\rho g x_{10}} & -\frac{0.2}{\rho A_1}c_1\sqrt{\rho g x_{10}} & -\frac{0.2}{\rho A_1}c_1\sqrt{\rho g x_{10}} & -\frac{0.2}{\rho A_1}c_1\sqrt{\rho g x_{10}} \\ 0 & 0 & \frac{0.2}{\rho A_2}c_1\sqrt{\rho g x_{10}} & \frac{0.2}{\rho A_2}c_1\sqrt{\rho g x_{10}} & \frac{0.2}{\rho A_2}c_1\sqrt{\rho g x_{10}} & \frac{0.2}{\rho A_2}c_1\sqrt{\rho g x_{10}} & \frac{0.2}{\rho A_2}c_1\sqrt{\rho g x_{10}} \\ \frac{1}{\rho A_3}c_2\sqrt{\rho g x_{10}} & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$