Nonlinear Model Predictive Control of the Four Tank Process

IVANA DRCA

KTH Electrical Engineering

Master's Degree Project
Stockholm, Sweden July 2007

XR-EE-RT 2007:016

Abstract

Model predictive control techniques are widely used in the process industry. They are considered methods that give good performance and are able to operate during long periods without almost any intervention. Model predictive control is also the only technique that is able to consider model restrictions.

Almost all industrial processes have nonlinear dynamics, however most MPC applications are based on linear models. Linear models do not always give a sufficiently adequate representation of the system and therefore nonlinear model predictive control techniques have to be considered. Working with nonlinear models give rise to a wide range of difficulties such as, non convex optimization problems, slow processes and a different approach to guarantee stability .

This project deals with nonlinear model predictive control and is written at the University of Seville at the department of Systems and Automatic control and at the department of Automatic Control at KTH. The first objective is to control the nonlinear Four Tank Process using nonlinear model predictive control. Objective number two is to investigate if and how the computational time and complexity can be reduced.

Simulations show that a nonlinear model predictive control algorithm is developed with satisfactory results. The algorithm is fast enough and all restrictions are respected for initial state values inside of the terminal set as well as for initial state values outside of the terminal set. Feasibility and stability is ensured for both short as well as for longer prediction horizon, guaranteeing that the output reaches the reference. Hence the choice of a short respectively long prediction horizon is a trade off between shorter computational time versus better precision.

Regarding the reduction of the computational time, penalty functions have been implemented in the optimization problem converting it to an unconstrained optimization problem including a PHASE-I problem. Results show that this implementation give approximately the same computational time as for the constrained optimization problem. Precision is good for implementations with penalty functions both for long and short prediction horizons and initial state values inside and outside of the terminal set.

Acknowledgements

I would like to thank my examiner at the Automatic Control Lab at KTH, Associate professor Karl Henrik Johansson, and the Head of department at the Department of System and Automatic Control at the University of Seville, Eduardo Fernandez Camacho, for making it possible for me to work with this project.

A special thanks to my supervisor at the Department of System and Automatic control at the University of Seville, Daniel Limón, for all the support, great ideas and guidance through out these months. Thank you Daniel!

Last but not least I would like to thank all the PhD students at the the Department of System and Automatic Control at the University of Seville for a great time and good friendship.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Introduction

Model predictive control techniques have been used in the process industry for nearly 30 years and are considered as methods that give good performance and are able to operate during long periods without almost any intervention. However the main reason that model predictive control is such a popular control technique in modern industry is that it is the only technique that allows system restrictions to be taken into consideration.

The majority of all industrial processes have nonlinear dynamics, however most MPC applications are based on linear models. These linear models require that the original process is operating in the neighborhood of a stationary point. However there are processes that can't be represented by a linear model and require the use of nonlinear models. Working with nonlinear models give rise to a wide range of difficulties such as, a non convex optimization problem, different approach to guarantee stability and in general a slow process.

1.2 Objective

In this project the objective is to make a nonlinear MPC algorithm for the four tank process and investigate if the computational cost, time and complexity of the optimization problem can be reduced. This involves implementation of penalty functions in the objective function converting the restricted optimization problem into an unrestricted optimization problem. A comparison between the two models has been done with respect to system performance, precision and computational time. The algorithm is implemented in MatLab.

1.3 Outline

The plant description is given in chapter 2. An introduction of MPC in general is shown in chapter 3. Chapter 4 shows the specific method and parameters chosen for this project. The theory and implementation of the penalty functions is given in chapter 5. A method of finding an initial feasible input is presented in chapter 6. All the results of this project are shown in chapter 7. Conclusions and some future work are presented in chapter 8.

2 The Four Tank Process

In this section the Four Tank Process is described. The dynamics and parameters are shown. The linearization of the model is also presented as it is used as an ingredient when modeling the nonlinear model predictive control algorithm.

2.1 Description of the real system

The four tank process was first proposed by [1] and consists of four connected tanks as shown in Fig 1. Pump A extracts water from the basin below and pours it to tank 1 and 4, while pump B pours to tank 2 and 3. The relationship between the flows at each outlet pipe and the total flow from pump A and pump B depends on the flow parameters γ_1 and γ_2 as:

$$\begin{aligned} q_1 &= \gamma_1 q_a \\ q_2 &= \gamma_2 q_b \\ q_3 &= (1-\gamma_2) q_b \\ q_4 &= (1-\gamma_1) q_a \end{aligned}$$

which are the water flows to each tank. The flow parameters γ_1 and γ_2 can vary between 0 and 1, i.e. $0 \leq \gamma 1 \leq 1$ and $0 \leq \gamma 2 \leq 1$.

The plant on which this project has been done on is a large scale model of the original plant. The plant can be seen in Fig 2. The differential equations that describe the system dynamics are taken from [1] and are a simplified description without for example friction. The dynamics look like:

$$\begin{aligned} \frac{dh_1}{dt} &= -\frac{a_1}{A}_1 \cdot \sqrt{2gh_1} + \frac{a_3}{A}_1 \cdot \sqrt{2gh_3} + \frac{\gamma_1}{A}_1 \cdot q_a \\ \frac{dh_2}{dt} &= -\frac{a_2}{A}_2 \cdot \sqrt{2gh_2} + \frac{a_4}{A}_2 \cdot \sqrt{2gh_4} + \frac{\gamma_2}{A}_2 \cdot q_b \\ \frac{dh_3}{dt} &= -\frac{a_3}{A}_3 \cdot \sqrt{2gh_3} + \frac{(1-\gamma_2)}{A}_3 \cdot q_b \\ \frac{dh_3}{dt} &= -\frac{a_4}{A}_4 \cdot \sqrt{2gh_4} + \frac{(1-\gamma_1)}{A}_4 \cdot q_a \end{aligned} \tag{1}$$

where the estimated parameter values of the real plant are shown in Table 1, see [5].

There are many different configurations available with the real plant, however the configuration applied in this project is the same as in the original one. For a more detailed description of the large scale plant see [4] and [5].

As earlier mentioned the objective of this project is to control the water level in the two lower tanks. This is done controlling the pump flow, q_a and q_b. Due to the strong coupling between the tanks this task results rather difficult to fulfill. If the original plant was considered a linearization of the model could be used, see [4]. However working with the big plant the nonlinearities are considered because of several reasons: the variation range of the water level in the big plant increases, a nonlinear model also gives a better representation of the system which in turn gives better performance and moreover the objective of this project is to develop a nonlinear model predictive control algorithm.

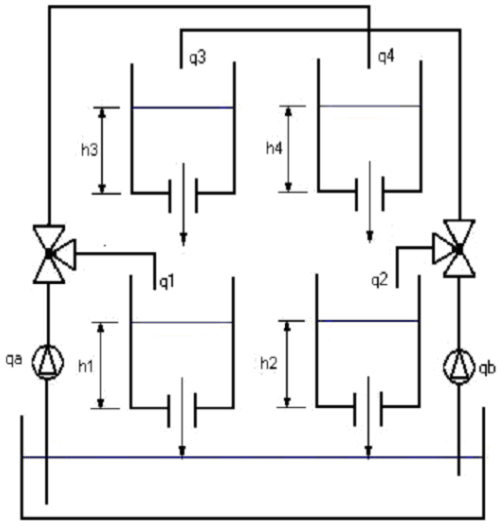

Figure 1: The original plant of the four tanks.

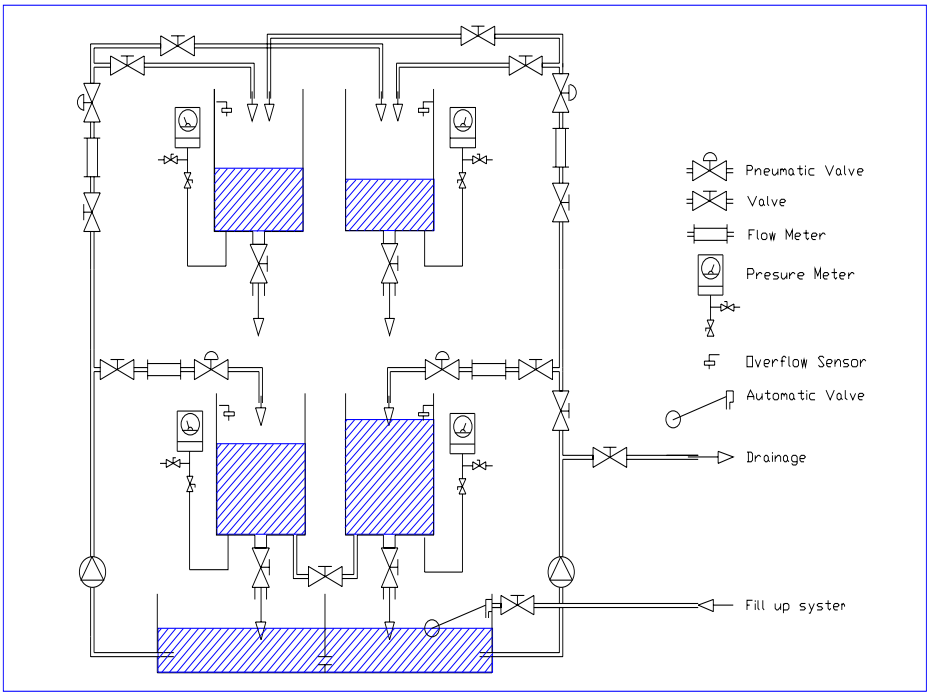

Figure 2: The real plant of the four tanks.

Cross section of the tanks$[m^2]$	Value
A_1, A_2, A_3, A_4	0.06
Cross section of the outlet hole$[m^2]$	
a_1	$8.7932 \cdot 10^{-}4$
a_2	$7.3772 \cdot 10^{-}4$
a_3	$6.3495 \cdot 10^{-}4$
a_4	$4.3567 \cdot 10^{-}4$
Flow parameters	
γ_1	0.3
γ_2	0.4
Gravity constant$[m/s^2]$	
g	9.81

Table 1: Parameter values.

2.2 Linearization

In the design of controllers, it results of interest to use simplified models, such as those obtained from linearization. This allows us to use well known techniques for the analysis and preliminary design of the controller. In this project the linearization had to me considered when modeling the stability part of the controller.

To work with a linear description the model has to be linearized around an operating point. Defining the operating point as h_i^0, the variables as $x_i = h_i - h_i^0$ and $u_j = q_j - q_j^0$ where $j = a, b$ and $i = 1, \ldots, 4$, leads to:

$$\begin{aligned} \frac{dx}{dt} &= \begin{bmatrix} \frac{-1}{T_1} & 0 & \frac{A_3}{A_1 T_3} & 0 \\ 0 & \frac{-1}{T_2} & 0 & \frac{A_4}{A_2 T_4} \\ 0 & 0 & \frac{-1}{T_3} & 0 \\ 0 & 0 & 0 & \frac{-1}{T_4} \end{bmatrix} x + \begin{bmatrix} \frac{\gamma_1}{A_1} & 0 \\ 0 & \frac{\gamma_2}{A_2} \\ 0 & \frac{(1-\gamma_2)}{A_3} \\ \frac{(1)-\gamma_1}{A_4} & 0 \end{bmatrix} u \\ y &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} x \end{aligned} \tag{2}$$

where

$$T_i = \frac{A_i}{a_i} \sqrt{\frac{2h_i^0}{g}}$$

To see a more detailed description of the linearization see [1] and [5].

3 Introduction to MPC

As mentioned in the earlier section the approach used to control the water level of the two lower tanks in the Four Tank process is based on Model Predictive Control methods. In this chapter an introduction to the applied MPC method for this project is given. For a more detailed step by step descriptions of MPC see [2].

3.1 The concept of MPC

The idea of Model Predictive Control is to choose the input signal(in this project q_a and q_b) that best corresponds to some criterium predicting how the system will behave applying this signal. The problem is converted into a mathematical programming problem at a given state. The feedback strategy is derived solving this problem at each sampling time and using only the current control action. This is called the Receding horizon technique. This process can be summarized into four steps:

1. At sample time t compute future outputs, i.e the prediction, $y_{t+i},\ i = 1, ..., N_p$ as function of future control inputs $u_{t+i},\ i = 0, ..., N_p - 1$.
2. Minimize the objective function re to $u_{t+i},\ i = 0, ..., N_p - 1$.
3. Apply u_t on the system.
4. At the next sampling time make $t = t + 1$ and go to Step 1.

3.2 The MPC elements

Observing the steps above there are three basic elements that together form the method of MPC, the prediction(step 1), the optimization problem(step 2) and the control law(step 3). How to compute and use these three elements is to be described in the following sections.

3.2.1 The prediction

The prediction is as the word itself says the prognostic of the future behavior of the system N_p sampling times ahead, where N_p is the prediction horizon chosen. From now on it will be referred to as the prediction or as $x_{\mathbf{F}}$. Computing the system dynamics recursively, $x_{\mathbf{F}}$ can easily be estimated. However the states have to measurable otherwise an Observer or a Kalman filter is needed, [2]. For the Four Tank Process considered, i.e the real plant, all states are measurable, therefore there is no need of an Observer or Kalman filter.

EX 1: Consider the linear state space representation:

$$\begin{cases} x_{t+1} = Ax_t + Bu_t \\ y_t = Cx_t \end{cases} \tag{3}$$

with a horizon N_p, the prediction would be:

$$x_{\mathbf{F}} = \begin{bmatrix} x_{t+1} \\ x_{t+2} \\ \vdots \\ x_{t+N_p} \end{bmatrix} = \begin{bmatrix} Ax_t + Bu_t \\ Ax_{t+1} + Bu_{t+1} \\ \vdots \\ Ax_{t+N_p-1} + Bu_{t+N_p-1} \end{bmatrix} = \ldots = Fx_t + Hu_{\mathbf{F}} \tag{4}$$

where $u_{\mathbf{F}} = [u_t\, u_{t+1} ... u_{t+N_p-1}]$ is the control sequence, for $x_{\mathbf{F}}$. For nonlinear system dynamics the prediction is represented by a function instead of an explicit expression.

EX 2: Consider the nonlinear representation:

$$\begin{cases} x_{t+1} = f(x_t, u_t) \\ y_t = g(x_t) \end{cases} \tag{5}$$

with prediction horizon N_p:

$$x_{\mathbf{F}} = \begin{bmatrix} x_{t+1} \\ x_{t+2} \\ \vdots \\ x_{t+N_p} \end{bmatrix} = \begin{bmatrix} f(x_t, u_t) \\ f(x_{t+1}, u_{t+1}) \\ \vdots \\ f(x_{t+N_p-1}, u_{t+N_p-1}) \end{bmatrix} = \Phi(x_t, u_{\mathbf{F}}) \tag{6}$$

where $\Phi(x_t, u_{\mathbf{F}})$ is a nonlinear function.

3.2.2 The Optimization problem

The optimization problem is the next step to solve in a MPC algorithm. The optimization problem consists of minimizing the objective function, re to the input sequence $u_{\mathbf{F}}$. This optimization can be constrained or unconstrained.

The Objective function
Controlling a system is in other words making the system evolve appropriately and follow a certain reference. This is done minimizing an objective function, which is a measure of the performance, typically penalizing the tracking error. The solution of the minimization gives the adequate input signal that makes the system output follow the reference. The reference is chosen or known a priory and here referred to as x_{ref}. Due to the system dynamics and assuming that the reference is reachable, each state reference has a corresponding input reference, u_{ref}. The standard representation of the objective function can be seen in equation (7). Here the objective function is quadratic. The objective function can have different forms, however a quadratic cost that penalizes the error is the most natural and simple way to chose it.

$$J(x_t, u_{\mathbf{F}}) = \sum_{i=1}^{N_p} L(x_{t+i}, u_{t+i}) + V_f(x_{t+N_p}) \tag{7}$$

where $L(x_{t+i}, u_{t+i}) = \| x_{t+i} - x_{ref} \|_Q^2 + \| u_{t+i} - u_{ref} \|_R^2$ and $V_f(x_{t+N_p}) = \| x_{t+N_p} - x_{ref} \|_P^2$

The minimum of function (7) is found when the states and the input signal reaches the corresponding reference, i.e. the total cost of the objective function is zero. However the system normally has different types of constraints that have to be considered then the reference may not longer be reachable making the optimization problem more complicated.

The matrices Q, R are the weighting factors. They show how the state error respectively the input error are valued. The final state cost, $V_f(x_{t+N_p})$ is added to the objective function to guarantee stability it also has an associated constraints, the terminal state constraints. The following discussion gives an example of how the weighting matrices Q and R can influence the minimization: Consider a single-input single-output system without terminal state cost $V_f(x_{t+N_p})$, let $Q = 1$ and $R = 0.0001$. This choice of weighting factors allows the difference $u_{\mathbf{F}} - u_{ref}$ to be greater then it would be if $Q = 1$ and $R = 1$, still making the cost of the objective function small.

Constraints
There are three types of constraints considered, input signal constraints, state constraints and terminal state constraints.
The input signal constraint is a lower bound/upper bound constraint for the actuator.
The state constraint is a interval constraint and the state terminal state constraint is normally a level set of the terminal cost. The terminal state constraint and the terminal cost are added to guarantee stability of the system.

- Input constraints: $u_t \in U = \{u : \ u_{min} \leq u_t \leq u_{max}\}$
- State constraints: $x_t \in X = \{x : \ x_{min} \leq x_t \leq x_{max}\}$
- Terminal state constraints: $x_{t+N_p} \in \Omega_\eta = \{x : \ V_f(x_{t+N_{N_p}}) \leq \eta\}, \forall t$

These constraints are represented mathematically as

$$\begin{aligned} &g_X(x_t) \leq 0, \ \forall t \\ &g_U(u_t) \leq 0, \ \forall t \\ &g_{\Omega_\eta}(x_{t+N_p}) \leq 0, \ \forall t \end{aligned}$$

3.2.3 The control law

The Constrained optimization problem (8) steems from the standard objective function (7) combined with the constraints mentioned in the previous section. The solution of this constrained optimization problem gives the input sequence for the next prediction horizon. However not the entire input sequence is applied at the next sample time, just the part of the solution that corresponds to the current time step is applied on the system. In other words the optimal control law for the actual sampling time is $K_{MPC}(x_t) = u_t^*$.

Constrained optimization problem:

$$\begin{aligned} min \ J(x_t, u_F) = \ & \textstyle\sum_{i=1}^{N_p} L(x_{t+i}, u_{t+i}) + V_f(x_{t+N_p}) \\ s.t. \\ & x_{t+i} = f(x_{t+i-1}, u_{t+i-1}) \\ & g_X(x_{t+i}) \leq 0, \ i = 1, \ldots, N_p \\ & g_U(u_{t+i}) \leq 0, \ i = 0, \ldots, N_p - 1 \\ & g_{\Omega_\eta}(x_{t+N_p}) \leq 0 \end{aligned} \tag{8}$$

To get a better understanding how this minimization works a simple unconstrained example is presented.

Ex3: Consider a simple non linear system

$$x_{t+1} = f(x_t) + g(x_t)u_t$$

with the prediction horizon $N_p = 1$ and the objective function

$$J(x_t, u_t) = x_t^2 + u_t^2 + (f(x_t) + g(x_t)u_t)^2$$

Then the optimal input signal is derived as

$$\frac{dJ}{du} = 2u_t + 2(f(x_t) + g(x_t)u_t)g(x_t) = 2u_t + 2f(x_t)g(x_t) + 2g(x_t)^2u_t = 0$$

$$u(2 + 2g(x_t)^2) + 2f(x_t)g(x_t) = 0$$

$$\Rightarrow \; u^* = -\frac{f(x_t)g(x_t)}{1+g(x_t)^2}$$

4 MPC design for the Four Tank Process

In this section the procedure of the control design for this project is tackled. At the end of the section the complete formulation is presented and a resume of the algorithm shown. All the basic steps are based on the MPC structure dealt with in the previous section. The algorithm is implemented in MatLab.

4.1 Discretization

The model provided for the Four Tank Process is a continuous time model and for a discrete MPC algorithm the model has to be discrete. The continuous model was discretized using an Euler forward approximation, see [11]. There are other types of discretization types, however Euler forward was chosen as it is straightforward and simple to use

Continuous time model:

$$\frac{dh_1}{dt} = -\frac{a_1}{A_1} \cdot \sqrt{2gh_1} + \frac{a_3}{A_1} \cdot \sqrt{2gh_3} + \frac{\gamma_1}{A_1} \cdot q_a$$

$$\frac{dh_2}{dt} = -\frac{a_2}{A_2} \cdot \sqrt{2gh_2} + \frac{a_4}{A_2} \cdot \sqrt{2gh_4} + \frac{\gamma_2}{A_2} \cdot q_b$$

$$\frac{dh_3}{dt} = -\frac{a_3}{A_3} \cdot \sqrt{2gh_3} + \frac{(1-\gamma_2)}{A_3} \cdot q_b$$

$$\frac{dh_3}{dt} = -\frac{a_4}{A_4} \cdot \sqrt{2gh_4} + \frac{(1-\gamma_1)}{A_4} \cdot q_a$$

Letting $\frac{dh}{dt} = \dot{x}$, $h_i = x_i$, $q_a = u_1$ and $q_b = u_2$ the continuous time model can be represented in a more condense form as:

$$\begin{array}{rcl} \dot{x} & = & f(x,u) \\ y & = & Cx \end{array} \tag{9}$$

where

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

The matrix C has this form as we just are interested in controlling the two lower tanks. Applying the forward Euler approximation on the Continuous model

$$\dot{x} = \frac{x_{t+1} - x_t}{\Delta T} \;\Rightarrow\; x_{t+1} = x_t + \Delta T f(x,u)$$

leads to the Discrete model.

Discrete model:

$$\begin{array}{l} x_{t+1} = x_t + T_s \cdot f(x_t, u_t) \\ y_t = Cx_t \end{array} \tag{10}$$

where ΔT is the integration time. However the sampling time T_s is correlated to the integration time as $T_s = m\Delta T$ typically with m=1. Here ΔT is 1 second and m is chosen to 5.

4.2 The prediction horizon

To analyze difference in system performance and precision the prediction horizon, N_p, is varied between 5 and 10 seconds.

An infinitely long prediction horizon would be the ideal choice which would give a perfect performance. As it is impossible to implement an infinitely long prediction horizon a finite one has to be considered. It is desirable, then, to use a large but finite horizon since it can be stated that the longer prediction horizon the better performance. However choosing a long prediction horizon the more variables have to be solved in the optimization and therefore the problem gets more complex and difficult to handle. A long prediction horizon also increases the domain of attraction. If the prediction horizon N_p is long then we have N_p steps to reach the desired area. Thus if N_p is long then we have more steps to reach the desired area than we would have if N_p is short, i.e we can begin further away from the desired area with a long prediction horizon.

4.3 The Optimization formulation

In this section the optimization ingredients are treated. Methods and choices of weighting functions and restrictions are presented.

4.3.1 The objective function

In the objective function the total cost of the objective is minimized. The objective function implemented has the form of the standard objective formulation, (7).

$$J(x_t, u_F) = \sum_{i=1}^{N_p} L(x_{t+i}, u_{t+i}) + V_f(x_{t+N_p})$$

First the weighting matrices Q and R are tuned until desired performance is achieved. This is a tradeoff between a smooth signal and a fast system performance. If a smooth signal is desirable then the quote $\frac{Q}{R}$ is to be low and if one wants a fast system then the quote should be greater. As only state 1 and state 2 are to be controlled the matrix Q is chosen as a diagonal matrix with nonzero values on the positions that correspond to state 1 and 2, R is a diagonal 2×2 matrix.

$$Q = \lambda_Q C^T C, \; R = \lambda_R I$$

The positive values λ_Q and λ_R are chosen to 0.0001 resp 1. The choice of terminal cost, $V_f(x_{t+N_p})$, is treated in a separate section, see section 4.4.

4.3.2 Constraints

The constraints considered, besides the model dynamics, are input signal constraints, output constraints and terminal state constraints. (The last one is treated in a separate section, see section 4.4).
The input constraints are considered due to the recommended input flow from the pump specifications. The specifications recommends as a maximum input flow 2 $\frac{m^3}{h}$. Therefore the chosen upper bound respectively lower bound is $u_{max} = 2\frac{m^3}{h} = \frac{2000}{3600}\frac{liter}{sec}$ and $u_{min} = 0\frac{liter}{sec}$. To make the problem well conditioned the input constraints have the dimension liter per second. The output state constraints are due to the height of the tanks. The maximum heights of the tanks are listed in Table 2. The minimum height is zero for all tanks.

Tank	Height$[m]$
1	1.36
2	1.37
3	1.3
4	1.3

Table 2: Maximum heights for the tanks.

4.4 Stability with terminal state conditions

To be able to guarantee stability for a nonlinear system a terminal state constraint can be added to the constraints and a terminal cost can be added to the cost function, see chapter 2 [6]. However to guarantee stability these two terms have to fulfil the following conditions. These are the conditions:

- The terminal set Ω_η is an admissible positive invariant set for the system controlled by $u_t = h(x_t)$.
- The terminal cost $V_f(x_{t+N_p})$ is a Lyapunov function such that the local control law is associated with the Lyapunov function as

  $$V(f(x_t, h(x_t))) - V(x_t) \leq -L(x_t, h(x_t))$$

  for all $x_t \in \Omega_\eta$. Which makes the nonlinear system asymptotically stable using the local control law.

To show why these formulations are needed to guarantee stability and feasibility for the closed loop system a small discussion is presented.

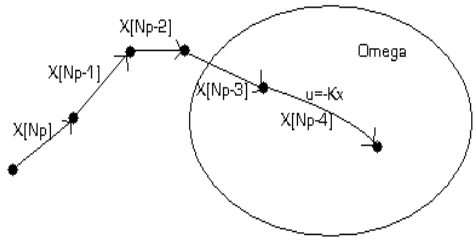

Figure 3: An illustrative example of the positive invariant set Omega.

Question 1: Why does the terminal set Ω_η have to be a positive invariant set?
Answer 1: If the terminal set Ω_η is a positive invariant set, then the set of feasible states, X_{N_p}, is the set of states that are stabilizable in N_p steps. In other words X_{N_p} is the set of all initial states that can reach the domain of attraction, Ω_η, in N_p steps. Consider that x_t is a stabilizable state, i.e. $x_t \in X_{N_p}$. Consider that there are no disturbances between the prediction and the system. Therefore the predicted state x_{t+1} can reach the terminal set Ω_η in $N_p - 1$ steps, hence $x_{t+1} \in X_{N_p-1}$. As the terminal set Ω_η is a positive invariant set it has the property that $X_{N_p-1} \subseteq X_{N_p}$. Consequently X_{N_p} is a positive invariant set for the closed loop system which guarantees feasibility for the controller at all times. Figure 3 gives an illustrative example of this invariant set.

Question 2: Why does the terminal cost have to be a Lyapunov function?
Answer 2: If the terminal cost is a Lyapunov function then the optimal cost will be strictly descending and therefore it is also a Lyapunov function, controlled by the MPC. This guarantees asymptotic stability for the closed loop system with constraints.

4.4.1 Finding the terminal set and the terminal cost

The operation explained in the following steps is the procedure that has been used in this project to obtain the terminal set Ω_η and the terminal cost $V_f(x_{t+N_p})$.

The procedure:

1. Let the linearized system $x_{t+1} = Ax_t + Bu_t$ be such that

$$A = \frac{\partial f(x,u)}{\partial x}\Big|_{(x_{ref})(u_{ref})} \; and \; B = \frac{\partial f(x,u)}{\partial u}\Big|_{(x_{ref})(u_{ref})}$$

   Calculate a auxiliary control law $u_t = -Kx_t$ using LQR, so that the linear system is asymptotically stabilizable.

2. Let $A_K = A - BK$ and let $Q^* = Q + K^T RK$. Take a matrix $\tilde{Q} > Q^*$(for example $\tilde{Q} = \lambda Q^*$ for a $\lambda > 1$) and calculate the matrix P such that

$$A_K^T P A_K - P = -\tilde{Q}$$

3. Let the terminal cost be $V_f(x_t) = (x_t - x_{ref})^T P(x_t - x_{ref})$. Calculate a constant $\eta > 0$ such that for all

$$x_t \in \Omega_\eta = \{x_t \in R^n : V_f(x_t - x_{ref}) \leq \eta\}$$

   the following conditions are fulfilled

$$\Omega_\eta \subseteq X^K = \big\{x_t \in X : u_t = (-K(x_t - x_{ref}) + u_{ref}) \in U\big\}$$
$$V(f(x_t, -Kx_t)) - V(x_t) \leq -x_t^T Q^* x_t$$

Hence the terminal state restriction, $x_t \in \Omega_\eta$ can be formulated as $(x_t - x_{ref})^T P(x_t - x_{ref}) - \eta \leq 0$ and the terminal cost $V_f(x_{t+N_p})$ as $(x_{t+N_p} - x_{ref})^T P(x_{t+N_p} - x_{ref})$.

4.5 Reference management

The reference x_{ref} is chosen a priori and calculated through out the corresponding final input reference u_{ref}. For every sampling time the objective function tries to reach a smooth temporary input target w_t. This temporary input target is the final input reference, u_{ref}, run through a filter. The filter makes the temporary reference a smooth function. In other words w_t is not as step function, but a gradually increasing function with final value u_{ref}. The filter dynamics are chosen as: $\frac{1}{\beta s+1}$, however in discrete time it takes the form as in equation (11). The parameter β determines how fast the filter reaches the constant reference. A small β gives a fast filter and a big β gives a slow filter.

$$\begin{aligned} w_{t+1} &= \beta w_t + (1-\beta)u_{ref} \\ \beta &= e^{-(T_s/50)} \end{aligned} \tag{11}$$

To get the corresponding temporal state reference, x_{w_t}, the calculated target, w_t, is applied on the static equation, i.e $x_{w_t} = f(x_{w_t}, w_t)$. Figure 4 illustrates the step function of the input reference run through the filter. It can seem unnecessary to make this operation, however

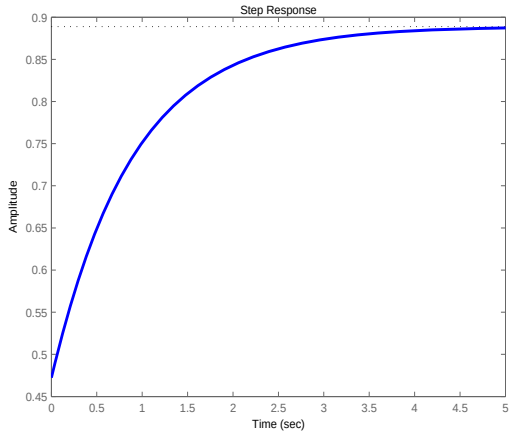

Figure 4: Step function of the input signal reference run trough the filter.

when model predictive control is used as a regulator, i.e when it steers the system to a steady state or a chosen target, abrupt changes in the derived set point may cause loss of feasibility, and hence, a erroneous operation of the controller. Therefore a reference filter has been used to avoid this. There reference filter makes a smooth change in reference values for the controller and hence reduces the possibility of unfeasible operations.

4.6 Complete algorithm

Once the model is discretized, the sampling time T_s and prediction horizon N_p chosen, all the terms in the objective function are calculated and the matrices (Q, R V_f) defined, the restrictions determined(X, U, Ω) and the reference chosen(x_{ref}) a constrained optimization problem is formulated as follows

$$
\begin{aligned}
(P_{Con}): \qquad min_{u_F} \quad &= \quad \sum_{i=1}^{N_p} L(x_{t+i}, u_{t+i}) + V_f(x_{t+N_p}) \\
s.t \quad & \\
& x_{t+i} \in X,\ i = 1, \ldots, N_p \\
& u_{t+i} \in U,\ i = 0, \ldots, N_p - 1 \\
& x_{t+N_p} \in \Omega_\eta
\end{aligned}
$$

Thus this is the optimization problem solved for every iteration in the designed MPC algorithm. The optimization solver used in MatLab is `fmincon.m`. All iterations require an initial point. For the very first initial point any initial input signal can be chosen, however for the following iterations the initial input is chosen as $u_0 = [u_{t+1},\ u_{t+2}, ..., u_{t+N_p}, -K(x_{t+N_p} - x_{w_t}) + w_t]$. This is a commonly used way to chose the initial input sequence as it makes use of known information. It uses the abundant calculated input sequence from the previous iteration and the local control law for the last part. In that way we know that the initial input, as it was calculated from the optimization problem, will be feasible and probably also quite close to the correct value. In A pseudo code illustrating the MPC algorithm is

`the first initial input, before starting the algorithm`
$u_0 = random$

for $k = 1, ...,$ `wanted number if iterations`

`calculate the temporary reference`

`calculate the next optimal input sequence`
$u =$ *min Restricted optimization problem* (12)

`calculate the state evolution for the next`
`prediction horizon using the calculated optimal input sequence`
$x_{\mathbf{F}} = \Phi(x(t), u_{\mathbf{F}})$

`define the next initial input sequence`
$u_0 = [u_{t+1},\ u_{t+2}, ..., u_{t+N_p}, -K(x_{t+N_p} - x_{w_t}) + w_t]$

end

Figure 5 illustrates the MPC algorithm in the form of a block diagram.

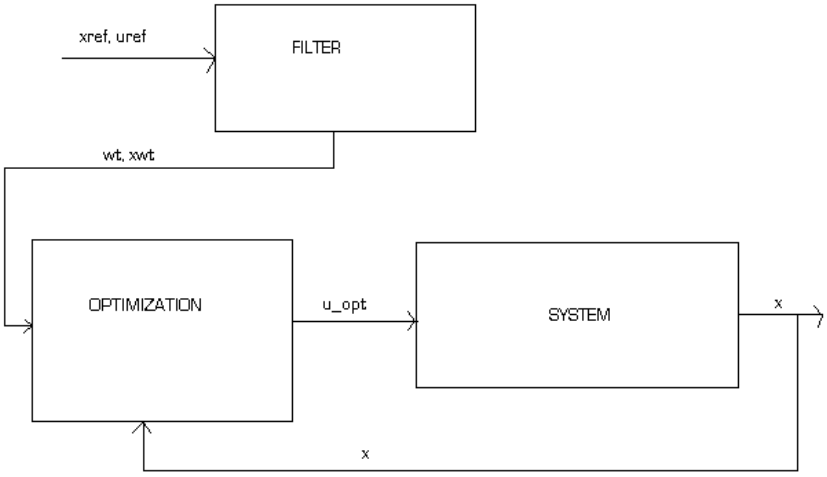

Figure 5: Block diagram of the calculation process for every iteration.

First the temporary references are calculated(FILTER). Then they are run on the model predictive controller where the optimal input sequence is found(OPTIMIZATION). The optimal input is run on the system(SYSTEM). The system responds on the input and for the next sampling time the new state values are measured.

5 Penalty functions

In this section there is first a brief description of the basics of penalty functions. Then it is shown how the penalty functions are implemented in the constrained optimization problem P_{Con} and how this changes the formulation.

5.1 Penalty function basics

Penalty functions are used in techniques to solve constrained optimization problems by means of unconstrained optimization problems, [7]. This type of method has the advantage of reducing the computational cost and complexity of the optimization problem. The idea with penalty functions is to approximate(substitute) the restricted problem, equation (13), with a problem without restrictions, equation (14).

Constrained optimization problem:

$$\begin{aligned} &min \quad f(x) \\ &s.t \\ &\qquad h_i(x) = 0,\ i = 1, \ldots, m \\ &\qquad g_j(x) \leq 0,\ j = 1, \ldots, p \end{aligned} \tag{13}$$

Unconstrained optimization problem:

$$min \quad f_p = f(x) + cP(x) \tag{14}$$

where c is a positive number $c > 0$ and $P(x)$ is the penalty function.

This function should be in such a way that the approximation is obtained by the penalty function. If the restrictions are violated a high cost on the objective function is added by the penalty function. The severity of the penalty is decided by the parameter c. In other words the parameter c characterizes the degree of approximation that the unrestricted penalty problem approaches the original restricted one. The more $c \to \infty$ the more the penalized unrestricted problem approximates the original restricted problem. However there are a type of penalty function called exact penalty functions and as the name tells this type of penalty function give an exact representation of the original constrained problem, see [7]. These exact penalty functions can be chosen as follows

$$P(x) = \sum_{i=1}^{m} |h_i(x)| + \sum_{j=1}^{p} max[0, g_j(x)] \tag{15}$$

To make sure that the approximation is exact the parameter c has to be greater that the absolute value of the largest Lagrangian multiplier associated to the constraints, that is $c > |\lambda_{max}|$, [7].

5.2 Implementation of penalty functions

The penalty functions applied in this project are the exact penalty functions. They are applied on P_{Con} (13), formulated in section 4. The constrained optimization problem, P_{Con} has the form:

$$\begin{aligned} min \quad & J(x_t, u_F) = \textstyle\sum_{i=1}^{N_2} L(x_{t+i}, u_{t+i}) + V_N(x_{t+N}) \\ s.t \quad & \\ & g_X(x_{t+i}) \leq 0,\ i = 1, \ldots, N_p \\ & g_U(u_{t+j}) \leq 0,\ j = 0, \ldots, N_p - 1 \\ & g_\Omega(x_{t+N_p}) \leq 0 \end{aligned} \tag{16}$$

Two types of approximations have been done. In the first one, referred to as the X-penalty problem (17), only the restrictions on the states(x_t) have been approximated with penalty functions. The second approximation is completely without information, referred to as the XU-penalty problem (18). In other words, in the XU-penalty problem, the restrictions on the states x_t and the restrictions on the input signal u_t have been approximated with exact penalty functions. The X-penalty problem is an example of how a search algorithm with input information works. When using the X-penalty formulation information is given about where the solution can be found. That is the input signal restrictions are not really restrictions, they are actually just information that reduces the set of possible solutions. The XU-penalty formulation, on the other hand is completely without any information. In this case the solution is found following the steepest descent. For the X-penalty problem the minimization is done with the Matlab function `fmincon` and for the XU-penalty problem `fminunc`.

X-penalty problem:

$$\begin{aligned} min \quad & J(x_t, u_F) - cP_X(x) - cP_\Omega(x) \\ s.t \quad & \\ & g_U(u) \leq 0,\ j = 0, \ldots, N_p - 1 \end{aligned} \tag{17}$$

where $P_X(x) = \sum_{i=1}^{N_p} max[0, g_X(x_{t+i})]$ and $P_N(x) = max[0, g_\Omega(x_{t+N_p})]$.

XU-penalty problem:

$$min\ J(x_t, u_F) - cP_X(x) - cP_\Omega(x) - cP_U(u) \tag{18}$$

where $P_U(u) = \sum_{j=0}^{N_p-1} max[0, g_U(u_{t+j})]$ and $P_X(x)$ and $P_\Omega(x)$ as for the X-penalty problem.

As the parameter c is unknown and difficult to calculate, iterations can been done until the solution is feasible. That is, to make sure that the chosen parameter c is really greater than the largest Lagrangian multiplier, λ_{max}, the problem has been solved until an exact match is found. Here the parameter c is chosen to 10000.
Both implementations have been compared with the original formulation regarding computational time, precision and performance. The results are seen in the section Results.

6 Initial feasible input

In this section it is shown how a initial feasible input is found. When penalty functions are implemented the optimization problem requires an initial feasible input, something that on the other hand is not necessary when using optimization with constraints. A feasible solution is a solution that fulfills all the constraints in a constrained optimization problem.

6.1 The PHASE-I problem

When penalty functions are used an initial feasible solution is needed. As the penalty function penalizes all solutions that are unfeasible it is better to start the optimization with a feasible solution guaranteeing a final feasible solution.

There is a method for finding a initial feasible solution. This method is based on an initial optimization problem called the PHASE-I problem, [8]. The PHASE-I problem is a optimization problem which solution gives the initial feasible input. The PHASE-I problem is described in the following lines.

Consider a set of inequalities and equalities

$$f_i(x) \leq 0,\ i = 1, \ldots, m \tag{19}$$

were f_i is a function with continuous second derivatives. Assume that the feasible set

$$\{(x, s) | f_i(x) \leq 0,\ i = 1, \ldots, m\} \tag{20}$$

is bounded and that we are given a point $x^{(0)} \in D$. To find a strictly feasible solution for the restrictions above, the following problem is solved

$$\begin{array}{ll} min & s \\ s.t & \\ & f_i(x) \leq s,\ i = 1, \ldots, m \end{array} \tag{21}$$

in the variables x and s. This optimization problem is called the PHASE-I problem and is always strictly feasible. We can choose $x = x^{(0)}$ and $s > max_{i=1,\ldots,m} f_i(x^{(0)})$. This problem can be solved by means of penalty function based methods reconstructing the PHASE I problem into an unconstrained optimization problem. When reconstructing the PHASE I problem into a unconstrained one, penalty functions from the previous section have been used. However not exact penalty functions, that is the parameter c does not longer have to be greater than the largest Lagrangian multiplier. It is now chosen to the first value that give a feasible solution. The solution of the penalized problem is feasible if $s < 0$, if $s > 0$ no feasible solution exists. For a detailed description of the PHASE-I method see [8].

7 Results

In this section all the results are presented. Simulations and results have been registered for two different initial state values x_{0A}, which is inside if the terminal set Ω_η and x_{0B}, which is outside. All simulations have the same reference value.

7.1 Simulation results

In Figure 6 and 7 the state evolution and the input signal are illustrated for the constrained optimization. Here the initial state is x_{0A} which is close to the reference value. This implies no active constraints.

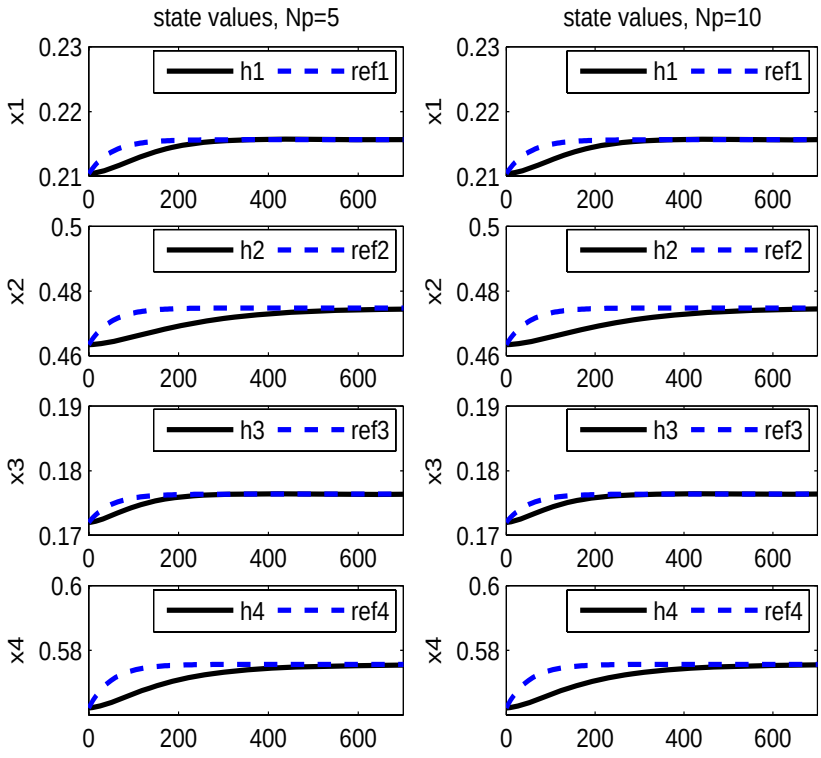

Figure 6: Constrained optimization when x_{0A}. Column 1: Output values for $N_p = 5$. Column 2: Output values for $N_p = 10$.

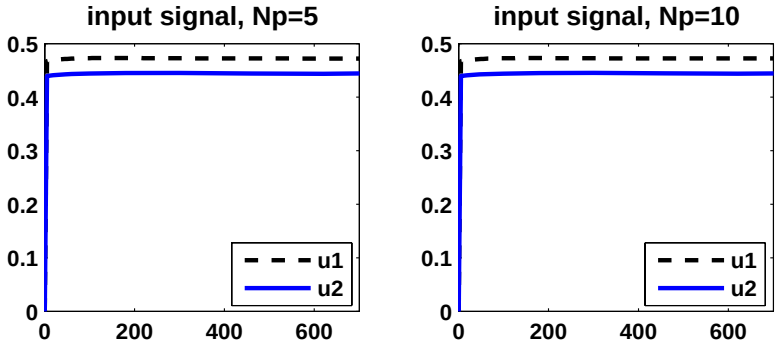

Figure 7: Constrained optimization when x_{0A}. Column 1: Input signal for $N_p = 5$. Column 2: Input signal for $N_p = 10$.

In Figure 8 and Figure 9 on the other hand the initial state value is x_{0B}, which is further away from the reference, we can see that the input signal has to rise slightly more until reaching a steady state. For P_{con}, (12), we can see that there is no need of having a larger prediction horizon, neither when the initial state is x_{0A} nor x_{0B}.

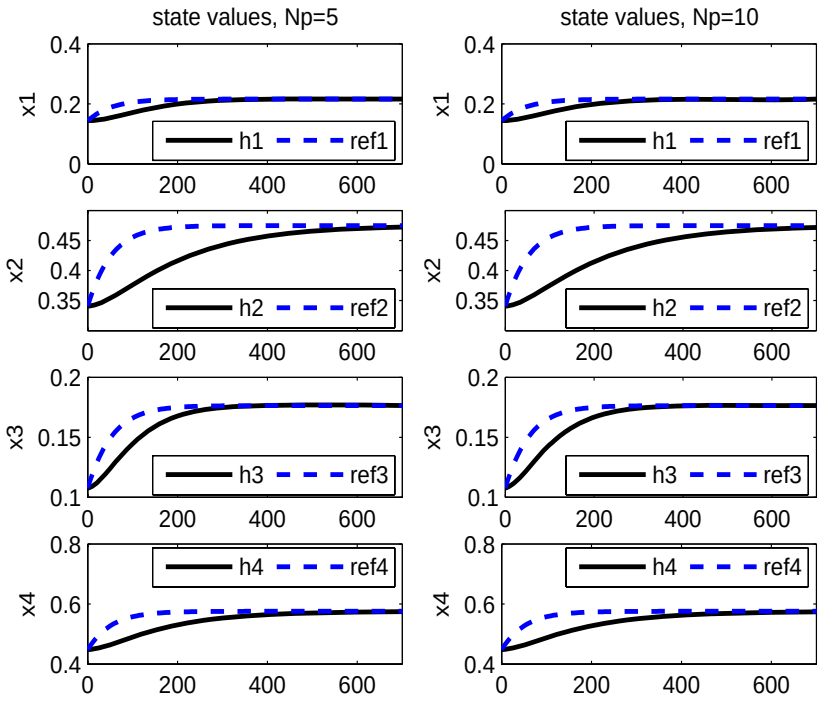

Figure 8: Constrained optimization when x_{0B}. Column 1: Output values for $N_p = 5$. Column 2: Output values for $N_p = 10$.

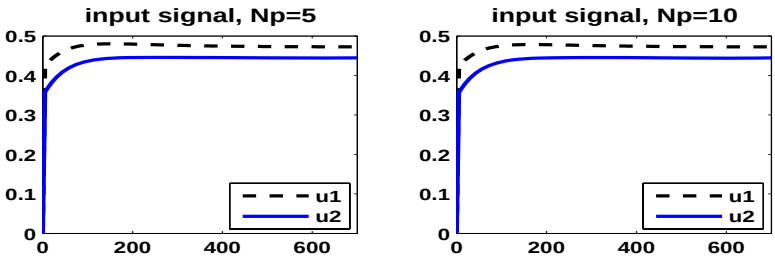

Figure 9: Constrained optimization when x_{0B}. Column 1: Input signal for $N_p = 5$. Column 2: Input signal for $N_p = 10$.

When the initial input is x_{0A} we can see that that X-penalty problem, Figure 10 and Figure 11, acts perfectly, with a smooth input signal and states reaching corresponding reference. As exact penalty functions are chosen the response should be the same as for P_{Con}, which they are. Compare Figure 11 with Figure 7.

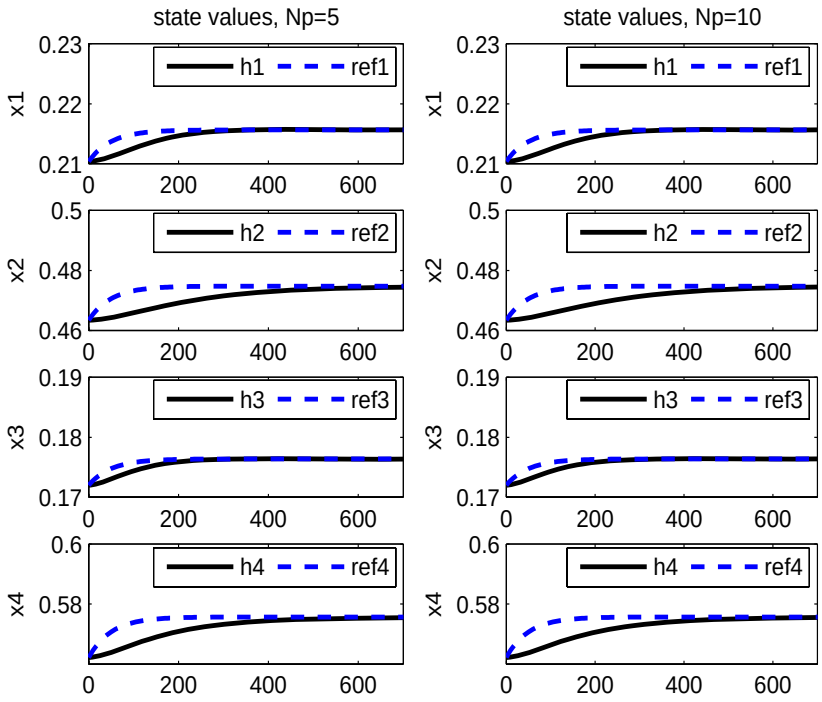

Figure 10: X-penalty when x_{0A}. Column 1: Output values for $N_p = 5$. Column 2: Output values for $N_p = 10$.

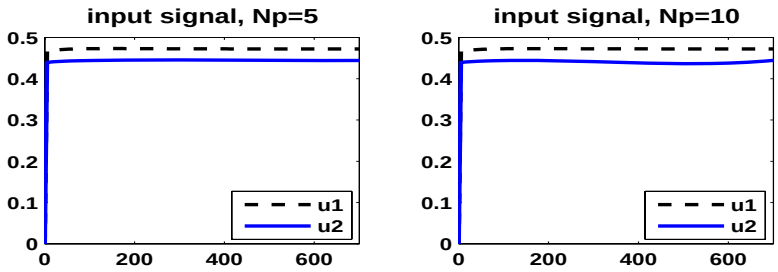

Figure 11: X-penalty when x_{0A}. Column 1: Input signal for $N_p = 5$. Column 2: Input signal for $N_p = 10$.

In Figure 12 and Figure 13 the response of the X-penalty problem with initial state value x_{0B} is illustrated. We can see that the final reference is reached without any violations on the restrictions. The same conclusion is drawn regarding the exact penalty function as for the case with x_{0A}, compare Figure 13 with Figure 9.

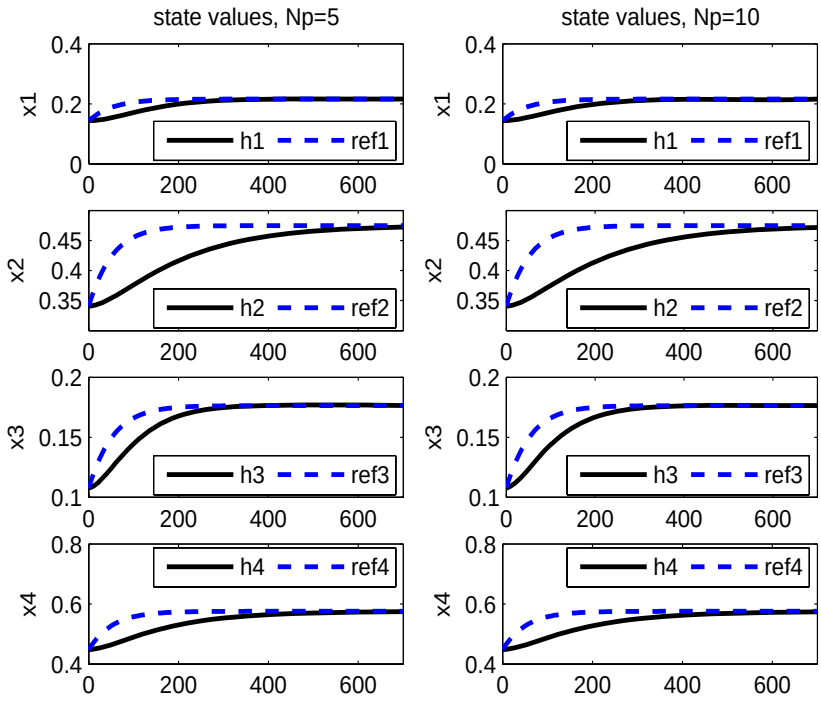

Figure 12: X-penalty when x_{0B}. Column 1: Output values for $N_p = 5$. Column 2: Output values for $N_p = 10$.

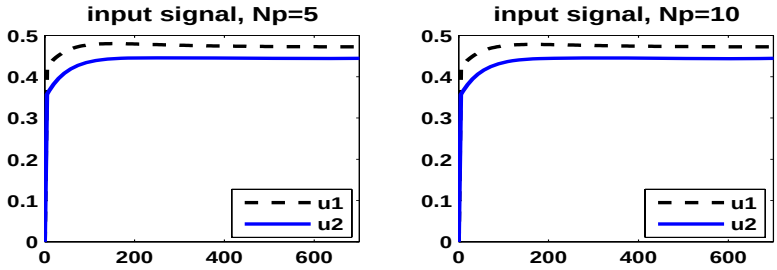

Figure 13: X-penalty when x_{0B}. Column 1: Input signal for $N_p = 5$. Column 2: Input signal for $N_p = 10$.

Figure 14 and Figure 15 illustrates the state evolution and the input signal for the XU-penalty problem when the initial state is x_{0A}. We can see that the system performance is completely satisfying.

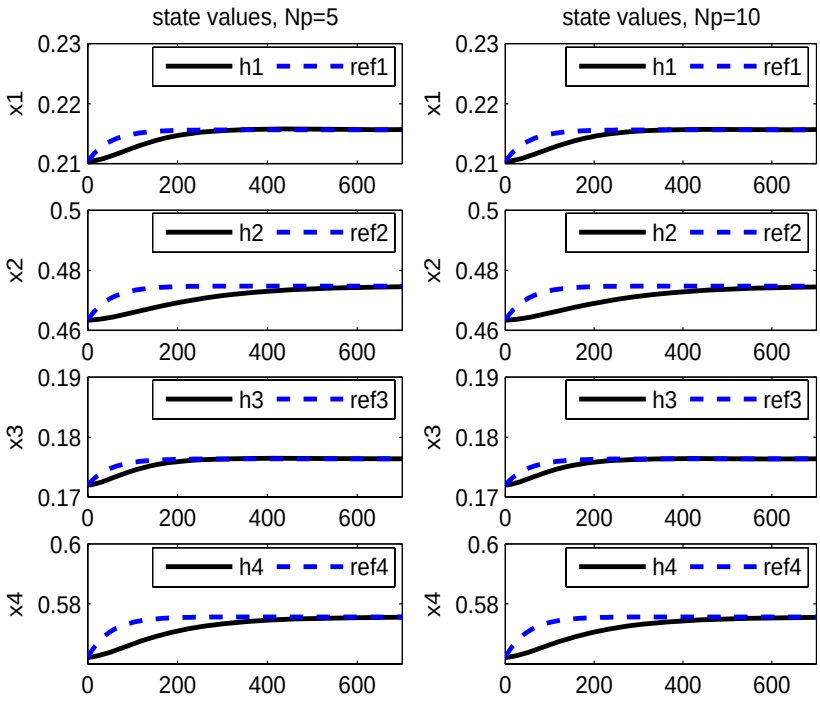

Figure 14: XU-penalty when x_{0A}. Column 1: Output values for $N_p = 5$. Column 2: Output values for $N_p = 10$.

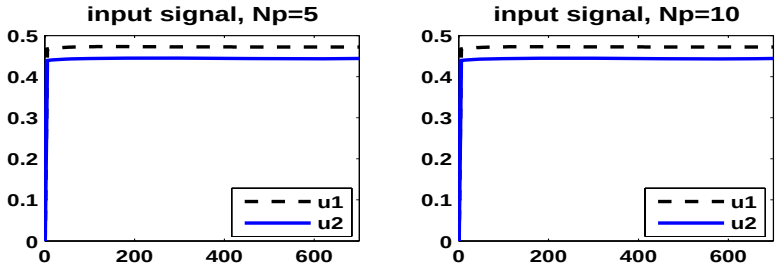

Figure 15: XU-penalty when x_{0A}. Column 1: Input signal for $N_p = 5$. Column 2: Input signal for $N_p = 10$.

In Figure 16 and Figure 17 the initial state is x_{0B}. It shows that all references are reached with the same input signal as for the constrained problem.

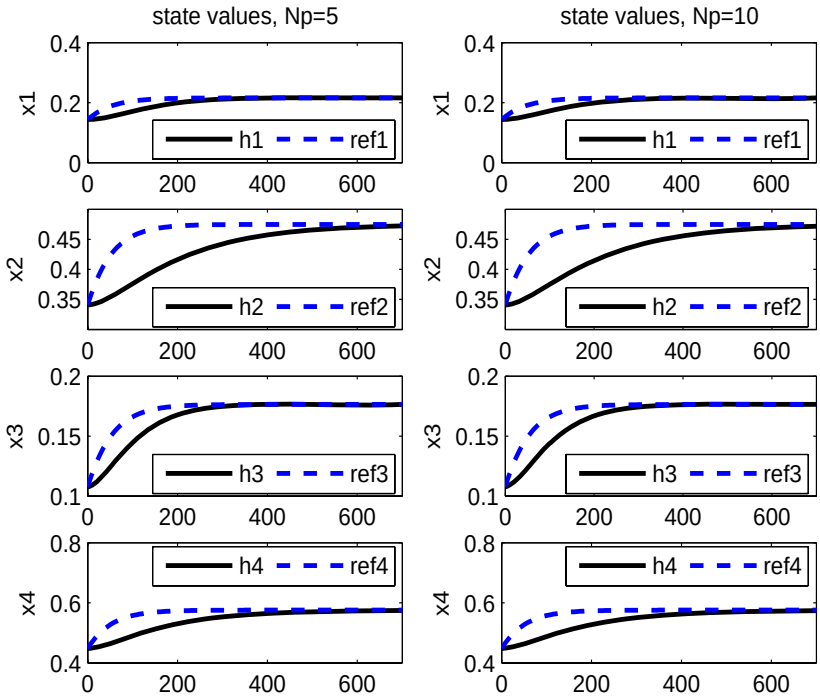

Figure 16: XU-penalty when x_{0B}. Column 1: Output values for $N_p = 5$. Column 2: Output values for $N_p = 10$.

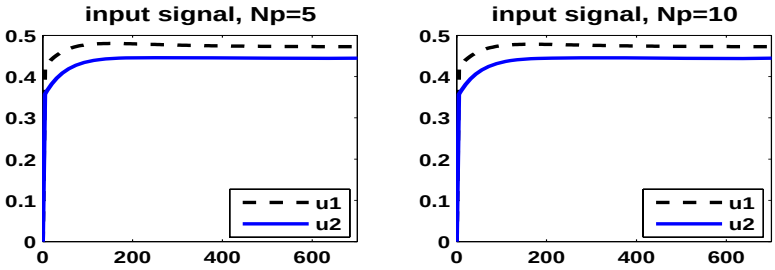

Figure 17: XU-penalty when x_{0B}. Column 1: Input signal for $N_p = 5$. Column 2: Input signal for $N_p = 10$.

All simulations from Figure 6 to Figure 17 show smooth behavior when the weighting quote $\frac{\lambda_Q}{\lambda_R}$ is 0.0001. There are no violation of the restrictions and all formulation show the same behavior. However when letting the quote increase we can see that the restrictions are indeed active. However the constrained optimization gets more difficult to solve and the "defalut" number of function evaluations are not sufficiently high to find a feasible solution. The number of function evaluations and iterations have to be increased for a feasible solution to be found when solving the constrained optimization. However when increasing the evaluation number the computational time is unacceptably high. Therefore a unfeasible solution is shown, see Figure 7.1. For the X-penalty and the XU-penalty formulations, see Figure 18 respectively Figure 19, a feasible solution is found without any violations. We can see that the results are not as "exact" as when the quote was 0.0001, however pretty good. For all cases we can for example see that state three has a small bump before reaching the reference value. There have not been done any comparatione regarding computational time for this quote as the number of iterations needed for a feasible solution vary for the Matlab functions used for the different formulations.

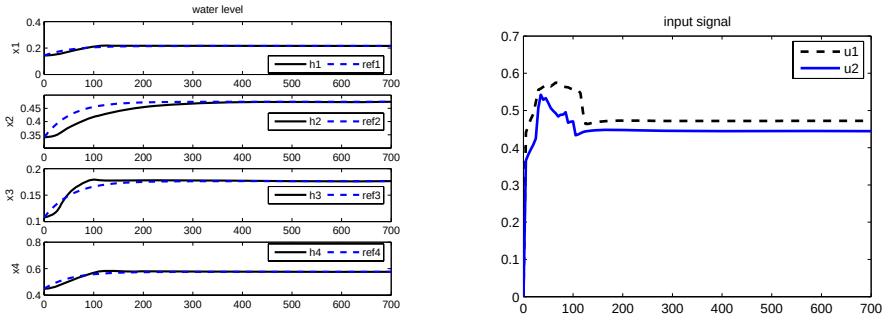

Figure 18: Sate values and input signal for the constrained optimization with initial state x_{0B} and quote 0.01.

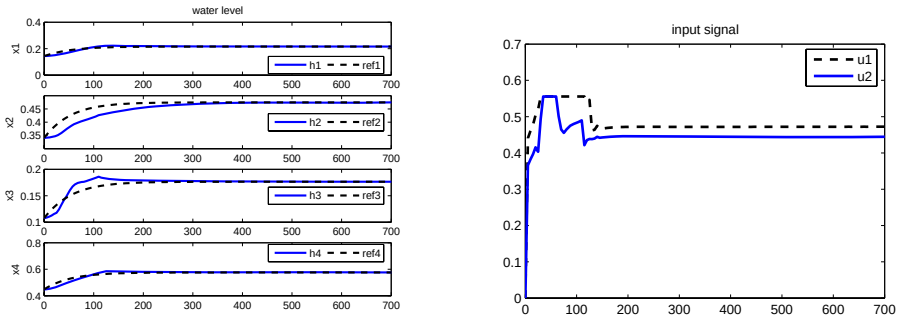

Figure 19: Sate values and input signal for the X-penalty optimization with initial state x_{0B} and quote 0.01.

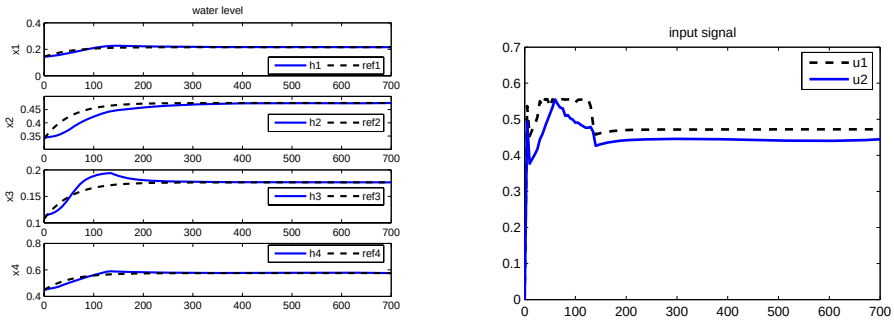

Figure 20: Sate values and input signal for the XU-penalty optimization with initial state x_{0B} and quote 0.01.

7.2 Numerical results

Numerical results for the total computational time and precision are shown in the following tables and figures.

The computational time is the total time it takes the control algorithm to compute the control action. For a fair comparison between the total computational times for the three different formulations the time for the PHASE I- problem is added when calculating the total computational time for the X-penalty and XU-penalty problem. This addition is done because the search algorithm in Matlab for the constrained optimization, `fmincon`, already has a "PHASE I-problem" built in that is included in the total computational time for that problem. The computational time has been measured with the MatLab functions `tic.m` and `toc.m`, as these functions vary the computational time from execution to execution a mean value has been calculated.

Regarding precision two comparisons are done. The first is between the total cost of the objective functions and the second between the last calculated input sequence. For the first one the absolute value is taken between the relative total cost of the constrained formulation and each one of the penalty formulations. The result of the constrained formulation is considered as the "correct" as its solution is expected to be a global optimum. The error between the "correct" value and the total cost of the penalty functions is denoted as $\frac{\Delta J}{J}$ and calculated,when XU-penalty for example, as $|(J_{Con} - J_{XU})/J_{Con}|$. A relative error for the input sequence, $\frac{\Delta u}{u}$ is also calculated in the same way. To make a more accurate investigation of the precision as well as of the computational time measures have been done not only for $N_p = 5$ and $N_p = 10$ as for the simulations, but also for $N_p = 1$ and $N_p = 3$.

Table 3 show the computational time for every formulation and prediction horizon. The results are illustrated in Figure 21 and Figure 22. In Figure 21 we can see that when the initial state value is x_{0A}, the best computational time is obtained with the X-penalty formulation, with an exception at $N_p = 1$. This result shows that theory and practice go hand in hand. As the X-penalty formulation is without restriction, but with information about where the solution will be found.
In Figure 22, when the initial state is outside of Ω_η, on the other hand, the best result is

shown for the XU-penalty formulation, with an exception at $N_p = 10$. Here we can see that even though the initial state is outside of the terminal set, both the penalty formulations and the constrained formulations give satisfactory results. Both Figures show that the XU-penalty formulation give best computational when $N_p = 1$ and the X-penalty when $N_p = 10$.

Computational time T_c

x_{0A}	$N_p = 1$	$N_p = 3$	$N_p = 5$	$N_p = 10$
Constrained	3.6	7.0	10.7	20.0
X-penalty	4.0	6.2	8.7	18.0
XU-penalty	2.8	6.9	9.7	21.4
x_{0B}	$N_p = 1$	$N_p = 3$	$N_p = 5$	$N_p = 10$
Constrained	3.9	8.6	12.7	20.0
X-penalty	4.0	7.4	11.3	17.0
XU-penalty	2.8	7.0	10.2	20.1

Table 3: Computational time T_c for the different formulations and initial state values.

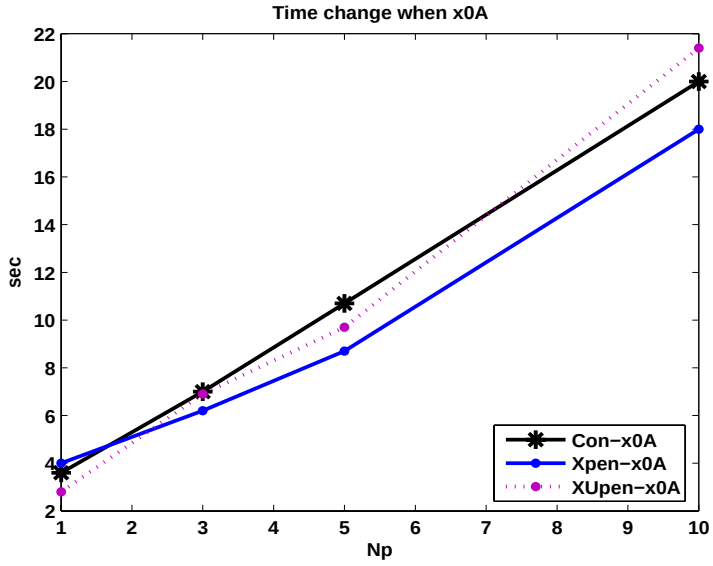

Figure 21: Time change for the different formulations when x_{0A}.

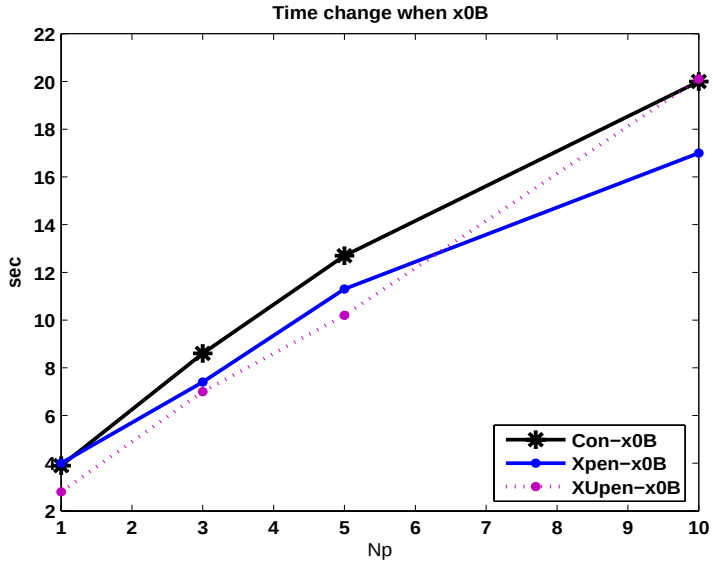

Figure 22: Time change for the different formulations when x_{0B}.

Regarding the precision Table 4 show good precision for the X-penalty formulation. Both when the initial state is inside as well as when outside of Ω, however slightly better for initial states outside of Ω.
For the XU-penalty formulation the error is not as good as for the X-penalty. It is approximately four times larger showing an error between 0.2% and 79%. However looking at the simulation results, Figure 14 and Figure 16, the states reaches their corresponding reference perfectly. The relative input error in Table 5 confirms that even though the error in the cost function is big the solution of the XU-penalty problem is close to the "exact" one. In Table 5 we can for example see, that the relative input error between the constrained formulation and the XU-penalty is indeed very small. The biggest error, approximately 0.1%, is found when $N_p = 10$ and the initial state value is x_{0B}. Hence precision is still very good for the XU-penalty formulation.

Relative error $\frac{\Delta J}{J}$

x_{0A}	$N_p = 1$	$N_p = 3$	$N_p = 5$	$N_p = 10$
X-penalty	$25.6241 * 10^{-4}$	$1.2181 * 10^{-4}$	$2.3980 * 10^{-4}$	$3.2947 * 10^{-4}$
XU-penalty	$3384.7924 * 10^{-4}$	$7932.5124 * 10^{-4}$	$7948.4660 * 10^{-4}$	$1487.1764 * 10^{-4}$
x_{0B}	$N_p = 1$	$N_p = 3$	$N_p = 5$	$N_p = 10$
X-penalty	$0.21116 * 10^{-4}$	$0.32413 * 10^{-4}$	$0.68197 * 10^{-4}$	$0.40421 * 10^{-4}$
XU-penalty	$5083.1656 * 10^{-4}$	$213.0555 * 10^{-4}$	$126.8942 * 10^{-4}$	$1319.9760 * 10^{-4}$

Table 4: Relative error $\frac{\Delta J}{J}$ between the constrained formulation and the penalty formulations.

Relative error $\frac{\Delta u}{u}$

x_{0A}	$N_p = 1$	$N_p = 3$	$N_p = 5$	$N_p = 10$
X-penalty	$5.76 * 10^{-8}$	$0.53 * 10^{-8}$	$1.22 * 10^{-8}$	$0.86 * 10^{-8}$
XU-penalty	$0.45 * 10^{-4}$	$1.19 * 10^{-4}$	$1.33 * 10^{-4}$	$0.920 * 10^{-4}$
x_{0B}	$N_p = 1$	$N_p = 3$	$N_p = 5$	$N_p = 10$
X-penalty	$0.64 * 10^{-8}$	$0.61 * 10^{-8}$	$1.35 * 10^{-8}$	$0.98 * 10^{-8}$
XU-penalty	$5.15 * 10^{-4}$	$0.93 * 10^{-4}$	$0.66 * 10^{-4}$	$10.38 * 10^{-4}$

Table 5: Relative error $\frac{\Delta u}{u}$ between the constrained formulation and the penalty formulations.

8 Conclusion

8.1 Conclusion

In this project a nonlinear model predictive control algorithm is developed for the Four Tank Process. Simulations show that parameters are adequately calculated and chosen. The control algorithm leads the system to the desired reference without any violations on the restrictions, both when the initial state is inside the terminal set as when it is outside. Regarding the prediction horizon, simulations shows that there is no need of a too long prediction horizon. Feasibility and stability is assured even for short horizons guaranteeing that the output reaches the reference. Therefore in this project short horizons are to prefer as the computational time is less.

Apart from obtaining a control algorithm that works well, another objective is to try to find a way to reduce the computational time and complexity of the optimization problem without loosing performance and precision. In this project the way that has been chosen to attack this task is to implement penalty functions converting the constrained optimization problem to an unconstrained optimization problem including a PHASE-I method. Simulation shows that the algorithm with implemented penalty functions gives a satisfactory result, both when all restrictions are substituted with penalty functions as well as when information about the input signal is given. It is shown that good precision is achieved even for a initial state value inside of the terminal set as for a initial state value outside of the terminal set, especially for the X-penalty formulation, i.e when input signal information is added. The computational time is reduced, even though just slightly, when using both penalty formulations. However the XU-penalty formulation show best computational time when $N_p = 1$ and the X-penalty when $N_p = 10$.

8.2 Future work

There are some interesting work that still remains to be done on this project. In specific it would be interesting to test the controller on the real plant and do further investigations about how to reduce the computational time.

Testing the controller: Regarding to test the controller on the real plant this task was not achieved due to lack of time and equipment that was out of function. If testing the controller on the real plant surely some disturbances have to be added to the system model.

Reduction of computational time: It would be interesting to find other types of penalty functions, as barrier function for example, that reduce the computational time while maintaining good performance and precision. Also other ways of reducing the computational time are of great interest. A method worth considering would be to stop the search algorithm for the optimization when the total current cost of the objective function is less than the initial one. This approach as well as the penalty functions does not find the global optimum, its solution is just suboptimal, however the question is if the goal is to always find a global optimum or if the goal is to find a solution that just is better than the previous one.

References

[1] K-H.Johansson: *The Quadruple-Tank Process: A Multivariable Process with an Adjustable Zero*, IEEE Transaction on control systems technology, 2000.

[2] E.F.Camacho: *Model Predictive Control*, Springer-Verlag, London Limited 2004

[3] T.Glad and L.Ljung: *Reglerteori-Flervariabla och olinjära metoder*, Studentlitteratur, Lund 2003

[4] I.Alvarado,D.Limon,W.Garcia-Gabin,T.Alamo and E.F.Camacho: *An Educational Plant Based on the Quadruple Tank Process*

[5] I.Alvarado: *Memoria de periodo de investigación*, Dpto deIngenería de sistemas y automática, Escuela Superior de Ingenieros, Sevilla 2004

[6] D.Limón Marruedo: *Control Predictivo de sistemas no lineales con restricciones: estabilidad y robustez*, Phd Thesis, Universidad de Sevilla, 2002

[7] David E.Luenberger: *Programación lineal y no lineal*, Addison-Wesley Iberoamerica, S.A, 1989 Wilmington, Delaware, USA.

[8] S.Boyd and L.Vandenberghe: *Convex Optimization*, Preface to 12/01 version, 2001, www.stanford.edu/class/ee364 and www.ee.ucla.edu/ee236b

[9] L.Magni,G.De Nicolao, L.Magnani, and R.Scattolini: *A Stabilizing Model Based Predictive Control Algorithm for Nonlinear Systems*, Automatica, 37:1351-1362,2001

[10] MatLab The Language of Technical Computing, Mathworks Inc, *www.mathworks.com/products/matlab/*, 2006

[11] L.Råde and B.Westergren, *BETA, Mathematics Handbook for Science and Engineering*, Fourth edition, Studentlitteratur

[12] Wikipedia the free Encyclopedia, *http://www.wikipedia.org/*.