DEVELOPMENT AND IMPLEMENTATION OF REAL-TIME MODEL PREDICTIVE CONTROL FOR A THREE-TANK CONTROL PROBLEM USING A DSPACE MULTI-PROCESSOR SYSTEM

D. Büchner, P. Skworcow

Water Software Systems, De Montfort University, The Gateway, Leicester LE1 9BH, U.K.

Keywords: Realt-Time Model Predictive Control, Three-Tank Control Problem, parallel computing

Abstract

This paper considers a three-tank control problem that is controlled using a Model Predictive Control (MPC) approach. A MPC algorithm was developed and implemented on a quad-core controller board. The ability to simultaneously process four tasks was used to parallelize the MPC algorithm. The paper investigates the utilisation of the processors and the control behaviour on the real plant using the parallel MPC algorithm.

1 Introduction

Since 2011 Water Software Systems research group has been constructing the Water Sustainability Laboratory (WSL). The WSL is equipped with devices that allow the physico-chemical examination of wastewater as well as the simulation of and experiments on water treatment processes. In this paper a three-tank level control problem is considered in order to investigate the capabilities and to determine the restrictions of the new equipment such as actuators, sensors and in particular the control system.
Model Predictive Control (MPC) was chosen as a modern control approach, which is commonly used in the process control industry and hundreds of other applications, see [1] and [2] and references therein. Furthermore, MPC is capable of utilizing the computing power of the state of the art control system hardware.
MPC is not a specific control strategy but rather an ample range of control methods developed around common ideas [3]. The three main features appearing in all the predictive control families are: (i) the explicit use of a model to predict the process response at future time instants (prediction horizon), (ii) the calculation of a control sequence by minimising a certain cost function and (iii) the use of a receding horizon strategy i.e. at each instant the horizon is shifted toward the future, which involves the application of only the first control signal from the sequence calculated at each step.
MPC is a more advanced controlling technique than the PID-Controller, since its characteristics overcome PID in many aspects. It takes limitations of the actuators and if required of the model states into account and it is easy to implement even for multivariable control problems - to mention only the main advantages of MPC [4]. However, one has to pay these advantages with the periodic calculation of a computationally demanding optimisation algorithm. Another disadvantage is that a model is required that satisfactorily represents the behaviour of the real system. Depending on the system this model can not always be obtained easily. This significant computational burden is one of the main reasons why MPC has not replaced the traditional PID-Controller.

The main objectives of this paper are:

- The development and implementation of a real-time MPC algorithm for a three-tank control problem.
- Parallelization of the control algorithm and implementation on a multi core system.
- Testing the behaviour of the MPC and analysing the utilisation of the controller on the physical plant.

Not only the on-line optimisation in MPC is time consuming but it is also showing significant variation in the execution time. The authors of [5] investigate and compare different scheduling methods that can be applied with the MPC approach in order to optimally utilize a processor. In this project however we worked with the common static task scheduling method as introduced in [6], since it is easy to implement and analyze.
The remainder of the paper is organised as follows: Section 2 describes the set up of the plant and the problems that had to be solved. In section 3 the material and the methods that had been used are presented. Section 4 describes the development steps that had been taken. Experiments are discussed in section 5 and conclusions are given in section 6.

2 Problem formulation

The model to be controlled is a system of three tanks that are set up in configuration similar to other works ([7], [8]). Figure 1 shows how the tanks were connected. Tanks one and three are buffer-tanks and have a capacity of 150l. Tank two is a sequencing batch reactor with a volume of 190l. It is also elevated by 20cm from the ground and has aerators attached to the bottom. The water level of the tanks is measured via ultrasound sensors mounted on top of each tank. The two peristaltic pumps have a bidirectional throughput (q_{c1}, q_{c2}) of maximal 3.3 l/min. The system dynamics are slow i.e. it takes a long

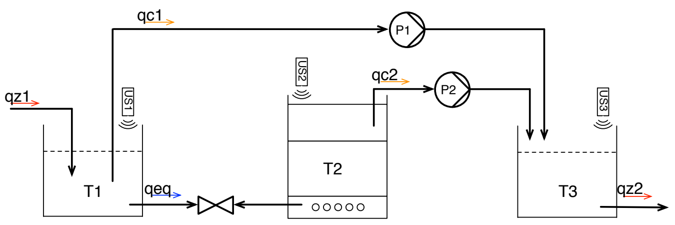

Fig. 1: The three tanks T1, T2, T3 were assembled as shown to form the experimental setup used in this paper. Mounted on top of each was a ultrasound sensor to measure the level of the water. The in- and outlet flow q_z1 and q_z2 are considered as disturbance. The water level in the tanks can be controlled by q_c1 and q_c2 through the pumps P1 and P2. q_eq describes the gravity driven equalization flow between tank 1 and tank 2.

time for control changes to take effect. The manual valves for the equalisation flow q_{eq} between tank 1 and 2 as well as the outlet valve q_{z2} of tank 3 have been set to a fixed position and were not changed. The valve opening was chosen such that the flow through the valve was always smaller than the pump throughput in order to ensure the plants controllability. The inlet flow q_{z1} comes directly from a connected water tap. The maximum Inflow is more than two times greater than the pump throughput.
To develop a MPC for this plant a model had to be obtained from the physical model, such that it can be described e.g. in the state space form

$$\dot{x} = Ax + Bu \tag{1}$$

$$y = Cx + Du. \tag{2}$$

After that, the cost function

$$\min_x \frac{1}{2} x^T Hx + G^T x \tag{3}$$

had to be developed and minimized using the model description. Furthermore, constraints for this cost function had to be defined such that

$$\Omega x \leq \omega. \tag{4}$$

Taking plant and control system, the cost function and its constraints as a given starting point, the development progressed in the following steps:

1. Developing a mathematical model description for the plant.
2. Implementing a MPC-Controller using the controller provided by the Simulink Model Predictive Control Toolbox.
3. Developing a MPC algorithm and splitting it into several tasks.
4. Implementing the algorithm in the control system.

3 Methods and Materials

In our disposal are a number of new sensors, actuators and reactors, that allow the simulation of different water treatment processes. The processes can be controlled via a high-end control system from dSPACE Ltd. The following hardware was used:

- Reactors
- Ultrasound Sensors
- Marlow 520U peristaltic pumps
- DS2003 MUX ADC Board
- DS2103 DAC Board
- DS1006 Processor Board, x86 Quad-Core AMD Opteron^{TM} processor 2.8 GHz
- PC as control and engineering station

The DA/AD boards send and receive signals in the range of maximal $\pm 10V$. The dSPACE system also comes with software that allows the programming and controlling of the platform. The four Real-Time Processor (RTP) on the processor board can be programmed graphically via MATLAB Simulink. dSPACE provides an additional so called Real-Time Interface (RTI) library to Simulink that allows the connection to in-/output boards and enables the compilation of models to the control board. Almost all standard Blocks are supported by the dSPACE system and some additional libraries like the MPC-Toolbox can be compiled to the controller. To observe and control the plant a software named ControlDesk is provided by dSPACE. It allows the development of GUIs as well as the capturing of data. For this paper the following softwares were used:

- MATLAB R2009b
- Simulink
- dSPACE Real-Time Interface Toolbox (RTI1006)
- Model Predictive Control Toolbox
- ControlDesk NG^{TM} Version 4

To distribute the calculation to multiple cores, the code was partially rewritten in C following the implementation in [9] and [10]. The implementation in Simulink was made using a C-Code S-Function. In order to minimize the cost function and to find the optimal control value for the linear model, the Dantzig Quadratic Programming (QP) algorithm by N. L. Ricker and A. Bemporad was used. It is the same algorithm that is used by the MPC-Toolbox in Simulink. For simplicity however the main part of the matrix operations that lead to the optimization are implemented using Embedded Matlab Functions.

4 Development

4.1 Modeling

To obtain the mathematical description of the plant, figure 1 was used to write down the differential equations of the system.

$$\dot{h_1} = (-q_{eq} - q_{c1} + q_z 1)/A_1 \tag{5}$$

$$\dot{h_2} = (q_{eq} - q_{c2})/A_2 \tag{6}$$

$$\dot{h_3} = (q_{c1} + q_{c2} - q_{z2})/A_1 \tag{7}$$

As a first approach the model was obtained by measuring parameters of the system directly. The throughput of the pumps as well as those of the valves were measured using a vessel and a stopwatch to measure the throughput in a fixed time span.

Since pumps have a linear behaviour this approach appeared to be the easiest. The gravity driven flow through the valves follows the equation

$$q = \sqrt{2 \cdot g \cdot h} \tag{8}$$

where g is the gravity constant and h is the distance from the surface to the outlet. The nonlinear function was linearised using the *polyfit* function in MATLAB. The model gained from this method however lacked accuracy and therefore had to be improved. To do so, a sequence of control actions was applied to the plant and the values measured by the ultrasound sensor were recorded. The parameters of the model were then slightly changed to fit its behaviour of the real plant.
Validation of the model proved it to be much more accurate than before. This is not surprising, since in a mere theoretic approach all hoses, connectors and valves as well as the characteristics of the actuators and sensors were not taken into account. Figure 2 shows the result of the validation.
Having found valid parameters to describe the model, the differential equations 5-7 were expressed in the state space using equations 1 and 2. The water level of each tank forms one state of the model and the flows q_{c1} and q_{c2} controlled by the pumps were treated as the two input values.

$$A = \begin{bmatrix} -\frac{a_1 \cdot q_{eq}}{A1} & \frac{a_2 \cdot q_{eq}}{A2} & 0 \\ \frac{a_3 \cdot q_{eq}}{A1} & \frac{a_4 \cdot q_{eq}}{A2} & 0 \\ 0 & 0 & \frac{a_5 \cdot q_{z2}}{A1} \end{bmatrix} \tag{9}$$

$$B = \begin{bmatrix} -\frac{b1 \cdot q_{c1}}{A1} & 0 \\ 0 & -\frac{b2 \cdot q_{c2}}{A2} \\ \frac{b3 \cdot q_{c1}}{A1} & \frac{b4 \cdot q_{c2}}{A1} \end{bmatrix} \tag{10}$$

Since all states are measured directly the output matrix C is a [3x3] identity matrix and the feedthrough matrix D is a [3x2] zero matrix.

4.2 Implementing a MPC using the MPC-Toolbox

In this work the MPC-Toolbox was used as a first attempt to control the plant with MPC. It was also used as a reference to compare later results of MPC implementations.
Having the state space model of the plant makes implementing the controller very simple, since Simulink provides a GUI that allows the definition of the model and the setup of all further parameters (such as prediction horizon or input weights) as required. The author of [9] was looking closely at the implementation of the MPC Toolbox on a dSPACE system.

4.3 Developing a MPC algorithm and defining subtasks

To build the cost function shown in equation 3 the H and G Matrices, had to be obtained from

$$H = \Theta^T Q \Theta + R \tag{11}$$

and

$$G = 2\Theta^T Q \varepsilon \tag{12}$$

where

$$\varepsilon = \Big(\Psi \cdot x(k-1) + \Upsilon \cdot u(k-1)\Big) - x_{ref}(k). \tag{13}$$

The matrices Ψ, Υ and Θ are gained from the state space model and describe the behaviour of the model within a future prediction horizon H_p. Therefore their value will not change during simulation. The tracking error matrix ε is gained by calculating the difference between reference trajectory (given by the user as set point) and the free response of the system i.e. the plants behaviour assuming that the last control action $u(k-1)$ will not be changed and therefore is kept constant over the prediction horizon. It therefore must be calculated periodically.
Q and R are diagonally shaped weighting matrices that allow to punish errors in curtain states or the control actions for each prediction step and with that change the behaviour of the MPC. Their values can be calculated online to allow the user to apply changes to them during the simulation. In the chosen implementation however Q and R can only be changed offline and therefore they stay fixed during simulation. The Q matrix was defined such that the deviation of the states from the set-point was punished more in the near future than at the end of the prediction horizon.

Since the optimisation problem in equation 3 has to be solved in subject to the constraints expressed as shown in equation 4, the restrictions of the plant have to be defined.
The constraints for control action were expressed as

$$-3.352 \leq u1, u2 \leq 3.352 \tag{14}$$

and describe the maximal throughput of the pumps. The limitations in the water level of the tanks were:

$$0 \leq x1, x3 \leq 65 \tag{15}$$

and

$$20 \leq x2 \leq 85. \tag{16}$$

The QP solver however calculates the change in the control action, that has to be applied to the previous control value to obtain the optimal control. Therefore all constraints have to be expressed a constraint of ΔU over the prediction horizon. The matrices Ω and ω have to be calculated each control step to update the information of the last control action.

Having the matrices H, G, Ω and ω calculated, they directly could be handed over to an opitimisation function such as *quadprog*. Finding the optimal solution using the Danzig algorithm implemented in the C-Code S-Function however requires several more matrix operations to calculate an initial tableau *tabi*, the initial setting of *il* and *ib* as well as a *initial basis* that are required by the Danzig algorithm. The matrices are build as follows:

$$tabi = \begin{bmatrix} -H^T & H^T \Omega^T \\ \Omega H^T & -\Omega H^T \Omega^T \end{bmatrix} \tag{17}$$

$$basisi = \begin{bmatrix} H^T \cdot a_{rhs} \\ c_{rhs} - \Omega H^T \cdot a_{rhs} \end{bmatrix} \tag{18}$$

where

$$c_{rhs} = \omega - \Omega \cdot x_{min}, \tag{19}$$

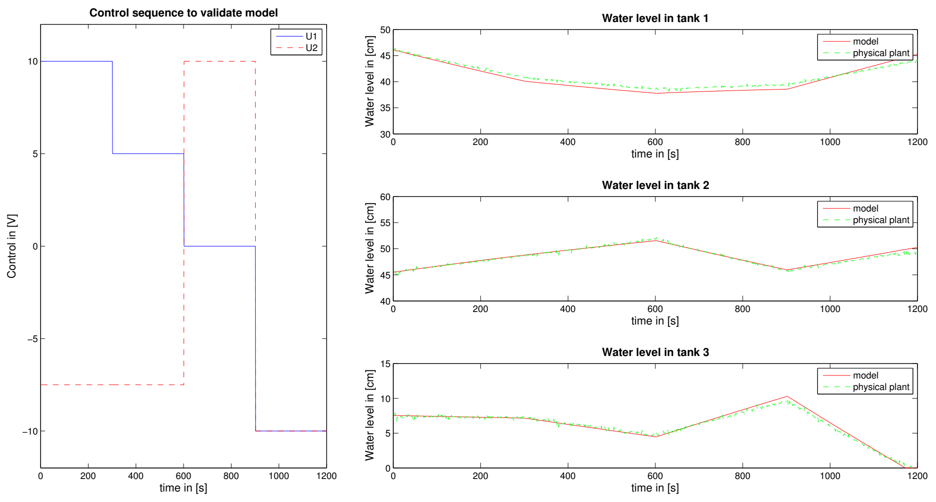

Fig. 2: Result of the model validation. The left graph shows the control action applied to the physical model and the simulated model over a timespan of 20 minutes. The three graphs on the right show the change in water level for the simulated model and the real plant.

$$a_{rhs} = H \cdot x_{min} - G \tag{20}$$

and x_{min} a vector with the same number of rows as H containing only the lefthand side of the constraints for each prediction step of the prediction horizon. *ibi* is a vector of a certain size, where each row is an iteration of the previous added by one starting at one. The number of rows n_{row} depends on the number of control signals n_{contr} the number of constraints n_{const} and the control horizon H_u i.e. the horizon within which the controller is allowed to apply changes to the control signal. The size of *ibi* is calculated using

$$n_{row} = n_{contr} H_u \cdot (1 + n_{const}). \tag{21}$$

ili is defined as $-ibi$. The variables as well as a value for the

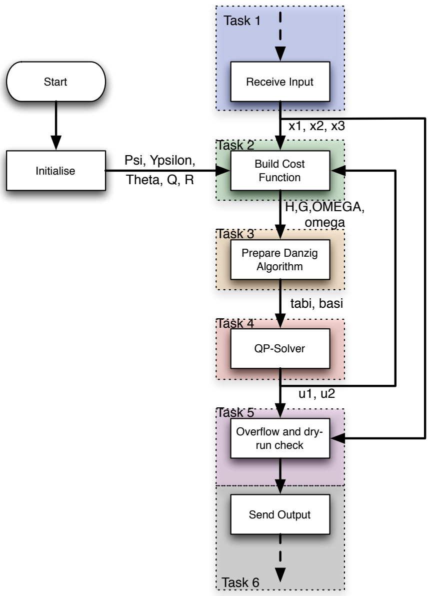

Fig. 3: The MPC algorithm was divided into subtasks as shown.

maximum number of iterations are handed over to the danzig algorithm that returns the optimal change of control action. After implementing and testing, the algorithm could be split into 6 subtasks as shown in Figure 3. The tasks are: (i) Receive signals from physical model (ii) Build the cost function (iii) Prepare matrices for the Danzig QP-Solver (iv) Minimise the cost function using the Danzig QP-Solver (v) Check for overflow or dry run on the physical system (vi) Send the optimal control signal to physical model.

5 Experiment and Discussion

A first experiment took the restrictions in the level of water in each tanks into account. Doing so however made the controller behave in rather undesired manner. Therefore the constraints of the states were removed as also suggested in [4]. Using 1 as initial value for all weights in Q and R, a prediction horizon of 10 steps and a sample time of 1 second, different weights were applied and tested on the plant. The prediction horizon could not be increased further, due to restrictions in the communication between multiple processors. See section 5.3 for further details.

5.1 Weight Tuning

As a first step, tank three was defined as a buffer tank. Its weight was therefore reduced by the factor of 0.1 compared to the other tanks. Assuming further that the weights for both pumps are always the same, different weights for the control action were set and the control behaviour as well as the Root Mean Square Error (RMSE) of tank one and two were analysed. The control action for different values of R is shown in Figure 4. When analysing the RMSE of the tanks, it was found that the dSPACE controller does not capture data periodically but instead with irregular gaps of up to 6 samples. It happen that more values have been captured during a curtain time interval than in another interval. The calculation of RMSE therefore becomes distorted and is unreliable.

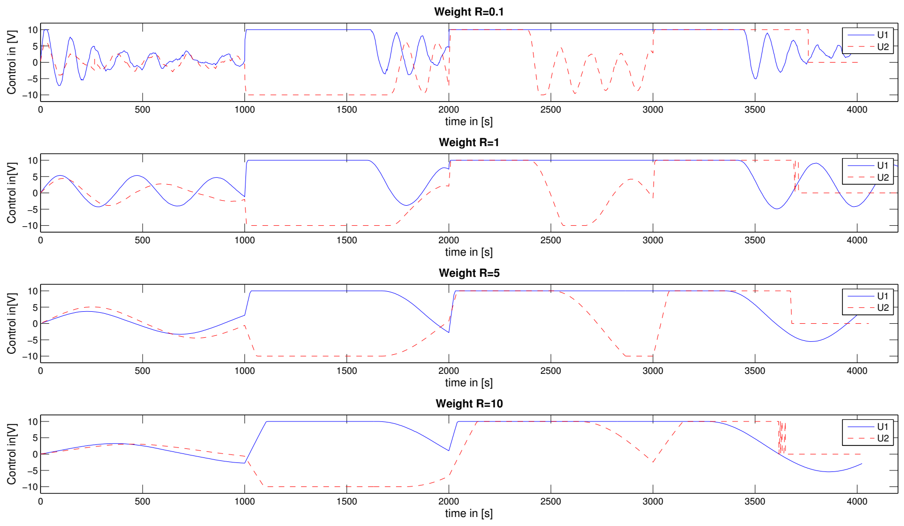

Fig. 4: Control behaviour using different values to punish control action. The values for R are top to bottom: 0.1; 1; 5; 10. The set point was changed every 1000 seconds

5.2 Comparing Simulink Toolbox MPC with S-Function MPC

Using the MPC-Block provided by the Simulink toolbox was a simple way of testing the suitability of the model with an MPC controller. Both the controller from the MPC-Toolbox and the developed MPC use the danzig algorithm to minimise the cost function. However giving them the same parameters for Q, R and the control and prediction horizons their behaviour significantly differs from each other. The Toolbox-MPC had to be tuned further using additional functions such as constraints softening and additional penalty for the control action itself.
Analysing the turn-around time after implementing the controller on the dSPACE system, the MPC-Block from Simulink was processed with a maximum turn-around time of 557,76 μs on a single RTP. This was almost 100 μs faster then the developed MPC-algorithm which had a max. turnaround-time of 653,57 μs on a single core.

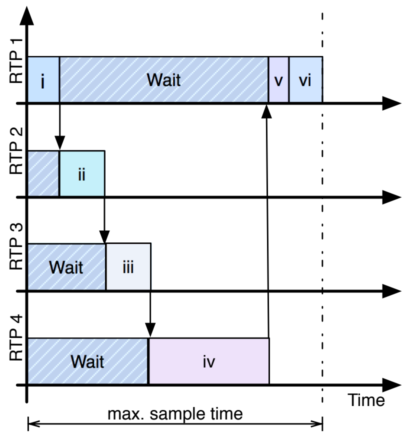

Fig. 5: The tasks (i) to (vi) as introduced in figure 3 have been scheduled as shown using the *synchronised swinging buffer* protocol for communication between the RTPs.

5.3 Controlling with Distributed Tasks

After defining the subtasks in section 4.3, different ways of distributing the tasks have been implemented and tested using also different protocols to pass information between the RTPs. The used protocols to handle data over to another processor were (a) the synchronised swinging buffer protocol and (b) the virtual shared memory protocol. Both are provided by RTI-toolbox and are part of the Virtual Gigalinks, that represents the connection between the cores on the DS1006 board. Each of the 18 Gigalink channels between two processors can transmit up to 8 KB per sample. This restriction of the communication was the reason why the prediction horizon had to be limited to 10 steps. Increasing the horizon entails a significant increase in the dimension of the H and G matrix which, after that, exceeded the restriction of the Gigalink channels.
In Protocol (a) the receiver is waiting for the sender to finish writing in the buffer before it starts reading data from it. This is slow compared to protocol (b), however it ensures the consistency of data. Since all tasks are started at the same time, tasks that require data from previous tasks have to wait until the previous calculation are completed. One can see the waiting time in Figure 5 and 6. In figure 5 the process of sending and receiving signals from the plant has been assigned to the RTP 1, which causes a long waiting time to that RTP. In figure 6 the configuration allows the core 1 to idle most of the time, however the waiting time from core 1 has been divided on to the three other cores that are all waiting a little longer. However a gain in the maximal turnaround time has been made using this configuration. The maximum turnaround time using the configuration shown in figure 5 is 797,98 μs where the one used in figure 6 is only 707,3 μs.
Using protocol (b) the sender and receiver share a virtual memory, that can be addressed as local memory from each CPU. It works without any synchronisation between the RTPs. This creates the possibility that the receiver reads the same date

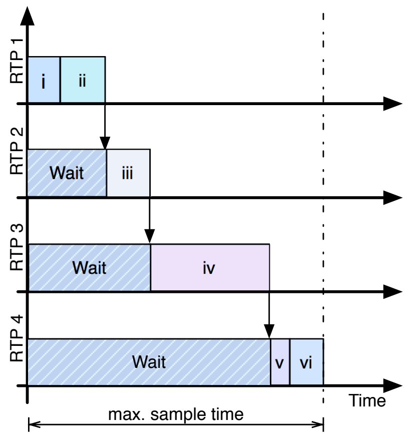

Fig. 6: The tasks (i) to (vi) as introduced in figure 3 have been scheduled as shown using the *synchronised swinging buffer* protocol for communication between the RTPs.

more than once or worse that it reads data that are inconsistent. Nevertheless it decreases the max. sample time significantly as shown in figure 7. Implementing this configuration, showed

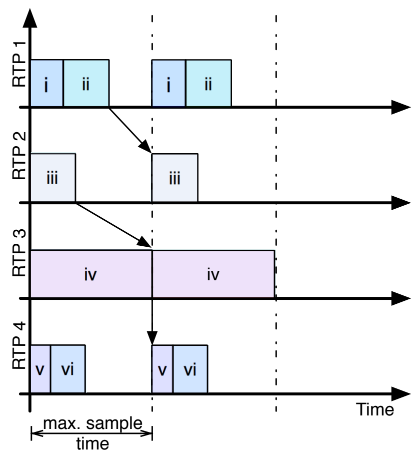

Fig. 7: The tasks (i) to (vi) as introduced in figure 3 have been scheduled as shown using the *virtual shared memory* protocol for communication between the RTPs.

that the max. turnaround time could be decreased to 375,38 μs using this protocol. However after less than 1000 seconds of simulation a deadlock occurred, that left the physical model uncontrolled.
Apart from the deadlock, no change in the control behaviour could be observed for any of the implemented parallel algorithms. If slight have changes occurred, the slow system dynamics and the sensor inaccuracy made them unnoticeable.

6 Conclusions

The successful implementation of an MPC algorithm on multiple processors controlling a physical three-tank model has been accomplished. In order to ensure data consistency, as often required in a real-time control system, the synchronized swinging buffer protocol has to be used. This slows down the communication between the RTPs significantly and makes the implemented parallel algorithm slower than the same running on a single core. To avoid this bottleneck, the amount of communication between the processors should be kept as small as possible. Another solution would be a parallel QP-algorithm as introduced in [11] to speed up the calculation significantly.

References

[1] M. Morar and J. H. Lee. Model predictive control: Past, present and future. *Computer and Chemical Engineering*, 23:667–682, 1997.

[2] S. J. Qin and T. A. Badgwell. A survey of industrial model predictive control technology. *Control Engineering Practice*, 11(7):733–764, 2003.

[3] E. Camacho, C. B. Alba. *Model Predictive Control*. Springer, 2004.

[4] J. Maciejowski. *Predictive control with constraints*. Pearson Education Ltd., 2002.

[5] D. Henriksson, A. Cervin, J. Akesson, and K.-E. Arzen. Feedback scheduling of model predictive controllers. In *Real-Time and Embedded Technology and Applications Symposium, 2002. Proceedings. Eighth IEEE*, pages 207 – 216, 2002.

[6] J. Sun and J. Liu. Synchronization protocols in distributed real-time systems. In *Distributed Computing Systems, 1996., Proceedings of the 16th International Conference on*, pages 38 –45, may 1996.

[7] M. C. Popescu and N. E. Mastorakis. Optimal flow control of a three tank system. *International journal of mathematics and computers in simulation*, 3(4), 2009.

[8] M.C. Popescu, G. Manolea, L. Perescu-Popescu, A. Drighiciu. *Implementation of New Solution Software for Three Tank System Control*. Technical report, University of Craiova, Faculty of Electromechanical and Environmental Engineering, 2009.

[9] D. Paluszczyszyn. *Real-Time implementation of a new control system utilising the Model Predictive Control and the Kalman Filter predictor for the Radiotherapy Patient Support System*. Master's thesis, University of Coventry, June 2008.

[10] T. Haugan. *Real-Time Model Predictive Control.*, pages 24, 25, 80–113. Diploma thesis. Swiss Federal Institute of Technology, June 2001.

[11] M. Brand, V. Shilpiekandula, S. A. Bortoff. A parallel quadratic programming algorithm for model predictive control. *World Congress of the Internatioal Federation of Automatic Control (IFAC)*, 10.3182/20110828-6-IT-1002.03222, 2011.