KU LEUVEN

ARENBERG DOCTORAL SCHOOL
Faculty of Engineering Science

Model Predictive Control in the Chemical Process Industry hosted by Industrial Controllers

Bart Huyck

Dissertation presented in partial fulfillment of the requirements for the degree of Doctor in Engineering

September 2013

Model Predictive Control in the Chemical Process Industry hosted by Industrial Controllers

Bart HUYCK

Supervisory Committee:
Prof. dr. ir. H. Van Brussel, chair
Prof. dr. ir. B. De Moor, supervisor
Prof. dr. ir. J. Van Impe, supervisor
Prof. dr. ing. J. De Brabanter, supervisor
Dr. ir. F. Logist, supervisor
Prof. dr. ir. J. Degrève
Prof. dr. ir. J. Vandewalle
Prof. dr. ir. L. Pinoy
(KAHO St-Lieven)
Prof. dr. ir. A.J.B. van Boxtel
(Wageningen Universiteit, NL)

Dissertation presented in partial fulfillment of the requirements for the degree of Doctor in Engineering

September 2013

© KU Leuven – Faculty of Engineering Science
Kasteelpark Arenberg 10, B-3001 Heverlee (Belgium)

Alle rechten voorbehouden. Niets uit deze uitgave mag worden vermenigvuldigd en/of openbaar gemaakt worden door middel van druk, fotokopie, microfilm, elektronisch of op welke andere wijze ook zonder voorafgaande schriftelijke toestemming van de uitgever.

All rights reserved. No part of the publication may be reproduced in any form by print, photoprint, microfilm or any other means without written permission from the publisher.

D/2013/7515/113
ISBN 978-94-6018-728-5

Voorwoord

Zo, de laatste verbeteringen zitten in de tekst. Nu is het tijd om als laatste onderdeel het eerste hoofdstuk van dit werk neer te pennen. Net iets minder dan acht jaar geleden ben ik gestart als assistent aan de KAHO St-Lieven met de vage verplichting om de helft van mijn tijd te spenderen aan het behalen van een doctoraat. Ik heb dat natuurlijk niet alleen kunnen doen. Heel wat mensen hebben mij gesteund gedurende die lange tijd met als resultaat dit mooie werkstuk.

Eerst en vooral wil ik mijn promotoren danken. Prof. De Moor om mij als doctoraatsstudent te aanvaarden en mee te denken rond een interessant onderwerp voor een doctoraatsstudent uit een academiserende hogeschool. Prof. Van Impe om voor mij een uitvalsbasis (een stoel en een tafel) te voorzien in zijn departement, voor de frequente begeleidingsmomenten en om mij de mogelijkheid te geven om tijdens mijn doctoraat op de destillatiekolom van het CIT te werken om zo mooie, praktisch resultaten te bekomen. Prof. De Brabanter om mij, zowel vanuit Gent als in Leuven, met raad en daad bij te staan en van tijd tot tijd mijn teksten na te lezen. Dr. Logist om mij, eerst als ervaren doctoraatsstudent en later als postdoc, tijdens mijn bezoeken aan Leuven door alle praktische en dagelijkse problemen te leiden. Ik kan niet genoeg benadrukken hoe belangrijk deze dagelijkse begeleiding geweest is voor mij! Als student uit een academiserende instelling, zoals de KAHO St-Lieven, waar onderzoek met behulp van doctoraten nog vrij recent is, is een goede dagelijkse begeleiding zeer nuttig geweest om alle kleine vaak slecht gedocumenteerde weetjes rond een doctoraat te weten te komen.

Verder wil ik ook de juryleden bedanken om in mijn jury te willen zitten, voor hun inbreng en opmerkingen. Prof. Van Brussel (voorzitter), Prof. De Moor, Prof. Van Impe, Prof. De Brabanter, dhr. Logist, Prof. Degrève, Prof. Vandewalle, Prof. Pinoy en Prof. van Boxtel.

Heel belangrijk tijdens mijn doctoraat waren ook mijn collega's van de vakgroep

E&A in Gent. Uiteindelijk hebben zij het doctoraat gedragen door ongeveer twee voltijdse lesopdrachten van mijn voorgangers onder elkaar te verdelen, waarvan ik slechts een halftijdse heb kunnen overnemen. Zonder hun hulp had ik nooit in alle vrijheid de helft van mijn tijd aan dit doctoraat kunnen wijden.

Verder wil ik ook alle collega's op het ESAT en CIT danken voor de hulp en mooie momenten die we beleefd hebben.

Ik wil ook mijn ouders en mijn vrouw Katrien en onze twee kleine schatten, Myrthe en Floris, bedanken voor de vele tijd die ze mij gegeven hebben om mijn doctoraat te voltooien en waarin ze mij hebben moeten missen.

Bart Huyck
Sint-Niklaas
Augustus 2013

Abstract

Linear Model Predictive Controllers (MPC) have been used in the chemical process industry for over forty years. They are currently implemented in a growing number of branches of the industry. The theoretical background of linear MPC is thoroughly investigated and recent evolutions focus on the speeding-up of algorithms to solve the quadratic optimization problem (QP), the heart of these controllers. Hence, MPC can be employed for fast systems as well as for embedded applications.

The aim of this PhD is to employ the recent online QP algorithms in order to minimize the use of computational power and memory to solve the MPC problem. Typical controllers used in industry, i.e., Programmable Automation Controllers (PACs) and Programmable Logic controllers (PLCs) have limited computational power and available memory. An in-depth investigation was performed to see if it is possible to run MPC hosted by a PLC. To this end, two practical set-ups were used, i.e., a mini set-up consisting of a fan and heating resistor, and a pilot-scale distillation column set-up. The complete chain starting from black-box linear system identification, simulations, hardware-in-the-loop experiments, up to experiments on the actual set-ups were carried out. On the mini set-up it has been proven that a PLC is able to solve the quadratic optimization problem accompanied by a model predictive controller. For the pilot-scale distillation column, it has been demonstrated that a PLC is not powerful enough to solve the QP problem with recent QP solvers. Nevertheless, a classical algorithm successfully passed hardware-in-the-loop simulations and experiments on the set-up. For PAC devices on the other hand, it has been verified that, on this pilot scale set-up, all investigated QP solvers can easily solve the optimization problem part of the MPC algorithm.

Beknopte samenvatting

Lineaire model predictieve regelaars (MPC) worden, sinds hun ontstaan in de chemische procesindustrie in de jaren 70 van vorige eeuw, veelvuldig en in verscheidene industrietakken gebruikt. De theoretische achtergrond rond lineaire MPC is grondig onderzocht en de huidige evolutie is vooral gericht op het versnellen van het algoritme voor het oplossen van het kwadratische optimaliseringsprobleem (QP), het hart van deze regelaars. Op die manier wordt het mogelijk MPC te gebruiken voor snelle systemen en in toepassingen waarbij een regelaar geïntegreerd zit in een machine, ook wel *embedded* toepassingen genoemd.

Het doel van dit doctoraat is het gebruik van deze recente QP-algoritmen om de benodigde rekenkracht en geheugen van de gebruikte regelaars te beperken. Typische regelaars in een industriële omgeving, zoals Programmable Automation Controllers (PAC's) en Programmable Logic Controllers (PLC's), hebben een beperkte rekenkracht en geheugen. Er wordt nagegaan in hoeverre een PLC gebruikt kan worden om MPC algoritmen uit te voeren. Dit wordt in dit doctoraat uitgetest in de praktijk op een mini opstelling bestaande uit een ventilator en verwarmingsweerstand, en een pilootschaal destillatiekolom. Daarbij wordt de volledige keten van identificatie van het model, ontwerp van een model predictieve regelaar, simulatie op PC en de uiteindelijke implementatie op een standaard industrieel regeltoestel, uitgevoerd. Op de mini opstelling is bewezen dat een PLC in staat is het kwadratische optimaliseringsprobleem verbonden aan een MPC regelaar op te lossen. Voor een pilootschaal destillatiekolom is gebleken dat, niettegenstaande experimenten succesvol voltooid zijn met een klassiek QP algoritme, recente QP algoritmen te veel rekenkracht vragen om uitgevoerd te worden op een de PLC. Met de huidige stand der techniek is algemeen gebruik van MPC op een PLC dus net een brug te ver, maar PAC's zijn in staat om het optimaliseringsprobleem betrouwbaar op te lossen.

Abbreviations

AI	Analogue Input
AIC	Akaike Information Criterion
ARMAX	Auto-Regressive Moving-Average with eXogeneous inputs
ARX	Auto-Regressive with eXogeneous inputs
BJ	Box-Jenkins
cFP	Compact FieldPiont
cRIO	CompactRIO
CV-MSE	Mean Squared Error of Crossvalidation
F	This experiment failed for the employed MPC approach/QP algorithm
FB	Function Block
FPGA	Field-Programmable Gate Array
HIL	Hardware-in-the-loop (experiments)
I/O	Input(s) and Output(s)
LS-SVM	Least Squares Support Vector Machine
MIMO	Multiple-Input Multiple-Output
MISO	Multiple-Input Single-Output
MPC	Model Predictive Control
MSE	Mean of Squared Errors
N/A	Not applicable
N/C	Not calculated

OB	Organization Block
OE	Output Error
PAC	Programmable Automation Controller
PID	Proportional-Integral-Derivative
PLC	Programmable Logic Controller
PRB	Pseudo Random Binary
QP	Quadratic Problem
RGA	Relative Gain Array
RTD	Resistance Temperature Detector
RTOS	Real-Time Operating System
s.t.	Subject To
SISO	Single-Input Single-Output
SSE	Sum of Squared Errors
STL	Statement List

List of Symbols

$\mathbf{A}(z)$, ...	Polynomials in a polynomial model
$\mathbf{u}(k)$	Input vector $\mathbf{u}$ on time instance k
$\mathbf{x}(k)$	State vector $\mathbf{x}$ on time instance k
$\mathbf{y}(k)$	Output vector $\mathbf{y}$ on time instance k
ω_n	Natural frequency of a transfer function
τ_d	Dead time in a transfer function
τ_x	$x = 1, 2, \ldots$ Time constant in a transfer function
ξ_d	Damping ratio of a transfer function
$A, B, ...$	Matrices
$H(s)$	Transfer function in the s-domain
$h(t)$	Transfer function in the time domain
k	Discrete time steps $k \in \mathbb{N}$
K_p	Proportional gain in a transfer function
t	Time (in seconds)
$\boldsymbol{\xi}(k)$	The augmented state

Contents

Contents xi

List of Figures xv

List of Tables xix

1 General Introduction 1

1.1 Motivation 1

1.2 Aim - Research 4

1.3 Content overview per chapter 5

1.4 Contribution 6

2 Model Predictive Control 9

2.1 A brain teaser 9

2.2 An appropriate model for control 10

2.2.1 Linear time-invariant models 11

2.2.2 Least Squares Support Vector Machines 14

2.2.3 Model selection 15

2.2.4 Model validation 16

2.3 Model predictive control 17

2.3.1 Basic idea . . . 17
2.3.2 Model predictive control formulation . . . 18
2.3.3 Solving a QP without constraints . . . 21
2.3.4 Solving a QP with inequality constraints . . . 21
2.4 Hardware and software . . . 23
2.4.1 Hardware . . . 23
2.4.2 Employed hardware . . . 29
2.4.3 Software . . . 31
2.4.4 Software to solve a QP . . . 33
2.5 Controller implementation features . . . 35
2.5.1 Model predictive controller implementation . . . 35
2.5.2 Features of the implementation on the PAC . . . 37
2.5.3 Features of the implementation on the PLC . . . 38
2.6 Conclusion . . . 41

3 Case Study I: A Small-Scale Air Heating Set-Up 43

3.1 Air heating set-up . . . 43
3.2 Model identification . . . 44
3.3 Model predictive control settings . . . 47
3.4 Model predictive control approaches . . . 49
3.5 Model predictive control simulations on a PC . . . 50
3.6 Model predictive control on a PAC . . . 55
3.6.1 Hardware-in-the-loop experiments . . . 55
3.6.2 Resulting temperature control on the set-up . . . 59
3.6.3 Properties of the QP solvers . . . 62
3.7 Model predictive control on a PLC . . . 66
3.7.1 Resulting temperature control on the set-up . . . 66

3.7.2 Properties of the QP solvers 66
3.8 Conclusion 69

4 Case Study II: A Pilot-Scale Distillation Column 71

4.1 Distillation column set-up 71
4.2 Model identification 74
4.2.1 Excitation experiments 75
4.2.2 Data preparation 77
4.2.3 Model validation and selection 77
4.3 Model identification - transfer functions 77
4.4 Model identification - subspace state-space 82
4.5 Model identification - polynomial models 84
4.6 Model identification - Least Squares Support Vector Machines . 87
4.7 Selected model - model analysis 92
4.8 Summary and remarks on the identification results 94
4.9 Model predictive control approaches 96
4.10 Model predictive control simulations on a PC 97
4.10.1 Model predictive control settings 97
4.10.2 Matlab simulation experiments 97
4.11 Model predictive control on a PC 101
4.11.1 Model predictive control settings 101
4.11.2 Simulation with the Matlab MPC toolbox 101
4.11.3 Experiments on the pilot-scale set-up 103
4.12 Model predictive control on a PAC 105
4.12.1 Model predictive control settings 105
4.12.2 Hardware-in-the-loop experiments 105
4.12.3 Experiments on the pilot-scale set-up 106

4.13 Model predictive control on a PLC 111
4.13.1 Hardware-in-the-loop experiments 111
4.13.2 Experiments on the pilot-scale set-up 115
4.14 Conclusion 117

5 Conclusions, Remarks and Future Research 119
5.1 Summary 119
5.1.1 Case I: air heating set-up 119
5.1.2 Case II: pilot-scale distillation column 121
5.2 Remarks and final conclusions 122
5.3 Future research 123

A MPC Algorithm 125
A.1 Addition of an integrator 125
A.2 Calculation of the future output 126
A.3 Controller objective 129

B Distillation column model 133
B.1 Transformation from the s-domain 133
B.2 The employed Matlab code 134
B.3 The employed state-Space model 134

Bibliography 137

C Curriculum vitae 145

D Publications by the author 147

List of Figures

1.1 Overview of the different control layers (a). The idea to combine a local loop controller with predictive control into one layer can be considered (b). 2

1.2 The aim of this work: Run to the complete chain to implement a model predictive controller on an (embedded) industrial controller. 4

1.3 Overview of the relations between the different chapters. 6

2.1 The basic idea behind Model Predictive Control. 18

2.2 Graphical comparison where to situate industrial control hardware, i.e., PLC and PAC to PC in view of FLOPS (Floating Point Operations Per Second) and memory. 24

2.3 The two PAC devices used in this dissertation. 30

2.4 The graphical symbol of the LabVIEW Call Library function. . 30

2.5 A Siemens CPU319-3DP/PN PLC. 30

2.6 Step 7 Professional 2010. 32

2.7 LabVIEW programming environment. On the left, a front panel is presented. On the right the block diagram is depicted. 32

2.8 Schematic overview of the different employed organization and function blocks in the PLC. 39

3.1 The air heating set-up. 44

3.2 Validation of the P1D and N4SID models on multisine validation data. 46

3.3 The effect of different horizons for the proposed model predictive controller. 48

3.4 The reference trajectory for the air heating set-up. 49

3.5 CVXGEN source code for the MPC controller referred to as CVXGEN MPC. 51

3.6 The in- and outputs for CVXGEN MPC and the Hildreth, CVXGEN and qpOASES algorithm for Matlab simulations on PC. 52

3.7 Number of iterations needed to solve the QP problem in a Matlab simulation on PC. 54

3.8 The in- and outputs for LabVIEW MPC, CVXGEN MPC and the Hildreth, CVXGEN and qpOASES QP algorithms for hardware-in-the-loop experiments on the CompactRIO. 56

3.9 The number of iterations and computation time needed for LabVIEW MPC, CVXGEN MPC, the Hildreth, CVXGEN and qpOASES QP algorithm for hardware-in-the-loop experiments on the CompactRIO. 58

3.10 The in- and outputs for the Hildreth, CVXGEN and qpOASES algorithm on the PAC for experiments in summer on the CompactRIO. 60

3.11 The in- and outputs for the CVXGEN MPC, LabVIEW MPC, Hildreth, CVXGEN and qpOASES algorithm on the PAC for experiments in winter on the CompactRIO. 61

3.12 The number of iterations and corresponding calculation time to solve the QP problem on the PAC system for the summer and winter experiments on the CompactRIO. 63

3.13 The in- and outputs for the Hildreth and qpOASES algorithm on the PLC. 67

3.14 The number of iterations and corresponding calculation time needed to solve the QP problem on the PLC for both algorithms. 69

4.1 Diagram of the pilot-scale distillation column. Nominal set-points for identification experiments are printed in bold and are followed by the maximum employed deviations during identification experiments. 72

4.2 Pictures of the pilot-scale distillation column: condenser (left), packed section and feed introduction (center), and reboiler (right). 73

4.3 Overview of the in- and outputs of the pilot-scale distillation column model. 74

4.4 The visual representation of the two different excitation experiments. 76

4.5 Validation of the top and reboiler temperature. Each figure depicts the simulation for the reduced state-space model and the identified PMIX model. 78

4.6 A plot of the Hankel singular values for the discrete state-space model. 81

4.7 Validation of the selected subspace state-space models (N4SID), the transfer function model employed in the MPC (TF SS), the selected OE models (OE) and the LS-SVM model (LS-SVM) for the top and reboiler temperature. 83

4.8 10-fold CV on training data and the MSE on test data for the LS-SVM models as original published in [26]. 89

4.9 10-fold CV on training data and the MSE on test data for the LS-SVM model with the same training and test data sets as used for linear identification. 91

4.10 The in- and outputs for the Hildreth and qpOASES algorithm in simulation on a PC. 98

4.11 Number of iterations needed to solve the QP problem for simulation on the PC for both the Hildreth an qpOASES algorithms. For qpOASES, the number of iterations in case of hot starting is also presented. 100

4.12 The in- and outputs for the MPC simulation experiment and the experiment on the set-up with the Matlab Model Predictive Control toolbox running on PC. 102

4.13 Measured outputs and inputs for experiments on the distillation column and HIL experiments tracking the desired reference temperature profile with an MPC controller running on the PAC. 107

4.14 Calculation time and number of iterations for the employed QP algorithms on the PAC. 110

4.15 Measured outputs for the top- and reboiler temperature for a hardware-in-the-loop experiment tracking a desired reference temperature profile with a MPC controller running on the PLC. 112

4.16 Calculation time and number of iterations for the employed QP algorithms for a hardware-in-the-loop experiment with a MPC controller running on the PLC. 113

4.17 Measured outputs for the top- and reboiler temperature for an experiment on the pilot-scale set-up tracking a desired reference temperature profile with a MPC controller running on the PLC. 116

4.18 Calculation time and number of iterations for an experiment on the pilot-scale set-up with a MPC controller running on the PLC. 117

List of Tables

2.1 Overview of the PAC market for a selected number of brands. . 27

2.2 Overview of the PLC market for a selected number of brands. . 28

3.1 A quality measure for the P1D, P1DSS and N4SID model by means of the FIT and MSE value. The Akaike Information Criterion (AIC) can be used to select the best model. 46

3.2 The MSE and SSE values calculated for Matlab simulations on PC. The total run time and the mean, minimum and maximum run times for one MPC iteration are presented to indicate the speed of the algorithms. 53

3.3 Comparison of the five temperature profiles to the reference trajectory based on SSE and MSE for hardware-in-the-loop experiments on the CompactRIO. The total run time for the experiments and a mean, minimum and maximum time for one MPC iteration step are also presented. 57

3.4 Comparison of three temperature profiles to the reference trajectory based on SSE and MSE for the experiments on the CompactRIO in summer. 62

3.5 Comparison of the five temperature profiles to the reference trajectory based on SSE and MSE for the experiments on the CompactRIO in winter. The total run time for the experiments and a mean, minimum and maximum time for one MPC optimization step are also presented. 64

3.6 Comparison of two temperature profiles to the reference trajectory based on SSE and MSE for the experiments on the PLC. The total run time for the experiments and a mean time for one optimization step are also provided. 68

4.1 Model quality and selection parameters for the transfer function models. 79

4.2 The FIT and MSE values for the reduced state-space model and identified PMIX models for both the reboiler and top temperature. 81

4.3 Quality and selection parameters for the subspace state-space identification. 82

4.4 Maximum value for each of the parameters employed in the Box-Jenkins model structures. 84

4.5 Model identification results of the polynomial models following the Box-Jenkins model structure for the top temperature. . . . 85

4.6 Model identification results of the polynomial models following the Box-Jenkins model structure for the reboiler temperature. . 86

4.7 The model quality parameters FIT and MSE on test data and the model selection criterium AIC on training data for LS-SVM models for the top and reboiler temperature as presented in [26]. 88

4.8 Model quality parameters FIT and MSE on test data and the model selection criterium AIC on training data for LS-SVM models for the top and reboiler temperature 92

4.9 Relative Gain Array for the MIMO transfer function model (Equation (4.1)). 93

4.10 The MSE and SSE values calculated for the simulation experiments in Matlab on PC. The total run time and the mean, minimum and maximum run times for one MPC iteration is presented to indicate the speed of the algorithms. 100

4.11 The MSE value calculated for hardware-in-the-loop experiments and experiments on the set-up on the PAC. The total run time as well as the mean, minimum and maximum run times for one MPC iteration are presented to indicate the speed of the algorithms. 109

4.12 The MSE value calculated for the hardware-in-the-loop and experiments on the column set-up hosted by the PLC. The total, mean, minimum and maximum run times for one MPC iteration is presented to indicate the speed of the algorithms. (F = Failed experiment). 114

5.1 Mean time and maximum time, the latter indicated between brackets, to calculate the input for the air heating set-up. All presented times are indicated in milliseconds. The abbreviations SIM, HIL and EXP stands for simulations on PC, hardware-in-the-loop experiments and experiments on the set-up, respectively. 120

5.2 Mean time and maximum time, the latter indicated between brackets, to calculate the input for the pilot-scale distillation column. All presented times are indicated in milliseconds. The abbreviations SIM, HIL and EXP stands for simulations on PC, hardware-in-the-loop experiments and experiments on the set-up, respectively. 121

Chapter 1

General Introduction

1.1 Motivation

One of the most successful and widespread advanced control methodologies in industry is *Model Predictive Control* (MPC). It has become a widely applied control technique in process industry for the control of large-scale installations for over forty years [2, 39, 67]. The advantages of MPC over classic control methodologies, e.g., PID control, are its ability (i) to steer the process in an optimal way while taking proactively desired future behavior into account, (ii) to tackle multiple inputs and outputs simultaneously and (iii) to incorporate constraints [38].

Model predictive control originates in the petrochemical industry and after several successful applications [11, 69], it is currently used in many other application areas, e.g., automotive industry [25], paper and pulp industry [84], power converters [31] and traffic control [3]. In the eighties, different academic branches picked up MPC and investigated its theoretical background, e.g., the stability and optimality [43, 68]. These works have mainly been finished by the start of this millennium. Since then, the focus of researchers working on MPC has moved to topics that are still under investigation, e.g., fast(er) algorithms [4, 16, 42] to increase and simplify the use of MPC in the industry.

Technical improvements, e.g., the increase of speed of microprocessors and the availability of cheap, powerful 16 and 32-bit microprocessors and Field-Programmable Gate Arrays (FPGAs), create opportunities to use MPC under different circumstances and in other application areas than the process industry. Historically, model predictive controllers have been used as supervisory

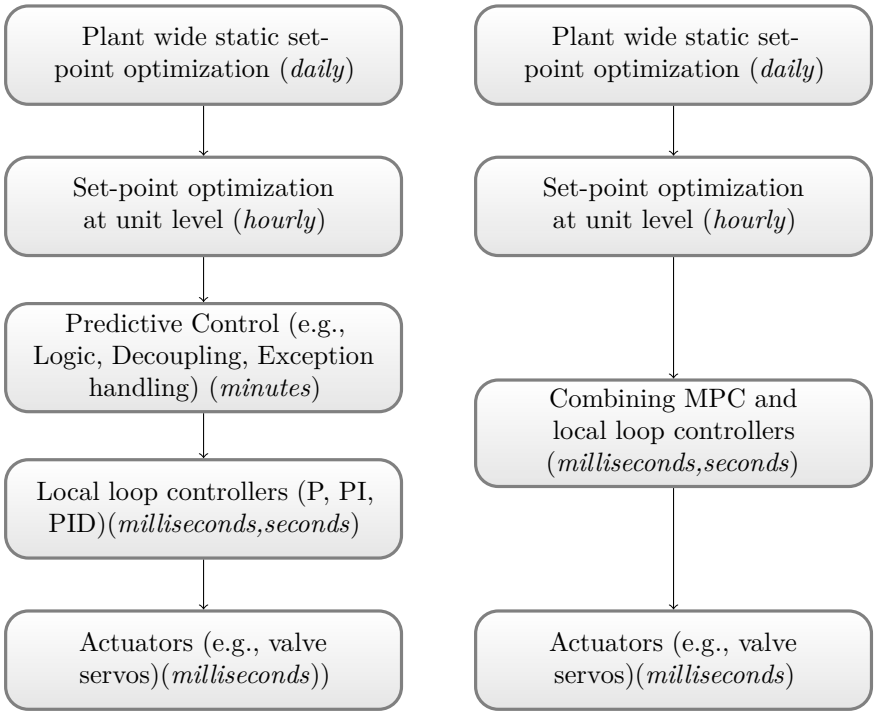


Figure 1.1: Overview of the different control layers (a). The idea to combine a local loop controller with predictive control into one layer can be considered (b).

controllers [38] adapting the set-point of underlying local loop controllers (see Figure 1.1a). Calculation of the set-point took a large amount of time in the early days of MPC, certainly with several in- and outputs. Given the evolution of increasing computing power and faster algorithms, calculation times have been drastically reduced, but still it can take quite some time to calculate an optimal solution.

Not only the process industry is interested in constraint-aware Multiple-Input, Multiple-Output (MIMO) control systems, but also, e.g., the automotive industry, where update intervals in the order of milliseconds are necessary. New and fast routines to solve the MPC optimization problem to meet these requirements have been developed in the last decade [4, 16, 88]. Many of these algorithms were developed with *embedded* applications in mind. Embedded applications are applications where the control scheme is running on an

autonomous platform, often integrated in a machine. Typical examples of such platforms are microcontrollers and FPGAs. They exist in a wide variety resulting in a high variability of available memory and computational power. To facilitate the use of MPC in embedded applications, not only robustness and stability of the theoretical framework has to be proven, but also an easy way to implement the solvers is necessary, e.g., through the use of code generators and ready-to-use code.

In the rush to faster MPC, one can forget that not every controller needs to be extremely fast. These fast algorithms can also be used in existing applications using available standard industrial controller hardware. When not using the top segment hardware from, e.g., Programmable Logic Controllers (PLCs), high computational power is not available. However, the robustness of this proven technology is highly appreciated in industry. These devices are robust and last the lifetime of an installation which eliminates expensive replacements of hardware. However, evolving legislation sometimes forces companies to meet new standards that cannot be always reached by the traditional controller schemes. An upgrade to, e.g., an MPC running on installed hardware can be considered. Whether this upgrade can avoid a long shutdown to install new, often expensive, controllers is part of this research.

On the other hand, also in industry, controller hardware evolves and operator demands change. Up to the start of this millennium, there was a clear distinction between powerful process computers running advanced process algorithms and low level equipment driving the process. The increasing computing power and the replacement of large operating panels with knobs and analog indicators by web panels has initiated the evolution to add new tasks and features to the hardware controlling the process itself. With higher computational power, one can also think to eliminate the local loop control layer and combine the predictive control and local loop layer in one predictive controller running on one device as presented in Figure 1.1b. Even if the local loop layer is not removed, the controller hardware might be powerful enough to run a local and predictive control layer on the same device.

The technical improvements, the mostly finalized theoretical background for linear MPC, recent online algorithms provided in an easy-to-use way to solve the Quadratic Problem (QP), often the heart of a model predictive controller, open a path to investigate the use of low-level controllers as an industrial PLC, to host a linear predictive controller. In the mean time the current evolution on this type of controllers is followed by also adding Programmable Automation Controller (PACs) to this investigation. Nonlinear MPC is not covered.

Its not the first time a PLC is suggested to host linear MPC algorithms. Bemporad et al. [4] used explicit MPC to solve quadratic problems. It has

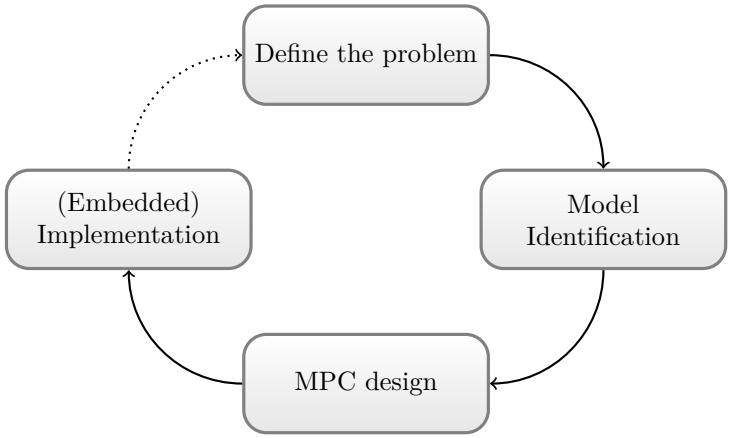


Figure 1.2: The aim of this work: Run to the complete chain to implement a model predictive controller on an (embedded) industrial controller.

been picked up and used for PLC applications [32, 82]. In this so called explicit MPC, the quadratic problem is solved offline. A piecewise linear solution to the constrained linear quadratic regulation problem is calculated and this piecewise linear solution is stored on the PLC in a look-up table. According to the current state, the right piecewise linear solution is selected and the corresponding system input is calculated.

Calculating the solution of the quadratic problem online for an model predictive controller hosted by a PLC is new. However, industrial applications for MPC on a PLC can be found. For instance, SIMATIC PCS 7 MPC (ModPreCon) from Siemens is an online model predictive controller hosted on a PLC. Application notes, e.g. [57], focus on the implementation of this controller. However, discovering how the controller works and what its limitations are, is difficult. In this work, the application of MPC on a PLC with online solution of the QP is investigated.

1.2 Aim - Research

The aim of this PhD is to investigate the practical feasibility of online linear MPC on typical industrial controllers, more specifically PACs and PLCs. Instead of focussing on a small aspect of MPC, this work encloses the complete chain starting with the problem definition, the model identification and model based controller design and finally the implementation of MPC (Figure 1.2).

As processes may change in time by, e.g., upgrades and different operating conditions, the model needs to be updated. In such a case, the chain is closed and becomes a cycle as presented in Figure 1.2. Two case studies are investigated, (i) a simple, easy-to-use heating device based on a hairdryer and (ii) a pilot-scale distillation column available at the chemical engineering department (CIT) in Leuven.

The intention is not to propose new methods to solve one or more subproblems, but to verify the applicability of recent advances and existing methods in linear MPC on the two experimental set-ups. This investigation will lead to suggestions and conclusions about the practical feasibility of online linear MPC on typical PLCs in an industrial environment. The current state-of-the-art on MPC will be viewed from the industrial point of view.

1.3 Content overview per chapter

This thesis consists mainly out of the description of the steps as presented in Figure 1.2 for both the air heating set-up and the pilot-scale distillation column. Figure 1.3 displays a general layout of the thesis structure. Details on the different chapters are given below.

Chapter 2 describes the concepts, methods, hardware and software used in this work. The process to obtain a model by means of system identification is presented together with the employed linear model structures and the nonlinear LS-SVM model structure. A basic description of Model Predictive Control is given together with the employed cost function and solution strategy. The employed QP solvers are highlighted.

Chapter 3 presents the complete cycle as presented in Figure 1.2 for a small air heating set-up. Starting with model identification, followed by the controller design and tuning of the model predictive controller, simulations on a computer are performed. After successful simulations, hardware-in-the-loop experiments on a PAC and experiments on the set-up with the PAC and finally with a PLC as controller hardware, are performed. In order to run MPC on a PLC, a typical industrial controller and translation of the algorithm has to be available in Statement List (STL), a textual PLC programming language. For the employed Siemens controller, the manufacturer's dialect, S7-SCL, has been used. Two algorithms are been investigated on this device: the Hildreth quadratic programming procedure and the qpOASES QP solver.

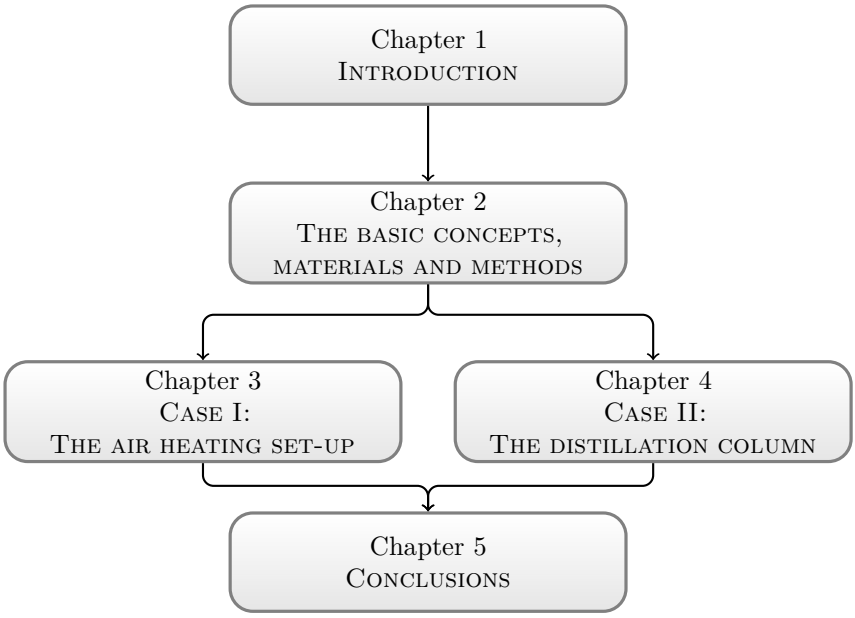


Figure 1.3: Overview of the relations between the different chapters.

Chapter 4 illustrates the identification of a controller model and implementation of a model predictive controller on a pilot-scale distillation column. In the first section the set-up is described. The identification of a model for control and actual model based control on different devices, is reported. First, some sections describe the results obtained by identification of linear black-box models based on transfer functions, subspace state-space techniques and polynomial models. One section in the model identification contains the results of an LS-SVM modeling procedure. The last sections in this chapter describes linear MPC hosted on a PC, PAC and PLC. The final section is used to come up with some conclusions on control on these devices.

Chapter 5 summarizes the main conclusions on this work and presents an outlook to future work.

1.4 Contribution

The contributions of this work are mainly situated in Chapters 3 and 4. Given the almost identical steps towards the implementation of online linear

model predictive control on an industrial PLC, only the application is different between the chapters.

Chapter 3. The small-scale air heating set-up has been used to investigate the different QP solvers. The implementation on the PAC is straightforward. An existing C implementation of qpOASES and C code generated by the CVXGEN web tool has been used to implement the QP solvers. The C-code for the Hildreth solver is translated from an existing Matlab implementation based on work by [87]. The qpOASES and Hildreth implementation on the PLC are written from scratch in S7-SCL, based on the Matlab implementation.
Although no *new* contribution to any theoretical framework is added, the contribution relates mainly to the implementation on PAC and PLC of *online* quadratic problem solvers required for MPC. The results of experiments carried out with a number of online QP algorithms hosted by the two employed controller devices, have been reported in:

- Bart Huyck, Hans Joachim Ferreau, Moritz Diehl, Jos De Brabanter, Jan Van Impe, Bart De Moor and Filip Logist. Towards online model predictive control on a programmable logic controller: practical considerations. *Mathematical Problems in Engineering*, 2012:20, 2012 (Article ID 912603, doi:10.1155/2012/912603.).
- Bart Huyck, Filip Logist, Jos De Brabanter, Jan Van Impe, and Bart De Moor. Constrained model predictive control on a programmable automation system exploiting code generation: practical considerations. In *18th World Congress of the International Federation of Automatic Control*, pages 12207–12212, Milano, Italy, August 28–September 2, 2011.
- Bart Huyck, Filip Logist, Hans Joachim Ferreau, Moritz Diehl, Jos De Brabanter, Jan Van Impe, and Bart De Moor. Implementation and experimental validation of classical mpc on programmable logic controllers. In *20th Mediterranean Conference on Control and Automation. Barcelona (MED 2012)*, pages 679–684, Barcelona, Spain, July 3–6, 2012.

Chapter 4. On the pilot-scale distillation column, the complete cycle starting with model identification, simulation and implementation, is repeated. The identification of the set-up based on LS-SVM models is performed, but is has to be noticed that using these models in control is not yet possible as the theoretical framework has not been fully elaborated. The pilot-scale set-up is a more realistic system for industry compared to the air heating set-up. The implementation on both PAC and PLC has

indicated that PLCs, as currently available, can be used, however, the lack of computing power and available memory hinder an easy applicability and restrict their area of application. PAC systems on the other hand are ready to be used in industry.

- Bart Huyck, Kris De Brabanter, Filip Logist, Jos De Brabanter, Jan Van Impe, and Bart De Moor. Identification of a pilot-scale distillation column: A kernel based approach. In *18th World Congress of the International Federation of Automatic Control*, pages 471–476, Milano, Italy, August 28–September 2, 2011.
- Bart Huyck, Jos De Brabanter, Bart De Moor, Jan Van Impe, and Filip Logist. Model predictive control of a pilot-scale destination column using a programmable automation controller *European Control Conference 13 (ECC13)*, Zurich (Switzerland), July 17-19, 2013

Chapter 2

Model Predictive Control

In this chapter, the basic elements to use a model predictive controller (MPC) are presented. First of all, as a model is needed, some techniques to obtain a model are summarized. Second, the theoretical background of MPC is introduced and the more recent evolutions important for MPC on embedded devices are highlighted. Next, PACs and PLCs are introduced and the most important features are listed together with a non-exhaustive market overview. A last section is dedicated to the employed hardware and software.

2.1 A brain teaser

Imagine, you have been asked to drive a car in the front direction with a non-transparent windscreen. Two excellent driving mirrors on both sides of the car make it possible to see what is behind you. Would you feel safe driving this way at 120 km/h on a motor way? I suppose not. No one will ever want to drive a car in such situation! To feel safe, you want to know where you are going to, to make sure you can anticipate what is in front of you, e.g., to see a corner or other vehicles.

But, be aware. Replace the car by an industrial process and the driver by a commonly used industrial controller, e.g., a PID controller. Such a controller is only aware of the past and current situations (what is behind and next to you) and has no idea what is coming (its non-transparent windscreen). Such a controller will never be able to anticipate on what is coming. However, for many applications this is not a problem.

Historically, typical process applications of MPC are slowly varying systems. Due to the globalization of the market, product changes for large-scale installations are more frequent and product distribution has become more narrow. To reduce the time to switch from one to another grade and to reduce the amount of off-spec production, MIMO and easy to tune controllers are required. When the controller knows in advance that the production of a new grade will start soon, the controller can anticipate to minimize the transient time, and so the off-spec production. The only way to have an idea of what will happen in the future is to have a description *of how the system behaves. This description can be a mathematical model of the system. In the car example there are several possibilities, e.g., a simple map of the environment, a camera and a windscreen. All these* models *describe the environment of the car, but none is perfect and none can warn for unforeseen obstacles.*

The idea behind model based control is to use a mathematical model to predict the behavior of the system [10]. This model is also used to calculate the current optimal *control action. What* optimal *exactly means is subject to choice of the designer of the process controller. Optimality of the process will most likely imply, e.g., maximal production, minimal energy consumption and minimal use of raw materials, but can also be chosen from an, e.g., economic, social, safety point of view or combination of these elements.*

It is clear that the accuracy of the employed model will define the accuracy of control, but a model that contains the important parts is already helpful. In the car example it is sufficient to know the location of the road and the other vehicles. We do not need the exact location of the swimming pool of the village 100 meter next to the road. This is unnecessary information while driving at 120 km/h. In the coming sections, possible methods to obtain a sufficiently accurate but not too overly detailed model are presented.

2.2 An appropriate model for control

A model for control needs to describe the process in a simple but sufficiently accurate way. The reason for this is to be able to decide fast, but accurately, what the next control input has to be. Basically, the following three modeling approaches can be distinguished [34, 63]:

White-box modeling White-box models are fully derived from first principles, i.e., physical, electrical, mechanical and chemical laws. All equations and parameters can be derived from theory. If one or more parameters are derived with the help of measured data this is still considered as

white-box, as long as there is a direct interpretation from known first principles.

Black-box modeling Black-box of data driven models are based on measurement data. Both model structure and parameters are determined from an experimental modeling strategy.

Grey-box modeling For grey-box models a model structure is proposed based on prior knowledge. The unknown model parameters are estimated based on experimental data.

Not only do different modeling approaches exist, also a choice has to be made between, e.g., linear and nonlinear models, parametric and non-parametric models. Due to the large variety in models, it is important to keep the purpose of the model in mind. For this dissertation, the following elements need to be respected.

1. The model will be used for control. An appropriate control methodology that can employ the obtained model has to be available.
2. Given the low computational power and limited amount of memory of the employed hardware, the model has to be as simple as possible, but still accurate.
3. Two processes have to be considered. The hairdryer set-up is easy and one can choose different modeling approaches. In contrast, the distillation column is a more complex (industrial) process application. It is in nature a nonlinear system. It is a slow system with time constants varying from seconds to more than one hour and has highly interconnected variables.

Combining the above elements, linear, time-invariant, parametric models obtained via black-box modeling have been chosen in this dissertation. There exist numerous excellent books that describe the background of these and other models [18, 34, 65, 79]. How to obtain the controller model is not a key element in this work, but an overview of the employed linear model classes is given below. Some background with regard to Least Squares Support Vector Machines is supplied as the pilot-scale distillation column has also been identified with this model class.

2.2.1 Linear time-invariant models

Different linear time-invariant models exist. In view of the current work, transfer function models in the frequency domain and polynomial models

and state-space models in the time domain will be briefly introduced. The identification of the these models is performed using the Matlab System Identification Toolbox [35].

Transfer function models

A common way to describe the relation between an input and an output of a system in terms of (temporal) frequencies is a transfer function. This transfer function $H(s)$ in the s-domain, $s \in \mathbb{C}$, is the unit impulse response $h(t)$ of a system in the time domain. In this work, only first and second order transfer functions are considered, but higher order can be constructed. For the continuous case, a first order, a damped second order and an undamped second order transfer function with time delay are represented by respectively Equations (2.1), (2.2) and (2.3):

$$H(s) = \frac{K_p}{1 + \tau_1 s} e^{\tau_d s} \tag{2.1}$$

$$H(s) = \frac{K_p}{(1 + \tau_1 s)(1 + \tau_2 s)} e^{\tau_d s} \tag{2.2}$$

$$H(s) = \frac{\omega_n^2}{1 + 2\xi_d \omega_n s + \omega_n^2} e^{\tau_d s} \tag{2.3}$$

with τ_1, $\tau_2 \in \mathbb{C}_0^+$ the time constants and $\tau_d \in \mathbb{R}^+$ the time delay of the transfer function. In the undamped case, $\omega_n \in \mathbb{R}^+$ is the undamped natural frequency and ξ_d the damping ratio, with $0 \leq \xi_d \leq 1$. For MIMO models, a transfer function is available for each input-output pair. This model is converted to a state-space formulation for use in the model predictive controller.

Polynomial models

Different polynomial model structures, e.g., Auto-Regressive with eXogeneous inputs (ARX), Auto-Regressive Moving-Average with eXogeneous inputs (ARMAX), Output Error (OE) and Box-Jenkins (BJ) have been used to construct discrete-time models. They can all be represented in the following format for the SISO case [34]:

$$\mathbf{A}(z)\mathbf{y}(k) = \frac{\mathbf{B}(z)}{\mathbf{F}(z)}\mathbf{u}(k - n_l) + \frac{\mathbf{C}(z)}{\mathbf{D}(z)}e(k). \tag{2.4}$$

Here, $\mathbf{A}(z)$, $\mathbf{B}(z)$, $\mathbf{C}(z)$, $\mathbf{D}(z)$ and $\mathbf{F}(z)$ are polynomial expressions in the shift operator z^{-1} which shifts samples back in time: $z^{-1}x(k) = x(k-1)$ with

$k \in \mathbb{N}$. The measurements are sampled at time instants $t = kT$, with $T \in \mathbb{R}_0^+$ the discretization time. The order of the polynomial expressions is indicated by respectively n_a, n_b, n_c, n_d and n_f. An additional shift back in time is introduced by n_l in order to incorporate system delays. To obtain the previously mentioned model structures, one or more polynomials are omitted in the general formulation. For instance, for the ARX model, the polynomial expressions $\mathbf{A}(z)$ and $\mathbf{B}(z)$ have to be identified in Equation (2.4). The ARMAX models consist of the polynomials $\mathbf{A}(z)$, $\mathbf{B}(z)$ and $\mathbf{C}(z)$. If only $\mathbf{B}(z)$ and $\mathbf{F}(z)$ are employed, these models are called OE models. For BJ models, all polynomials in Equation (2.4) are populated.

In this dissertation, MISO models are identified. The formulation in Equation (2.4) can be expanded for MISO models. In that case, the polynomials $\mathbf{B}(z)$ and $\mathbf{F}(z)$ have to be identified for each input. The order of the polynomial and shift back can be different for each input [35]. Equation (2.4) becomes for the MISO case:

$$\mathbf{A}(z)\mathbf{y}(k) = \sum_{i=1}^{n_u} \frac{\mathbf{B_i}(z)}{\mathbf{F_i}(z)}\mathbf{u_i}(k - n_{l_i}) + \frac{\mathbf{C}(z)}{\mathbf{D}(z)}e(k), \tag{2.5}$$

with n_u the number of inputs.

Linear state-space models

There exist methods to estimate the controller model directly in a discrete state-space formulation:

$$\begin{aligned} \mathbf{x}(k+1) &= A\mathbf{x}(k) + B\mathbf{u}(k) + K\mathbf{e}(k) \\ \mathbf{y}(k) &= C\mathbf{x}(k) + D\mathbf{u}(k) + \mathbf{e}(k) \end{aligned} \tag{2.6}$$

with A, B, C, D, and K the parameter matrices. The parameter matrices are identified in this dissertation using the subspace identification method [83]. The measurements are sampled at time instants $t = kT$, with $k \in \mathbb{N}$ and T the discretization time.

When using a model for control in this work, discrete state-space formulations are used. Transfer function models and polynomial models are therefore converted to the state-space representation of Equation (2.6).

2.2.2 Least Squares Support Vector Machines

The standard framework for LS-SVM is based on a primal-dual formulation. Given a training data set $\mathcal{D}_n = \{(u_k, y_k) : u_k \in \mathbb{R}^d, y_k \in \mathbb{R}; k = 1, \dots n\}$ of size n:

$$y_k = m(u_k) + \epsilon_k, \qquad k = 1, \dots, n, \tag{2.7}$$

where $\epsilon_k \in \mathbb{R}$ are assumed to be independent and identically distributed zero mean random errors with finite variance. The optimization problem of finding the vector w and $b \in \mathbb{R}$ for regression can be formulated as follows [81]:

$$\min_{w,b,e} \mathcal{J}(w, e) = \frac{1}{2} w^T w + \frac{\gamma}{2} \sum_{k=1}^{n} e_k^2 \tag{2.8}$$

$$\text{subject to :} \quad y_k = w^T \varphi(u_k) + b + e_k, \quad k = 1, \dots, n, \tag{2.9}$$

where $\varphi : \mathbb{R}^d \to \mathbb{R}^{n_h}$ is the feature map to the high dimensional feature space [85], $w, b \in \mathbb{R}$ are the unknown variables and e is the residual. However, w and $\varphi(\cdot)$ do not need to be evaluated explicitly. By using Lagrange multipliers for the optimization problem in Equation (2.8), the following Lagrangian function is obtained:

$$\mathcal{L}(w, b, e; \lambda) = \frac{1}{2} w^T w + \frac{\gamma}{2} \sum_{k=1}^{n} e_k^2 - \sum_{k=1}^{n} \lambda_k (w^T \varphi(u_k) + b + e_k - y_k), \tag{2.10}$$

where λ_k are the Lagrange multipliers. The Karush-Kuhn-Tucker (KKT) first order necessary conditions for optimality are given by:

$$\frac{\partial \mathcal{L}}{\partial w} = 0; \frac{\partial \mathcal{L}}{\partial b} = 0; \frac{\partial \mathcal{L}}{\partial e_k} = 0; \frac{\partial \mathcal{L}}{\partial \lambda_k} = 0. \tag{2.11}$$

After elimination of the variables w and e, the solution is given by the linear system:

$$\left(\begin{array}{c|c} 0 & 1_n^T \\ \hline 1_n & \Omega + \frac{1}{\gamma} I_n \end{array}\right) \left(\frac{b}{\lambda}\right) = \left(\frac{0}{Y}\right), \tag{2.12}$$

with $Y = (y_1, \dots, y_n)^T$, $1_n = (1, \dots, 1)^T$, $\lambda = (\lambda_1, \dots, \lambda_n)^T$ and $\Omega_{kl} = \varphi(u_k)^T \varphi(u_l) = K(u_k, u_l)$, with $K(u_k, u_l)$ a positive definite kernel $(k, l = 1, \dots, n)$. According to Mercer's theorem [62], the resulting LS-SVM model for new inputs becomes:

$$\hat{m}(u^\star) = \sum_{k=1}^{n} \hat{\lambda}_k K(u^\star, u_k) + \hat{b}, \tag{2.13}$$

where K is any positive definite kernel and in this work K is chosen $K = \exp(-\|u_k - u_l\|_2^2/\sigma^2)$. The training of the LS-SVM model involves an optimal selection of the tuning parameters, e.g., σ and γ, which are tuned via a 10-fold cross-validation. More background information on LS-SVM for regression can be found in [12].

2.2.3 Model selection

To compare different models and to derive the correct model order, the Akaike Information Criterion (AIC) [1, 34] is adopted. A lower value of AIC indicates a more accurate model. Here, the AIC for linear models is defined as:

$$\mathrm{AIC} = \log(V) + \frac{2d}{n} \tag{2.14}$$

where V is the loss function, d the number of estimated parameters, and n the number of values in the estimation data set. The loss function V is equal to the residual sum of squares:

$$V = \frac{1}{n}\sum_{1}^{n} \epsilon_i^2. \tag{2.15}$$

The loss function V is proportional with the error between the simulations and the measured variables. The term $\frac{2d}{n}$ corrects this loss function while taking into account the number of parameters in the model and the number of data points. This term will be high in case the number of data points is low and the number of parameters is high. However, if the value of two times the number of parameters is similar to the number of data points, the AIC value is mainly determined by the loss function. For instance, if $2d \approx n$, the model is likely to be overfitted. In such a case, the model describes random noise instead of the relationship between the in- and outputs. To identify this overfit, a correction to AIC is suggested [9, 80]:

$$\mathrm{AICc} = \mathrm{AIC} + \frac{2d(d+1)}{n-d-1} \tag{2.16}$$

In this corrected AIC, the number of estimated parameters is more important than the number of data points, reducing the probability of overfitting.

For LS-SVM models, the number of estimated parameters is replaced by the degrees of freedom or effective number of parameters given by the trace of the smoother matrix. The smoother matrix L is defined as [13]:

$$L = \Omega\left(Z^{-1} - Z^{-1}\frac{J_n}{c}Z^{-1}\right) + \frac{J_n}{c}Z^{-1}. \tag{2.17}$$

with $Z = \Omega + \frac{I_n}{\gamma}$, $c = 1_n^T(\Omega + \frac{I_n}{\gamma})^{-1}1_n$ and J_n is a square matrix with all elements equal to 1. Hence, for an LS-SVM model the number of estimated parameters is the trace of L.

AIC, as well as AICc, need to be as low as possible. The model with the lowest AIC or AICc is expected to be the model that most accurately describes the original data with a minimal number of parameters.

2.2.4 Model validation

To check whether a selected model is valid for a data set different from the estimation data, a FIT measure, identical to the one employed in the Matlab System Identification Toolbox, the sum of squared errors and the mean squared errors are calculated on a new validation data set.

FIT measure

The FIT measure as used in the Matlab System Identification Toolbox [35] is defined as:

$$\text{FIT} = \left(1 - \frac{\|\hat{\mathbf{y}}(k) - \mathbf{y}(k)\|_2}{\|\mathbf{y}(k) - \bar{\mathbf{y}}(k)\|_2}\right) 100\% \tag{2.18}$$

where $\hat{\mathbf{y}}(k)$ is the simulated or predicted output, $\mathbf{y}(k)$ the measured output and $\bar{\mathbf{y}}(k)$ the mean of the measured output. A FIT value of 100% means that the simulation is identical to the measured output. If the simulated value is equal to the mean value of the validation data set, the result is 0%. Values below zero mean that the simulation or prediction is even worse than a constant mean value. For control purposes, this is to be avoided.

Sum of Squared Errors (SSE)

This is one of the possibilities to quantify the size of the error. Comparison on the SSE is only possible for an equal number of data points. With n the number of data points, the sum of squared errors is defined as:

$$\text{SSE} = \sum_{k=1}^{n} (\hat{y}(k) - y(k))^2. \tag{2.19}$$

Mean Squared Error (MSE)

This is another possibility to quantify the size of the error. It is the mean of the squared errors. By taking the mean, this quantification is independent of the number of data points n. The mean squared error is defined as:

$$\text{MSE} = \frac{1}{n}\sum_{k=1}^{n}(\hat{y}(k) - y(k))^2. \tag{2.20}$$

2.3 Model predictive control

The key element in MPC is to repeatedly solve an optimization problem based on available measurements of the current state of the process. The result of the optimization problem is a new set of inputs. Only the first input is applied to the process. For more in-depth information on MPC, the reader is invited to read some excellent books concerning this topic [10, 38, 70, 87]. Below, the basic idea of MPC is explained and the formulation employed in this dissertation is briefly introduced.

2.3.1 Basic idea

Figure 2.1 depicts the basic idea of MPC for a SISO System. The optimal input has to be calculated at the current time k. A set-point or reference trajectory r has been defined beforehand. This is the desired value for the output of the system. The part $r(k|k)$ to $r(k+H_p|k)$ of this reference trajectory r is supplied to the controller and has to be followed as close as possible. The notation $r(k + H_p|k)$ points to the data point $k + H_p$ of r at time step k. H_p is the so called *prediction horizon* of the controller. This is the time the controller can *look into in the future*. The (internal) model in the controller is used to calculate the inputs $\hat{u}(k)$ over the control horizon H_c. The control horizon is lower than or equal to the prediction horizon and it is the interval in which the inputs can differ from each other. The inputs in the interval between the end of the control horizon up to the prediction horizon, are kept constant to the last calculated input value $\hat{u}(H_c)$.

To calculate the appropriate inputs, a *cost function* is defined. This cost function is used to define the objectives the controller has to meet, e.g., the predicted output $\hat{y}(k)$ has to be as close a possible to the reference $r(k)$ in the selected interval $r(k|k)$ to $r(k + H_p|k)$. The optimal input is the input that makes the cost function at the current time k minimal. Only the first

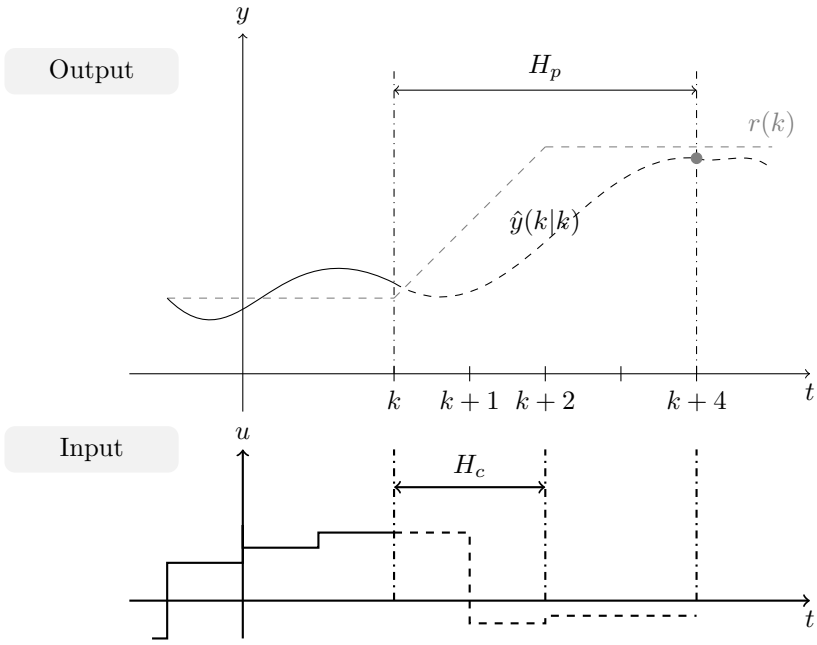


Figure 2.1: The basic idea behind Model Predictive Control.

input is applied to the system. At the next time instance $k+1$, this process in repeated and a new set of optimal inputs is determined. How the optimization is performed, is explained in the following sections.

2.3.2 Model predictive control formulation

Linear Model Predictive Control is well known in literature [10, 38, 70, 87]. The basic formulation for linear MPC used for the practical implementation in this work is given in Appendix A and is summarized below.

A linear, time-invariant discrete-time system is described by:

$$\begin{aligned} \mathbf{x}(k+1) &= A\mathbf{x}(k) + B\mathbf{u}(k) \\ \mathbf{y}(k) &= C\mathbf{x}(k), \end{aligned} \tag{2.21}$$

with $A \in \mathbb{R}^{p\times p}$, $B \in \mathbb{R}^{p\times m}$ and $C \in \mathbb{R}^{q\times p}$. Here m, p and q are the number of inputs, states and outputs, respectively. The objective of the controller is to find the optimal input for this system by means of minimizing a cost

function, a user-defined function taking into account the objective(s) to be reached by the controller. In this work two objectives have been selected. First, the predicted output $\hat{\mathbf{y}}(k)$ should be as close as possible to the reference $\mathbf{y}_{\text{ref}}(k)$ over the prediction horizon. The second objective is to minimize the difference between $\Delta\mathbf{u}(k)$ and its reference $\Delta\mathbf{u}_{\text{ref}}(k)$ over the control horizon. The reference $\Delta\mathbf{u}_{\text{ref}}(k)$ can be used to force the input to track a desired input as close as possible. This leads to the cost function:

$$J = \sum_{i=1}^{H_p} \|\hat{\mathbf{y}}(k+i|k) - \mathbf{y}_{\text{ref}}(k+i|k)\|^2_{W_y} + \sum_{j=0}^{H_c-1} \|\Delta\mathbf{u}(k+j|k) - \Delta\mathbf{u}_{\text{ref}}(k+j|k)\|^2_{W_u} . \tag{2.22}$$

$\mathbf{y}_{ref}$ is the output reference and $\hat{\mathbf{y}}$ is the predicted output. The formulation $\mathbf{y}(k+i|k)$ represents the vector $\mathbf{y}$ on the sample time $k+i$ at calculation time k. $\Delta\mathbf{u}(k+j|k) = \mathbf{u}(k+j|k) - \mathbf{u}(k+j-1|k)$ is the difference between two successive inputs. When $\Delta\mathbf{u}_{\text{ref}}(k+j|k)$ is employed, this reference is such that the future inputs will be as close as possible to a nominal working point of the employed set-up. H_p and H_c (with $H_c \leq H_p$) are respectively the prediction and control horizon of the controller. $W_y \in \mathbb{R}^{q\times q}$ and $W_u \in \mathbb{R}^{m\times m}$ are positive definite weighting matrices.

One of the key elements of MPC is the possibility to handle constraints. For this work only input constraints are taken into account:

$$\mathbf{u}(j) \leq \mathbf{u}_{\text{Max}} \tag{2.23}$$
$$\mathbf{u}(j) \geq \mathbf{u}_{\text{Min}}.$$

Output constraints are omitted in the current study, but can easily be introduced (see Appendix A). The optimization problem can be formulated as the minimization of Equation (2.22), subject to Equations (2.21) and (2.23). In order to solve the optimization problem, it is reformulated by elimination of the states resulting in a Quadratic Program (QP) of the following form:

$$\min_{\boldsymbol{\theta}} J = \frac{1}{2}\boldsymbol{\theta}^T H \boldsymbol{\theta} + \mathbf{g}^T \boldsymbol{\theta} \tag{2.24}$$

$$\text{subject to : } P\boldsymbol{\theta} \leq \boldsymbol{\alpha} \tag{2.25}$$

with Equation (2.24) the quadratic objective function and Equation (2.25) the linear inequality constraints and $\boldsymbol{\theta}$ the decision variables. The decision variables

employed in this dissertation are $\Delta\mathbf{u}(k+j|k)$ that are rearranged into a vector $\Delta\mathbf{U}$.

To reformulate the problem, following steps have been taken. First, the state-space model in Equation (2.21) is rewritten in terms the augmented state $\boldsymbol{\xi}$:

$$\boldsymbol{\xi} = \begin{bmatrix} \Delta\mathbf{x} \\ \mathbf{y} \end{bmatrix} \tag{2.26}$$

with $\Delta\mathbf{x}(k) = \mathbf{x}(k) - \mathbf{x}(k-1)$ and $\mathbf{y}$ the current measured output [87]. The prediction over the prediction horizon is written in matrix formulation and is expressed as:

$$\hat{\mathbf{Y}} = F\boldsymbol{\xi} + \Phi\Delta\mathbf{U}, \tag{2.27}$$

with $\mathbf{Y}$ and $\Delta\mathbf{U}$ column matrices of predicted outputs and the difference of successive inputs, respectively. $\Delta\mathbf{U} \in \mathbb{R}^{mH_c \times 1}$ composed of $\Delta\mathbf{u}(k|k)$ to $\Delta\mathbf{u}(k+H_c-1|k)$. The matrices F and Φ can be found in Equation (A.11) and (A.12). Based on these matrices, the hessian H becomes:

$$H = \Phi^T W_{y_{bd}} \Phi + W_{u_{bd}} \tag{2.28}$$

with $W_{y_{bd}}$ and $W_{u_{bd}}$ block diagonal matrices of W_y and W_u respectively. The hessian can be computed offline as no online information is required. As for the purpose of this work, the aim is to minimize online calculation work, matrices that can be computed in advance will be calculated offline and stored in memory.

The gradient vector $\mathbf{g}$ in Equation (2.24) has to be calculated online. It contains three parts depending on the current state and the reference of the in- and outputs. So, $\mathbf{g}$ has to be calculated according to:

$$\mathbf{g} = G_1\boldsymbol{\xi} - G_2\mathbf{Y}_{\text{ref}} - G_3\Delta\mathbf{U}_{\text{ref}} \tag{2.29}$$

where G_1, G_2 and G_3 are gradient matrices to be calculated according to Appendix A. $\mathbf{Y}_{\text{ref}}$ is an $\mathbb{R}^{qH_p \times 1}$ matrix of the references $\mathbf{y}_{\text{ref}}(k|k)$ to $\mathbf{y}_{\text{ref}}(k+H_p|k)$. The gradient matrices G_1, G_2 and G_3 are constant and are computed offline.

The constraints, i.e., the minimum and maximum admissible values for $\mathbf{u}(k+j|k)$, $j = 0 \dots H_c - 1$, require online information and, hence, are calculated online. To solve the QP, these minimum and maximum admissible values are needed in terms of ΔU. With $\mathbf{u}(k+j|k) = \mathbf{u}(k-1|k) + \sum_{j=0}^{H_c-1} \Delta\mathbf{u}(k+j|k)$, the constraints can be reorganized. $\mathbf{U}_{\text{Max}}$ is a column matrix of H_c times $\mathbf{u}_{\text{Max}} = \mathbf{u}_{\text{Max}} - \mathbf{u}(k-1)$. $\mathbf{U}_{\text{Min}}$ is a column matrix of H_c times $\mathbf{u}_{\text{Min}} = \mathbf{u}_{\text{Min}} - \mathbf{u}(k-1)$. A lower triangular matrix with identity matrices, C_1, is employed to build the

vector $\mathbf{U}$ from $\Delta\mathbf{U}$ (See Appendix A on page 130). Finally, the QP problem to be solved is:

$$\min_{\Delta\mathbf{U}} \frac{1}{2}\Delta\mathbf{U}^T H \Delta\mathbf{U} + \mathbf{g}\Delta\mathbf{U} \tag{2.30}$$

$$\text{subject to :} \mathbf{U}_{\text{Min}} \leq C_1 \Delta\mathbf{U} \leq \mathbf{U}_{\text{Max}} \tag{2.31}$$

The Hildreth algorithm [23], qpOASES [16] and CVXGEN [41] will be used to solve this QP problem.

2.3.3 Solving a QP without constraints

If no constraint is active, the solution to the problem in Equation (2.30) can be calculated in a closed-form:

$$\mathbf{u}_{opt} = -H^{-1}\mathbf{g} \tag{2.32}$$

with the H the hessian and $\mathbf{g}$ the gradient as defined in Appendix A. In such a case, the solution to the problem is easily and rapidly obtained.

2.3.4 Solving a QP with inequality constraints

The constraints of the quadratic problem are expressed in terms of inequalities. A general closed-form solution of the constrained control problem is not possible, unless the set of active constraints is known in advance. Several methods to solve a QP exist. Two classes of methods, often used in online MPC, are *active-set* methods [17, 20] and *interior point* methods [91].

Briefly, an active-set method is an iterative method that solves a sequence of equality-constrained quadratic subproblems. The aim of this method is to predict the active set. This is the set of constraints that are satisfied with equality at the solution of the problem. The conventional active-set method is divided into two phases: the first focuses on feasibility, while the second focuses on optimality. An advantage of active-set methods is that the methods are well-suited for hot starting, where a good estimate of the optimal active set is used to (re-)start the algorithm. This is particularly useful in applications where a sequence of quadratic programs is to be solved [89], e.g., MPC. Examples of active-set implementations are SQOPT [24] and qpAOSES [16].

Interior-point methods compute iteration values that lie in the interior of the feasible region, rather than on the boundary of the feasible region. The method computes and follows a continuous path to the optimal solution. One way of

looking to interior point methods [8, 38] is to minimize the function $V(\boldsymbol{\theta})$ over $\boldsymbol{\theta}$ subject to constraints $P\boldsymbol{\theta} \leq \boldsymbol{\alpha}$. This is reformulated as:

$$f(\boldsymbol{\theta}, \gamma) = \gamma V(\boldsymbol{\theta}) - \underbrace{\sum_i \log(\alpha_i - P_i^T \boldsymbol{\theta})}_{\text{barrier function}} \tag{2.33}$$

with P_i^T the ith row of P and α_i the ith element of $\boldsymbol{\alpha}$. γ is a positive scalar that is continuously increased during the search operation to the optimal solution. The barrier function prevents the search to leave the feasible region. A well known interior-point implementation is IPOPT [86].

In the case where a limited number of combinations of active constraints exist, one can build a look-up tree populated with linear models that each corresponds to a specific combination of active constraints. This look-up tree can be built in advance to speed up calculations when the actual QP needs to be solved. This approach has been used in, e.g., [4, 30, 66, 74]. A drawback of this approach is that the number of possible active constraints has to be small to remain effective.

A model predictive controller controlling a process involves a sequence of QPs to be solved. Traditionally, the QP is solved by an online algorithm. Depending on the formulation of the MPC problem, the QP is condensed[1] (most elements of the hessian are different from zero) or sparse (most elements are zero). In this dissertation, the resulting QP is condensed. Both, active-set and interior-point methods can be used to solve the QP. The solvers employed in this work are qpOASES and the Hildreth quadratic problem algorithm, which are both active-set solvers [16, 87] and CVXGEN, a solver based on an interior-point method [41]. If the QP is sparse, the same methods optimized for sparse matrices, are used.

MPC implementations based on offline calculations of the QP, known as explicit MPC, can be made very fast for small systems. However, the off-line complexity is high. Suboptimal strategies have been followed to reduce the complexity, e.g., [29], but some researchers state that it is hardly possible to use this strategy for large systems [64].

[1] *Condensed* does not refer to the physical phenomenon of *condensation*. A *condensed* QP consist of a *dense* hessian matrix with most elements different form zero. Condensing here involves the reformulation of the QP problem such that a smaller size hessian matrix is obtained, in which most elements are different form zero.

2.4 Hardware and software

For the implementation of the model predictive controller, a top-down path starting with a high-level controller to an industrial controller is followed. This means that there is a gradually decreasing amount of computing power and available memory when changing to a different controller device.

First, a standard personal computer is used simulate the model predictive controller. For the distillation column this PC is also used to control the set-up. A first decrease of computing power is obtained by using a Programmable Automation Controller (PAC). In a last step, a Programmable Logic Controller (PLC) is used to control the different set-ups. For the air heating set-up, only a PAC and a PLC have been used as controller hardware.

2.4.1 Hardware

Positioning the hardware

Figure 2.2 depicts the three different hardware classes employed to implement MPC in this work in view of their computing power and available memory. Typical personal computers (PC) are nowadays equipped with several GB of memory and the CPU runs at a speed from one GHz up to several GHz. CPU speed does not tell the whole story, so the devices are compared based on their Floating Point Operations Per Second (FLOPS). For a Programmable Automation Controller, the available memory and speed is slightly less compared to a PC. Programmable Logic Controller devices have significantly less memory compared to personal computers.

A standard PC is typically equipped with an operating system such as, e.g., Windows and Linux that is not real-time. This means that a task receives all the necessary time to finalize that task. Real-time operation systems have features to force tasks to finish within a given time. In control applications, control actions are to be provided to the system at predefined time instances. To safely use a PC in control applications, software to ensure real-time behavior is required.

In 2001 the ARC advisory group was the first company that used the term Programmable Automation Controller. This is a PLC with enhanced capabilities, or from a different point of view, a PC with real-time capabilities and standard in- and outputs. It is an answer to the demand of the industry for fast, robust devices with open (communication) standards, standard processors, standard programming languages, enhanced logging and visualization.

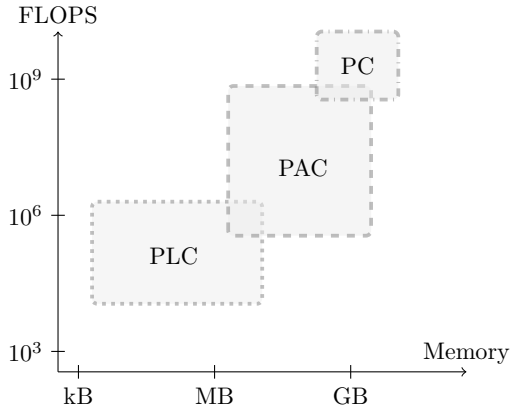


Figure 2.2: Graphical comparison where to situate industrial control hardware, i.e., PLC and PAC to PC in view of FLOPS (Floating Point Operations Per Second) and memory.

A Programmable Logic Controller is a typical industrial device used to automate a process application. *To automate* does not necessarily imply *to control*. This type of device was intentionally created to replace Relay Ladder Logic (RLL). This can still be seen in the oldest, but still frequently used, programming language *Ladder*. This language is graphically very similar to classical relay schemes. Although high-level programming languages exist in the international standard IEC 61131-3, the use of Ladder is still widespread [6, 90].

Practical use of PLCs for control is mainly offered by, e.g., PID and Fuzzy logic controller blocks implemented by the hardware manufacturer [53, 56]. The programmer only needs to supply appropriate parameters to the block according to the manual. An implementation of a model predictive controller is uncommon. After a private communication with a Siemens engineer, only one example in the paper pulp industry (ModPreCon MPC controller in the SIMATIC PCS 7 APC software [57]) could be found for this brand. All other installed MPC controllers are based on the concept of a computer running the MPC software which supplies set-points to a PLC in the field. Not surprisingly, MPC implementations on PLCs reported in research are scarce (see, e.g., [32, 82] for explicit MPC).

A pre-requisite in industry is the use of reliable, easy-to-maintain control hardware. This explains the dominance of PLCs in process applications and automation. The standard programming languages, the robustness of the devices and the long-term hardware support make that PLCs fit these

requirements. However, the long-term hardware support slows down the hardware evolution compared to, e.g., personal computers. By the end of the 90's a cycle of adding all kinds of industrial communication has ended. The current refresh cycle, initiated around 2007, starts to incorporate ethernet network capabilities combined with, e.g., ftp servers, web servers and communication with Human Machine Interface (HMI) panels. This evolution follows the need for simplified logging and enhanced Human Machine Interfacing. Adding these functionalities makes recent PLCs more complex and more powerful.

The defined zones in Figure 2.2 are subject to constant change as the hardware world evolves constantly. Nevertheless, it sets the scene. For PACs, the available memory ranges from 16 MB up to 1 GB and the CPU speed varies in a wide range from approximately $1 \cdot 10^5$ to $1 \cdot 10^8$ FLOPS. Local storage of a few GB is not uncommon. PACs have similar specifications to state-of-the-art PCs of a decade ago. The high availability of memory and computing power is an eyecatcher compared to PLCs. However, these complex devices need a real-time operation system such as, e.g., VxWorks, embedded Linux or embedded Windows, which all need a lot of computing and memory resources. Nevertheless, these devices are highly flexible and can handle a large amount of data and in- and outputs. The real-time operating system is also required to ensure that the in- and outputs of the device are working properly. PLCs are typically equipped with 64 kB up to 8 MB of memory (see the market overview in the following subsection). The maximum supplied memory for a PLC is 64 MB, although this number fluctuates throughout the different manufacturers and cannot be considered as a hard limit. The CPU speed is normally indicated in micro-/nanosecond per floating point operation. As PLCs have been used to replace relays, a significant amount of memory and instructions are bit-based. Bit operations are at least ten times faster than floating point operations, so this difference is always indicated in the manual. An example of a recent (introduced in 2010) mainstream PLC is a Siemens S7-1200 [58] CPU 1214C. It has 50 kB of working memory (8 kB is dedicated bit memory) and 2 MB of local storage. Bit operations take 0.1 μs per instruction and a *real math* operation takes 18 μs/instruction. For 10 year old CPUs these values may easily be multiplied by 1000 [55].

(Partial) PAC and PLC market overview

There are many companies that have PLC and/or PAC devices in their portfolio. Some of them cover a part of this market segment, others cover all. A selection of companies selling these products are ABB, B&R Industrial Automation, Beckhoff Automation, Bosch Rexroth, Eaton/Moeller,

Fuji Electric, GE Intelligent Platforms, Hitachi/Hitachi Industrial Equipment Systems, IDEC, Koyo Electronics Industries, LS Industrial Systems (LSIS), Mitsubishi Electric, OMRON, Panasonic Electric Works, Rockwell Automation, Saia-Burgess Controls, Schneider Electric, Sharp, Siemens, Toshiba, VIPA, Yokogawa.

Tables 2.1 and 2.2 present a selected number of properties useful to compare PLCs and PACs in this dissertation. Only a couple of the above brands are selected to give an impression on some PLC and PAC features. The tables focus on the speed and memory characteristics. The speed of a PLC is indicated in different ways. Manuals focus mostly on bit operations per second. Only occasionally, the floating point operations per second is indicated, or if none of the former properties are present, a minimum cycle time is given. The latter is only a vague speed indication. Cycle times for a PLC depend on the number of instructions, and thus on the program. As a program is often a mix of bit and floating point operations, defining the floating point calculation capacity based on the cycle time is uncertain.

Programmable Automation Controllers exist in wide varieties. Table 2.1 lists a selection for a couple of brands. It can be noticed that manufacturers supply different information depending on the market position of their devices. Those who claim a PAC to be a very powerful PLC also provide information concerning bit operations, those claiming a PAC is a rugged PC with I/O possibilities provide processor speed. In Table 2.1, the PACs of GE and Phoenix contact can be considered to be fast PLCs, The Siemens and National Instruments PACs are rugged PCs, both running a (third party) real-time operation system, Windows CE and VxWorks respectively. An extra OS requires more memory, which is seen for the corresponding devices in Table 2.1.

For PLCs, the processing speed of floating point numbers is the most important feature in this dissertation. When provided by the manufacturer, this number is indicated in Table 2.2. It can be noticed that floating point arithmetics is always slower than bit operations per second, but it is also clear that it is not possible to calculate a form of multiplication factor to calculate the floating point capacity of a device based on the number of bit operations per second. Conclusions can hardly be taken about the relationship between bit operations and floating point operations per second. One conclusion that can be drawn is that there is a wide variety of processing speed available for PLCs, e.g., from 4 μs to almost 2 ms for floating point arithmetic for the selected brands in the table. The mentioned CPU in the table is always the fastest CPU available in that PLC series.

The available memory in each PLC series also varies largely. The main/work memory column contains the sum of the program and data memory for each

Table 2.1: Overview of the PAC market for a selected number of brands.

Company	Controller Series	CPU	Speed: Execution time (μs) Bit operations	Floating point	Processor Speed	Memory Main/work	Load/mass
- Beckhoff - [44, 45, 46, 47]:							
	embedded PC	CX1000			266 MHz	128 MB	64 MB + 4 GB
	embedded PC	CX1010			500 MHz	128 MB	64 MB + 4 GB
	embedded PC	CX2040			2.1 GHz	4 GB	8 GB
	embedded PC	CX5020			1.6 GHz	512 MB	64 MB
- GE - [49]:							
	PACsystems	RXi3 CPU315	0.047			64 MB	64 MB
	PACsystems	RXi7 IC698CPE040				64 MB	64 MB
- Phoenix Contact - [52]							
	400	S-MAX 400 CE PN MC	0.002	0.007	1.8 GHz	24 MB	256 MB
- National Instruments- [51]							
	CompactRio	cRIO-9024			800 MHz	512 MB	4 GB
- Siemens - [59]:							
	S7-mEC	EC31	0.004			1 GB	4 GB

Table 2.2: Overview of the PLC market for a selected number of brands.

Company	Controller Series	CPU	Speed: Execution time (μs) Bit operations	Floating point	Scan time	Memory Main/work	Load/mass
- Eaton - [48]							
	ELC Series	ELC-PV28NNDR/DT	0.24			20 kB	
	XC Series	XC202	0.025			4 MB	
- GE - [49]							
	VersaMax	Micro 64			1.1	24 kB	32 kB
	series 90-30	CPU 374			0.15	240 kB	4 MB
	series 90-70	IC697CPM790	0.4			1 MB	
- Phoenix Contact - [52]							
	100	ILC 190	0.09	1.3		1024 kB	1024 kB
	300	ILC 390 PN	0.006	0.2		2 MB	4 MB
- Saia - [71]							
	PCD2	M5xxx	1.5	0.9		1 MB	4 MB
	PCD3	Mxx60	0.1	0.3		1 MB	4 MB
- Siemens - [54, 55, 58, 59, 60]							
	S7-1200	1215C	0.08	2.3		50 kB	2 MB
	S7-300	CPU 319	0.004	0.04		2 MB[a]	8 MB
	S7-400	CPU 417	0.018	0.054		30 MB	64 MB
	S7-1500	CPU1516	0.01	0.064		6 MB[b]	2 GB

[a] DB max. 64 kB
[b] DB max. 512 kB

PLC. This memory ranges from several kilobyte up to tens of megabytes. If extra memory can be added by, e.g., SD-cards, this is mentioned in the load/mass memory column. This memory ranges from 1 megabyte up to several megabytes. However, more recent PLC series, e.g., Siemens S7-1500, support SD-cards with storage in the gigabyte range.

2.4.2 Employed hardware

In the next paragraphs an overview is given of the hardware employed in this dissertation.

Personal Computer

As personal computers are common and much used today, these devices do not need much explanation. The employed PCs and laptops all have a memory between 2 and 8 GB and a CPU running at a speed close to 3 GHz. During this dissertation the employed Intel processor family changed from core duo to core-i5 processors.

Programmable Automation Controller

As a PAC, a National Instruments CompactRIO has been used (Figure 2.3a). This PAC controller consists of a NI cRio–9024 Real-Time Controller (800 MHz CPU, 512 MB of memory) [51], a cRIO–9114 Reconfigurable Chassis, a NI 9265 Analogue Current Output module and a NI 9217 RTD 24-Bit Analogue Input Module. The real-time controller is programmed with LabVIEW and is able to run a software library compiled from C/C++ code via a *call library function* (see Figure 2.4). The library is compiled for the VxWorks 6.3 operating system with the gcc-compiler version 3.4.4.

The second experimental set-up used is this dissertation, the pilot-scale distillation column, is equipped with an additional second PAC system: a Compact FieldPoint, which is the predecessor of the CompactRIO. This device consists of a NI cFP-2020 Real-Time Controller, a NI cFP-AIO-610 8-Channel Combination Analog Input/Output Module, NI cFP-RTD-124 8-Channel 4-Wire RTD Temperature Module, NI cFP-AI-110 8-Channel Analog Voltage and Current Input Module and NI cFP-AIO-600 8-Channel Combination Analog Input/Analog Output Module. As this device is not powerful enough to run the controller program and certainly not the controller program with an incorporated model predictive controller, it is only used as a service-hatch to



(a) A CompactRIO **(b)** A Compact FieldPoint

Figure 2.3: The two PAC devices used in this dissertation.



Figure 2.4: The graphical symbol of the LabVIEW Call Library function.

pass the measured inputs to the controller and desired outputs to the actuators of the column.

Programmable Logic Controller

As PLC, a Siemens CPU319-3DP/PN PLC is used (Figure 2.5). The base memory of this CPU is increased to the maximum allowed 8 MB. This CPU is the fastest Siemens S7-300 CPU. It takes 40 ns for one floating-point operation [55]. An additional SM334 analogue card is employed to connect to the solid state relays and DC drive for the air heating set-up. The Siemens CPU is programmed using the Step 7 Professional 2010 software. To code the problem, the *Structured Control Language* (S7-SCL) is used. This programming language corresponds to Structured Text (STL) in the standard IEC 61131-3.



Figure 2.5: A Siemens CPU319-3DP/PN PLC.

2.4.3 Software

Advanced hardware needs appropriate software. The following software packages have been used: Step 7 Professional 2010 from Siemens, LabVIEW from National Instruments and a GCC C/C++ compiler. To simulate the controller and build models, the System Identification Toolbox, the LS-SVM Toolbox 1.7 and the Control Toolbox in Matlab are used.

Step 7 Professional 2010

This program is the standard editor and device manager for Siemens S7-300/400 PLCs. A screen shot is given in Figure 2.6. Step 7 Professional 2010 is used to program the S7-319 CPU. The selected programming language is S7-SCL, which corresponds to the Structured Text (STL), a textual programming language for PLC defined in IEC 61131-3. An example of S7-SCL code is given below.

```
FUNCTION FC1:  VOID

VAR
   Array_Variable:  ARRAY[1..30] OF INT;
   X_Var:  INT;
END_VAR

BEGIN
   X_Var:= 1;
   X_Var:= Array_Variable[X_Var];
END
```

LabVIEW

LabVIEW is a graphical programming language. Measurement and automation hardware from National Instruments and other third party vendors can be connected and programmed with LabVIEW. The combination of hard- and software makes it a powerful combination for controlling a set-up. The LabVIEW programming concept is based on a Virtual Instrument (VI). A graphical user interface is composed of a front panel on a block diagram (Figure 2.7). These blocks can be graphically connected to a user program. If necessary the program is connected to the real world by means of a variety of devices ranging from low budget USB Data acquisition, PACs to high-end measurement devices.

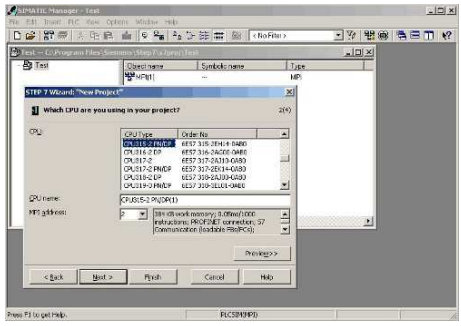


Figure 2.6: Step 7 Professional 2010.

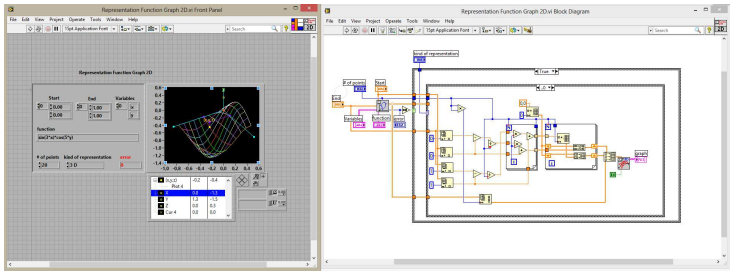


Figure 2.7: LabVIEW programming environment. On the left, a front panel is presented. On the right the block diagram is depicted.

GCC

GCC, the GNU Compiler Collection is a freeware compiler. National Instruments provides a copy with the correct programming scripts to compile working C and C++ libraries for the VxWorks 6.3 Operating system on the CompactRIO. The compiler, with version number 3.4.4, has been used to create libraries containing the different QP solvers used by the model predictive controller. The libraries exchange data with the LabVIEW user program by means of a *call library function*.

Matlab

Matlab has been used to investigate, test and debug the MPC controller and to identify the different models. The most important toolboxes that have been used are:

System Identification Toolbox. This toolbox [35] has been used to identify the different linear models.

Control Toolbox. This toolbox [5] has been exploited for simulation and practical experiments on the distillation column.

LS-SVM Toolbox 1.7. This toolbox [14] has been employed to identify the nonlinear LS-SVM model.

2.4.4 Software to solve a QP

In order to solve the QP problem, several algorithms and software tools have been used. The Hildreth QP algorithm, CVXGEN and qpOASES are described below. Although not QP solvers on their own, the built-in LabVIEW MPC controller and Matlab MPC controller are equipped with a QP solver and are also described in this section.

The Hildreth QP algorithm

The Hildreth algorithm has been chosen for its limited number of code lines which makes it easy to implement. It has been written based on [87] in C for the PAC and in S7-SCL for the PLC. This algorithm calculates the solution in two steps. First, the unconstrained solution is calculated based on Equation (2.32) and if no constraints are violated, this solution is adopted. If a constraint is violated, a constrained QP is solved. The solution of the QP is then passed to the inputs of the set-up. For more information about the solution, see [23, 87, 37]. If a solution to the QP could not be found, the unconstrained solution is compared to the constraint. If a constraint is violated, that entry of the unconstraint solution is limited to the constraint.

If a QP needs to be solved, the Hildreth programming procedure identifies the active constraints via Lagrange multipliers (or the dual variables). Although it is still a quadratic programming problem on the dual variables, the constraints are much simplified, so a simple iterative procedure was used to obtain the optimal solution of the multipliers. This iterative solution does not involve matrix inversion, so in the situation of conflict constraints, the algorithm still delivers a compromised, sub-optimal solution without being numerically unstable. This is particularly important in the real-time implementation of the predictive control system. An implementation of this procedure on devices with no provisions for matrix implementations is easy as no matrix operations, e.g., calculation of the inverse of a matrix, are required.

The CVXGEN QP algorithm

CVXGEN is a web interface (http://www.cvxgen.com) to generate fast, custom C-code for small, QP-representable convex optimization problems, using an online interface with no software installation [41]. With minimal effort, a mathematical problem description can be turned into a high speed solver. The generated code is C-code that should run on any device supporting this programming language. It works best for small problems, where the final KKT-matrix has up to 4000 non-zero matrix entries. CVXGEN does not work well for larger problems due to the choice of explicit factorization in the generated code, i.e. the decomposition into a product of elements [41]. The mathematical representation of the QP problem is presented to the web interface and the generated code is compiled for the VxWorks operating system of the CompactRIO. Similar to the Hildreth algorithm, the unconstrained solution, bounded to the constraints, is employed if a solution to the QP cannot be found.

The qpOASES QP algorithm

qpOASES is an open-source C++ implementation of the recently proposed online active-set strategy [16]. It builds on the idea that the optimal sets of active constraints do not differ much from one QP to the next. At each sampling instant, it starts from the optimal solution of the previous QP and follows a homotopy[2] path towards the solution of the current QP. Along this path, constraints may become active or inactive as in any active-set QP solver and the internal matrix factorizations are adapted accordingly. While moving along the homotopy path, the online active-set strategy delivers sub-optimal solutions in a transparent way. Therefore, such sub-optimal feedback can be reasonably passed to the process in case the maximum number of iterations is reached as the solution always satisfies the constraints and has moved closer to the optimal solution.

A simplified version of qpOASES has been translated to S7-SCL and was used for controlling the heating device and hardware-in-the-loop experiments for the distillation column set-up. Note that the simplified implementation does not allow for hot starting the QP solution and is not fully optimized for

[2] *Homotopy* strategies are often exploited in applied mathematics for the solution of, e.g., nonlinear systems. The idea is to gradually evolve from a known solution to the desired unknown solution by making the problem dependent on a parameter (often called *homotopy parameter*) and solving a series of problems for increasing/decreasing parameter values, while initializing each time the next problem with the solution of the previous one. The series of solutions that is obtained constitutes the so-called *homotopy path*.

speed. Moreover, it only handles bounds on the control inputs but no general constraints. On the PAC, the plain ANSI C implementation of qpOASES has been used. Although the full version of qpOASES is perfectly suited for hot starting, this feature is not used. Furthermore, based on the knowledge that a solution is found in one step if no constraints are active, the algorithm is only used with cold starts. This makes it possible to start the search for a solution with offline computed matrices. On the other hand, qpOASES will most probably benefit from hot starting if constraints are active as the number of required iterations decreases.

The LabVIEW QP solver inside the built-in MPC controller

The LabVIEW programming language contains a set of blocks making it possible to build a model predictive controller [50]. This controller has only been used on the CompactRIO device as it is not possible to use these blocks on a PLC. In the blocks actually calculating the next input, an (undocumented) QP solver solves the quadratic program.

The Matlab QP solver inside the built-in MPC controller

Matlab contains a toolbox for model predictive control [5]. The Matlab MPC controller is equipped with a *Specialized model predictive control quadratic programming (QP) solver optimized for speed, efficiency, and robustness* according to the Mathworks data sheet. According the manual this is the KWIK algorithm [72].

2.5 Controller implementation features

Every device has its own characteristics. The programmer needs to be aware of these to create a properly working program. This section presents how the QP solvers have been implemented and what the consequences are of some of the typical features of the employed PAC and PLC devices.

2.5.1 Model predictive controller implementation

To compute a new input for the process, the sequence of actions presented in Algorithm 1, is followed. Constant matrices are computed in advance and the reference trajectory for the output is selected. This sequence of steps to solve

the MPC problem has been used for both controller devices and the employed QP algorithms.

```
Offline: Calculate H, G_1, G_2 and G_3.
Online: Start PLC or PAC.
Store all precomputed matrices in working memory, together with the reference for inputs
and output.
while CPU is running do
    if controller interval passed since last call then
        Scale the in- and outputs.
        Calculate current state.
        Calculate g, U_min and U_max.
        case Hildreth algorithm
            Calculate the unconstrained inputs of the process.
            if unconstrained inputs violate constraints then
                while maximum number of iterations is not reached and solution not
                found do
                    Solve one iteration of the QP.
                end
                if maximum number of iterations is reached then
                    Use unconstrained solution, but inputs violating a constraint are
                    limited to that constraint.
                end
            end
        end
        case qpOASES
            while maximum number of iterations is not reached and solution not found
            do
                Solve one iteration of the QP.
            end
            if maximum number of iterations is reached then
                Use last sub-optimal solution.
            end
        end
        case CVXGEN (Only on the PAC)
            while maximum numbers of iterations is not reached and solution not found
            do
                Solve one iteration of the QP.
            end
            if maximum numbers of iterations is reached then
                Use unconstrained solution with inputs violating the constraint limited to
                the constraint.
            end
        end
        Scale and apply the calculated inputs to the system
    end
end
```

Algorithm 1: Steps to compute the inputs on the experimental set-up for the PAC and PLC.

2.5.2 Features of the implementation on the PAC

The practical limitations while using PAC and PLC controllers is different. Programmable automation controllers employ a Real-Time Operations System (RTOS). It is an extra layer between the user code and the hardware. Consequently, more resources are to be provided to run both this RTOS and the user code. On the other hand, any operating system handles all basic tasks in a more user friendly way, e.g., file system operations, parallelization of user programs and memory management. The CompactRIO runs VxWorks as its RTOS. A compiler exists to convert (existing) C/C++ code. All implemented QP solvers are originally written in C or C++ and are converted into a library, supported by the RTOS.

During run-time, two programs run in parallel on the distillation column. The first LabVIEW program, further called *Human Machine Interface Virtual Instrument* or *HMI VI*, performs the data exchange between the program and the in- and output cards of the Compact Fieldpoint, the scaling of the in- and outputs, the visualization of the process variables and logging of the data. De PI-controllers are also part of this program. A second LabVIEW program, further called *Model Predictive Control Virtual Instrument* or *MPC VI*, performs the preparative calculations for the QP, e.g., the estimation of the state and selection of the current reference. The QP algorithm is located in a separated library, called from within the MPC VI. The HMI VI runs at a rate of 10 Hz. The MPC VI is called every minute. For the air heating set-up, the HMI VI and MPC VI are merged into one program. This program calls the QP library every second.

Although of no importance for this work, but worthwhile mentioning when using QP solvers, is that a library loaded from within a LabVIEW program is removed from memory as soon as the integrated function is finished. If one wants to keep data for later use, the data must be passed to the LabVIEW program. Consequently, in case hot starting is wanted, the necessary information has to be passed to the LabVIEW program. For the available qpOASES implementation, hot starting is possible, but the hot starting routines store their data in memory currently not accessible for the programmer. To use hot start in conjunction with the LabVIEW call library function, an extra function to collect the info needed for hot starting has to be added in the qpOASES source code. The available working memory of the employed PAC is 512 MB. Uploading all necessary files and the already available OS-files fills approximately 30% of the 512 MB.

2.5.3 Features of the implementation on the PLC

No compiler exists to transform C/C++ source code to a running binary on a Siemens PLC. Therefore, the C/C++ code has to be translated into S7-SCL (STL). Although possible, this is a time consuming step. In this dissertation, the qpOASES and Hildreth solver are translated to S7-SCL. The qpOASES solver is translated without the hot starting possibilities and the general constraint handling code. Instead, only the part that handles bounds is translated. CVXGEN cannot generate STL code and as a manual translation of the generated code is impossible, it is not used.

To calculate the appropriate inputs of the system and solve the QP, a number of built-in *function blocks* (FB) and *organization blocks* (OB) are programmed. Organization blocks are built-in functions called by hardware interrupts. Function blocks are user defined functions with corresponding data stored in a data block (DB) with the same number. Figure 2.8 depicts the order in which these blocks are called.

Every function block has a corresponding data block. A data block cannot exceed 64 kB on the Siemens S7-319 CPU. The editor and compiler force some more limitations. Initialization of a variable must be on a line not exceeding 255 characters. This is not a big issue, but only awkward when initializing matrices. More important is the limitation that an array initialized with unique elements, cannot exceed 255 elements. Strangely enough, an array can be as large as desired as long as it fits into the 64 kB of available memory. To overcome this limitation, each array containing more than 255 elements has to be filled during runtime (e.g., in OB100) by arrays of maximum 255 elements. As can easily be understood, this situation is very unpractical. Linked with this situation is the need for absolute addressing if a function needs data in a data block different from its own data block. For example, to initialize matrices containing more than 255 elements, OB100 writes data to DB2. This operation requires the fault sensitive absolute addressing. The employed FBs and OBs to code the program for experiments on both the air heating set-up and for hardware-in-the-loop (HIL) experiments on the pilot-scale distillation column are presented below. A graphical representation of the order of calling is given in Figure 2.8.

OB100 - Cold Start. This block is called once when the controller is started. It calls function FB1 and FB5, which are used to initialize the precomputed matrices containing more than 255 elements which are stored in DB2 linked to FB2. This procedure is followed to overcome the limitation that an array cannot exceed 255 elements at compilation time. In the case a matrix needs to contain more than 255 elements, more arrays are combined in this block at runtime into a combined array. For

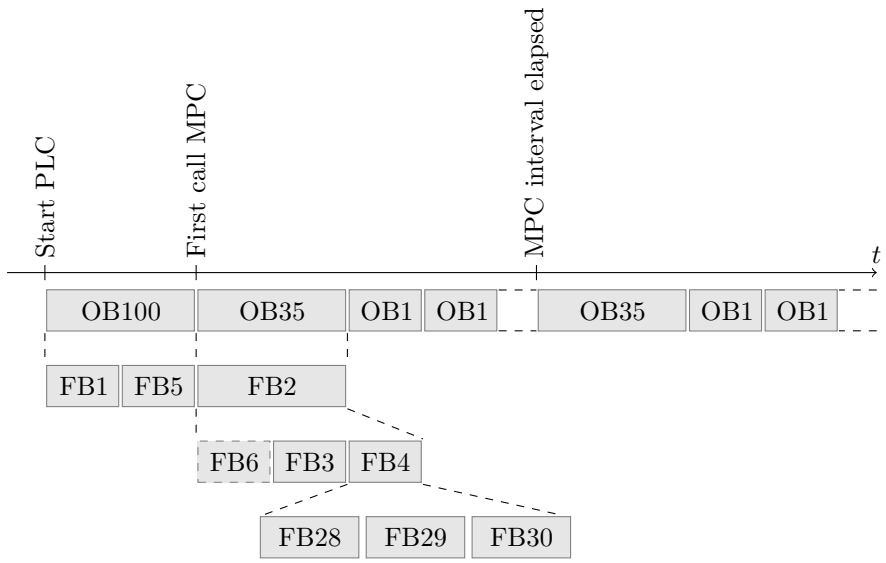


Figure 2.8: Schematic overview of the different employed organization and function blocks in the PLC.

the experiments on the air heating set-up, only matrix G_2 is initialized in this function. For the distillation column HIL experiments, the hessian H and matrices G_1, G_2 and G_3 are initialized with OB100.

OB1 - Main loop. This loop is started as soon as OB100 is finished. When this function finishes, it restarts. This loop is used to program standard tasks of the PLC. In this experiment, it is not used. It will run during the idle time of the CPU between two OB35 calls.

OB35 - Timed loop. This timed organization block is called every second. It contains the necessary code to read the current inputs. This information is scaled and employed to calculate the current state (FB3). Together with the reference for the in- and outputs, the state is used to update vector **g** (FB2). Now, the QP is solved and the scaled solution will be passed to the outputs of the PLC.

FB1 - Initialize large matrices for the QP. This block builds the G_1, G_2 and G_3 matrices to calculate the gradient of the QP from a set of matrices of maximum size of 255. For the air heating set-up, only G_2 is built by this function.

FB2 - Main program. This function scales the in- and outputs, calls the estimator block, calls the QP solver itself and finally presents the calculated inputs to the system.

FB3 - State estimator. This block estimates the current state to be used to calculate the current gradient for the QP solver.

FB4 - QP solver. This block calculates the input of the system. For the qpOASES algorithm, a number of additional function blocks are called. The block FB29 requires the most memory and, hence, determines the practical limits. For the Hildreth algorithm, FB4 is populated with the matrices to solve the QP. For a hessian of size 40×40, the total required memory is 8 matrices of 1600 real elements, plus 7 real numbers, two vectors of 40 and two vectors of 80 real numbers. This results in 53124 bytes needed. Increasing to a hessian of 44×44 elements exceeds the 64 kB limit of the FB. It has to be noticed that based on symmetry 4 matrices can be eliminated. This makes it possible to increase the size of the hessian to 64×64 elements which can be considered as the maximum practical limit for using the Hildreth algorithm in a PLC. Surpassing this limit is possible by storing matrices in different function blocks, but as absolute addressing is required, this is very fault sensitive and has to be avoided.

FB5 - Initialize other large matrices. The hessian is built up from a set of matrices of maximum size of 255 elements. For the experiments with the heating set-up, this block has been merged with the FB1 block.

FB6 - Column model. For hardware-in-the-loop experiments on the PLC, the state-space model of the column is implemented in this block. This is not used for the air heating set-up.

FB28 - Calculate relative distance. This function is only needed for the qpOASES algorithm and calculates the relative distance to the final solution [16].

FB29 - QR and Cholesky decomposition. This function is only needed for the qpOASES algorithm and calculates the QR decomposition based on Givens rotations and the Cholesky decomposition [21]. Calculation of these decompositions needs 12 matrices of the size of the hessian. For the heating set-up, the hessian consists out of 196 elements and the limit of 64 kB for a FB is still far away. For the pilot-scale distillation column, a hessian with 1600 elements is constructed. 11 matrices of 1600 elements violate the 64 kB limit. However, reusing temporary matrices employed in the QR decomposition for the Cholesky decomposition, reduces the number of required matrices so experiments can be performed.

In the current implementation, 8 matrices of 1600 elements are needed. This feature poses a strong practical constraint on using MPC based on qpOASES on a PLC. The total required memory in the employed formulation for hardware-in-the-loop experiments is: 8 matrices of 1600 real elements plus 1 vector of 40 real elements and 15 real numbers which makes in total 51420 required bytes. Increasing the size of the hessian to 44×44, results in 62188 bytes. This is the absolute maximum to fit the QR and Cholesky decomposition into one function block. Separating both QR and Cholesky decomposition is not worthwhile, as the latter needs no extra memory to be allocated in the current implementation.

FB30 - Matrix inversion. This function is only needed for the qpOASES algorithm. This block calculates the inverse of a lower triangular matrix by substitution.

2.6 Conclusion

In this chapter, the black-box model structures used further in this dissertation are briefly introduced. A model is needed to formulate a model predictive controller. Next, the formulation of this controller that is implemented on a programmable automation controller and programmable logic controller is given. Further, an overview of commonly used QP solvers is given as the heart of the presented MPC formulation is a quadratic program. The actual employed QP solvers are introduced and with it, the implementation of MPC on the different devices is explained. Finally, the characteristic features of the different employed devices is presented together with a non-exhaustive market overview.

Chapter 3

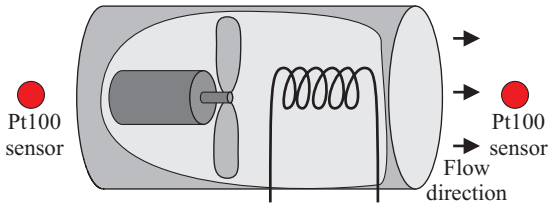
CASE STUDY I: A Small-Scale Air Heating Set-Up

This chapter includes the complete cycle of identification, simulation, parameter selection and implementation of a model predictive controller on embedded devices for a small experimental set-up. This set-up has been selected to safely and quickly investigate different conditions, devices and QP algorithms for possible future use on the pilot-scale distillation column.

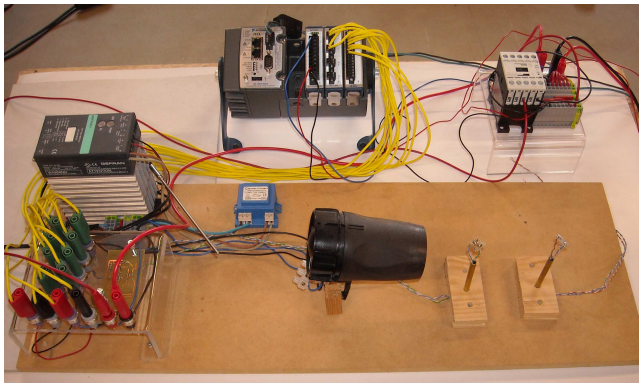
First, the set-up will be introduced. In the second section, a model for control will be identified based on system identification. The next section describes settings and parameters of the controller. This is followed by a section with simulations and hardware-in-the-loop experiments for the different MPC approaches employed in this dissertation. The following two sections elaborate the results obtained from experiments on the set-up for the PAC and PLC devices. The last section presents the conclusions.

3.1 Air heating set-up

The air heating set-up presented in Figure 3.1 consists of a resistor and a fan that can be independently manipulated. The fan is driven by a 24 V DC motor and the resistor can dissipate nominally 1400 W of heat. The heating



(a) Schematic overview of the set-up.



(b) A photo of the set-up.

Figure 3.1: The air heating set-up.

power delivered by the resistor can be adapted by a solid state relay with analogue control (Gefran GTT 25 A 480 V AC - analogue control voltage 0-10 V). The fan is controlled by a custom made DC drive based on a Texas Instruments DRV102T chip and directed by an analogue control voltage of 0-10 V. Temperature sensors measure the ambient temperature and the temperature of the heated air as indicated in Figure 3.1a. Both sensors are of the PT100 type class B. These sensors were chosen for their short time response due to their low thermal mass as they are not embedded in a enclosure.

3.2 Model identification

The Matlab System Identification Toolbox [35] is used to identify the two inputs (fan speed and resistor power) and single output (temperature) linear black-box model. A linear, first order, continuous-time transfer function (Equation (2.1)) is fitted for each of the inputs to the output. This model is called *P1D*.

Afterwards, this model is discretized and converted to a state-space formulation as in Equation (2.6). This resulting model is called *P1DSS*.

The excitation signal of the identification experiment is a multisine with frequencies within the range 0.01 - 0.002 Hz. After subtraction of the ambient temperature from the measured temperature, the data set is detrended, normalized and divided into an estimation and validation data set. Each set contains 250 data points. To determine the model quality, the FIT measure and the *mean squared error* (MSE) are calculated.

The final identified P1D model is:

$$T_{\mathrm{n}} = \begin{bmatrix} \frac{-0.98(\pm 0.03)}{1+5.17(\pm 0.39)s} e^{-1.34(\pm 0.22)s} & \frac{0.83(\pm 0.03)}{1+7.16(\pm 0.51)s} e^{-1.53(\pm 0.25)s} \end{bmatrix} \begin{bmatrix} u_{\mathrm{Fan,n}} \\ u_{\mathrm{Power,n}} \end{bmatrix} \tag{3.1}$$

where the index n indicates a normalized variable. $u_{\mathrm{Fan,n}}$ and $u_{\mathrm{Power,n}}$ are respectively the normalized and detrended actuator voltage for the motor drive and the solid state relay of the resistor. T_n is the normalized temperature. Between brackets, the standard deviation is indicated for each parameter. The predicted temperature in degrees Celsius can be calculated with Equation (3.2):

$$\begin{aligned} T_{pred}[^{\circ}\mathrm{C}] &= \sigma_{\mathrm{detrending}} T_n + \mu_{\mathrm{detrending}} + T_{\mathrm{environment}}[^{\circ}\mathrm{C}] \\ &= 7.13 T_n + 24.42 + T_{\mathrm{environment}}[^{\circ}\mathrm{C}] \end{aligned} \tag{3.2}$$

with $\sigma_{\mathrm{detrending}}$ and $\mu_{\mathrm{detrending}}$ the standard deviation and mean value of the temperature of the excitation experiment, respectively. The ambient temperature has to be added as it has been subtracted before detrending in order to make the model independent of the ambient temperature. For the inputs, a zero input corresponds to 5 V. Conversion to a state-space model of the P1D model with a discretization interval of one second, results in a discrete state-space model of order 4:

$$\begin{aligned} \mathbf{x}(k+1) = & \begin{bmatrix} 0.906 & 0.205 & 0.007 & -0.008 \\ -0.200 & 0.293 & -0.372 & 0.227 \\ -0.017 & 0.081 & 0.682 & 0.492 \\ -0.013 & -0.282 & 0.303 & -0.187 \end{bmatrix} \mathbf{x}(k) \\ & + \begin{bmatrix} -0.299 & 0.209 \\ -0.282 & 0.229 \\ -0.106 & -0.11 \\ -0.020 & 0.068 \end{bmatrix} \begin{bmatrix} u_{\mathrm{Fan,n}}(k) \\ u_{\mathrm{Power,n}}(k-1) \end{bmatrix} \\ \mathbf{y}(k) = & \begin{bmatrix} 0.365 & -0.371 & -0.050 & 0.041 \end{bmatrix} \mathbf{x}(k) \end{aligned} \tag{3.3}$$

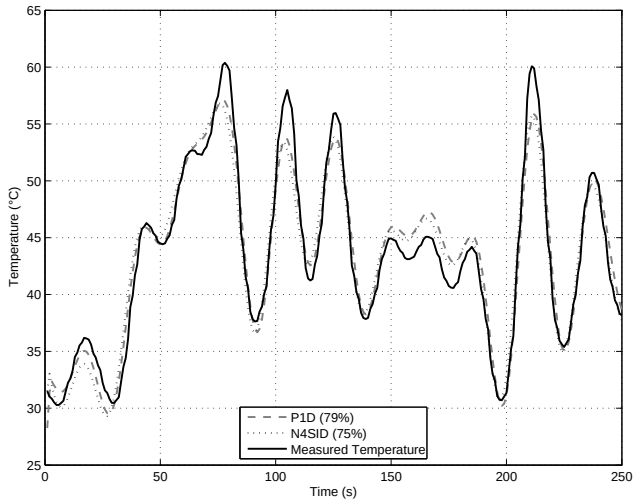


Figure 3.2: Validation of the P1D and N4SID models on multisine validation data.

This model is controllable, observable and stable. The validation plot of the P1D model is depicted in Figure 3.2. The numerical FIT value is 79% and it is identical for both the P1D and P1DSS models. The MSE is indicated in Table 3.1.

Table 3.1: A quality measure for the P1D, P1DSS and N4SID model by means of the FIT and MSE value. The Akaike Information Criterion (AIC) can be used to select the best model.

	P1D	P1DSS	N4SID
FIT	79%	79%	75 %
MSE	0.051	0.049	0.070
AIC	-3.16	-2.97	-2.79

A different approach to obtain the model, namely direct subspace identification, leads to a discrete state-space model of order one. The resulting model is called N4SID and is presented in Equation (3.4). Notice the time shift of one time step for the resistor power. Incorporating this shift into the model increases

the order to two.

$$x(k+1) = \begin{bmatrix}0.847\end{bmatrix} x(k) + \begin{bmatrix}-0.021 & 0.016\end{bmatrix} \begin{bmatrix} u_{\text{Fan,n}}(k) \\ u_{\text{Power,n}}(k-1) \end{bmatrix} \quad (3.4)$$

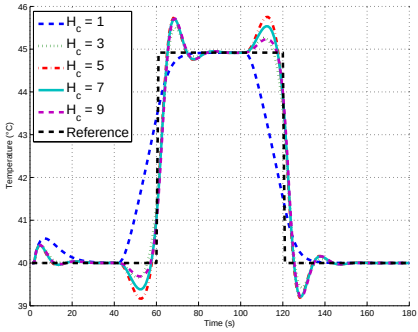
$$y(k) = \begin{bmatrix}7.417\end{bmatrix} x(k) \quad (3.5)$$

The validation plot is given in Figure 3.2. The FIT value and MSE are slightly worse compared to the P1D model (see Table 3.1). Based on the *Akaike Information Criterion* the P1D has to be selected as it has the lowest AIC value. Despite this advice, the P1DSS model is selected for the controller design. Three motives support this choice. First of all, a state-space model is needed for the employed MPC formulation. Second, the P1DSS is derived from the P1D model selected by AIC. This P1D model is easier to understand as the different parameters can be easily interpreted and changed slightly to check the behavior of the controllers in the presence of model uncertainties. Furthermore, the P1DSS model has the lowest AIC from both state-space models. Thirdly, the size of the controller matrices does not depend on the number of states in the employed controller formulation.

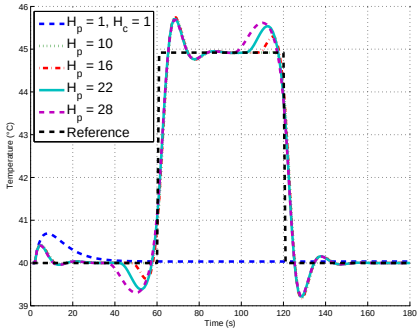
3.3 Model predictive control settings

As stated above, the *P1DSS* model is selected as controller model. The control horizon H_c is 7 and the prediction horizon H_p is 22. These horizons have been selected similar to those in [27], to compare the different controllers and control algorithms for this temperature control set-up. These horizons turned out to be the maximum settings if an MPC controller is built with CVXGEN for use on the PAC at the time of the experiments. Recently, it has become possible to pass the limit of 2000 non-zero KKT matrix entries. This maximum is now, at the time of writing, 4000 non-zero KKT matrix entries, which makes it possible to increase H_c and H_p. Nevertheless, these horizons were not changed as the value for H_p can also be motivated by the fact that it is 3 times the largest time constant, after which a system can be considered in steady-state. A control horizon $H_c = 7$ is preferred over a lower control horizon to make the computational load of the experiments as high as possible for the employed controller devices. Although there is a different response for the different control horizons (see Figure 3.3a), the effect will be of minor importance for experiments on the real set-up. All depicted plots behave similarly. These figures are created with the models in Equation (3.1) and (3.7) as control model and model to represent the system, respectively. Only an control horizon of 1 results in a clearly different response of the controller on step changes compared

to a prediction horizon higher than 1. A prediction horizon larger than one makes the controller react earlier on the future set-point change (Figure 3.3b). In case of the control and prediction horizon are both equal to one, the used system is not stable. In all other cases, the presented system consisting of first order models is stable.



(a) Different control horizon ($H_p = 22$).



(b) Different prediction horizon ($H_c = 7$, except for the plot with $H_p = H_c = 1$) .

Figure 3.3: The effect of different horizons for the proposed model predictive controller.

On each controller device, a number of experiments are performed. The weight matrix W_u is the identity matrix:

$$W_u = I_{2\times 2} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}. \tag{3.6}$$

W_y is set to $\begin{bmatrix} 1 \end{bmatrix}$. Each experiment consists of a reference trajectory for the temperature of 10 minutes as plotted in Figure 3.4. This reference is first at a constant temperature of 40°C during 100 s. To ensure the data logging program is ready and the estimator has reached a steady-state value on both PAC and PLC, the inputs calculated by the controller are only applied from 30 s on. During the first 30 s, the fan speed is set to 20% of its maximum speed and the resistor power is zero. After 100 s, the set-point jumps to 45°C and 60°C for 60 s each. This stairs is followed by quarter of a period of a cosine that has to bring the temperature at 20°C. This is below the ambient temperature of approximately 22°C for some experiments on the set-up and therefore unreachable. This part of the trajectory is added to make the controller hit the constraints for a number of seconds. From 320 s on, the temperature reference is set to 30°C for 60 s, followed by a ramp of 30 s with a slope of 0.1°C/s. At the end of the ramp there is a jump towards 50°C. This

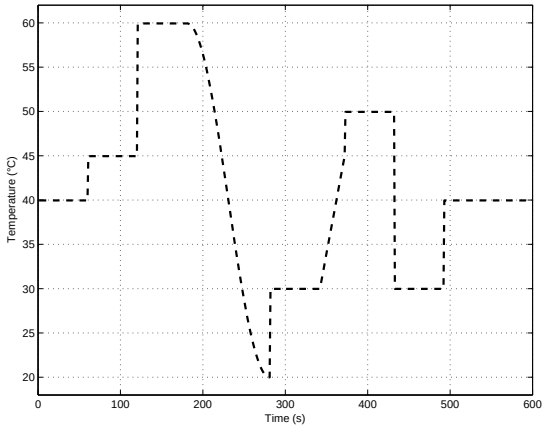


Figure 3.4: The reference trajectory for the air heating set-up.

temperature is kept constant for 60 s and followed by another set-point change to 30°C for 60 s. To end the experiment the temperature is fixed at 40°C.

The motivation to take this trajectory is mainly based on the presence of step changes for the temperature in an upward and downward direction which are common in industrial applications. The cosine part is selected to move slowly downwards to a temperature below typical ambient temperatures. Consequently, these temperatures are impossible to reach as the device can only heat up the air. The slope is employed to see if the controller can handle a slowly changing set-point without delay.

The constraints on the inputs are set to a minimum of 2 V and a maximum of 7.8 V for the fan speed. The lower constraint is set to ensure that there is a minimum flow to bring the heated air towards the temperature sensor. The constraints for the heating are 0 V and 10 V. These are the limitations of the output, corresponding with minimal and maximal power.

3.4 Model predictive control approaches

Five different approaches to calculate the optimal input of the system are investigated:

Approach 1: LabVIEW MPC There is a built-in MPC controller in LabVIEW that can be performed on the CompactRIO Real-time target. The manual describes the way it has to be implemented but very little is

known about the implementation of the controller, e.g., what algorithm is used by the solver. Despite this lack of information, it would be a pity not to compare this controller to other approaches running on the same target devices.

Approach 2: CVXGEN MPC CVXGEN is a web-based tool to convert a mathematical formulation of an optimization problem to C-code. A mathematical formulation of an MPC controller is provided to this tool. This formulation is depicted in Figure 3.5. The corresponding C-code is implemented.

Approach 3: CVXGEN An MPC controller formulated as in Section 2.3 leads to a quadratic problem to be solved. In this approach, all preparatory calculations are computed in LabVIEW, and the solution to the QP is calculated by the C-code obtained by the mathematical formulation of a QP in CVXGEN.

Approach 4: qpOASES This approach is similar as the former one. The QP is now solved by qpOASES, an open source implementation of an active-set QP solver.

Approach 5: Hildreth In a last approach, the same QP obtained as in the former two approaches is solved by the Hildreth algorithm.

All three QP algorithms used with the MPC formulation in Section 2.3 and the two MPC implementations are tested in advance in a Matlab simulation for the approaches 2 to 5 and in hardware-in-the-loop (HIL) experiments on the CompactRIO for all approaches. Hardware-in-the-loop experiments in this dissertation employ the final implementation of the controller on the hardware under investigation. The experimental set-up, however, has been replaced by a slightly changed linear model of the corresponding set-up. Matlab simulations for the LabVIEW MPC approach are not possible.

3.5 Model predictive control simulations on a PC

Before the actual implementation of the different controllers, the controllers are simulated in Matlab. A simulation for approaches 2 to 5 in Section 3.4 is performed. The different ways to calculate the optimal input, subject to constraints may cause different system behavior, although they cannot differ that much as the solution to the QP is unique. The model representing the set-up during the Matlab simulations as well as the HIL experiments, is the state-

```
# This formulation employes the small State-space model
# for a MPC formulation - 17/1/12
dimensions
  m = 2
  n = 3
  p = 1
  Hc = 7
  Hp = 22
end

parameters
  A (n,n)
  B (n,m)
  C (p,n)
  Q (p,p) psd
  R (m,m) psd
  x[0] (n)
  yref[t] (p), t=0..Hp
  duref[t] (m), t=0..Hp
  uMax (m)
  uMin (m)
  uFormer(m)
end

variables
  x[t] (n), t=1..Hp+1
  du[t] (m), t=0..Hc
  y[t] (p), t=0..Hp
end

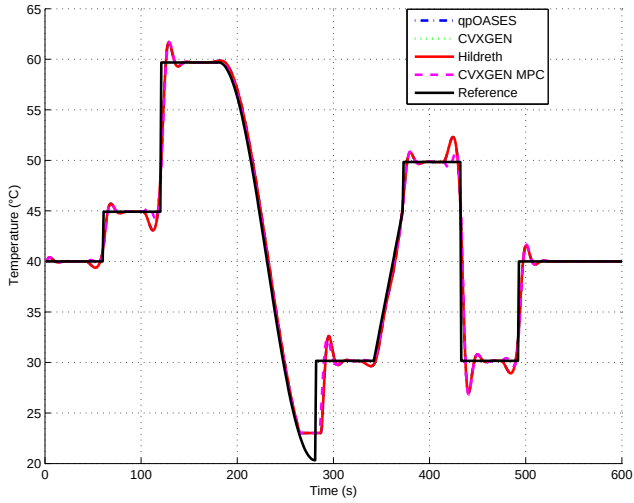
minimize
  sum[t=1..Hp](quad((y[t]-yref[t]), Q)) + sum[t=0..Hc-1](quad(du[t]-duref[t], R))
subject to
  x[t+1] == A*x[t] + B*du[t], t=0..Hc-1
  x[t+1] == A*x[t] + B*du[Hc-1], t=Hc..Hp-1
  y[t] == C*x[t], t=1..Hp
  du[0] < uMax - uFormer
  du[1] < uMax - uFormer - du[0]
  du[2] < uMax - uFormer - du[0] - du[1]
  du[3] < uMax - uFormer - du[0] - du[1] - du[2]
  du[4] < uMax - uFormer - du[0] - du[1] - du[2] - du[3]
  du[5] < uMax - uFormer - du[0] - du[1] - du[2] - du[3] - du[4]
  du[6] < uMax - uFormer - du[0] - du[1] - du[2] - du[3] - du[4] - du[5]
  du[0] > uMin - uFormer
  du[1] > uMin - uFormer - du[0]
  du[2] > uMin - uFormer - du[0] - du[1]
  du[3] > uMin - uFormer - du[0] - du[1] - du[2]
  du[4] > uMin - uFormer - du[0] - du[1] - du[2] - du[3]
  du[5] > uMin - uFormer - du[0] - du[1] - du[2] - du[3] - du[4]
  du[6] > uMin - uFormer - du[0] - du[1] - du[2] - du[3] - du[4] - du[5]
end
```

Figure 3.5: CVXGEN source code for the MPC controller referred to as CVXGEN MPC.

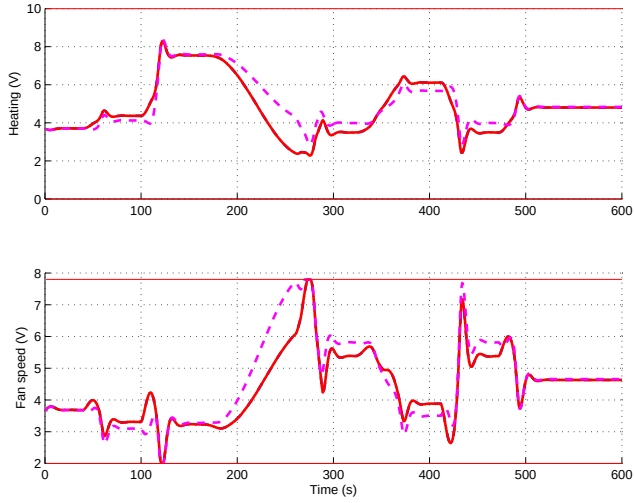
space representation of a slightly adapted P1D model given in Equation (3.7):

$$T_{\mathrm{n}} = \left[\frac{-1.01}{1+5.09s}e^{-1.48s}\ \frac{0.87}{1+7.04s}e^{-1.39s}\right]\begin{bmatrix} u_{\mathrm{Fan,n}} \\ u_{\mathrm{Power,n}} \end{bmatrix} \tag{3.7}$$

In Figure 3.6, the four test runs are depicted. The reference trajectory to be followed, is identical to the path to be followed for experiments on the set-up. It is to be expected that all four depicted cases are identical. Based on Figure 3.6a, three temperature plots are identical but the CVXGEN MPC algorithm differs slightly. At each set-point change with a large step, the controller prepares the step in a different way. CVXGEN MPC starts with a small oscillation close to the set-point. All other algorithms first move in the wrong direction before



(a) Simulated temperature, the controlled output of the system.



(b) The calculated inputs to the simulated system.

Figure 3.6: The in- and outputs for CVXGEN MPC and the Hildreth, CVXGEN and qpOASES algorithm for Matlab simulations on PC.

Table 3.2: The MSE and SSE values calculated for Matlab simulations on PC. The total run time and the mean, minimum and maximum run times for one MPC iteration are presented to indicate the speed of the algorithms.

	run-time (ms)				error	
	total	mean	min.	max.	MSE	SSE
qpOASES	1554	2.57	2.43	18.95	2.3	1376
CVXGEN	1636	2.73	2.48	16.90	2.3	1376
Hildreth	1566	2.72	2.44	20.26	2.3	1376
CVXGEN MPC	9277	15.45	14.69	31.75	2.1	1286

they actually start to follow the step for the temperature. Table 3.2 gives both the MSE and SSE values for the four simulated trajectories. From this table, it is clear that the qpOASES, CVXGEN and Hildreth algorithm present identical results. As the termination criterium for these three algorithms is set identically to 10^{-4}, this result was to be expected. CVXGEN MPC has lower SSE and MSE values and can be considered to have a more accurate control. Based on the small difference between CVXGEN MPC and the other MPC approaches, one can expect a different controller specification, but up to now, no difference could be found. Processes during code generation that might affect the result are not known to the author, but appear to be present.

Figure 3.6b depicts the inputs of the simulated approaches. Also here, it is clear that CVXGEN MPC behaves differently. The other three approaches are identical. The constraints are only reached for the fan. In simulation, only one constraint is active at the same time.

Figure 3.7 presents the number of iterations needed to calculate a solution. The Hildreth algorithm needs significantly more iterations, while the more recent state-of-the-art algorithms, qpOASES and CVXGEN, need less iterations. None of the algorithms need the maximum allowed number of iterations which is set to 60, 20 and 20 for Hildreth, CVXGEN and qpOASES respectively. The maximum number of iterations is significantly higher for the Hildreth algorithm as the convergence rate of this algorithm is slower than for CVXGEN and qpOASES.

Comparing qpOASES with CVXGEN indicates that CVXGEN always needs 5 to 7 iterations, qpOASES needs maximum 4 iterations for this trajectory. Not surprisingly, the number of iterations for the CVXGEN MPC case is very similar to CVXGEN. To have an impression of the speed of the MPC controllers, the total time for a complete run of the reference trajectory and the mean, minimum and maximum time for one MPC iteration is presented in Table 3.2.

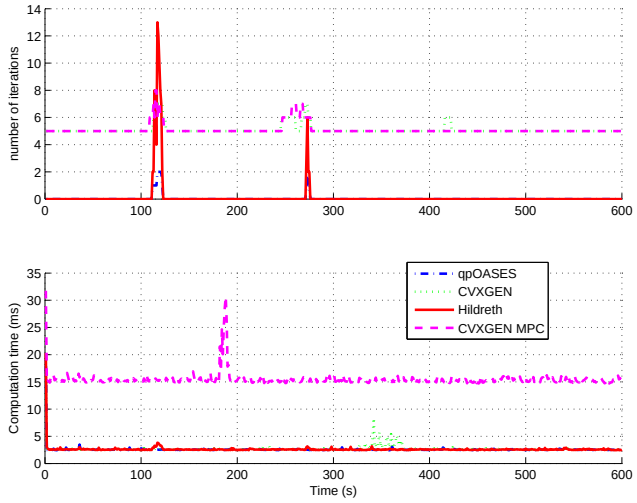


Figure 3.7: Number of iterations needed to solve the QP problem in a Matlab simulation on PC.

The total required computation times for Hildreth, CVXGEN and qpOASES are too similar to draw a conclusion based on simulations. Several reruns result in a difference of more than 0.2 seconds, resulting in a different speed ranking. According to the maximum required time for one MPC iteration, the Hildreth is the slowest algorithm and the more recent qpOASES and CVXGEN algorithms are slightly faster. For CVXGEN MPC, the state estimation is part of the QP. As the QP in this approach is larger than in the other cases, the time needed to present a solution more than triples.

None of the employed algorithms reaches its maximum number of allowed iterations, so no tricks to present a solution to the QP are needed. This explains the similar or identical results. When tightening the constraints, two constraints could be activated at the same time. The first algorithm that ran into trouble was Hildreth. As soon as the maximum number of iterations is reached, an unconstrained saturated solution is provided, causing MSE and SSE to increase. For these simulations, the recent algorithms, qpOASES and CVXGEN always stop at a maximum of 14 and 10 iterations, respectively. The fact that these algorithms always could provide a solution makes them a better candidate for being the heart of a model predictive controller.

3.6 Model predictive control on a PAC

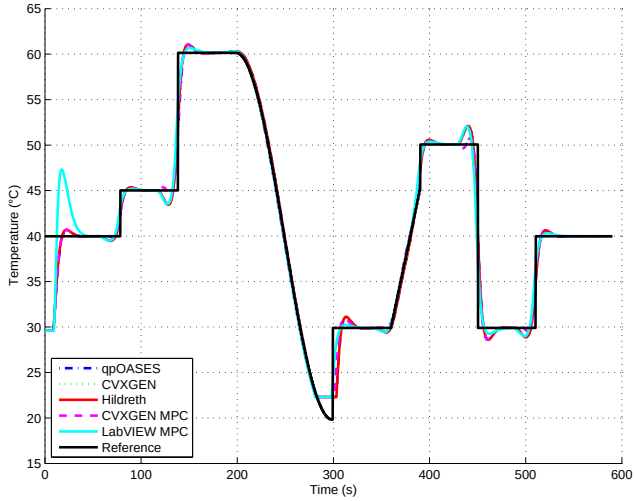
All five approaches described in Section 3.4 are tested on the CompactRIO PAC system connected to the experimental set-up. Experiments are performed in winter and summer. In summer, the experiment involves only the approaches 3 to 5, where only the QP solver differs. For experiments in winter, all approaches are tested. The ambient temperature differs almost 10°C between both experiments. This causes a different control behavior for the winter and summer experiments, showing the ambient temperature dependence of the experiments performed.

However, to ensure the correct implementation of the different MPC approaches under investigation, hardware-in-the-loop experiments on the PAC system are carried out.

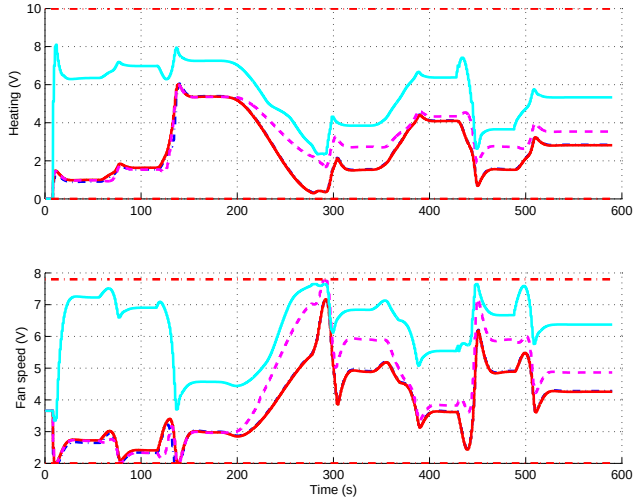
3.6.1 Hardware-in-the-loop experiments

The hardware-in-the-loop experiments consist out of all five approaches listed in Section 3.4. Note that the set-up is controlled by controller hardware which is not capable of running Matlab. A fault-sensitive change of program is needed, hence, hardware-in-the-loop experiments are recommended. The idea behind a HIL simulation is that the controller and controller hardware is used as it will be used on the real system. The in- and outputs, however, are connected to a model simulating the system. If possible, the simulated system runs on a different machine. For this heating set-up, the controller is implemented as it will be used for experiments on the set-up, but the inputs and outputs are connected to the state-space representation of the linear model presented in Equation (3.7). Of course, this model is identical for the different approaches.

Although HIL experiments can be performed on both the PLC and the PAC, only the PAC system will be used. The main reason is that this controller is more sensitive to errors, as the final controller program is a combination of a LabVIEW VI and a library written in C or C++. Thorough tests are recommended as a lot of errors can occur between these two programs. On the other hand, logging is easy. A text file can be created and stored on the device itself. For the PLC a network connection with SCADA software is necessary which affects the speed of logging (minimal 1 s), and, more importantly, arrays cannot be logged, which makes it difficult to track errors. For the PLC, only a textual translation to a different programming language is necessary. To reduce errors, all code modifications have been tested in advance in Matlab.



(a) Simulated temperature, the controlled output of the system.



(b) The calculated inputs.

Figure 3.8: The in- and outputs for LabVIEW MPC, CVXGEN MPC and the Hildreth, CVXGEN and qpOASES QP algorithms for hardware-in-the-loop experiments on the CompactRIO.

Table 3.3: Comparison of the five temperature profiles to the reference trajectory based on SSE and MSE for hardware-in-the-loop experiments on the CompactRIO. The total run time for the experiments and a mean, minimum and maximum time for one MPC iteration step are also presented.

	run-time (ms)				error	
	total	mean	min.	max.	MSE	SSE
qpOASES	155.23	0.26	0.26	0.35	3.7	2194
CVXGEN	1606.40	2.72	2.46	4.33	3.7	2194
Hildreth	22.88	0.04	0.03	0.27	3.7	2194
CVXGEN MPC	7678.30	13.01	12.25	18.03	3.5	2069
LabVIEW MPC	4741.00	8.04	8.00	11.00	4.2	2495

Figure 3.8 presents the simulated temperature for the five different MPC strategies for the HIL experiments. The output (Figure 3.8a) is similar for all strategies. Only the LabVIEW MPC is the outsider. The difference to the other four approaches is clear at time steps 30 and 300, but all step set-point changes are affected. The settings for all MPC controllers are the same, but it is not possible to supply the desired input reference to the LabVIEW MPC block. The reference for $\Delta\mathbf{U}_{\text{ref}}$ is zero for all MPC controllers in Equation (A.24). As only references to $\mathbf{U}$ can be offered to the LabVIEW MPC block, this causes the different response.

Less obvious, but also present in Figure 3.8, is the different behavior for CVXGEN MPC. Analogous to the simulation case, a small oscillation close to the reference trajectory is seen before each step. It is clearly seen at, e.g., time 125, 440 and 500 s. Comparing the different temperature profiles based on a mathematical base, indicates that all profiles are different except for qpOASES, CVXGEN and Hildreth solvers. Table 3.3 presents SSE and MSE values. The conclusions are similar as in the simulations. The qpOASES, CVXGEN and Hildreth solvers perform equally well and the SSE and MSE for CVXGEN MPC is slightly lower. LabVIEW MPC is performing worse, but as the input reference is different, a fair comparison to the other algorithms based on Table 3.3 is not possible.

The inputs reveal a similar conclusion. The identical results for CVXGEN, qpOASES and Hildreth from the table result in identical input profiles. CVXGEN MPC and LabVIEW MPC have different input profiles.

Figure 3.9 depicts the required time to calculate the solution. It has to be noticed that the time measurement for LabVIEW MPC has a resolution of 1 ms. The other time measurements have a resolution of 1 ns. Regardless of

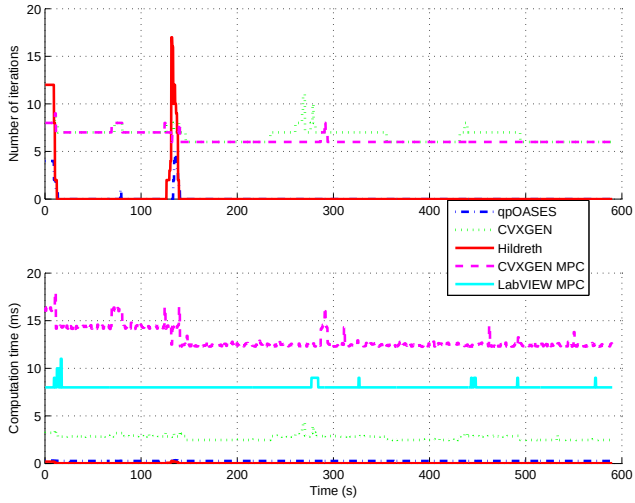


Figure 3.9: The number of iterations and computation time needed for LabVIEW MPC, CVXGEN MPC, the Hildreth, CVXGEN and qpOASES QP algorithm for hardware-in-the-loop experiments on the CompactRIO.

this difference in resolution, it is clear that LabVIEW MPC needs more time to present a solution than the trio CVXGEN, qpOASES and Hildreth. Notice, however, that state estimation is part of the recorded time for the LabVIEW MPC approach. Compared to the simulation results where the total runtime and mean run-time for one iteration hardly differs for CVXGEN, qpOASES and Hildreth, these times are clearly different on the CompactRIO device. As the measurement of the different times now only contains the calculation of the input and not any preparative calculation as, e.g., state estimation, the speed of the algorithm is compared. Based on Table 3.3 the Hildreth algorithm is the fastest, qpOASES is second and CVXGEN is the slowest. The translation to C leads to significantly different calculation times. The CVXGEN MPC approach is clearly the slowest. The added complexity increases the calculation time significantly. The researchers who created CVXGEN are aware of this problem and advise the use of FastMPC [88], a solver exploiting the structure of MPC. This solver is not taken into consideration for this dissertation as external libraries are needed, which are to be recompiled for VxWorks, the real-time operation system of the CompactRIO.

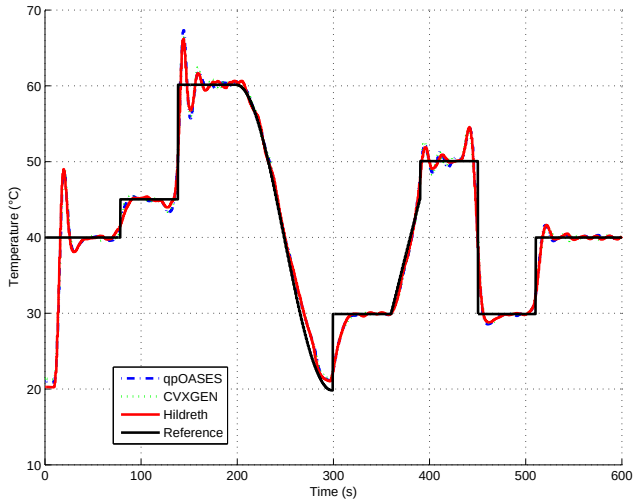
Despite the higher number of iterations for Hildreth, it is currently the fastest algorithm as can be seen at the bottom of Figure 3.9. However, it has to be kept in mind that this is a small-scale set-up and no hard conditions (constraints) had to be overcome. The number of iterations is not depicted for LabVIEW

MPC as it is not possible to track this variable. As for LabVIEW MPC not all parameters are available and there was no port available to (Siemens) PLC hardware, this controller will be neglected for PLC experiments on the real set-up. As CVXGEN MPC appears to be the slowest, also this controller will be omitted as time is considered to be an important issue on slow hardware. Also for this MPC approach, no port to PLC hardware exist.

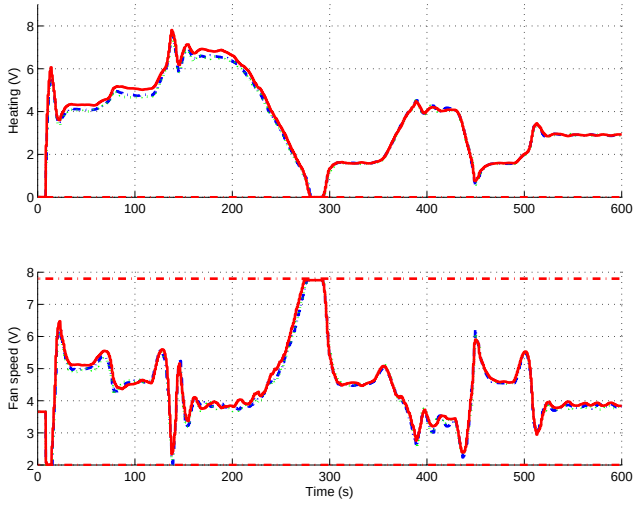
3.6.2 Resulting temperature control on the set-up

The experiments on the set-up were performed in summer (Figure 3.10) and during the winter (Figure 3.11), whereby the influence of the ambient temperature can be taken into account. For the summer experiments, the controlled temperature follows the reference accurately when all transient effects have faded. The incorporated integrator eliminates the steady-state error. The plots show oscillating behavior at steps when the reference is above 45°C. This is explained by the limited validity of the linear model at these temperatures. This oscillation behavior is undesired and has to be corrected with different controller settings, e.g., an increasing W_u. In order to make a fair comparison with the simulations these settings are not changed. All three algorithms start the experiment with a large overshoot. It is caused by the combination of the large step from the ambient temperature, for this experiment 20°C, to 40°C and the mismatch between the estimated state from the linear model and the real system. For the next step around 80 seconds, an overshoot is hardly seen. The mismatch between the model and the real heating set-up is small at this point. The next jump towards 60°C is outside the validity region of the model and this is clearly observed as an overshoot followed by oscillations. Next, the cosine function is followed accurately, except at the end, where the set-point is below the ambient temperature. This makes it impossible to track the desired temperature for the system. After 300 s, the set-point evolves slowly to 30°C as both constraints are active. The next ramp is followed with a delay of one to two seconds. At the end of the ramp, the 10°C jump causes again a large overshoot as the mismatch between the model and real system is large at 50°C. The transition to 30°C leads to a small non-oscillating undershoot.

Important to notice is that all three algorithms behave almost identically for the experiments in summer. The small differences in the output (Figure 3.10a) and inputs (Figure 3.10b) are caused by different environmental conditions. For instance, the open window during the experiments may be responsible for a soft breeze in the room from time to time. The MSE and SSE values are presented in Table 3.4. In this table, qpOASES presents the lowest MSE and SSE. However, as there is no difference between the three algorithms in

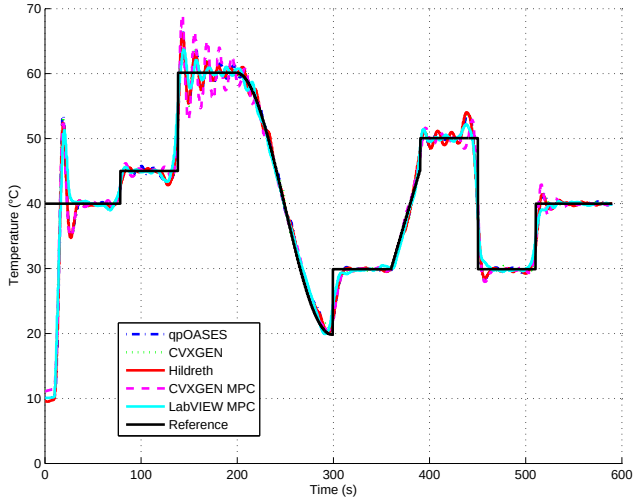


(a) Measured temperature. The controlled output of the system.

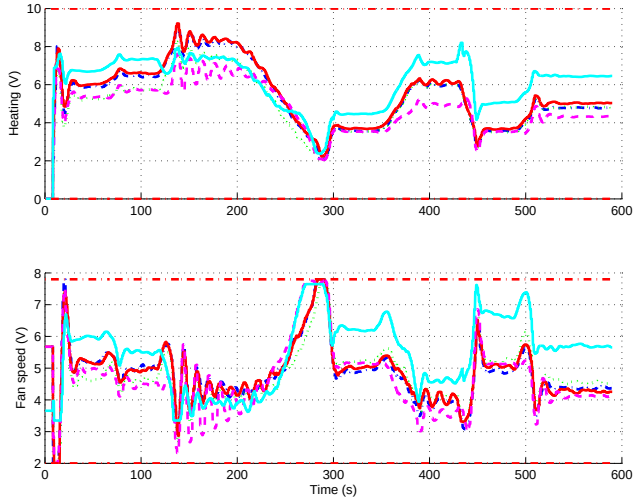


(b) The applied inputs to the system.

Figure 3.10: The in- and outputs for the Hildreth, CVXGEN and qpOASES algorithm on the PAC for experiments in summer on the CompactRIO.



(a) Measured temperature. The controlled output of the system.



(b) The applied inputs to the system.

Figure 3.11: The in- and outputs for the CVXGEN MPC, LabVIEW MPC, Hildreth, CVXGEN and qpOASES algorithm on the PAC for experiments in winter on the CompactRIO.

Table 3.4: Comparison of three temperature profiles to the reference trajectory based on SSE and MSE for the experiments on the CompactRIO in summer.

	error	
	MSE	SSE
qpOASES	11.1	6531
CVXGEN	10.7	6329
Hildreth	11.5	6766

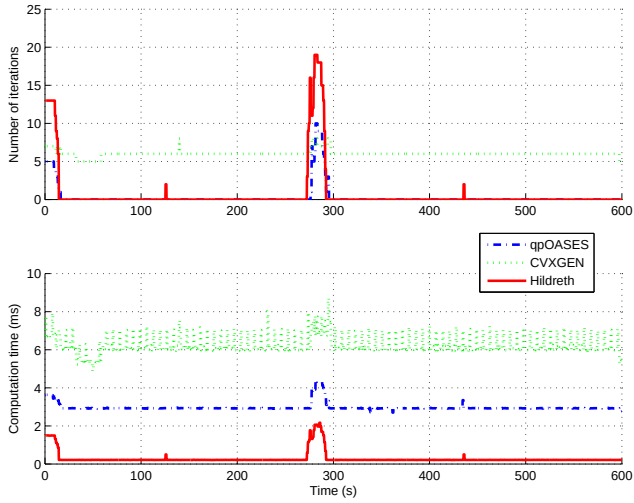
simulation and hardware-in-the-loop experiments, the differences are attributed to the different environmental conditions.

For the winter experiments the ambient temperature is close to 11°C. Notice the higher overshoot at the step at the start. The oscillations appearing for the step to 60°C are now also more distinct. This is caused by the identified model to be outside its validity region is at 60°C combined with an ambient temperature almost 7°C below the ambient temperature of the excitation experiment for model identification. This demonstrates one of the disadvantages of MPC. The model has to be valid for the conditions in which it is used. The model is identified with an ambient temperature of 18°C, and the system is excited between approximately 30 and 60°C. If the ambient temperature changes significantly and the controlled temperature reaches these extremes, the model validity is doubtful, which shows itself by means of large, almost unstable oscillations. As the desired reference temperature is higher than the ambient temperature, only one constraint is active for some time instances (Figure 3.11b). As can be seen for the summer experiment in Figure 3.10b, two constraints are active between 280 and 290 s after the start of the experiment.

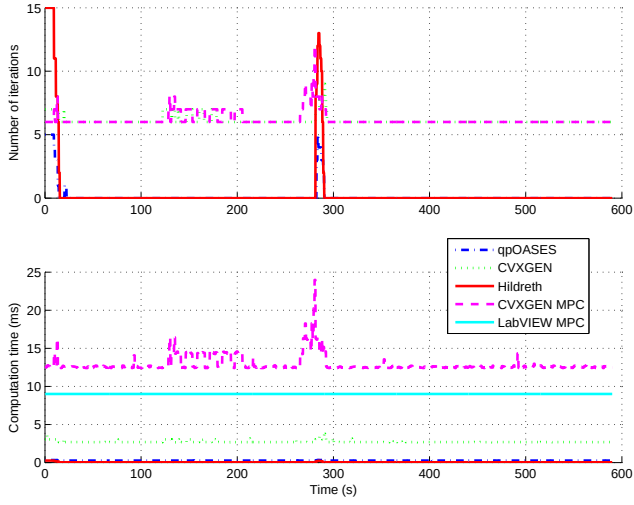
Table 3.5 presents the run-times for the different MPC approaches and their corresponding MSE and SSE values. From the plots in Figures 3.10 and 3.11, it is clear that the quality of control for the winter case is worse than for the summer experiment. This is confirmed by the MSE and SSE values which are more than doubled compared to the summer experiment.

3.6.3 Properties of the QP solvers

Figure 3.12a depicts the number of iterations needed to solve the QP for the summer experiment. The maximum was set to 20 for qpOASES, 20 for CVXGEN and 60 for Hildreth. The latter is so high as the number of iterations grows linearly with the problem size [28]. None of these maxima were reached, so the QP solvers always converged during this experiment. qpOASES needs



(a) Experiments during summer



(b) Experiments during winter

Figure 3.12: The number of iterations and corresponding calculation time to solve the QP problem on the PAC system for the summer and winter experiments on the CompactRIO.

Table 3.5: Comparison of the five temperature profiles to the reference trajectory based on SSE and MSE for the experiments on the CompactRIO in winter. The total run time for the experiments and a mean, minimum and maximum time for one MPC optimization step are also presented.

	run-time (ms)				error	
	total	mean	min.	max.	MSE	SSE
qpOASES	159.40	0.27	0.26	0.46	23.3	13757
CVXGEN	1610.99	2.73	2.66	3.78	20.9	12339
Hildreth	23.73	0.04	0.03	0.26	23.4	13804
CVXGEN MPC	7604.17	12.89	12.41	23.97	22.2	13093
LabVIEW MPC	5310.00	9.00	9.00	9.00	21.8	12849

only 5 iterations at the start whilst only one constraint is active. When two constraints are active, the maximum number of iterations is 10. If no constraint is active, no iterations are needed to solve the problem. The corresponding time is 0.48 ms for 10 iterations, 0.37 ms for 5 iterations and if no iteration is performed, the solution is presented after 0.29 ms. CVXGEN needs at least 6, but never more than 8 iterations during this experiment. The corresponding time is 0.62 and 0.82 ms. Hildreth needs at maximum 19 iterations during this experiment, which takes 0.22 ms.

Figure 3.12a depicts the time needed to present a solution by the different algorithms. As there is no difference between one and no active constraints for CVXGEN, it is not possible to determine whether there is zero or one active constraint from this plot. Only with two active constraints the number of required iterations increases. The number of required iterations and the corresponding time fluctuates more frequently for the other two algorithms. It is not possible to determine from these plots if one or two constraints are active. For this set-up, the Hildreth algorithm needs less time to solve the QP than the other algorithms. As seen in Table 3.4, CVXGEN needs the most time to present a solution, the Hildreth algorithm the least.

Figure 3.12b depicts the number of iterations for the experiment in winter. The maximum number of iterations is lower compared to the summer experiment as now at maximum one constraint is active. Again here, the Hildreth algorithm needs the highest number of iterations. CVXGEN fluctuates between 5 to 8 iterations and qpOASES is the algorithm with the lowest number of required iterations. In the bottom plot of Figure 3.12b the high computation times for LabVIEW MPC and even more striking for CVXGEN MPC catch the eye. Using these approaches is clearly more expensive in computational cost. As can be seen in Table 3.5, the time needed for CVXGEN MPC multiplies at

least by a factor 4 compared to CVXGEN and which is again almost 10 times slower than qpOASES and nearly 100 times slower than Hildreth. Even if it were possible to transform this code to working PLC code, this approach would not be recommended because it would be too slow. LabVIEW MPC is more than three times slower than the CVXGEN algorithm. For a built-in MPC approach this seems slow, although one can imagine, additional fault checking to make this controller available to different national instruments devices, can slow down the calculations.

All approaches, except LabVIEW MPC, are integrated in a C/C++ library called by the LabVIEW code. The design choices of the QP developers have important consequences. Although CVXGEN delivers ready-to-use code in a fast and straightforward way, it has to be mentioned that the choice for memory pre-allocation results in large code if the problem size increases. A test to solve a QP problem with a hessian of size 30 resulted in a code of size 430 kB and a hessian of size 40 even resulted in 996 kB, while only 80 kB were needed with a hessian of size 14 in these experiments. In [27] it has been observed that the code of the problem cannot be higher than 900 kB as the code simply will not run due to limitations of the PAC. This means a hessian of size 40 is the limit with CVXGEN. In the case where one wants MPC to control a system with multiple inputs or outputs, this limit is most likely too low for practical use on the CompactRIO. This code generator delivers C-code for the declared mathematical description. A small adaptation of this description means a regeneration of the C-code. If a different language is needed, this code generator cannot be used. For CVXGEN MPC, a similar problem occurs. For this small set-up, the current size of the library is 496 kB, corresponding to almost 2500 non-zero KKT entries. This is almost the maximum of 4000 non-zero KKT entries for the code generator at the time of writing and close to the observed (and undocumented) limit of 1 MB for the library. This approach is likely to fail with a growing problem size.

An increase in the size of the hessian hardly increases the code size of qpOASES. The compiled library is about 112 kB in size. The compiled code for Hildreth is only 7 kB. The size of the latter two algorithms will only grow with increasing size of the hessian which is coded in the library. These algorithms are programmed generally, so that it can be applied to both varying hessian sizes and can be translated into a different language.

To employ these algorithms on a PLC, the limited speed and memory size have to be taken into account. As CVXGEN and CVXGEN MPC require the most time to solve the problem and consumes the most memory, this algorithm is not preferred for this problem. Moreover, together with LabVIEW MPC the missing port to S7-SCL prohibits the use of these algorithms on the PLC.

3.7 Model predictive control on a PLC

Following the conclusions for the PAC controller, only the Hildreth and qpOASES algorithm are investigated on the PLC. Figure 3.13a depicts the measured temperature and its reference for the Hildreth and qpOASES algorithms. The corresponding inputs are displayed in Figure 3.13b.

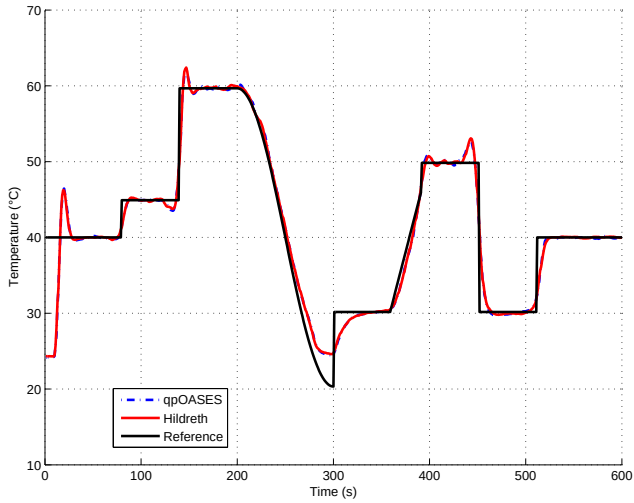
3.7.1 Resulting temperature control on the set-up

Both the Hildreth and qpOASES algorithm, follow the desired reference temperature accurately. The large overshoot in the beginning is again caused by an inaccurate estimation of the temperature, combined with a large step from the ambient temperature towards 40°C. The jump towards 50°C is taken without overshoot. The set-point change to 60°C causes an overshoot but after about 20 s, the reference is accurately followed again. Next, also the cosine function is tracked accurately. Around 300 s, at the end of the cosine, the reference temperature is lower than the ambient temperature, which makes it impossible to reach this temperature without cooling, causing both the heating and fan constraint to be hit (Figure 3.13b). The next set-point of 30°C is reached, but as still both constraints are active, the temperature evolves slowly to the desired set-point. The ramp and step towards 50°C is followed with a small delay. The large decrease of the temperature from 50 to 30°C is reached with a very small undershoot. The final jump toward 40°C is smooth and without overshoot.

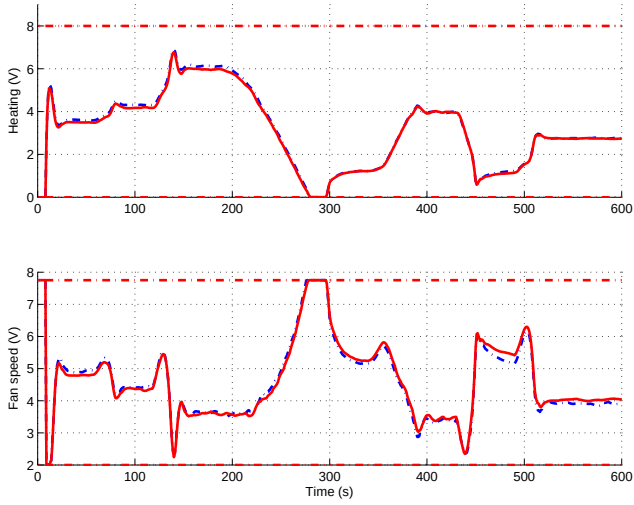
Both algorithms behave similarly. The MSE and SSE calculated on the PLC experiments can be found in Table 3.6. The difference on the SSE is less than 3% with qpOASES having the highest value. As the termination criterion for both QP algorithms is identical, the different environmental conditions are the main reason for this difference. In case a constraint is active, the delay caused by the calculation time needed to solve the QP problem differs nearly 150 ms between both algorithms (Figure 3.14). This has also a small influence on the resulting temperature control.

3.7.2 Properties of the QP solvers

The top plot in Figure 3.14 depicts the required number of iterations for the employed algorithms. The bottom plot of Figure 3.14 plots the corresponding calculation time. For Hildreth, the maximum number of iterations is 60 which is reached at 279 s. As this is the maximum allowed number of iterations, no



(a) Measured temperature. The controlled output of the system.



(b) The applied inputs to the system.

Figure 3.13: The in- and outputs for the Hildreth and qpOASES algorithm on the PLC.

Table 3.6: Comparison of two temperature profiles to the reference trajectory based on SSE and MSE for the experiments on the PLC. The total run time for the experiments and a mean time for one optimization step are also provided.

	run-time (ms)				error	
	total	mean	min.	max.	MSE	SSE
qpOASES	9292	15.72	10.00	206.00	4.1	2418
Hildreth	1076	1.82	1.00	52.00	4.0	2365

solution was found to the QP. An unconstrained saturated solution is delivered where the calculated inputs are limited to the constraints for all entries violating their constraints. It is to be expected that this causes no harm, but it is not the optimal solution. The Hildreth algorithm needs approximately 1 ms to calculate one iteration for the QP on this set-up.

If no constraint is reached for the qpOASES algorithm, no iteration of the QP is needed. Due to the more complex code of the algorithm, it takes about 10 ms to calculate the input. If a constraint is hit, it takes between 15 and 16 ms to process 1 iteration. At the start, at maximum 8 iterations, corresponding to 140 ms of calculation time, are needed to come up with a solution. The maximum number of iterations is set to 20. This number is never reached, so an optimal solution is always provided to the system for these experiments. If only one constraint is hit, 7 iterations are sufficient. No constraints are violated.

Both algorithms solve the QP online. The Hildreth algorithm is faster in time compared to qpOASES. The mean run-time for one MPC iteration as presented in Table 3.6, is more than 10 times faster for Hildreth compared to qpOASES. On the other hand, the maximum number of iterations is less for qpOASES and still an optimal solution is always found, in contrast to the Hildreth algorithm. This makes the latter more suited for MPC. The maximum required time mentioned in the table for qpOASES is 206 ms. As every second a new input has to be available to the system, the maximum available run time is almost reached for the qpOASES algorithm. It is of no importance for these experiments, but the hardware constraint that a function cannot run longer than *the scan cycle monitoring time* of 6 s, is to be taken into account if the QP matrices are larger.

Both algorithms are active-set algorithms. This means that it is possible to stop the algorithms early. It has to be noted that a minimum number of iterations is required. Although the qpOASES algorithm should be able to reach the optimum in maximum 2 to 5 iterations according to [16] if hot starting

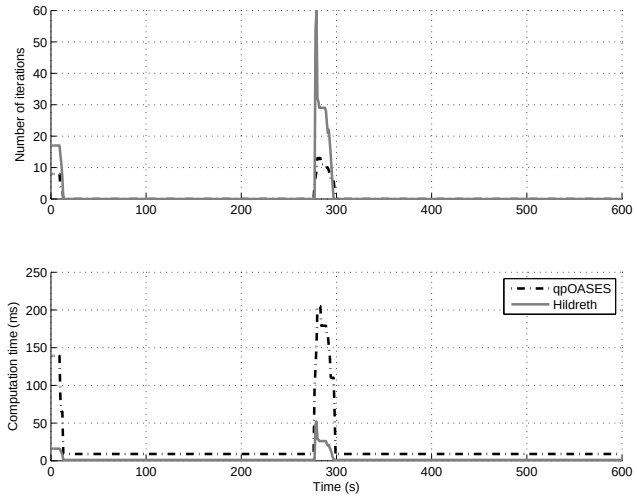


Figure 3.14: The number of iterations and corresponding calculation time needed to solve the QP problem on the PLC for both algorithms.

is applied, at least 7 iterations if one constraint is active or 14 iterations if two constraints are active, are needed for these experiments if no hot start is performed. With a cold start, at least all constraints are to be checked once over the control horizon at every time instance. For one active constraint this means 7 checks are to be performed. For two active constraints 14 checks are required. qpOASES is in this case able to solve all optimization problems in one check of the constraints, as it needs at most 14 iterations. At 279 s, Hildreth needs to check more than 4 times all constraints and is still not at the optimal solution. It is clear that qpOASES evolves faster to the optimum than Hildreth. For practical use, qpOASES is therefore preferred as it evolves faster to the optimal solution. If systems with a larger QP are to be solved, qpOASES is expected to require less time as it needs less iterations to solve the QP [16]. However, for this small air heating set-up, the size of the QP is too small to also benefit in time from the advanced solving strategy of qpOASES.

3.8 Conclusion

The full cycle, starting with model identification, followed by the controller design, defining of the MPC settings, simulations on computer, safe hardware-in-the-loop experiments on a PAC, real experiments on a PAC and finally real experiments on a PLC, is completed in this chapter to implement an MPC

controller on a PLC device. The employed system involves a small-scale heating set-up.

For the PAC system, the selection of the *best* approach is difficult. The built-in controller is not the fastest, nor the slowest, but if support is important, this controller has to be taken in consideration. The combination of software and hardware from the same vendor simplifies the direction of support questions. For most applications, this controller will be sufficient. For reasons of speed the CVXGEN MPC controller should be avoided. It is (too) slow and, also important, a slightly larger problem will already reach the maximum non-zero KKT entries in the web tool for this approach. The CVXGEN, Hildreth and qpOASES algorithm are employed the same way. Despite the higher number of required iterations for solving the QP, the Hildreth algorithm is the fastest. qpOASES is slightly slower and CVXGEN is the slowest algorithm. For this small problem, the Hildreth algorithm is sufficient, but with increasing size, qpOASES is preferred.

In order to run MPC on a PLC, a typical industrial controller and translation of the algorithm has to be available in STL. For the employed Siemens controller, the manufacturer's dialect, S7-SCL, has been used. Based on the experiments on the PAC, two algorithms have been retained for investigation on PLC. The Hildreth and qpOASES QP solver have been used and are considered to be useful. For this set-up, the Hildreth algorithm is the fastest algorithm, but suffers from the fast increasing number of iterations needed to present a solution. Hence, it is to be expected that this high number of iterations will be a disadvantage when scaling up. If the maximum number of iterations is reached, the presented solution is suboptimal. The employed unconstrained saturated solution is a fall-back solution to continue to the next MPC step. qpOASES is slower, but solves, even in the constrained case the problem with very few iterations. If hot starting can be integrated, which is currently not done, the number of required iterations can be reduced further. For larger systems, qpOASES is recommended. For this small set-up, Hildreth is sufficient.

The two code-generated approaches are slow compared to Hildreth and qpOASES. CVXGEN always needs more time. CVXGEN MPC multiplies that time by a factor of 3. Given the calculation times on the PLC for Hildreth and qpOASES, the CVXGEN MPC is expected not to be able to calculate the solution in the given time.

Based on this conclusion, Hildreth and qpOASES are the two approaches that will be taken into consideration for a controller hosted by a PLC for use on the pilot-scale distillation column. This second case study will be investigated in the next chapter.

Chapter 4

CASE STUDY II: A Pilot-Scale Distillation Column

This chapter describes the identification of a controller model and implementation of a model predictive controller on a pilot-scale distillation column. First, the set-up is described. Next, a model for control is identified. The results obtained by identification of linear black-box models based on transfer functions, subspace state-space techniques and polynomial models are provided. The model identification is finalized with the identification results of a nonlinear LS-SVM modeling procedure.

The selected linear model has been used in a linear model predictive controller hosted on a PC, PAC and PLC. Finally, some conclusions on linear MPC hosted by the employed industrial hardware are summarized.

4.1 Distillation column set-up

The pilot-scale experimental set-up involves a packed distillation column (see Figures 4.1 and 4.2). The column is about 4.5 m high and has an internal diameter of 6 cm. The column works under atmospheric conditions and contains three sections of about 0.95 m with Sulzer CY packing (Sulzer, Winterthur) to facilitate the separation process by increasing the contact

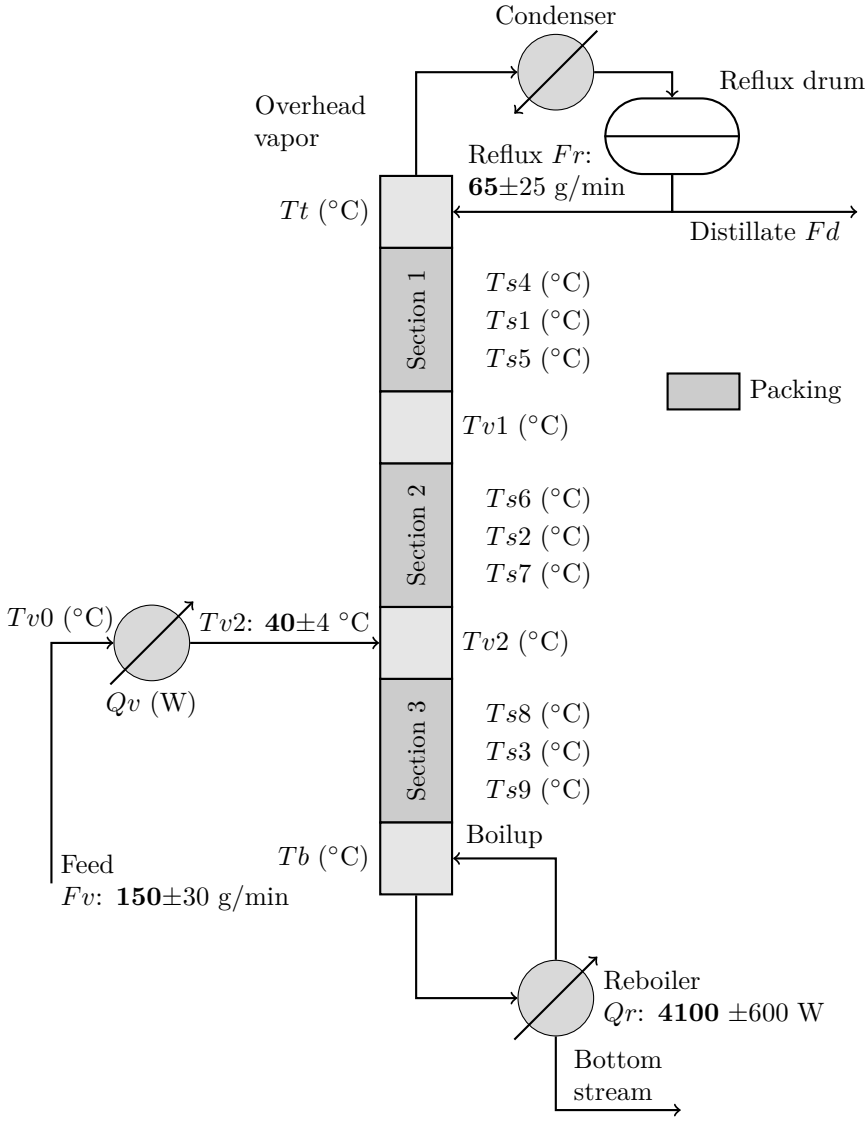


Figure 4.1: Diagram of the pilot-scale distillation column. Nominal set-points for identification experiments are printed in bold and are followed by the maximum employed deviations during identification experiments.



Figure 4.2: Pictures of the pilot-scale distillation column: condenser (left), packed section and feed introduction (center), and reboiler (right).

surface area. This packing has a contact surface area density of 700 m^2/m^3 and each meter packing is approximately equivalent to 3 theoretical trays. The feed stream containing a mixture of methanol and isopropanol is fed into the column between packed sections 2 and 3. The temperature of the feed can be adjusted by an electric heater which can deliver power up to 250 W. At the bottom of the column a reboiler is present containing two electric heaters, each of 3000 W. These can be manipulated to deliver heat to the reboiler from zero up to 6 kW. In the reboiler, a part of the liquid is vaporized while the remainder is removed as the bottom stream. At the top of the column, a total condenser allows the condensing of the entire overhead vapor stream, which is then collected in a reflux drum. A part of the condensed liquid is fed back to the column as reflux, while the remainder leaves the column as the distillate stream.

In this set-up the following four variables can be manipulated: the reboiler duty Qr (W), the feed rate Fv (g/min), the duty of the feed heater Qv (W) and the reflux flow rate Fr (g/min). The distillate flow Fd (g/min) is adjusted to maintain a constant reflux drum level. Measurements are available for the reflux flow rate Fr, the distillate flow rate Fd, the feed flow rate Fv and 15 temperatures. The temperature probes are placed at the top of the column Tt and at three places on every packing section ($Ts1$ to $Ts9$). Temperature measurements $Tv1$ and $Tv2$ are available between section 1 and 2 and between section 2 and 3 respectively. Finally, the temperature measurement Tb in the reboiler of the column, and the temperatures of the feed before and after heating ($Tv0$ and $Tv2$, respectively) complete the list of measured temperatures. All temperatures are measured in degrees Celsius with a resolution of 0.01°C.

The actuators and sensors are connected to a Compact Fieldpoint (National Instruments, Austin) with a cFP-2020 controller interface and in- and output

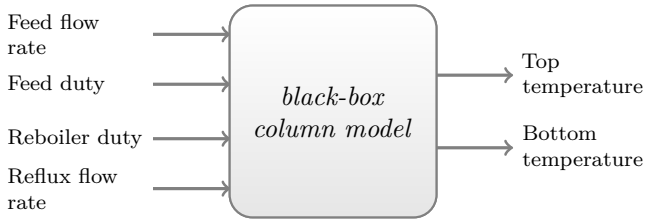


Figure 4.3: Overview of the in- and outputs of the pilot-scale distillation column model.

modules cFP-RTD-124, cFP-AIO-610, cFP-AIO-600 and cFP-AI-110. Due to the lack of computing power, this Compact Fieldpoint is linked to a powerful CompactRIO cRIO-9024 CPU or personal computer to run the controller programs.

There is no on-line measurement of the concentrations in the distillate and bottom streams although these are very important for the chemical process. The concentrations can be measured off-line, e.g., based on a their refractive index using a refractometer. However, since the concentrations for this binary system under study can be easily inferred from the boiling temperatures when the pressure is known, only a controller for the temperatures is implemented.

4.2 Model identification

For linear MPC, a linear process model is required. To describe the behavior of the column, a model is created that connects the four manipulated variables, i.e., the feed flow rate (Fv), the feed duty (Qv), the reboiler duty (Qr) and the reflux flow rate (Fr) with two temperatures, the top temperature (Tt) and the reboiler temperature (Tb). Given the nonlinear nature of the distillation process, the linear model is identified around the nominal working point of the pilot-scale distillation column. Although the concentration of the end product is actually a desired variable to be controlled, this can easily be inferred from the temperatures when the system pressure is known [73]. Hence, a MIMO model (Figure 4.3) that predicts the reboiler and top temperature is implemented.

A distillation column is a complex system with a lot of interconnected subsystems. A white-box model will be very complex [7] and most likely too large for control on low level hardware. In this dissertation, only black-box models are created for control purposes. Based on recorded data, four different modeling structures are employed to obtain a model:

1. A model consisting of a transfer function for each of the input-output pairs.
2. A subspace state-space model.
3. A polynomial model following the Box-Jenkins model structure.
4. A Least Squares Support Vector Machine (LS-SVM).

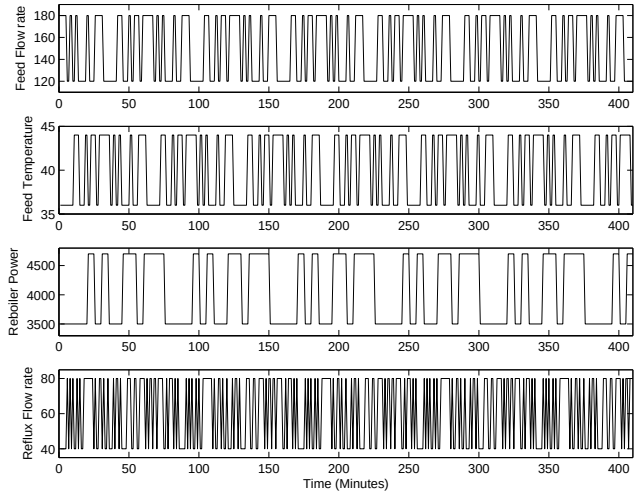
The first three approaches create a linear model, which can be used for linear MPC. The employed MPC needs a state-space representation. Hence, all linear models are converted to state-space. The last structure describes a nonlinear model. This model is identified to compare it to the linear models to see if it delivers more accurate prediction. However, it will not be used for control.

4.2.1 Excitation experiments

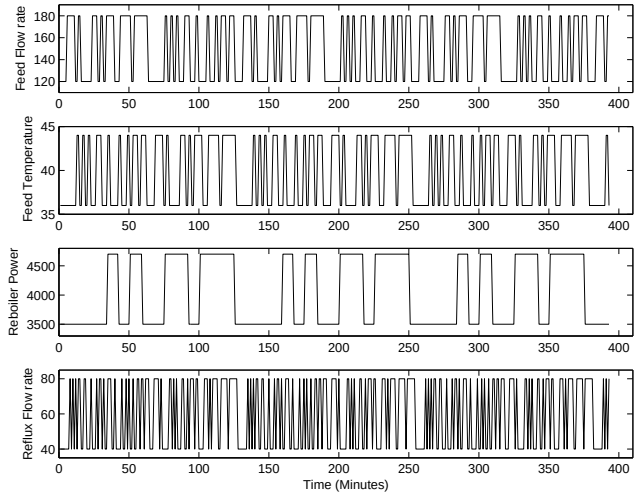
In order to generate estimation and validation data for the system identification, two experiments are performed. The excitation signal is built up with Pseudo Random Binary (PRB) signals for the different manipulated variables. Before the excitation signals are applied, the column is kept at a constant operating point for two hours to ensure the column is in steady-state. The nominal steady-state values of the different manipulated variables are: a reflux flow rate Fr of 65 g/min, a feed flow rate Fv of 150 g/min, a feed heater duty Qv of 152 W to maintain a feed temperature $Tv2$ of 40°C and a reboiler power Qr of 4100 W. These nominal values are known to yield an appropriate operating point for the column [36]. All manipulated variables are controlled by PI controllers except for the heating duties which are controlled directly.

When the column has reached steady-state after two hours, the experiment is started. When the excitation signals are applied, all manipulated variables switch between two values. The reflux flow rate Fr fluctuates between 40 and 90 g/min, while the feed flow rate Fv changes between 120 and 180 g/min. The feed heater duty Qv is manipulated between 0 and 250 W and the reboiler power Qr switches between 3500 and 4700 W. These values are displayed on Figure 4.1. The distillate flow rate Fd is manipulated in order to keep the content of the reflux drum at 40% of its maximum volume. All data are recorded with a sampling period of 100 ms. Two different experiments are performed with these settings.

The first PRB input signal is constructed in the following manner. The reboiler duty Qr is a repeated periodic signal of 6000 s. The clock period, i.e., the minimum time before the signal is allowed to switch, is 300 s. From previous



(a) Excitation experiment 1.



(b) Excitation experiment 2.

Figure 4.4: The visual representation of the two different excitation experiments.

experiments [36], it is known that the dynamics of the system are faster at the top of the column. Therefore, the clock period of the other inputs is slightly smaller. For the feed flow rate Fv and feed temperature $Tv2$ a clock period of 120 s is taken with a period of length 3720 s and for the reflux flow rate Fr, the clock period is 20 s with a period length of 5100 s. These input signals are combined into one experiment with a time span of 25000 s. The second experiment is slightly slower. The periodic signal has a length of 8000 s for all manipulated variables and the clock period for Qr, Fv, $Tv2$ and Fr is 500 s, 120 s, 120 s and 60 s respectively. The duration of the experiment is also 25000 s in total.

4.2.2 Data preparation

The sampling period of the two recorded data sets is reduced to 60 s. Therefore, every 60 s a sample is taken from the original recorded data that is first filtered with zero-phase digital filter (-3dB frequency = 0.3 Hz). Next, an identification and a validation data set have been created. The first PRB data set described in the former section is selected to be the estimation data set, the second is the validation data set.

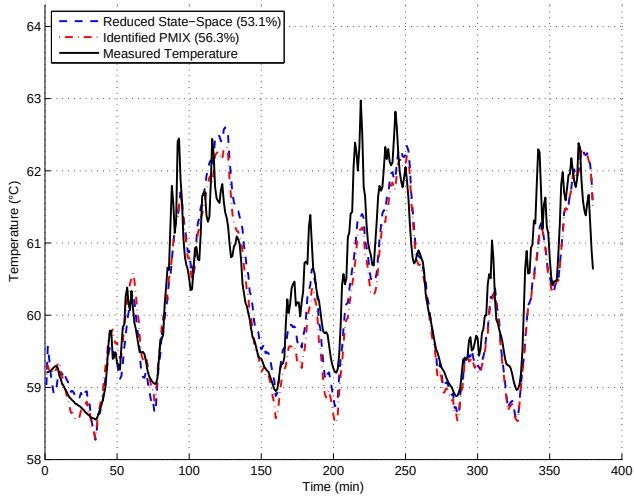
4.2.3 Model validation and selection

During validation, different quality indexes are calculated on the validation data set, e.g., the FIT value (Equation (2.18)) and the MSE (Equation (2.20)). The model selection is based on model selection criteria as, e.g., the Akaike Information Criterion. However, operator experience is also taken into account.

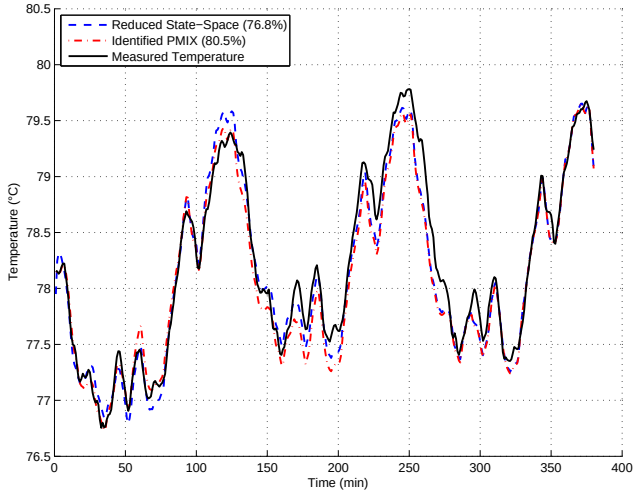
4.3 Model identification - transfer functions

Distillation columns can be described by linear low order systems [76]. Identification based on low order transfer functions is the first step on the path to a model for control. An ideal model for control has to combine an accurate tracking of the main column behavior with a low model complexity.

The following approach has been used. For each output, a MISO model consisting of low order transfer functions as presented in Section 2.2.1, is fitted. For each MISO model, each of the four transfer function structures are identical. Five different model structures are estimated for both output temperatures: A first order model (further called *P1*), a first order model with time delay (*P1D*),



(a) Top temperature



(b) Reboiler temperature

Figure 4.5: Validation of the top and reboiler temperature. Each figure depicts the simulation for the reduced state-space model and the identified PMIX model.

Table 4.1: Model quality and selection parameters for the transfer function models. Transfer function model structures marked with an * in the FIT column have parameters higher than $1 \cdot 10^4$ or lower than $1 \cdot 10^{-4}$.

	Top temperature			Reboiler temperature		
Model type	FIT	AIC	MSE	FIT	AIC	MSE
P1	51.59	*-2.84*	0.3258	66.03*	-2.26	0.204
P1D	60.99	-2.66	0.2116	75.20	-2.27	0.109
P2	54.82*	-1.98	0.2838	80.95*	-2.51	0.064
P2D	60.40*	-2.03	0.2180	84.15*	-2.09	0.044
P2DU	66.16*	-2.69	0.1592	73.28	*-2.65*	0.126
PMIX	56.29	-3.83	0.2656	80.68	-1.66	0.066

a second order model (*P2*), a second order model with time delay (*P2D*) and an undamped second order model with time delay (*P2DU*). Table 4.1 presents the model quality and model selection indicators for both the top and reboiler temperatures. Based on the Akaike Information Criterion, the P1 model for the top and the P2DU model for the reboiler have to be selected.

The P1 model, selected for the top temperature by the AIC value indicated in Table 4.1, presents a FIT of only 51% and MSE of 0.32. The model quality is poor. The almost 9% better performing P1D model with only slightly higher AIC can be considered as a replacement. However, as the model quality based on FIT and MSE is still low, the 4 extra parameters needed for this model are neglected and the P1 model, indicated by the AIC value, is suggested.

The AIC, MSE and FIT values for the reboiler temperature in Table 4.1, reveal more accurate models compared to the top temperature. Based on AIC, the P2DU model is selected. With a slightly higher AIC index and slightly higher FIT and MSE, the P1D model can be suggested as an alternative candidate controller model. The even higher FIT and lower MSE, together with a AIC comparable to P2DU suggest P2 might also be an alternative controller model. However, the unrealistic time constant of $8 \cdot 10^5$ s and gain of 500, indicated in the table by a *, for the feed power makes this an uncertain candidate. As seen later in Section 4.7, the feed power is of minor importance for control. Regarding this information, P2 can be considered. Notwithstanding that different candidates exist, the P2DU model is suggested as the controller model.

Although the AIC values select useful models, an attempt has been made to improve the selected models. Based on operator knowledge, some identified time constants appear to be too large. To improve the model, the identification is repeated but this time the 4 transfer functions in each MISO model do not

need to be of the same model structure. Based on trial and error, a *PMIX* model is created for both temperatures. For the top temperature, this model has a lower AIC and MSE than the selected P1 model. A similar trial and error procedure for the reboiler temperature resulted in a model with slightly higher fit and lower MSE, but higher AIC. Although AIC does not suggest this model, it has been selected to be part of the candidate controller model as the (dominant) time constants appear to be closer to the operator expectations. From physical insight, one can assume all subsystems have a time delay. This is not the case in the PMIX models, as several second order models without time delay appear. In these cases, the second order pole masks the time delay. Re-estimation based on a first order transfer function with time delay resulted often in a worse AIC or FIT value. Is such a case, the second-order subsystem has been preserved. The two selected PMIX models have been merged into one MIMO transfer function presented in Equation (4.1) and converted to state-space.

$$\begin{bmatrix} Tt \\ Tb \end{bmatrix} = \begin{bmatrix} \frac{-2.26}{(1+2565s)(1+135s)} & \frac{0.53}{1+735s} & \frac{3.74}{(1+1803s)(1+78s)} & \frac{-3.45}{(1+2698s)}e^{-72.8s} \\ \frac{-1.87}{(1+890s)(1+169s)} & \frac{0.62}{1+1465s}e^{-307s} & \frac{5.21}{(1+2864s)}e^{-43s} & \frac{-2.36}{(1+1275s)(1+525s)} \end{bmatrix} \begin{bmatrix} \mathrm{Fv} \\ \mathrm{Qv} \\ \mathrm{Qr} \\ \mathrm{Fr} \end{bmatrix} \tag{4.1}$$

Conversion to discrete state-space is required for the model predictive control formulation. The discretization interval is one minute. To obtain this discrete state-space model, the continuous transfer functions have been transformed from the s to the z-domain. The time delays have been integrated by adding k poles at z=0 for each delay of k sampling periods. In a last step, conversion to state-space results in a model with 20 states. Model reduction has been performed based on Hankel singular value decomposition. The Hankel singular value plot in Figure 4.6 does not present a clear step for the energy of the singular values. Model reduction is therefore performed by removing all states with energy lower than 0.01. After performing this procedure, 13 states remain. This model and some more information on the conversion to state-space of the identified transfer function model, is available in Appendix B. The reduced state-space model has been checked to be observable and controllable. In Figure 4.5 two plots are depicted. One presents the simulation of the reduced state-space and original selected transfer function model for the top temperature, the second does the same for the reboiler temperature. For the top temperature, both models follow the trend of the measured temperature well, although some peaks are not captured. Especially, sharp peaks and fast variations are not covered by the models. The fast varying peaks are believed not to disturb the control action of the controller due to the long time constants

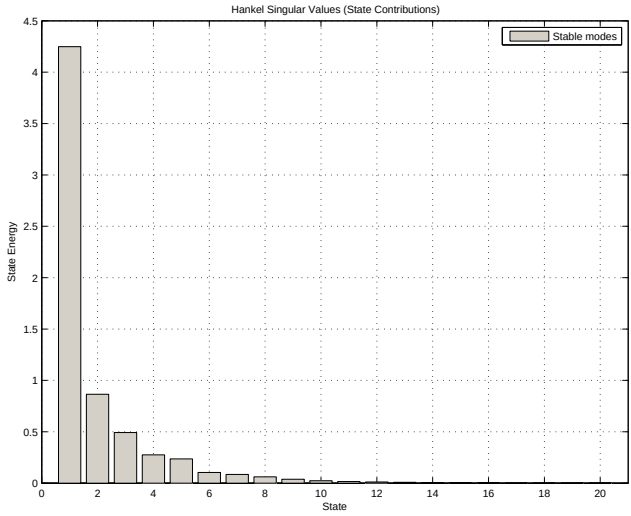


Figure 4.6: A plot of the Hankel singular values for the discrete state-space model.

Table 4.2: The FIT and MSE values for the reduced state-space model and identified PMIX models for both the reboiler and top temperature.

model		FIT	MSE
top:			
	Reduced State-Space	53.1%	0.2608
	PMIX transfer function	56.3%	0.2261
reboiler:			
	Reduced State-Space	76.8%	0.0810
	PMIX transfer function	80.5%	0.0566

of the model. For the reboiler temperature, the measured temperatures are followed accurately.

Table 4.2 presents the FIT and MSE values for the plotted models. Conversion to state-space and model reduction results in a small loss of accuracy of the model. The reduced state-space model is selected to be the candidate model created with system identification based on transfer function models.

4.4 Model identification - subspace state-space

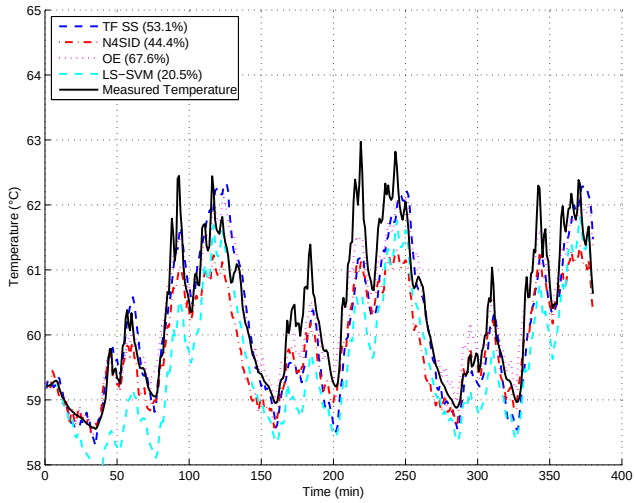
This identification method estimates the parameters A, B, C, D and K of a state-space model according to the formulation of Section 2.2.1. The matrices are identified with the N4SID algorithm, extensively described in [83].

The models are identified for model orders 1 to 20. The focus of the algorithms is set to *simulation* to ensure stability. In Table 4.3 a number of identified models are presented.

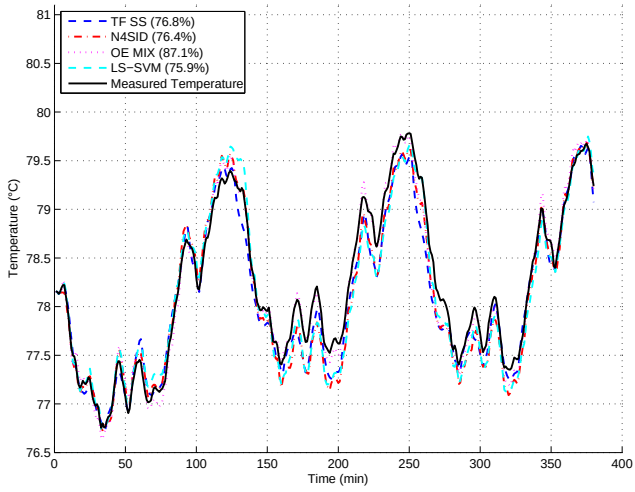
Table 4.3: Quality and selection parameters for the subspace state-space identification.

	Top temperature			Reboiler temperature		
Model order	FIT	AIC	MSE	FIT	AIC	MSE
3	34.41	-4.00	0.5982	70.93	-5.68	0.1495
4	24.99	*-4.08*	0.7823	73.08	-5.72	0.1281
5	30.17	-4.04	0.6780	*76.38*	-5.67	0.0987
6	44.45	-3.90	0.4291	76.08	*-5.78*	0.1012
7	*45.52*	-3.79	0.4127	74.88	-5.64	0.1116
8	29.97	-3.80	0.6820	76.16	-5.58	0.1005
9	25.62	-3.71	0.7692	72.73	-5.35	0.1315
10	44.13	-3.76	0.4340	68.11	-5.51	0.1799

Based on the AIC, the fourth order model has to be selected for the top temperature. The best FIT on validation points to the seventh order model. When combining a high FIT and low AIC, the sixth order model is selected. For the reboiler the maximum FIT is seen for the 5th order model. The AIC value selects the sixth order. Given the almost identical values for selection and quality indices, the fifth order model is selected. The plot of the selected models is depicted in Figure 4.7, together with the reduced state-space model created from the transfer function models. The subspace models perform slightly worse compared to the transfer function models. Combined into one MIMO state-space model to integrate this model into the model predictive control formulation used in this work, the number of states is 11. This is slightly less than 13 for the transfer function model. If the number of states does matter for the MPC controller, the state-space model identified by N4SID can be taken into consideration.



(a) Top temperature



(b) Reboiler temperature

Figure 4.7: Validation of the selected subspace state-space models (N4SID), the transfer function model employed in the MPC (TF SS), the selected OE models (OE) and the LS-SVM model (LS-SVM) for the top and reboiler temperature.

4.5 Model identification - polynomial models

The polynomial models following the Box-Jenkins model structure consist of the Auto-Regressive with eXogeneous inputs (ARX), Auto-Regressive Moving-Average with eXogeneous inputs (ARMAX), Output-Error (OE) and Box-Jenkins (BJ) model classes. For each model class and for each temperature, the table presents the model selected based on lowest AIC, on lowest AICc and highest FIT on validation data. In some cases a different model candidate is proposed based on a combination of two or three of these criteria. The corrected Akaike Information Criterion, AICc, is employed as the number of parameters in the Box-Jenkins structure can increase rapidly. Given that there are only 400 data points in the identification data set, models with a high number of parameters, e.g., 150 parameters, are most likely overfitted and do not give an accurate model for the concerned temperature. Nevertheless, they are selected by AIC. The limited number of data points is responsible for an incorrect AIC as the correction term in AIC supposes far more data points than parameters. In the current situation, the number of parameters is only half the number of data points. In such cases AICc is more reliable. Table 4.4 presents the maximum value for each of the parameters in the polynomials used during the identification process. All parameters range from the value of one to the value indicated in the table. In Table 4.5 and Table 4.6 the results are presented for the identification procedure for the top and bottom temperature, respectively.

Table 4.4: Maximum value for each of the parameters employed in the Box-Jenkins model structures.

	Parameters					
Model	n_a	n_b	n_c	n_d	n_f	n_k
ARX	60	60	-	-	-	4
ARMAX	20	20	20	-	-	4
OE	-	20	-	-	20	4
BJ	-	20	10	10	20	4

Table 4.5 lists a selection of models for the top temperature for the four model classes. For ARX, the maximum FIT is 43% and the lowest AIC results in a 10% FIT. Given the huge number of parameters for these two models, ARX is supposed to be the wrong model class. For ARMAX the FIT_{max} on validation data decreases 10% compared to ARX. The FIT calculated for the presented models is close to 30%.

The high number of parameter needed, combined with low FIT and high MSE, indicates that also ARMAX is probably not the right model class. For OE

Table 4.5: Model identification results of the polynomial models following the Box-Jenkins model structure for the top temperature.

					Parameters					
	FIT	AIC	AICc	MSE	n_a	n_b	n_c	n_d	n_f	n_k
ARX model:										
FIT_{max}	43.5	-4.2611	81.68	0.44	17	22	-	-	-	1
AIC_{min}	10.8	-4.5946	410.16	1.10	54	35	-	-	-	1
$AICc_{min}$	36.4	-4.1134	-3.53	0.56	2	1	-	-	-	1
ARMAX model:										
FIT_{max}	33.0	-3.7128	19.41	0.62	15	5	14	-	-	3
AIC_{min}	30.9	-4.4889	30.86	0.66	17	9	17	-	-	1
$AICc_{min}$	31.6	-4.0671	-3.48	0.65	1	1	1	-	-	1
OE model:										
FIT_{max}	67.6	-2.8205	3.11	0.14	-	1	-	-	6	1
AIC_{min}	52.7	-3.4639	104.89	0.31	-	6	-	-	19	1
$AICc_{min}$	42.6	-2.7799	-1.95	0.45	-	1	-	-	1	1
BJ model:										
FIT_{max}	73.2	-3.0328	15.46	0.09	-	6	4	7	4	1
AIC_{min}	45.3	-4.9752	99.03	0.41	-	7	3	3	20	1
$AICc_{min}$	47.9	-4.0609	-2.93	0.37	-	1	1	1	1	1

the maximum FIT for FIT_{max} increases up to 67% which is even more than 20% better than the best subspace state-space models and 10% better than the best transfer function model. The model selection based on AIC and AICc reduces the FIT again to the level of the state-space model structure. However, the high FIT and low MSE combined with a low AIC and low number of parameter in the FIT_{max} model, suggest this OE model is a good candidate model. The BJ models present similar results as the corresponding OE models. As BJ models are more complex, the OE model FIT_{max} is selected as candidate model representing the top temperature.

The results for the reboiler temperature in Table 4.6 present for the ARX model class a beautiful FIT_{max} of 75%, but a reasonably low AIC is only obtained by removing all models with a FIT value lower than 0. With only FIT values above 0, the AIC number selects a model with a FIT of 22%. This suggests this is not the right model class. The existence of numerous models with large number of parameters selected by AIC supports this hypothesis. With the AICc criterium the selected model presents a better FIT, but it is still far from the maximum possible for this class. Although most likely not the correct model class, a model with a low number of parameters could be found that presents accurate

Table 4.6: Model identification results of the polynomial models following the Box-Jenkins model structure for the reboiler temperature.

					Parameters					
	FIT	AIC	AICc	MSE	n_a	n_b	n_c	n_d	n_f	n_k
ARX model:										
FIT_{max}	75.5	-6.3760	-0.81	0.10	19	2	-	-	-	1
AIC_{min}	22.4	-6.7740	371.34	1.06	60	32	-	-	-	1
AICc_{min}	55.4	-5.8972	-5.35	0.35	2	1	-	-	-	1
mix	73.5	-6.3210	-5.19	0.12	2	2	-	-	-	1
ARMAX model:										
FIT_{max}	78.3	-6.3396	-0.78	0.08	6	2	13	-	-	1
AIC_{min}	29.3	-6.7046	88.98	0.88	20	18	18	-	-	1
AICc_{min}	57.3	-5.8903	-5.19	0.32	2	1	1	-	-	1
mix	75.0	-6.3278	-5.04	0.11	2	2	1	-	-	1
OE model:										
FIT_{max}	88.3	-2.4601	89.25	0.07	-	9	-	-	18	1
AIC_{min}	78.5	-3.5449	38.53	0.06	-	3	-	-	16	1
AICc_{min}	72.9	-3.0052	-1.54	0.13	-	2	-	-	1	1
mix	86.3	-3.4806	2.45	0.02	-	3	-	-	4	1
BJ model:										
FIT_{max}	93.0	-2.5925	85.2493	0.01	-	10	5	5	14	1
AIC_{min}	76.2	-6.8963	48.0702	0.07	-	3	4	2	17	1
AICc_{min}	73.4	-6.1041	-4.0445	0.12	-	1	2	1	2	1
mix	83.7	-6.2691	-3.5040	0.04	-	3	1	1	1	1

predictions. It is referred to as the *mix* model.

The results for ARMAX are very similar to ARX. A high FIT_{max} can be obtained, but AIC and AICc select models with a significantly lower FIT. Again here, a good candidate model can be obtained with low number of parameters and reasonable FIT. The OE presents a very high FIT_{max} for the reboiler temperature, higher than transfer and subspace state-space models. The AIC and AICc select, however, a model with a FIT comparable to transfer and subspace state-space models. A good candidate model is the *mix* model. It combines the very high FIT with nearly the lowest AIC and a low AICc. The selected models for the BJ model class have similar values to the models selected by AIC and AICc, but a significantly higher FIT_{max} can be obtained. Given the higher number of parameters for the BJ model compared to the OE model for similar FIT and MSE, the OE *mix* model is selected to be the final model for the BJ class.

The plots for both selected OE models are presented in Figure 4.7. The reboiler is simulated very accurately. The model for the top performs better than its transfer function and subspace state-space competitor, but still has difficulties with the sharp peaks. Converting the OE models to state-space results in a model of order 24 for the top and order 16 for the reboiler. A combined model will be of order 40. This is more than three times as large as the converted transfer function model. As a result, this model is not taken into consideration for use in the MPC controller. More than doubling the order of the state-space representation cannot be justified, even though a 10% higher FIT on validation data for both top and reboiler temperature is achieved.

4.6 Model identification - Least Squares Support Vector Machines

Up to now, linear black-box models have been identified to obtain a model for control. The LS-SVM model identified in this section will not be used for control. The LS-SVM models are used to investigate whether it is possible to increase the accuracy of the simulation for the top temperature. The model identification of the pilot-scale distillation column based on an LS-SVM has been described previously in [26]. The excellent results in that paper have been obtained using the estimation data set as described in Section 4.2.1. However, the original data is resampled with a sample interval of 10 instead of 60 seconds employed for the linear models in this work. Nevertheless a sample interval of one minute is used for the model predictive controller, a more accurate model can be obtained using more intermediate data points. This data set is split up into an estimation data set of the first two thirds and a validation data set of the last one third. In this field of research, the estimation data set and validation data set are also called the training data set the test data set, respectively. As a result, the training and test data set have identical excitation frequencies [22]. A more realistic situation for control, however, is a training and test data set that have (slightly) different excitation frequencies. When using the same training and test data set as for the linear models, two data sets with slightly different excitation frequencies are used. Both cases will be compared. Table 4.7 presents the results for the identification procedure in [26].

On the training data set, a 10-fold crossvalidation is performed. 10-fold crossvalidation divides a data set in 10 equal parts and uses 9 as training data, while the remaining part is used as validation data. This is repeated 10 times (folds) with each time a different validation data set. The validation results are averaged over the different training rounds. For the top temperature, the mean

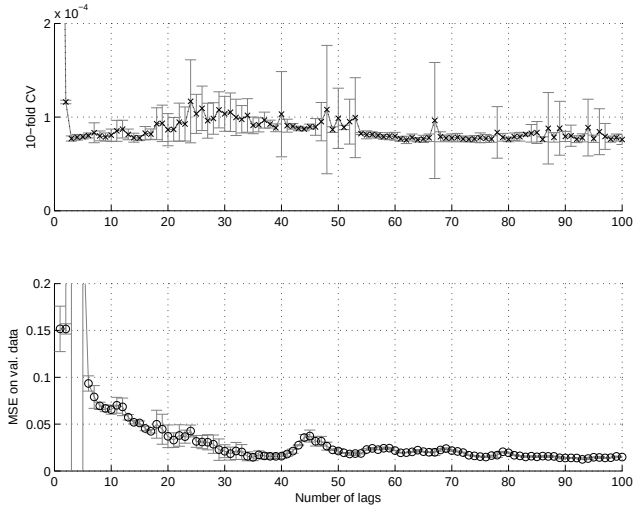
Table 4.7: The model quality parameters FIT and MSE on test data and the model selection criterium AIC on training data for LS-SVM models for the top and reboiler temperature as presented in [26].

Lags	FIT	AIC	MSE
$-top-$			
21	70.5	-2.77	0.0329
35	78.1	-2.84	0.0244
45	59.2	-1.45	0.0373
54	70.6	-1.74	0.0186
$-reboiler-$			
13	80.0	-3.80	0.0536
34	77.4	-2.69	0.00209
43	71.7	-2.69	0.00206
100	80.2	-2.388	0.00109

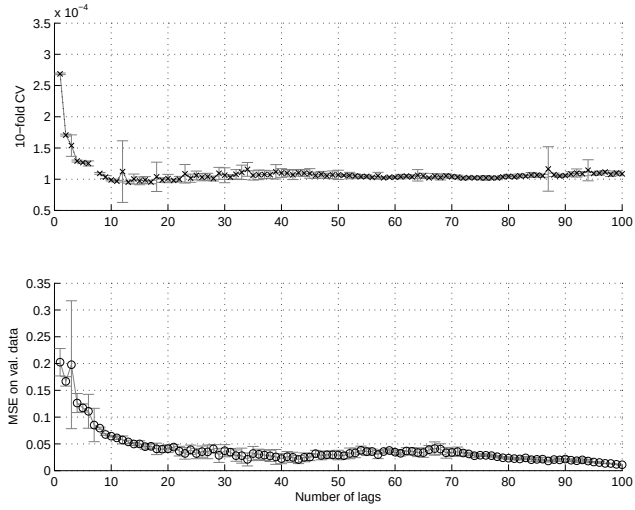
squared error of cross-validation (CV) is plotted in Figure 4.8a (top). The CV value evolves very fast to a value around $7.5 \cdot 10^{-5}$. Already at lag three this is achieved. Augmenting the lags does not decrease the CV value anymore. The observation that there is no clear minimum, indicates the possibility of the model class not being entirely correct. Although one cannot define a clear minimum, one can choose a lag value higher than three to represent a model in this model class. The error bars on the Figures 4.8 and 4.9 are the standard deviation on 10 repeated identifications for the same lag.

Based on the MSE on the test data set, depicted in Figure 4.8a (bottom), the model with a number of lags equal to 35 can be selected. The MSE value is 0.0244 and the FIT is 78%, which is 10% better than the best linear models. The main trend, seen in this figure, is a decrease of the MSE up to lag 40. Between 30 and 40 lags, a valley is observed. From lag 40 on, the MSE increases to lag 45 but afterwards decreases again. A low MSE, combined with a low lag number is situated between lags 30 and 40. Hence, the model with the lags equal to 35 is selected. The error bar, representing the standard deviation for 10 experiments, for the MSE suggests that the error on the MSE decreases for higher lags.

Figure 4.8b (top) displays the 10-fold CV for different lags of the training data set representing the bottom temperature. A minimum is observed at lag 13. The corresponding MSE on test data is 0.053. The 10-fold CV is continually decreasing until lag 13 and slowly increasing for higher lags. In Figure 4.8b, the MSE on test data is plotted. The lowest MSE values are found between lag 30 and 50. The CV and MSE fluctuate hardly in this region. For the model with



(a) Top temperature.



(b) Reboiler temperature.

Figure 4.8: 10-fold CV on training data and the MSE on test data for the LS-SVM models as original published in [26].

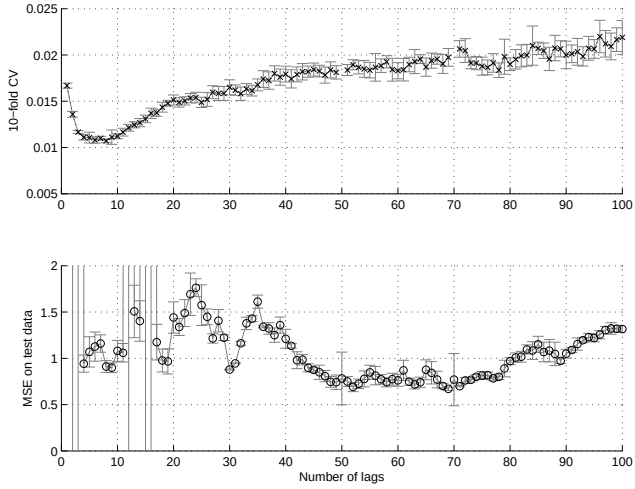
lag 34, a FIT value, AIC value and the MSE are presented in Table 4.7. Based on the plots in Figure 4.8b and supported by the values presented in Table 4.7, the model with lag 13 is selected. Only when high accuracy is needed one can opt for the model with lag 34 as the MSE on test data decreased with a factor two compared to the selected LS-SVM model with lag 13.

When the training and test data set is changed from the sets with discretization interval of 10 s to these used for the linear identification in this dissertation with discretization interval of 60 s, a test data set excited at different frequencies has been used. Again, 10-fold crossvalidation is performed on the training data set and plotted together with the corresponding MSE on test data in Figures 4.9a and 4.9b for the top and reboiler temperature respectively. For each lag, the model identification is repeated 10 times. The plots present the mean value and the error bar representing the standard deviations for the CV and the MSE.

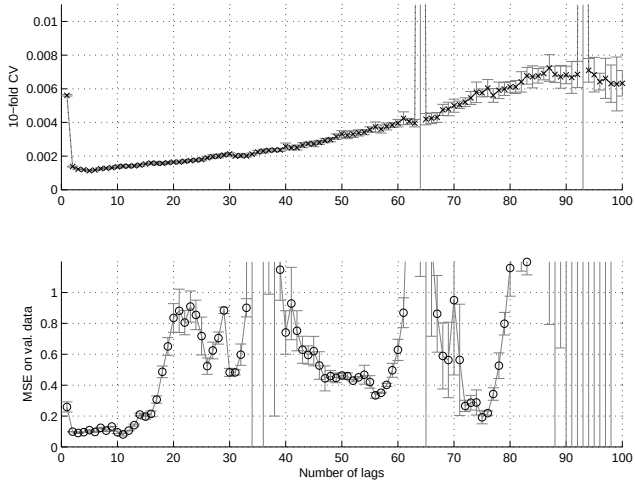
The CV for the top (Figure 4.9a, top) presents, in contrast to the former results, a clear minimum for lag 8. For two lags, the standard deviation is much higher compared to the other lags caused by some outliers. As the CV is already higher than for the selected model, these lags are of less importance for model selection, given the upward trend for the CV value. The MSE on test for the top temperature is unsteady with a large variance for the first 5 lags and lags 12, 15 and 16. For lag 8 the standard deviation is limited and thus the plotted MSE can be trusted. However, the MSE for this lag is not minimal. This is the case for lag 69, but the CV and AIC value suggest this is not a good candidate model. The results are presented in Table 4.8, where also AIC and FIT are calculated. For lag 8, the corresponding MSE and FIT values on test data are 0.91 and 17%. This is nowhere near the more than 70% FIT reached by the linear models. It is clear from the MSE and FIT values in Table 4.8 that the LS-SVM model is not a good candidate for simulation if the system is not excited with an excitation signal containing frequencies also present in the training data set.

In Figure 4.7a on page 83, the validation plot is displayed. Although the overall shape of the plot resembles the measured temperature well, the plot is shifted down almost 1°C. Most probably, a to the model unknown excitation frequency causes this shift. For use in control applications, different excitation frequencies during control and model estimation are to be expected. As the identified LS-SVM model reacts in an undesired way to these unknown frequencies, this observation suggests more research has to be done on LS-SVMs before they can be used safely for control of distillation columns. Table 4.8 also presents the maximum FIT and minimum MSE for 69 lags, but this model also performs worse compared to linear models.

For the reboiler temperature, the lowest CV in Figure 4.9b (top) selects a model



(a) Top temperature.



(b) Reboiler temperature.

Figure 4.9: 10-fold CV on training data and the MSE on test data for the LS-SVM model with the same training and test data sets as used for linear identification.

Table 4.8: Model quality parameters FIT and MSE on test data and the model selection criterium AIC on training data for LS-SVM models for the top and reboiler temperature .

lags	FIT	AIC	MSE
−top−			
6	10.1	0.84293	1.1266
8	17.8	0.66086	0.9117
10	11.9	1.2653	1.0804
69	27.6	1.8208	0.6710
−reboiler−			
5	74.6	-1.5781	0.1090
11	78.4	-1.6822	0.0802
26	45.6	0.92874	0.5234
33	28.7	1.9141	0.9012

with lag 5. In the bottom plot the minimum MSE on test data selects a lag 11. For both CV and MSE, a clear minimum is observed in this region. Based on the AIC value as presented in Table 4.8, the model with lag 11 is preferred. The presented FIT is similar to the linear models. In Figure 4.7 the simulation plot resembles the validation plots of the linear models. The MSE on the test data set starts fluctuating for lags higher than 30. When calculating the MSE and FIT on the training data set, the FIT is nearly 100 and the MSE is $1 \cdot 10^{-8}$. Lags higher than 30 are to be neglected as the data set is too small to identify accurate models. Given the higher FIT and lower MSE for the OE model, there is no need for an LS-SVM when accurate simulation results are needed. The linear OE model is sufficient in the current situation.

The identification on both the 10 s and 60 s interval learns that the LS-SVM is not significantly more accurate than the linear models for the pilot-scale distillation column. Certainly while using excitation signals with different excitation frequencies, the LS-SVM is not able to present a more accurate simulation for the top temperature. For the reboiler temperature, the LS-SVM model simulates not significantly better than the linear models, but the results are similar.

4.7 Selected model - model analysis

The model selected for use in the model predictive controller, is the reduced state-space model obtained by conversion to state-space of the MIMO transfer

function model presented in Equation (4.1). To investigate the coupling between the input and output for the actually employed model, the Relative Gain Array (RGA) [19, 61, 75] is used. This array is a measure for the influence of a control variable, relative to the other controller variables. Elements close to one are the preferred control variables for an output. Elements close to zero are of less influence for the corresponding output. Elements much higher than one have a large influence [77]. Very high or low influences are not desired for control. Input-output connections with negative elements indicate an inverse reaction, but similar to positive values, the elements close to -1 are preferred. The RGA from Equation (4.1) is presented in Table 4.9. It indicates a strong

Table 4.9: Relative Gain Array for the MIMO transfer function model (Equation (4.1)).

	Feed flow rate	Feed power	Reboiler power	Reflux flow rate
Top temperature	0.3688	-0.0411	-1.0201	1.6923
Reboiler temperature	-0.1792	0.0783	1.9547	-0.8537

influence of the reflux flow rate and the reboiler power on the top temperature. The indicated values correspond to the normalized in- and outputs. Increasing the reboiler power reduces the top temperature in an almost one to one relation. This can be physically explained. Due to the increase of energy (heat) a better separation of the two products is obtained which is seen as a lower top temperature (and higher reboiler temperature). According to the RGA, an increase of the reflux flow rate increases the top temperature. It would be expected that if the distillate flow rate would have been constant, an increase of reflux will result in a decrease of the top temperature as a result of an increase of the inflow of the cooled reflux flow at the top of the column. More recirculation of the reflux normally also means a better separation of the two product. This is an inconsistency in the relations between the in- and outputs. However, note that the employed model is a *dynamic* model identified *around an operating point*. Moreover, the identification experiments excite *all input variables at the same time* and the distillate flow rate is adapted to maintain a constant reflux drum level. This can explain the inconsistency in the RGA. Individual relations between the different in- and outputs are not recorded separately and can therefore not be experimentally verified with the employed experiments.

Both the feed flow rate and feed power are of minor importance for the top temperature. The feed power RGA value is almost zero and this indicates it can hardly be used to control the top temperature. Their relation can be explained logically. Increasing the feed flow rate adds extra (cold) fluid to the system which result is a worse separation seen as an increase of the top temperature.

An extra heated feed adds energy to the system resulting in a better separation seen as a lower top temperature.

The reboiler temperature is again strongly determined by the reboiler power and by the reflux flow rate. A change in normalized reboiler power nearly doubles the normalized reboiler temperature. An increase of the reflux flow rate decreases the reboiler temperature in the same order of magnitude. These relations can be explained physically. An increase of reboiler power will increase the energy and thus the temperature in the reboiler vessel. Adding extra cooled reflux fluid will decrease the temperature of the boiler. Both feed variables hardly influence the reboiler temperature. The physical explanation of the RGA elements for these relations are consistent. Extra feed flow will decrease the reboiler temperature as more product is supplied in the same time for the same amount of energy. Increasing the temperature of the feed causes the reboiler temperature to be higher if the same amount of energy is supplied to the reboiler.

To summarize, when adapting the top temperature, only the reflux flow rate has to be manipulated and the reboiler power has to be kept constant. This manipulation is exactly the opposite for the adaptation of the reboiler temperature, which demonstrates that control is difficult. Manipulating one of the two temperatures will always disturb the other, as the temperatures are linked. Manipulation of both the feed flow rate and feed power has a low impact on the controlled temperatures. As a result of the very low impact of the feed power, one can suggest to eliminate this input for the model. The preferred inputs to manipulate both temperatures are the reboiler power and the reflux flow rate.

The condition number, the ratio of maximum singular value divided by the minimum singular value of the gain matrix for this transfer function is 6.49. This high number indicates an ill conditioned control problem [78]. As a result, control will be sensitive to model uncertainties.

4.8 Summary and remarks on the identification results

Several linear model classes have been used to identify a linear model. A transfer function model has been chosen as the controller model for model predictive control experiments. The identification results demonstrate that the top temperature is the most difficult to model. Some nonlinearities for the top temperature are difficult to cover. On the other hand, the reboiler temperature

is easier to model accurately. The implemented model for MPC will be a state-space model with 13 states. If needed, a subspace state-space model with 11 states which offers slightly lower performance is available. This model can be used if the number of states needs to be as low as possible. For the polynomial models, the OE model class seems to be the most appropriate. Given the large order of the resulting state-space model after conversion, it will not be used. Nevertheless, these models present the best performance on the validation data.

The performance of the LS-SVM model for the reboiler is similar to the performance of the linear models, but the results for the top temperature are less accurate in case the excitation frequencies of the validation and excitation data differ. If the validation and estimation data are excited with the same frequencies, the LS-SVM model performs 10% better than the linear models. In a more realistic case where excitation frequencies not available in the estimation data set are likely to appear on the distillation column, the identified linear models are expected to be more accurate than the LS-SVM.

Analysis on the final model based on the relative gain array demonstrates that control of the column will be difficult. There exists a strong coupling between the different in- and outputs. Manipulating one input will result in a change on both outputs.

4.9 Model predictive control approaches

The reduced state-space model obtained from the MIMO transfer function model has been selected as the model to be implemented in the model predictive controller. In the following sections, this model will be used for model predictive control hosted by a PC, PAC and PLC. Some background information on the MPC experiments is justified as it can explain certain, maybe unexpected, evolutions in the following sections.

Collecting the identification data and first MPC experiments on the pilot-scale distillation column was started in summer 2009 and the MPC experiment hosted by the PAC took place in autumn 2012. During this period, the column has been operated several times by students causing a concentration change by removing distillate. In spring 2010, the collecting vessel of the column was refilled to a concentration of 50% for both methanol and isopropanol. Consequently, the concentration of the feed for the different experiments is different due to the refilling, but also by evaporation. The identification and MPC experiments with MPC hosted by a PC took place in 2009. It can be expected that the identified model matches the column as accurately as the model structure could reach. The MPC experiments hosted by a PAC took place in the autumn of 2012. Due to the changing safety regulations at KU Leuven, the latter experiments had to be restricted in number and in time. Although a re-identification of the column was recommended, the model identified in 2009 has been reused. Given the changed concentration of the column feed, this model is most likely not the most accurate, but neither completely wrong. However, it will certainly affect the quality of control. To anticipate, the settings of the MPC controller are different for the PC and PAC experiments. Note, that the aim of this dissertation is not to obtain the best possible control for the pilot-scale distillation column, but to investigate the use of industrial hardware for MPC control. Differences in the quality of control are to be expected, but they do not interfere with conclusions concerning the hardware.

It is also important to note that the experiments on the distillation set-up hosted on PC are performed to investigate and list the problems that can occur while performing MPC experiments on this set-up that was intentionally built to run steady state experiments. Experiments on the PAC focus more on the aspect of MPC control on industrial hardware.

The following MPC experiments are discussed:

1. An MPC simulation on PC, focussing on the different algorithms that can be used on the PAC.

2. An MPC experiment on the pilot-scale distillation column with an MPC hosted by a PC, focussing on the problems that might occur when using MPC control on this set-up.

3. An MPC experiment on the pilot-scale distillation column with an MPC hosted by a PAC. The focus of these experiments is on the online behavior of the algorithms hosted on industrial hardware.

4. Hardware-in-the-loop experiments on PLC. These experiments focus on the feasibility of employing MPC on a PLC for the pilot-scale distillation column.

4.10 Model predictive control simulations on a PC

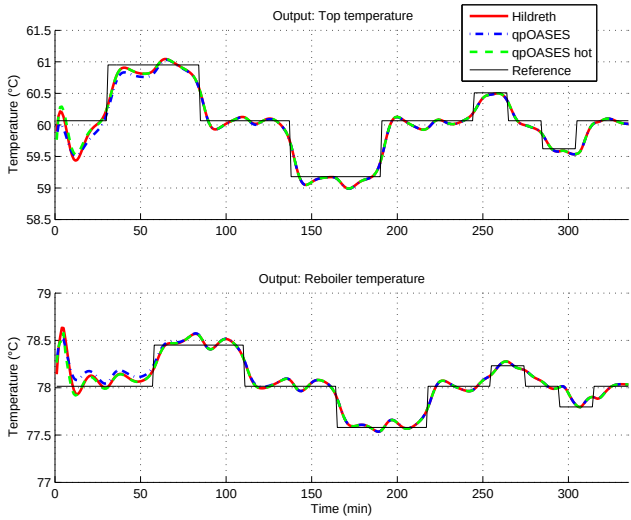
4.10.1 Model predictive control settings

The different MPC horizons are determined in a trial-and-error experiment and are set to 10 steps for the control horizon ($H_c = 10$) and 50 steps for the prediction horizon ($H_p = 50$). Each step takes one minute. This leads to a hessian of size 40×40 in Equation (2.30). The weight matrix W_y is a diagonal matrix with elements $W_{y_{11}} = 1$ and $W_{y_{22}} = 0.9$. This punishes each deviation from the top temperature reference slightly more than a deviation from the reboiler temperature. The weight matrix W_u has four elements on the diagonal $W_{u_{11}} = 0.8$, $W_{u_{22}} = 1$, $W_{u_{33}} = 0.8$ and $W_{u_{44}} = 1$ for the feed flow rate, the feed duty, the reboiler duty and the reflux flow rate, respectively. This choice has been made to encourage the use of the flow rates, which are the fastest varying control variables. All of the off-diagonal elements are taken equal to 0.

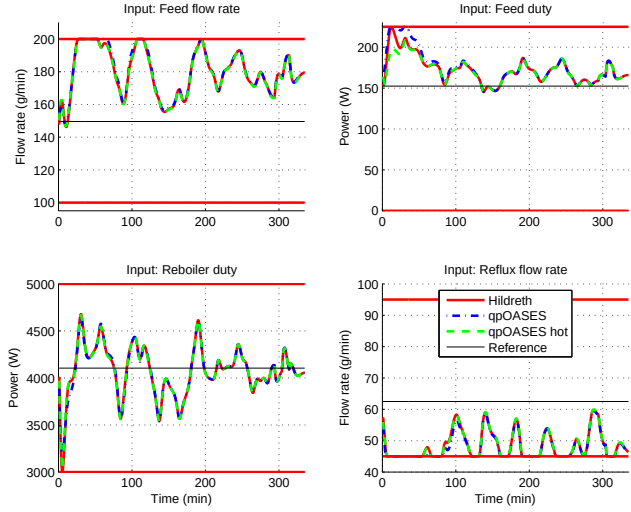
As a reference trajectory for both the top and reboiler temperature, a sequence of steps is applied. The size of these steps for the top temperature are 0.88°C and 0.62°C. For the reboiler the large and small steps are 0.44 and 0.31°C, respectively.

4.10.2 Matlab simulation experiments

Figure 4.10 depicts the in- and outputs of the MPC simulation experiments for three different QP algorithms: the Hildreth algorithm, the qpOASES algorithm and the qpOASES algorithm using a hot start. Unfortunately, the latter cannot be used on the PAC system. The combination of the graphical LabVIEW VI



(a) The measured temperatures, the controlled outputs of the distillation column.



(b) The inputs applied to the distillation column.

Figure 4.10: The in- and outputs for the Hildreth and qpOASES algorithm in simulation on a PC.

and C-code programming causes the memory of the C-coded library to be cleared every time the MPC function is finished. Storing data for use in the next cycle is impossible. Despite this limitation of the PAC, hot starting is investigated in simulation. To make the control a little bit more challenging, the reference for the top and reboiler temperature are shifted respectively +0.3 and -0.3°C compared to the mean temperature of the identification experiment.

The two controlled temperatures are plotted in Figure 4.10a. The three different experiments start from a similar, although not identical state. The effect of this different initial condition has faded after 60 minutes. The overall shapes of both the top and reboiler temperature follow their respective references correctly, however it is clear that the system is in constant change. This result is not surprising, as the largest time constant in the linear model is in the order of 2500 seconds. Steady-state is only expected after three time constants. Consequently, steady-state is never reached. It has to be noted that experiments on the pilot-scale set-up were restricted in time fore safety reasons. The total time of an experiment had to fit in 10 hours. Stretching the depicted reference track would lead to an experiment of more than eight hours plus the required additional start-up time.

In Figure 4.10b, the inputs are plotted. In contrast with the experiments on the air heating set-up, a set-point for the inputs is supplied. The nominal working conditions are considered to be the input reference. The constraints for each input are depicted. These are never violated, but the flow rates hit the constraint several times. The lower constraint for the reflux flow rate is hit at least six times, all together for 134 minutes. This is more than one third of the length of the simulation. The control also never reaches the nominal flow rate of 65 g/min. The feed flow rate varies between the nominal working flow rate of 150 g/min and the upper constraint. Only in the first minutes of the simulation the flow rate is below 150 g/min. The upper constraint is reached three times. The first time for 23 minutes and the next times only for short periods compared to the reflux flow rate. The input feed duty resides between the upper constraint and the nominal value. These control actions are needed to keep the feed that enters the column sufficiently warm. The reboiler power, one of the most influencing inputs, only reaches the constraints shortly at the start and fluctuates thereafter around the nominal working point.

Figure 4.11 presents the number of iterations needed to solve the problem. Similar to the air heating set-up, the Hildreth algorithm needs more iterations to solve the problem. To have an impression of the speed of the different simulations, the total runtime and the mean time for one iteration for Matlab experiments are presented in Table 4.10. The MSE and SSE values in Table 4.10 are calculated on the difference between the temperatures and their corresponding reference. They are exactly the same for the three algorithms as

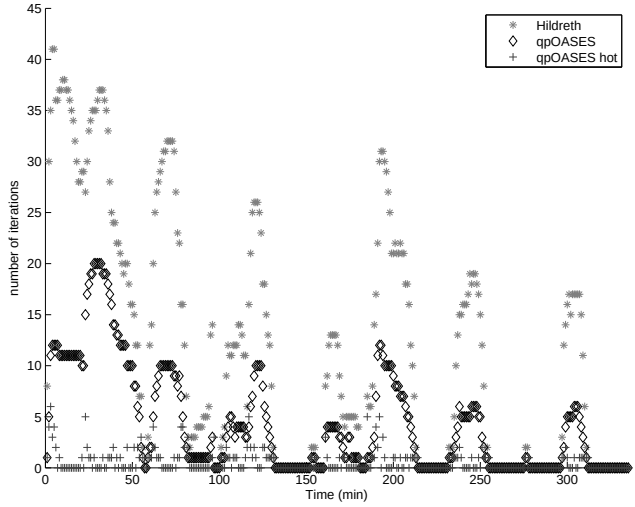


Figure 4.11: Number of iterations needed to solve the QP problem for simulation on the PC for both the Hildreth an qpOASES algorithms. For qpOASES, the number of iterations in case of hot starting is also presented.

is to be expected because the MPC setting are identical.

Table 4.10: The MSE and SSE values calculated for the simulation experiments in Matlab on PC. The total run time and the mean, minimum and maximum run times for one MPC iteration is presented to indicate the speed of the algorithms.

	run-time (ms)				error	
	total	mean	min.	max.	MSE	SSE
- Top temperature -						
Hildreth	3130	6.67	0.21	108	0.0254	8.52
qpOASES	114	0.24	0.11	2.15	0.0254	8.52
qpOASES with hot start	54	0.11	0.06	3.16	0.0254	8.52
- Reboiler temperature -						
Hildreth					0.0105	3.50
qpOASES					0.0105	3.50
qpOASES with hot start					0.0105	3.50

Two important observations for future experiments can be seen in Table 4.10. First, in contrast to the similar run times for the small heating set-up in

simulation, Hildreth now needs significantly more time to present the solution than qpOASES. Second, the clear reduction of required iterations for qpOASES with hot starting leads again to a clear reduction in required calculation time. As mentioned earlier, hot starting is not possible on the employed PAC.

4.11 Model predictive control on a PC

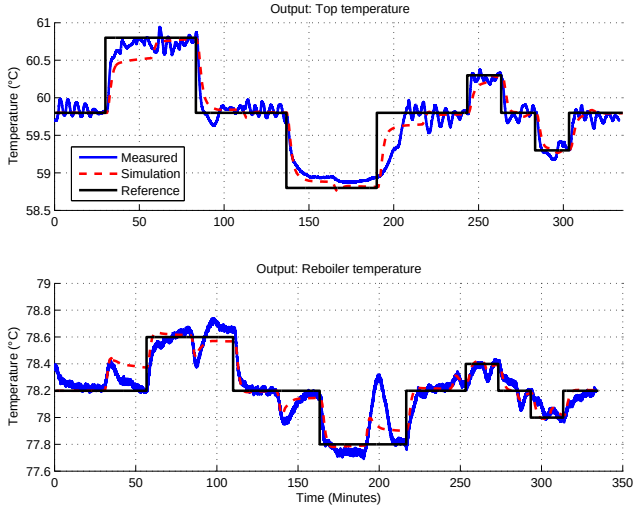
4.11.1 Model predictive control settings

These experiments are the first MPC experiments on the pilot-scale set-up. The employed MPC controller is the one integrated in the Matlab Model Predictive Control Toolbox. To make sure the nonlinear column can be controlled with the linear model, the control and prediction horizon are selected based on the time delays and time constants appearing in transfer function model. Three times the longest time constant is close to 150 minutes. The actual prediction horizon is restricted to 100 minutes. The control horizon is set to 30 minutes. In the cost function (Equation (2.22)), the weight matrices W_y is a diagonal matrix with elements $W_{y11} = 1$ and $W_{y22} = 0.9$. This punishes each deviation from the top temperature reference slightly more than a deviation from the reboiler temperature. The weight matrix W_u has four elements on the diagonal $W_{u_{11}} = 0.1$, $W_{u_{22}} = 0.2$, $W_{u_{33}} = 0.2$ and $W_{u_{44}} = 0.1$ for the feed flow rate, the feed power, the reboiler power and reflux flow rate, respectively. For this experiment, the Matlab MPC control toolbox also needs a weight matrix on the change of the input. A diagonal matrix with elements $W_{\Delta u_{11}} = 0.05$, $W_{\Delta u_{22}} = 0.1$, $W_{\Delta u_{33}} = 0.1$ and $W_{\Delta u_{44}} = 0.1$ has been used.

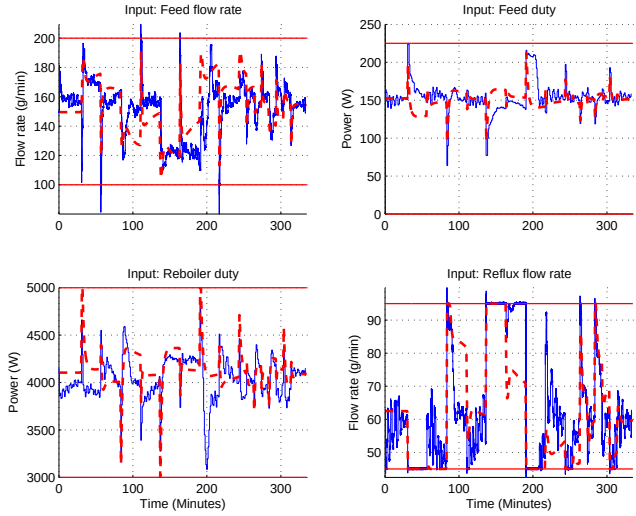
The values of the different input signals are constrained. The reboiler power is bound between 3000 and 5000 W and the reflux flow rate is allowed to move in a range between 45 and 95 g/min. The constraints for the feed flow rate are 100 and 200 g/min and the feed duty varies between 0 and 225 W. No constraints are set for the states or outputs.

4.11.2 Simulation with the Matlab MPC toolbox

Before the actual experiments, Matlab simulations are carried out. As mentioned earlier, the Matlab simulation and experiment on the pilot-scale distillation column, both using the Matlab Model Predictive Control Toolbox, are performed before any other experiment described in this chapter. The aim was to investigate the behavior of the column during dynamic experiments.



(a) The measured temperatures, the controlled outputs of the distillation column.



(b) The inputs applied to the distillation column.

Figure 4.12: The in- and outputs for the MPC simulation experiment and the experiment on the set-up with the Matlab Model Predictive Control toolbox running on PC.

Both the simulation model and the controller model are the reduced state-space model obtained by identification with transfer functions. Figure 4.12a illustrates with a dashed line the corresponding simulation for the top and reboiler temperature. The different MPC settings compared to the former section cause the simulated temperatures to be different. The overall shape of the reference temperatures is followed correctly, but also here, the fast changing set-point, alternating for top and reboiler temperature, makes the system to be in constant change. In the interval 30 to 56 minutes, the temperature simulations deviate nearly 0.2°C (reboiler) and 0.3°C (top) from the set-point. In the next interval from 56 to 83 minutes, the reference is followed more accurately. This pattern is repeated during this experiment. A step to a higher/lower temperature for the top temperature with a fixed reboiler temperature, is most difficult to track and causes both bottom temperature to leave the reference. A change in top temperature with fixed bottom temperature is barely seen in the top temperature. According to the RGA, both the reflux flow rate and reboiler power strongly influence the reboiler temperature. Unfortunately, the reflux flow rate influences the top more than the reboiler temperature, so it cannot compensate for the change in reboiler power. Temperature runaways halfway are seen for the top temperature for every set-point change of the reboiler temperature, but as the top temperature is highly influenced by both the reboiler temperature and reflux flow rate, the latter compensates the change in reboiler power quickly. For the reboiler temperature, these deviations from the set-point halfway the steps are not compensated. Consequently, each set-point change for the top temperature causes a clear deviation from the set-point for the reboiler temperature.

The inputs in Figure 4.12b hardly hits the constraint except for the reflux flow rate. This flow rate reaches often the upper or lower constraint. Adapting the constraints is not a good idea for experiments on the distillation set-up, as the flow rate depends on the volume of liquid in the reflux drum. If this drum is empty (or below a safety level), the reflux pump is shut down. To avoid such shutdowns, the lower constraint is left unchanged.

4.11.3 Experiments on the pilot-scale set-up

On the pilot-scale set-up the same reference trajectory as provided in Matlab simulations is applied to both temperatures. As the column is a nonlinear system controlled by a linear controller, significant deviations from the simulation can be expected. Figure 4.12a depicts both simulation and experimental measurements for the outputs of the MPC experiment of the distillation column. The corresponding inputs are plotted in Figure 4.12b.

On the top temperature, a high-frequency oscillation is present during the experiment. This is caused by the PI level controller of the reflux drum. The current PI settings are sufficient for a fixed operating point however, the large hysteresis of the regulating valve forces the tuning of the PI settings for dynamical experiments. In Figure 4.12b, depicting the inputs, all flow rates also violate their constraints. This violation is not a problem of the MPC controller, but is a result of the overshoot of the underlying PI controllers. As a consequence PI controllers at the column were retuned in 2010 [15] to a step response without overshoot in the working range of this MPC controller.

The top temperature follows the reference well, even better than the simulation during the first two hours of the experiment. As soon as the temperature is below the nominal working temperature, the oscillations disappear around 135 minutes, indicating a drastic set-point change for the reflux drum level controller. This change is caused by the reflux flow rate, that switches from a flow rate close to the lower constraint, to a flow rate for a long time at the upper constraint. The temperature itself follows the simulation closely, but since the upper constraint of the reflux flow rate is hit, the cooling caused by the high flow rate is insufficient to bring the temperature to the desired reference temperature. At 190 minutes, the temperature increases again, but this time the lower constraint is hit, slowing down the increase of the temperature. The next set-point changes are limited in size and therefore the reference is followed more closely.

It has to be noticed that the set-point changes are limited. For the top and reboiler temperature, the nominal working temperature is increased and decreased with 0.6°C and 0.4°C, respectively. Notice again, this experiment is originally performed to investigate how the column behaves in case it is MPC controlled.

The measured signal for the reboiler temperature contains no oscillations. The large peaks are caused by a set-point change of the top temperature. As mentioned for the top temperature, the strong dependency between top and reboiler temperature, both mainly controlled by the reflux flow rate and reboiler power, shown by the RGA, leads to a deviation from the set-point for the reboiler temperature as soon as the top temperature changes. For example, at approximately 200 minutes a deviation of 0.5°C can be seen in the reboiler temperature from its set-point. This deviation is an even larger change than the set-point changes for the reboiler temperature. In contrast to the top temperature, the reboiler temperature is mainly determined by the reboiler power during this experiment with its fast changing set-points. The large time constants for the reflux flow rate and the strong link between this flow rate and the top temperature, makes the reflux flow rate of secondary importance for the reboiler temperature. Consequently, similar to the simulation, deviations

for the reboiler temperature are larger than for the top temperature.

Although the quality of the top temperature model is below the model quality of the reboiler temperature, the model is powerful enough to build a controller. No run time measurements for the QP, nor other MPC parameters are logged during the experiment as this experiment was intentionally performed to observe the column behavior in case of dynamic experiments.

One conclusion on this experiment is that the PI controllers had to be tuned for a step answer without overshoot. This was done in spring 2010. Another conclusion is that the model is of sufficient quality for linear MPC control on this set-up. The chosen reference trajectory is certainly not ideal to obtain perfectly followed references, as the frequent set-point changes follow each other rapidly. However, this reference trajectory will be preserved for all experiments, due to the time available for an experiment and the focus on running the model based controller on industrial hardware. This is of course more interesting when the control is challenging.

4.12 Model predictive control on a PAC

4.12.1 Model predictive control settings

The control horizon and the prediction horizon are, together with the weight matrices in the cost function, taken identical to the Matlab simulation experiments in Section 4.10. The only difference to that section is that the 0.3°C shift is removed for the reference trajectory.

4.12.2 Hardware-in-the-loop experiments

To make sure the controlling LabVIEW VI and corresponding C-libraries for the MPC controller are implemented correctly, hardware-in-the-loop experiments are performed. The VI running the MPC controller is connected to a linear model of the pilot-scale distillation column. The employed linear model is a state-space representation of the MIMO transfer function in Equation (4.1). The number of states of this model is 20. The HIL simulations are plotted together with the experiments on the real system in Figures 4.13a and 4.13b. Both the Hildreth and qpOASES algorithm have been employed for HIL experiments. Given the (almost) identical result and corresponding indistinguishable plots, only one HIL experiment is plotted. Similar to other simulations, the overall shape of the reference is followed well but also from

these plots it is clear that the column will never reach a steady-state situation for this experiment. These plots resemble the Matlab simulations, but as the latter are shifted 0.3°C, the results are different. After the HIL experiments, it can now be expected that the LabVIEW VI is working properly and, hence, it is employed on the pilot-scale set-up.

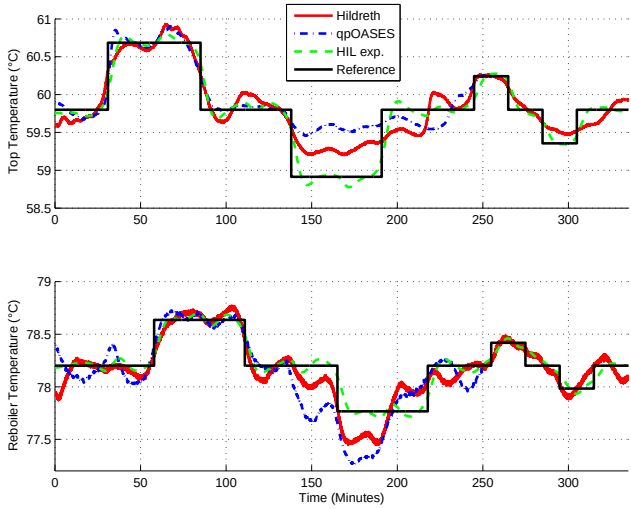
4.12.3 Experiments on the pilot-scale set-up

Figure 4.13a depicts the measured top and reboiler temperature during the experiment. Both controlled variables follow the same reference trajectory as in the HIL experiment. The numerous changes, hence, yield a challenging reference path to be tracked. As mentioned earlier, the time constants of the different subsystems are longer than half an hour, causing the system to never reach steady-state. These references are selected in order to combine the safety regulations of the column with a sufficient number of different steps.

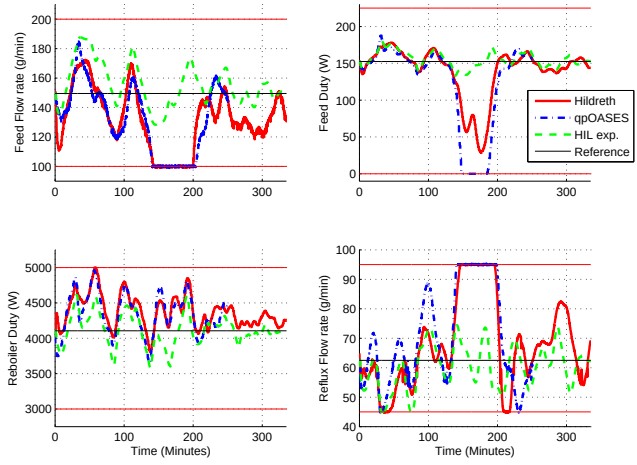
Two experiments are recorded. First, an experiment with the Hildreth algorithm used as QP solver. A second experiment employs qpOASES as solver. Unfortunately, the environmental conditions were different. This is of major importance as the pilot-scale distillation set-up is not thermally insulated from its environment. As a consequence heat losses depend largely on the environmental conditions. For the Hildreth experiment the environmental temperature was 17°C at noon. At night the temperature was not lower than 14°C. For the second experiment, the temperature at noon was 10°C and close to 0° at night.

In the first 120 minutes of the experiment, both the top and reboiler temperatures are followed quite accurately (Figure 4.13a). The plotted temperature obtained from hardware-in-the-loop experiments can be considered as the temperature to be tracked. The differences between the HIL experiments and the experiments on the set-up are limited for both QP solvers. The inputs, presented in Figure 4.13b, also differ only slightly form the HIL experiment. Although more than three years have passed since the identification experiment, the model is apparently still accurate enough for control on the real set-up.

However, between 120 and 240 minutes, both temperatures have to follow a step down in the temperature. As soon as the top temperature reference starts decreasing, both the top and reboiler temperature decrease. For the experiments with the Hildreth solver, corresponding to an ambient temperature of 17°C at noon, the top temperature only decreases half the applied temperature step for the top temperature. This temperature stays constant for half an hour and increases slightly to 220 minutes. This is the moment both the top and reboiler reference are back at the nominal temperatures. The top



(a) The measured temperatures, the controlled outputs of the distillation column.



(b) The inputs applied to the distillation column.

Figure 4.13: Measured outputs and inputs for experiments on the distillation column and HIL experiments tracking the desired reference temperature profile with an MPC controller running on the PAC.

temperature evolves now quickly to this temperature. The measured reboiler temperature is approximately 0.25°C below the reference in the interval 140 to 190 minutes. From 200 minutes on, this temperature is close to the nominal reboiler temperature, approximately 0.25 °C above its reference.

The experiment with the qpOASES presents an even more problematic tracking in the interval 120 to 240 minutes. The top temperature hardly leaves the nominal temperature, the reboiler temperature is even further below the reference in the interval 140 to 190 minutes, but similar to the Hildreth algorithm for 200 minutes on. The low environmental temperature for the qpOASES experiment causes the controller variables to hit the constraint faster than for the Hildreth experiment as can be seen in Figure 4.13b. From 200 minutes on, the constraints are left for the flow rates and feed temperature. It is expected that the MPC controller recovers quickly. However, due to safety regulations the experiment had to be stopped here. As both the experiment with Hildreth and the HIL experiments deliver almost identical results the qpOASES experiment is most likely capable of tracking the small sequence of steps from 250 minutes on.

It is clear from the plot in Figure 4.13a that the top and reboiler temperature do not follow their references accurately in this interval. From the input plots presented in Figure 4.13b, one can see that two constraints are reached for Hildreth and three constraints are reached for qpOASES. This, together with an ambient temperature below the ambient temperature of the excitation experiment, causes the inaccurate control. Nevertheless, this difficult control is interesting to investigate the behavior of the different QP solvers.

In Table 4.11 the MSE value for the HIL experiment and experiments on the columns is present. As all three experiments are of different length, the SSE is to be discarded and not displayed. The MSE demonstrates the experiment with the Hildreth algorithm as QP solver is much closer to the reference as the experiment with qpOASES as solver. The large deviations from the reference seen in Figure 4.13 are demonstrated by the numerical values in the table.

Based on these two control experiments, it can be concluded that for this pilot-scale set-up the environmental temperature is of high importance. Although difficult to incorporate in the model, except if a (large) sequence of experiments throughout the year can be set-up, the quality of control can only be considered *good* when the environmental temperature differs only at most 3°C from the one present during the identification experiment. The environmental temperature was 20.6°C during the identification experiment for the distillation column set-up.

Figure 4.13b depicts the four inputs of the distillation column. The two

Table 4.11: The MSE value calculated for hardware-in-the-loop experiments and experiments on the set-up on the PAC. The total run time as well as the mean, minimum and maximum run times for one MPC iteration are presented to indicate the speed of the algorithms.

	run-time (ms)				error
	total	mean	min.	max.	MSE
- Top temperature -					
Hildreth	1655	4.58	0.28	67.59	0.0448
qpOASES	333	1.33	0.84	3.11	0.1078
qpOASES (HIL)	314	0.89	0.83	1.69	0.0154
- Reboiler temperature -					
Hildreth					0.0156
qpOASES					0.0468
qpOASES (HIL)					0.0045

heating powers do not reach the constraints for the Hildreth experiment. Both flow rates touch the constraints. For the qpOASES experiment, all controller variables hit the constraints except the reboiler duty.

The bottom plot in Figure 4.14 depicts the number of iterations needed to solve the QP problem for both the Hildreth and qpOASES experiment together with the HIL experiment. All algorithms need between 10 and 15 iterations to solve the MPC problem around 30 minutes after the start. One active constraint on the reflux flow rate is responsible for this peak in iterations. For the HIL experiment, no more than one iteration is needed to present a solution, demonstrating no constraints are hit anymore. For the two experiments on the real set-up, the number of iterations increases fast as soon as two or more constraints are hit in the interval from 140 to 210 minutes. The maximum number of iterations for Hildreth is 70 and for qpOASES 30. The conclusions drawn from the experiments on the small air heating set-up are confirmed. The number of iterations for qpOASES is lower to solve the same problem. For practical use, the required time is more critical. The number of elements in the hessian of the QP problem increases from 196 to 1600 compared to the air heating set-up. Based on the two plots displaying the required time, the upper left plot in Figure 4.14 presents the needed time for each calculation of an optimal input, the upper right is a blow-up of the left plot, limited to the first 0.4 ms on the vertical axis. As can be seen, the Hildreth algorithm needs more time than the qpOASES algorithm if the number of iterations is above 20. qpOASES needs maximum 0.32 ms to solve the problem. Hildreth multiplies this value at some iterations by almost 10. This confirms the suggestion made

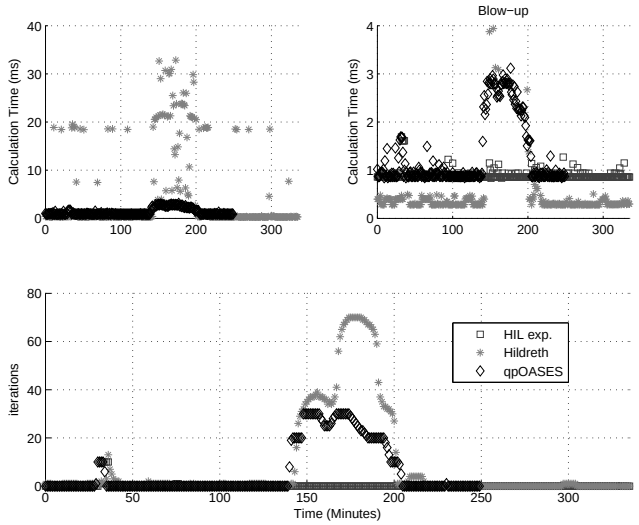


Figure 4.14: Calculation time and number of iterations for the employed QP algorithms on the PAC.

in the conclusions of the previous chapter. Large systems, and this column can be considered between small and average scale on industrial level, benefit from this new, state-of-the-art solver qpOASES.

Table 4.11 presents the mean time for one iteration and total runtime for the experiment. The mean time for one iteration for Hildreth is higher than for the qpOASES algorithm. This demonstrates that the increased complexity for qpOASES leads to a decrease in computation time, but the system has to be *sufficiently large*. The cause of the peaks seen in the calculation time (upper left plot in Figure 4.14) is not clear, but most likely a hardware interrupt for communication or module detection causes the algorithm to pause for some time. This phenomenon is not seen for the qpOASES algorithm as these modules were removed from the CompactRIO device. Table 4.11 also indicates some calculation times for the QP solvers. It is clear that the Hildreth algorithm needs significantly more time to solve the QP. Note that the maximum time required for Hildreth is more than 20 times higher than the maximum required time for qpOASES.

The PAC has clearly sufficient computation power to run the MPC controller and the PI controllers together with the data logging routines. Whichever QP solver is used, the computation time is below 5 ms, far below the one minute update interval between two MPC inputs. The working memory used, is around

70 MB. Note that this value contains the controller programs and the operation system memory demands. The question of whether MPC can be applied on a PAC for this pilot-scale distillation column, can be answered positively.

It has to be noted that the difficult control of the temperatures, seen in sometimes large deviations from the temperature reference, was predicted by the relative gain array. Moreover, as two of the four inputs hardly have any effect on the controlled temperatures and the system is under constant change, the temperature reference cannot be tracked perfectly.

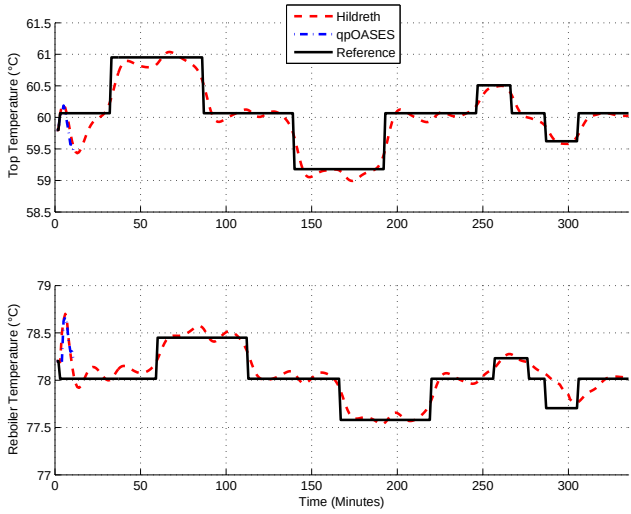
4.13 Model predictive control on a PLC

The ultimate goal of this work is to run MPC on a PLC. After succeeding in this for the small air heating set-up, the pilot-scale column is the next, and for industry use more realistic, target. However, given the safety regulations, hardware-in-the-loop experiments are carried out first. Only after successful HIL experiments, practical experiments on the set-up can be considered.

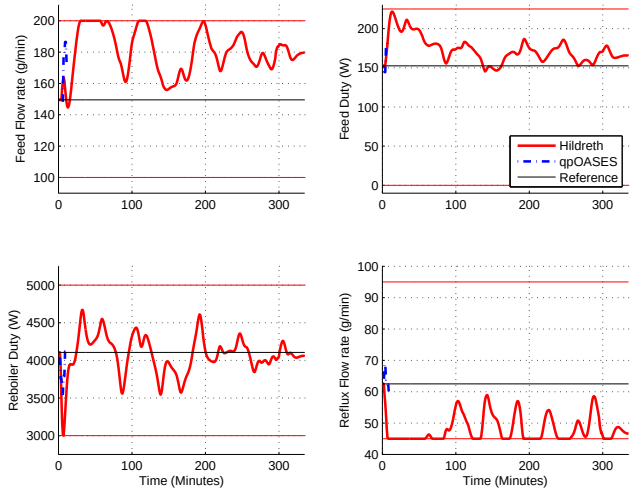
4.13.1 Hardware-in-the-loop experiments

In Figure 4.15, the in- and outputs are plotted for the HIL experiments on PLC. The settings for the MPC are taken identical to the Matlab experiments. The shift of plus 0.3 and -0.3°C for the top and reboiler references respectively is also applied to make the control a little bit more challenging.

For the Hildreth algorithm, the results are close to the Matlab simulations. There is a small difference as clarified by the MSE values for the Matlab and HIL experiments on PLC as presented in Table 4.12. Also the number of iterations presented in Figure 4.16 is slightly different. With the identical start-up conditions, MPC settings and the stop criterium of the algorithm, these differences are traced down to the different numerical precision, i.e., the difference between the 32-bit and 64-bit floating point numbers for the PLC and PC, respectively. The required time to solve the QP is plotted in the upper plot of Figure 4.16. The maximum required time is around 556 ms for 35 iterations. The indicated minimum value for the run times of 0 ms is caused by the fact that the QP solver is not called. This does not mean the calculation time to present a new input is zero. The MSE value in Table 4.12 is different from these in Table 4.10, certainly for the reboiler temperature. This is caused by the way the reference trajectory is passed to the model predictive controller at the end of the experiment. Unfortunately this reference is different for time



(a) The measured temperatures, the controlled outputs of the distillation column.



(b) The inputs applied to the distillation column.

Figure 4.15: Measured outputs for the top- and reboiler temperature for a hardware-in-the-loop experiment tracking a desired reference temperature profile with a MPC controller running on the PLC.

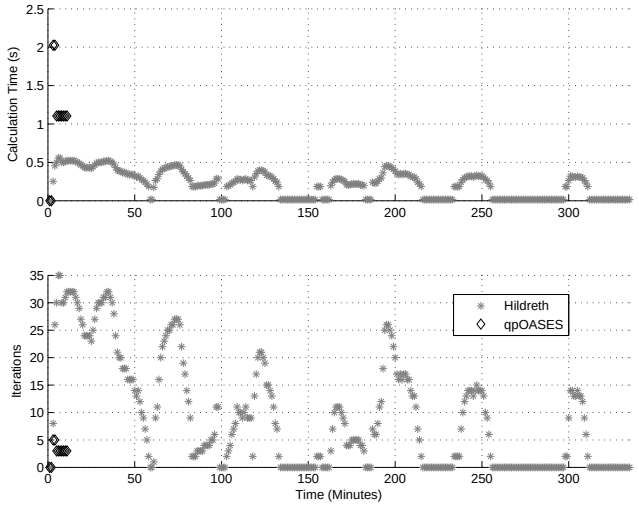


Figure 4.16: Calculation time and number of iterations for the employed QP algorithms for a hardware-in-the-loop experiment with a MPC controller running on the PLC.

steps beyond 380 s after the start of the experiment compared to the Matlab experiments in Section 4.10. This influences the controlled temperature slightly in the last steps at the end of the experiment.

Worthwhile to mention is that in trials done as preparation for this experiment, small mistakes caused the QP solver to reach its maximum number of iterations. For the pilot-scale distillation column HIL experiments on PLC, this maximum is set to 250 iterations. This corresponds to a calculation time of 3970 ms. With a cycle time of 4 seconds for the organization block initiating the MPC solver, this is the absolute maximum allowed calculation time for the maximum number of iterations. On the pilot-scale set-up, an MPC update cycle of 1 minute is selected. The calculation times to solve the QP are far below this cycle time. In case a PLC would be used to control the pilot-scale set-up, the control program running at cycle rate of 100 ms containing the PI controller and logging should run on the same PLC. This will increase the run time of OB35, but it is expected that this increase will be limited to less than 50%. The calculation time of the QP is therefore not considered to be a bottleneck for the implementation of MPC on a PLC for the pilot-scale set-up.

As already mentioned in Section 2.5.3, the other programming issues, e.g., maximum amount of 64 kB for each function block and the difficult initialization of array larger than 255 elements are much harder to take into

Table 4.12: The MSE value calculated for the hardware-in-the-loop and experiments on the column set-up hosted by the PLC. The total, mean, minimum and maximum run times for one MPC iteration is presented to indicate the speed of the algorithms. (F = Failed experiment).

	run-time (ms)				error
	total	mean	min.	max.	MSE
- Top temperature -					
Hildreth on PLC (HIL)	70956	212.4	15	556	0.0253
qpOASES on PLC (HIL)	11780	1308.8	1104	2026	F
Hildreth on PLC	31123	91.0	14	532	0.0445
- Reboiler temperature -					
Hildreth on PLC (HIL)					0.0139
qpOASES on PLC (HIL)					F
Hildreth on PLC					0.0255

account. Based on the latter issues, a practical limit for a QP to be solved with the Hildreth algorithm is a hessian with 64×64 elements. It is possible to increase that size, but then the matrix data are to be spread over different data blocks which makes it even more difficult to manage as a programmer. The memory constraint is not yet reached. According to the online memory status information, the current implementation needs only 4% of the 8 MB available Random Access Memory, 16% of the 1.4 MB working memory and 26% of the 0.7 MB retentive data memory. Hence, there is enough space to add additional control programs and to increase the QP problem size to 64×64 elements as suggested.

For the qpOASES algorithm, the experiments ended abruptly after 8 time steps with a system failure. According to the diagnostic registers, the error was caused by a *scan cycle monitoring time* violation, which means the algorithm needed more than 3.95 s. Increasing the cycle time to 10 seconds, or even one minute, caused the scan cycle monitoring time to be set to its maximum setting of 5999 ms. Unfortunately, even this is not high enough for the qpOASES algorithm. When taking a margin of about 10 percent, one iteration needs between 350 and 450 ms. According to the Matlab simulation on this reference trajectory (Figure 4.11), up to 25 iterations are to be expected. This is at least 8.75 s and far above the maximum allowed calculation time, the scan cycle monitoring time, of 6 s for the PLC. The additional calculations required for this algorithm, need too much time to calculate the solution online. The presented time values in Table 4.12 are calculated on the first 7 time steps. The time step causing an error is not added. The maximum calculation time

indicated in the table corresponds to 5 iterations.

As already mentioned in Section 2.5.3, the practical upper memory limit for this algorithm would have been a hessian of size 44. The required calculation time, however, reduces this upper limit. An experiment with control horizon set to 5 instead of 10, corresponding to a hessian size of 20 was successful. This successful HIL experiment for qpOASES demonstrates the code is working properly. The observation that a control horizon of 10 steps reaches the computing power limits of the employed PLC, which is the most powerful of its series, demonstrates that a PLC does not benefit from a sophisticated solver as, e.g., qpOASES. The speed constraint is the bottleneck for this algorithm. The Hildreth algorithm on the other hand succeeds to solve the MPC problem with control horizon 10. For this algorithm, the memory constraint is reached first.

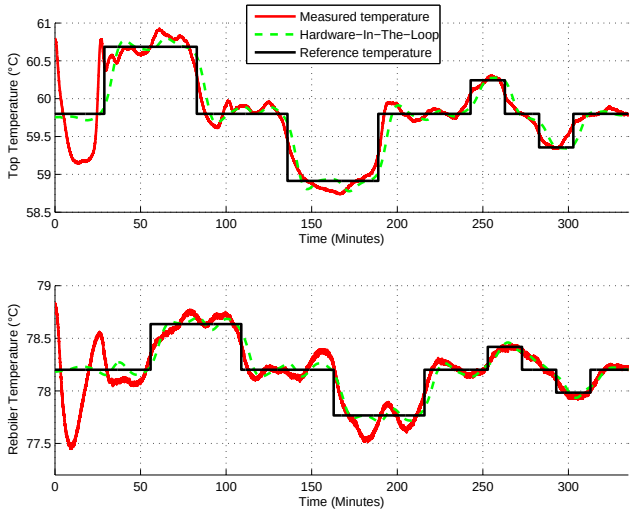
4.13.2 Experiments on the pilot-scale set-up

Nevertheless the unexpected failure of the qpOASES algorithm during the hardware-in-the-loop experiments, experiments on the set-up are performed. Using the Hildreth algorithm, the results plotted in Figure 4.17 are obtained. The additional shift for the reference of 0.3°C employed in the hardware-in-the-loop experiments on the PLC, has been removed. The ambient temperature was 14°C at the start of the experiment and 19°C at noon.

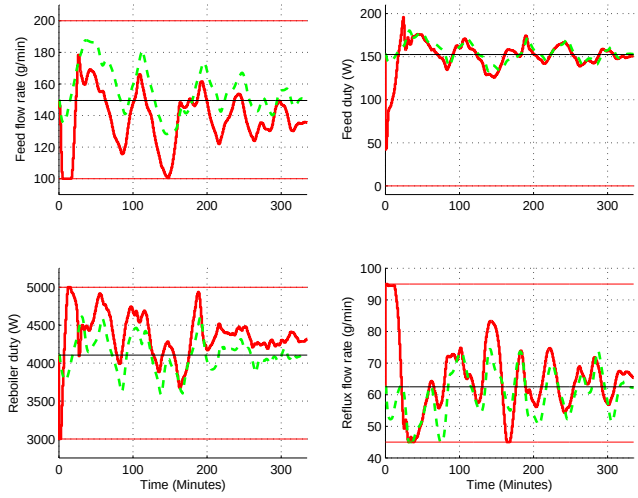
After a transient from the start-up temperatures above the set-point during the first 30 minutes (Figure 4.17a), the reference is followed clearly more accurately than for the experiments on the PAC. The different environmental conditions during the PLC experiment mainly cause the better control, which is confirmed by the MSE values in Tables 4.11 and 4.12. In contrast to the PAC experiments, the reference trajectory is now also followed when the temperature decreased below the reference starting temperature for both the top and reboiler temperature.

The inputs plotted in Figure 4.17b are also more similar to the hardware-in-the-loop experiments. The constraints are hit only during a few minutes for the flow rates.

The calculation time to present a solution to the column and the number of iterations required to calculate the QP are plotted in Figure 4.18. Some more run-times are presented in Table 4.12. Compared to the hardware-in-the-loop simulations, the desired reference trajectory was less demanding which is seen in lower maximum and mean run times for algorithm. The maximum number



(a) The measured temperatures, the controlled outputs of the distillation column.



(b) The inputs applied to the distillation column.

Figure 4.17: Measured outputs for the top- and reboiler temperature for an experiment on the pilot-scale set-up tracking a desired reference temperature profile with a MPC controller running on the PLC.

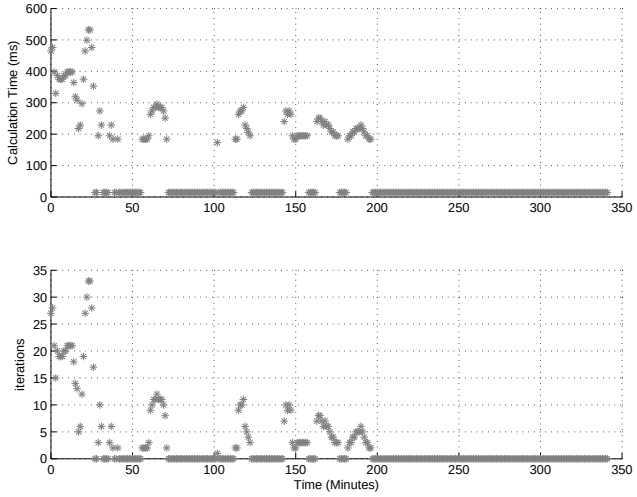


Figure 4.18: Calculation time and number of iterations for an experiment on the pilot-scale set-up with a MPC controller running on the PLC.

of iterations is 33 and at maximum a calculation time of 532 ms is required during this experiment.

4.14 Conclusion

In this chapter, the use of an online model predictive control algorithm hosted by a programmable automation controller (PAC) and a programmable logic controller (PLC) for a pilot-scale binary distillation column has been investigated. Experiments have been designed to track a reference temperature.

First, a model for control has been identified. A transfer function model has been selected. This model is expected to be sufficiently accurate to control the column. Conversion to state-space resulted in a model with 13 states. The Output-Error model, performing almost 10% better for the FIT on both top and reboiler temperature, results in a state-space model with 40 states. More than doubling of the number of states does not correspond to a significantly more accurate model. Hence, the latter model is not used.

It has been demonstrated that it is possible to run an online MPC algorithm on standard industrial PAC hardware which is not only able to run the supervisory algorithm, but is also responsible for data logging, human-machine interfacing

and low level control. This proves that a strategy where all procedures and computations needed to control and interface processes for an average size process model (e.g., the pilot-scale distillation column), can be integrated in an industrial embedded device.

Using a PAC, time and memory constraints are still *far away*. Using these or similar devices in industry to implement an MPC controller will be successful. Switching to a PLC is not to be encouraged. Nevertheless, it has been experimentally proven by hardware-in-the-loop tests that the pilot-scale set-up can be MPC controlled using the Hildreth algorithm. Unfortunately, the failure to solve the problem using the qpOASES algorithm using the same MPC settings, proves that the limits of these devices are reached. The programming issues, e.g., the limited amount of memory for each function and the difficult initialization of arrays, are also not conductive for an easy use in an industrial environment.

Chapter 5

Conclusions, Remarks and Future Research

5.1 Summary

In this work, the practical feasibility of online model predictive control hosted by low level industry standard control hardware was investigated. To this end, two case studies, i.e., a small air heating set-up and a pilot-scale distillation column were elaborated. The complete chain starting from (black-box linear) system identification, simulation experiments, hardware-in-the-loop experiments, up to experiments on the actual set-ups were carried out.

5.1.1 Case I: air heating set-up

This small set-up has been used to easily investigate several QP solvers. A model predictive controller has been created. The controller objective is to minimize the difference between the controlled temperature and its reference trajectory, together with minimal change of the inputs between two successive MPC steps. The control and prediction horizon were set to 7 and 22, respectively. The diagonal elements of the weight matrices W_u and W_y were set to one. The employed two input, one output black-box model is identified based on recorded data. The model structures for identification are first order transfer functions. Afterwards, this model was converted to a state-space representation. The resulting state-space model has 4 states. Integrated in

Table 5.1: Mean time and maximum time, the latter indicated between brackets, to calculate the input for the air heating set-up. All presented times are indicated in milliseconds. The abbreviations SIM, HIL and EXP stands for simulations on PC, hardware-in-the-loop experiments and experiments on the set-up, respectively.

	qpOASES	CVXGEN	Hildreth	CVXGEN MPC	LabVIEW MPC
- PC -					
SIM	2.57 (18.9)	2.73 (16.9)	2.61 (20.2)	15.45 (31.7)	N/A
- PAC -					
HIL	0.26 (0.4)	2.72 (4.4)	0.04 (0.3)	13.01 (18.0)	8.04 (11.0)
EXP	0.27 (0.5)	2.73 (3.8)	0.04 (0.3)	12.97 (24.0)	9.00 (9.0)
- PLC -					
EXP	15.72 (206)	N/A	1.82 (52)	N/A	N/A

the model predictive controller, the resulting hessian of the QP to be solved was 14×14, corresponding to 196 (non-zero) elements.

The mean time (in milliseconds) to calculate one MPC step is indicated in Table 5.1. This small-scale air heating set-up can easily be controlled by MPC on the different controller devices. The indicated times for the PC cover the QP routines, but also the calls to the compiled C libraries from within the Matlab programming environment. The times are only an indication of relative speed of the different algorithms. For the PAC and PLC only the time required to solve the QP is indicated. For the two algorithms tested on all devices, the classic Hildreth algorithm wins the speed race convincingly based on the mean time for the PAC. However, the maximum required time to calculate one MPC iteration is only 30% higher for qpOASES. The maximum required time is a crucial element to check, whether the device is capable of running the MPC algorithm in the available time. For the PLC, the conclusions are similar to the conclusions on the PAC.

The CVXGEN algorithm and CVXGEN MPC approach are, based on the currently available web tool to generate the code, impossible to use on a PLC. A second reason not to use them is that they are the slowest variants. The LabVIEW MPC controller is a built-in MPC controller on the CompactRIO. It is a slow MPC controller compared to Hildreth and qpOASES approach for this small set-up, but nevertheless it is an interesting alternative if a CompactRIO device is to be used. For this small set-up, both the qpOASES and Hildreth

Table 5.2: Mean time and maximum time, the latter indicated between brackets, to calculate the input for the pilot-scale distillation column. All presented times are indicated in milliseconds. The abbreviations SIM, HIL and EXP stands for simulations on PC, hardware-in-the-loop experiments and experiments on the set-up, respectively.

	Hildreth	qpOASES	qpOASES hot start
- PC -			
SIM	6.67 (108)	0.24 (2.15)	0.11 (3.16)
- PAC -			
HIL	N/C	0.89 (1.69)	N/A
EXP	4.58 (68)	1.33 (3.11)	N/A
- PLC -			
HIL	211 (556)	F	N/A
EXP	91 (532)	F	N/A

algorithm are able to successfully solve the QP, part of MPC, on the PLC.

5.1.2 Case II: pilot-scale distillation column

A more realistic industrial control application is the pilot-scale distillation column. After black-box linear system identification, the obtained reduced state-space model has been used in the MPC formulation. The 4 input, 2 output linear state-space model with 13 states is converted from an identified MIMO transfer function model. The control and prediction horizon of the controller are 10 and 50, respectively. The controller objective is to minimize the difference between the controlled temperature and its reference trajectory, together with a minimal difference between the inputs and their corresponding reference values. The weight matrix W_u for the inputs is adapted to encourage the use of the flow rates of the column. The weight matrix W_y for the outputs punished the deviation for the top temperature slightly more than that of the reboiler temperature. The obtained hessian size is 40×40, corresponding to 1600 (non-zero) elements.

For a distillation column, the aim is to control the concentration of the end products. As these can be inferred if pressure and temperature are known, only the temperatures along the column are controlled.

In Table 5.2, the mean time for one MPC iteration is presented for the different algorithms running on the different devices. For the PC, the indicated times cover extra time needed to call the different libraries from within the Matlab programming environment. Nevertheless, it is clear that the Hildreth algorithm is slower than the qpOASES algorithm. This indicates that the more complex solver benefits from the added complexity only for *sufficiently large* QPs. A similar trend is observed for experiments on the PAC. At some time steps, more than 20 times higher computation times are seen for the Hildreth algorithm for exactly the same number of iterations. These long calculations are caused by the presence of (unused) input and output modules. This disturbs the calculation of the mean time for one MPC iteration. However, based on the maximum required time, the observation that Hildreth is the slowest algorithm is confirmed yet again.

On the PLC, hardware-in-the-loop experiments failed for the qpOASES algorithm. The added complexity results in much more algebraic calculations and the time needed to solve these exceeded the available time. Only the Hildreth algorithm successfully completed hardware-in-the-loop experiments. The mean time for one MPC iteration for the Hildreth algorithm is 211 ms. At most 556 ms is required, which is still below the hardware constraint of 6 s, which is the maximum run time for completion of a function. Experiments on the pilot-scale set-up with the Hildreth algorithm hosted by a PLC have been successfully ended and the calculation times and required number of iterations resemble the hardware-in-the-loop experiments.

5.2 Remarks and final conclusions

To encourage the use of MPC in industry, the ease of implementation of model predictive controllers on commonly used devices has to be investigated. In this dissertation two standard industrial devices have been employed. A PAC device, which can be seen as a robust computer with real-time properties and additional in- and outputs, and a PLC, a standard industrial automation device. Experiments with MPC on the PAC demonstrate that MPC can easily be implemented. Memory nor speed constraints are encountered in this dissertation and it is believed that even much larger systems can be MPC-controlled on this type of devices, however, absolute limits were not investigated. The practical implemention of MPC by an engineer is realizable in a limited amount of time. The software is well documented, is easy to use and has many features. In summary, MPC implemented on PAC devices is ready for industry.

The second investigated device is a PLC. Historically, this device is situated in

the automation branch of the industry. The evolution in industry has been to request additional features, e.g., logging and human-machine interaction, which go together with the increase in computation power of the PLC. The research in this work to employ MPC on a PLC demonstrated that these devices, however capable, are not the ideal devices to run MPC. The limited amount of memory and especially the limited speed of the devices makes it difficult to implement MPC. Based on the experience obtained from this work, one can conclude that online MPC on a PLC is only applicable for problems resulting in a QP of limited size. For the PLC used for this study, a hessian of size 64×64, or 4096 elements is the absolute maximum. Important to notice is that the classical Hildreth algorithm is able to successfully run a hardware-in-the loop test for the pilot-scale set-up. The qpOASES algorithm, created with MPC applications in mind, fails this test due to its complexity. The hardware constraint that a function cannot run longer than 5999 ms is reached very fast for qpOASES. For the Hildreth algorithm, the 560 ms required for the HIL experiment in this dissertation, is still far way from the above-mentioned time constraint. Nevertheless, perfectly suited QP algorithms exist for use in MPC applications, where simple algorithms do the jobs where a PLC can be used as a host for the MPC algorithm. Moreover, the uncommon programming language is an extra barrier for the easy applicability of MPC on PLCs, as the employed algorithms are complex and require thorough testing. Given the easy applicability for MPC on a PAC, PLCs are not advisable for MPC applications in industry. To make a PLC a valuable alterative for a PAC, not only speed and memory limits need to increase drastically, e.g. by a factor of 100, but also the programming languages employed in a PLC need to be more commonly used languages as, i.e., C or C++.

5.3 Future research

Notwithstanding that this work is dedicated to the practical applicability of MPC on industrial control devices, some ideas for future research are put forward:

Investigate the limits for MPC on a PAC PAC devices are realistic examples for use of MPC in industry. It is interesting to know what the limits are in terms of speed and size of possible MPC problems. The presence of in- and outputs can influence the feasibility of MPC on a PAC. More research can be done on, for example, the unexpectedly large amount of extra time required for the same number of iterations. Also faster systems can be considered, given the low calculation times required to solve the QP.

Include output constraints In this dissertation, output/state constraints are omitted. In Appendix A is indicated how output constraints can be added. It would be interesting to investigate the effect of these constraints in experiments on different practical set-ups.

Investigate the effect of early stopping of an QP algorithm in practice It would be interesting to investigate on a practical set-up the effect of early stopping, employed to make sure a solution is presented within a maximum calculation time, of a QP algorithm.

Appendix A

MPC Algorithm

In this appendix, the complete formulation of the employed model predictive controller is highlighted. A large part of this formulation is mainly based on [87], but similar information and formulations can be found in [10, 38].

A.1 Addition of an integrator

A linear, time invariant discrete-time system is described by:

$$\begin{aligned} \mathbf{x}(k+1) &= A\mathbf{x}(k) + B\mathbf{u}(k) \\ \mathbf{y}(k) &= C\mathbf{x}(k), \end{aligned} \tag{A.1}$$

with $A \in \mathbb{R}^{p\times p}$, $B \in \mathbb{R}^{p\times m}$ and $C \in \mathbb{R}^{q\times p}$. Here m, p and q are the number of inputs, states and outputs, respectively.

There are several ways to add an integrator into this state-space model, but all lead to an increase of the number of states of the original system. For this reason it is called an *augmented state-space system*. An augmented state, often indicated with $\boldsymbol{\xi}$, defines the new state in terms of the state and the former input [38]:

$$\boldsymbol{\xi}(k) = \begin{bmatrix} \mathbf{x}(k) \\ \mathbf{u}(k-1) \end{bmatrix}. \tag{A.2}$$

The resulting augmented system is:

$$\underbrace{\begin{bmatrix}\mathbf{x}(k+1)\\ \mathbf{u}(k)\end{bmatrix}}_{\boldsymbol{\xi}(k+1)} = \underbrace{\begin{bmatrix}A & B\\ 0 & I\end{bmatrix}}_{\tilde{A}} \underbrace{\begin{bmatrix}\mathbf{x}(k)\\ \mathbf{u}(k-1)\end{bmatrix}}_{\boldsymbol{\xi}(k)} + \underbrace{\begin{bmatrix}B\\ I\end{bmatrix}}_{\tilde{B}} \Delta\mathbf{u}(k) \tag{A.3}$$

$$\mathbf{y}(k) = \underbrace{\begin{bmatrix}C & 0\end{bmatrix}}_{\tilde{C}} \begin{bmatrix}\mathbf{x}(k)\\ \mathbf{u}(k-1)\end{bmatrix}$$

Another commonly used augmented state is composed of $\Delta\mathbf{x}(k) = \mathbf{x}(k) - \mathbf{x}(k-1)$ and the current measured output:

$$\boldsymbol{\xi}(k) = \begin{bmatrix}\Delta\mathbf{x}(k)\\ \mathbf{y}(k)\end{bmatrix}. \tag{A.4}$$

The corresponding resulting augmented system is:

$$\underbrace{\begin{bmatrix}\Delta\mathbf{x}(k+1)\\ \mathbf{y}(k+1)\end{bmatrix}}_{\boldsymbol{\xi}(k+1)} = \underbrace{\begin{bmatrix}A & 0\\ CA & I\end{bmatrix}}_{\tilde{A}} \underbrace{\begin{bmatrix}\Delta\mathbf{x}(k)\\ \mathbf{y}(k)\end{bmatrix}}_{\boldsymbol{\xi}(k)} + \underbrace{\begin{bmatrix}B\\ CB\end{bmatrix}}_{\tilde{B}} \Delta\mathbf{u}(k) \tag{A.5}$$

$$\mathbf{y}(k) = \underbrace{\begin{bmatrix}0 & I\end{bmatrix}}_{\tilde{C}} \begin{bmatrix}\Delta\mathbf{x}(k)\\ \mathbf{y}(k)\end{bmatrix}.$$

Notice that the structure of both augmented systems is given by:

$$\boldsymbol{\xi}(k+1) = \tilde{A}\boldsymbol{\xi}(k) + \tilde{B}\Delta\mathbf{u}(k) \tag{A.6}$$

$$\mathbf{y}(k) = \tilde{C}\Delta\mathbf{x}(k)\boldsymbol{\xi}(k).$$

In the following sections, this augmented system is employed for prediction and to calculate the inputs of the model predictive controller. Both augmented systems, given in Equations (A.3) and (A.5) can be employed, but in this dissertation, the latter is implemented on the practical set-ups, as only measured outputs of the system are available. To obtain the states as used in Equation (A.3), an additional calculation based on the measured outputs is required. Moreover, the states of the black-box model do not have a physical meaning in this dissertation.

A.2 Calculation of the future output

To predict the future behavior of the system and to calculate the optimal input, the future output has to be calculated based on the current state and future

inputs. This is done for an augmented system, but if the tilde is removed from matrices A, B and C, this also holds for the linear system as defined in (A.1). The formulation $\mathbf{y}(k+i|k)$ will be used to represent the vector $\mathbf{y}$ on sample time $k+i$ at calculation time k. Starting from the augmented system, the future predictions can be calculated from:

$$\begin{aligned}
\boldsymbol{\xi}(k+1|k) &= \tilde{A}\boldsymbol{\xi}(k|k) + \tilde{B}\Delta\mathbf{u}(k|k) \qquad (A.7)\\
\boldsymbol{\xi}(k+2|k) &= \tilde{A}\boldsymbol{\xi}(k+1|k) + \tilde{B}\Delta\mathbf{u}(k+1|k)\\
&= \tilde{A}[\tilde{A}\boldsymbol{\xi}(k|k) + \tilde{B}\Delta\mathbf{u}(k|k)] + \tilde{B}\Delta\mathbf{u}(k+1|k)\\
&= \tilde{A}^2\boldsymbol{\xi}(k|k) + \tilde{A}\tilde{B}\Delta\mathbf{u}(k|k) + \tilde{B}\Delta\mathbf{u}(k+1|k)\\
\cdots &= \cdots\\
\boldsymbol{\xi}(k+H_p|k) &= \tilde{A}^{H_p}\boldsymbol{\xi}(k|k) + \tilde{A}^{H_p-1}\tilde{B}\Delta\mathbf{u}(k|k)\\
&\quad + \tilde{A}^{H_p-2}\tilde{B}\Delta\mathbf{u}(k+1|k) + \cdots\\
&\quad + \tilde{A}^{H_p-H_c}\tilde{B}\Delta\mathbf{u}(k+H_c-1|k)
\end{aligned}$$

leading to:

$$\begin{aligned}
\hat{\mathbf{y}}(k+1|k) &= \tilde{C}\boldsymbol{\xi}(k+1|k) \qquad (A.8)\\
&= \tilde{C}[\tilde{A}\boldsymbol{\xi}(k|k) + \tilde{B}\Delta\mathbf{u}(k|k)]\\
&= \tilde{C}\tilde{A}\boldsymbol{\xi}(k|k) + \tilde{C}\tilde{B}\Delta\mathbf{u}(k|k)\\
\cdots &= \cdots .\\
\hat{\mathbf{y}}(k+H_p|k) &= \tilde{C}\tilde{A}^{H_p}\boldsymbol{\xi}(k|k) + \tilde{C}\tilde{A}^{H_p-1}\tilde{B}\Delta\mathbf{u}(k|k)\\
&\quad + \tilde{C}\tilde{A}^{H_p-2}\tilde{B}\Delta\mathbf{u}(k+1|k) + \cdots\\
&\quad + \tilde{C}\tilde{A}^{H_p-H_c}\tilde{B}\Delta\mathbf{u}(k+H_c-1|k)
\end{aligned}$$

with H_p the prediction horizon and H_c the control horizon. Rearranging this into matrices, the prediction $\hat{\mathbf{Y}}$ over the prediction horizon H_p in matrix formulation for the augmented system is:

$$\hat{\mathbf{Y}} = F\boldsymbol{\xi}(k) + \Phi\Delta\mathbf{U}, \qquad (A.9)$$

with $\hat{\mathbf{Y}}$, F, Φ and $\Delta\mathbf{U}$, respectively:

$$\hat{\mathbf{Y}} = \begin{bmatrix} \hat{\mathbf{y}}(k|k) \\ \hat{\mathbf{y}}(k+1|k) \\ \hat{\mathbf{y}}(k+2|k) \\ \vdots \\ \hat{\mathbf{y}}(k+H_p|k) \end{bmatrix} \tag{A.10}$$

$$F = \begin{bmatrix} \tilde{C}\tilde{A} \\ \tilde{C}\tilde{A}^2 \\ \tilde{C}\tilde{A}^3 \\ \vdots \\ \tilde{C}\tilde{A}^{H_p} \end{bmatrix} \tag{A.11}$$

$$\Phi = \begin{bmatrix} \tilde{C}\tilde{B} & 0 & 0 & \cdots & 0 \\ \tilde{C}\tilde{A}\tilde{B} & \tilde{C}\tilde{B} & 0 & \cdots & 0 \\ \tilde{C}\tilde{A}^2\tilde{B} & \tilde{C}\tilde{A}\tilde{B} & \tilde{C}\tilde{B} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \tilde{C}\tilde{A}^{H_p-1}\tilde{B} & \tilde{C}\tilde{A}^{H_p-2}\tilde{B} & \tilde{C}\tilde{A}^{H_p-3}\tilde{B} & \cdots & \tilde{C}\tilde{A}^{H_p-H_c}\tilde{B} \end{bmatrix} \tag{A.12}$$

$$\Delta\mathbf{U} = \begin{bmatrix} \Delta\mathbf{u}(k|k) \\ \Delta\mathbf{u}(k+1|k) \\ \Delta\mathbf{u}(k+2|k) \\ \vdots \\ \Delta\mathbf{u}(k+H_c-1|k) \end{bmatrix}. \tag{A.13}$$

To calculate the future inputs in Equation (A.13), the following equation can be employed:

$$\begin{bmatrix} \mathbf{u}(k|k) \\ \mathbf{u}(k+1|k) \\ \vdots \\ \mathbf{u}(k+H_c-1|k) \end{bmatrix} = \begin{bmatrix} I \\ I \\ \vdots \\ I \end{bmatrix} \mathbf{u}(k-1|k) \tag{A.14}$$

$$+ \begin{bmatrix} I & 0 & \dots & 0 \\ I & I & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ I & I & \dots & I \end{bmatrix} \begin{bmatrix} \Delta\mathbf{u}(k|k) \\ \Delta\mathbf{u}(k+1|k) \\ \Delta\mathbf{u}(k+2|k) \\ \vdots \\ \Delta\mathbf{u}(k+H_c-1|k) \end{bmatrix}.$$

A.3 Controller objective

The objective of the controller is to find the optimal input $\Delta\mathbf{u}(k+j|k)$, $i = 1 \dots H_p$ for this system by means of minimizing the cost function J:

$$J = \sum_{i=1}^{H_p} \|\hat{\mathbf{y}}(k+i|k) - \mathbf{y}_{\text{ref}}(k+i|k)\|^2_{W_y} \tag{A.15}$$
$$+ \sum_{j=0}^{H_c-1} \|\Delta\mathbf{u}(k+j|k) - \Delta\mathbf{u}_{\text{ref}}(k+j|k)\|^2_{W_u}.$$

with $\Delta\mathbf{u}_{\text{ref}}$ and $\mathbf{y}_{\text{ref}}$ respectively the input and output reference and $\hat{\mathbf{y}}$ the predicted output. H_p and H_c (with $H_c \leq H_p$) are respectively the prediction and control horizon of the controller. $W_y \in \mathbb{R}^{q\times q}$ and $W_u \in \mathbb{R}^{m\times m}$ are positive definite weighting matrices.

One of the key elements of MPC is the possibility to handle constraints. For this dissertation, only input constraints are taken into account:

$$\mathbf{u}(j) \leq \mathbf{u}_{\text{Max}} \tag{A.16}$$
$$\mathbf{u}(j) \geq \mathbf{u}_{\text{Min}}.$$

Output constraints and constraints on $\Delta\mathbf{u}$ are omitted, but can easily be introduced as seen later in this section. The optimization problem can be formulated as the minimization of Equation (A.15), subject to Equation (A.5) and Equation (A.16). In order to solve this problem, the optimization problem is reformulated by elimination of the states in the form of a QP:

$$\min_{\boldsymbol{\theta}} J = \frac{1}{2}\boldsymbol{\theta}^T H \boldsymbol{\theta} + \mathrm{g}^T \boldsymbol{\theta} \tag{A.17}$$

$$\text{subject to : } P\boldsymbol{\theta} \leq \boldsymbol{\alpha} \tag{A.18}$$

with Equation (A.17) the quadratic objective function, Equation (A.18) the linear inequality constraints and $\boldsymbol{\theta}$ the decision variables.

The cost function presented in Equation (A.15) has to be rearranged into the form of Equation (A.17). Using the notation from Section A.2, the cost function becomes:

$$J = (\hat{\mathbf{Y}} - \mathbf{Y}_{\text{ref}})^T W_{y_{bd}} (\hat{\mathbf{Y}} - \mathbf{Y}_{\text{ref}})^T + \Delta\mathbf{U}^T W_{u_{bd}} \Delta\mathbf{U} \tag{A.19}$$

with $W_{y_{bd}}$ and $W_{u_{bd}}$ block diagonal matrices of W_y and W_u, respectively. When $\hat{Y}$ is replaced by Equation (A.9), the cost function becomes:

$$J = (F\boldsymbol{\xi}(k|k) + \Phi\Delta\mathbf{U} - \mathbf{Y}_{\text{ref}})^T W_{y_{bd}} (F\boldsymbol{\xi}(k|k) + \Phi\Delta\mathbf{U} - \mathbf{Y}_{\text{ref}})^T \tag{A.20}$$

$$+ (\Delta\mathbf{U} - \Delta\mathbf{U}_{\text{ref}})^T W_{u_{bd}} (\Delta\mathbf{U} - \Delta\mathbf{U}_{\text{ref}}). \tag{A.21}$$

In this equation is $F\boldsymbol{\xi}(k) - \mathbf{Y}_{\text{ref}}$ constant over the prediction horizon. Simplifying and grouping this equation leads to:

$$J = \Delta\mathbf{U}^T (\Phi^T W_{y_{bd}} \Phi + W_{u_{bd}}) \Delta\mathbf{U} \tag{A.22a}$$

$$+ 2[\Phi^T W_{y_{bd}} F\boldsymbol{\xi}(k|k) - \Phi^T W_{y_{bd}} \mathbf{Y}_{\text{ref}} - W_{u_{bd}} \Delta\mathbf{U}_{\text{ref}}] \Delta\mathbf{U} \tag{A.22b}$$

$$+ (F\boldsymbol{\xi}(k|k) - \mathbf{Y}_{\text{ref}})^T W_{y_{bd}} (F\boldsymbol{\xi}(k|k) - \mathbf{Y}_{\text{ref}}) \tag{A.22c}$$

$$+ \Delta\mathbf{U}_{\text{ref}}^T W_{u_{bd}} \Delta\mathbf{U}_{\text{ref}} \tag{A.22c}$$

with Subequation (A.22a), (A.22b) and (A.22c) respectively the quadratic, linear and constant parts of the cost function. The constant part can be neglected in the (online) calculation of the decision variable $\Delta\mathbf{U}$. The hessian H and gradient g to be supplied to Equation (A.17) are:

$$H = \Phi^T W_{y_{bd}} \Phi + W_{u_{bd}} \tag{A.23}$$

$$g = \underbrace{\Phi^T W_{y_{bd}} F}_{G_1} \boldsymbol{\xi}(k|k) - \underbrace{\Phi^T W_{y_{bd}}}_{G_2} \mathbf{Y}_{\text{ref}} - \underbrace{W_{u_{bd}}}_{G_3} \Delta\mathbf{U}_{\text{ref}}. \tag{A.24}$$

The gradient is clearly divided in three parts: a constant matrix G_1 to be multiplied by the current state, a constant matrix G_2 to be multiplied by the reference output(s) and a matrix G_3 to be multiplied by the input change reference. The matrices H, G_1, G_2 and G_3 can be computed offline.

The inequality constraints in Equation (A.18) are the constraints of the MPC problem. The minimum and maximum value for $\Delta\mathbf{U}$ is determined by the input and output constraints and constraints on $\Delta\mathbf{U}$. To take into account input constraints, the following two inequalities have to be calculated and integrated in the QP:

$$\underbrace{\begin{bmatrix} \mathbf{u}_{\text{Min}}(k|k) \\ \mathbf{u}_{\text{Min}}(k+1|k) \\ \cdots \\ \mathbf{u}_{\text{Min}}(k+H_c-1|k) \end{bmatrix}}_{\mathbf{U}_{\text{Min}}} \leq \underbrace{\begin{bmatrix} I & 0 & \cdots & 0 \\ I & I & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ I & I & \cdots & I \end{bmatrix}}_{C_1} \underbrace{\begin{bmatrix} \Delta\mathbf{u}(k|k) \\ \Delta\mathbf{u}(k+1|k) \\ \Delta\mathbf{u}(k+2|k) \\ \vdots \\ \Delta\mathbf{u}(k+H_c-1|k) \end{bmatrix}}_{\Delta\mathbf{U}} \tag{A.25}$$

$$\underbrace{\begin{bmatrix} I & 0 & \dots & 0 \\ I & I & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ I & I & \dots & I \end{bmatrix}}_{C_1} \underbrace{\begin{bmatrix} \Delta\mathbf{u}(k|k) \\ \Delta\mathbf{u}(k+1|k) \\ \Delta\mathbf{u}(k+2|k) \\ \vdots \\ \Delta\mathbf{u}(k+H_c-1|k) \end{bmatrix}}_{\Delta\mathbf{U}} \leq \underbrace{\begin{bmatrix} \mathbf{u}_{\text{Max}}(k|k) \\ \mathbf{u}_{\text{Max}}(k+1|k) \\ \vdots \\ \mathbf{u}_{\text{Max}}(k+H_c-1|k) \end{bmatrix}}_{\mathbf{U}_{\text{Max}}} \tag{A.26}$$

with $\mathbf{u}_{\text{Max}}(k|k)$, $\mathbf{u}_{\text{Max}}(k+1|k)$, …, $\mathbf{u}_{\text{Max}}(k+H_c-1|k) = \mathbf{u}_{\text{Max}} - \mathbf{u}(k-1)$ and $\mathbf{u}_{\text{Min}}(k|k)$, $\mathbf{u}_{\text{Min}}(k+1|k)$, …, $\mathbf{u}_{\text{Min}}(k+H_c-1|k) = \mathbf{u}_{\text{Min}} - \mathbf{u}(k-1)$. Hence, $\mathbf{U}_{\text{Max}}$ is a column matrix of H_c times $\mathbf{u}_{\text{Max}} - \mathbf{u}(k-1)$ and $\mathbf{U}_{\text{Min}}$ is a column matrix of H_c times $\mathbf{u}_{\text{Min}} - \mathbf{u}(k-1)$. $\mathbf{u}_{\text{Min}}$ and $\mathbf{u}_{\text{Max}}$ are to be supplied by the designer of the MPC problem and will be selected based on, e.g., the characteristics of the system. In a more compact form and ready to be used in the inequality formulation in Equation (A.18), Equations (A.25) and (A.26) can be written as:

$$\underbrace{\begin{bmatrix} -C_1 \\ C_1 \end{bmatrix}}_{P_1} \Delta\mathbf{U} \leq \underbrace{\begin{bmatrix} -\mathbf{U}_{\text{Min}} \\ \mathbf{U}_{\text{Max}} \end{bmatrix}}_{\boldsymbol{\alpha}_1}. \tag{A.27}$$

Adding constraints on the change of $\Delta\mathbf{U}$ is easily done by using:

$$\underbrace{\begin{bmatrix} -I \\ I \end{bmatrix}}_{P_2} \Delta\mathbf{U} \leq \underbrace{\begin{bmatrix} -\Delta\mathbf{U}_{\text{Min}} \\ \Delta\mathbf{U}_{\text{Max}} \end{bmatrix}}_{\boldsymbol{\alpha}_2} \tag{A.28}$$

where I is the identity matrix and $\Delta\mathbf{U}_{\text{Min}}$ and $\Delta\mathbf{U}_{\text{Max}}$ are column matrices of respectively $\Delta\mathbf{u}_{\text{Min}}$ and $\Delta\mathbf{u}_{\text{Max}}$. $\Delta\mathbf{u}_{\text{Min}}$ and $\Delta\mathbf{u}_{\text{Max}}$ are selected by the designer of the model predictive controller.

Similar, output constraints can be formulated based on Equation (A.9). With $\mathbf{Y}_{\text{Min}}$ and $\mathbf{Y}_{\text{Max}}$ column matrices of respectively $\mathbf{y}_{\text{Min}}$ and $\mathbf{y}_{\text{Max}}$ the output constraints can be written as:

$$\mathbf{Y}_{\text{Min}} - F\boldsymbol{\xi}(k|k) \leq \Phi\Delta U \leq \mathbf{Y}_{\text{Min}} - F\boldsymbol{\xi}(k|k) \tag{A.29}$$

or in a matrix formulation:

$$\underbrace{\begin{bmatrix} -\Phi \\ \Phi \end{bmatrix}}_{P_3} \Delta\mathbf{U} \leq \underbrace{\begin{bmatrix} -\mathbf{Y}_{\text{Min}} + F\boldsymbol{\xi}(k|k) \\ \mathbf{Y}_{\text{Max}} - F\boldsymbol{\xi}(k|k) \end{bmatrix}}_{\boldsymbol{\alpha}_3}. \tag{A.30}$$

The inequality constraint matrices P and α from Equation (A.18) can now be composed of the desired Equations (A.27), (A.28) and (A.30):

$$P = \begin{bmatrix} P_1 \\ P_2 \\ P_3 \end{bmatrix} \tag{A.31}$$

$$\boldsymbol{\alpha} = \begin{bmatrix} \boldsymbol{\alpha}_1 \\ \boldsymbol{\alpha}_2 \\ \boldsymbol{\alpha}_3 \end{bmatrix} \tag{A.32}$$

For this dissertation, only input constraints have been employed. Hence, the QP problem to be solved online is:

$$\min_{\Delta \mathbf{U}} \frac{1}{2} \Delta \mathbf{U}^T H \Delta \mathbf{U} + \mathbf{g} \Delta \mathbf{U} \tag{A.33}$$

$$\text{subject to : } \mathbf{U}_{\text{Min}} \leq C_1 \Delta \mathbf{U} \leq \mathbf{U}_{\text{Max}}. \tag{A.34}$$

Appendix B

Distillation column model

In this appendix, the state-space model employed in this dissertation is given. The different steps taken to transform the identified transfer function model to the employed reduced discrete state-space model, are summarized.

B.1 Transformation from the s-domain to a discrete state-space model

The transfer functions obtained by the identification process, are transformed from the s-domain to the z-domain. Discretization is only possible after the selection of a discretization time. For the distillation column 1 minutes has been selected. This choice is based on operator knowledge and the observation that the time delays and the time constants in the estimated transfer function are larger than 60 seconds. The spectrum of the both the reboiler and the top temperature present no peaks above the frequency of 0.005 Hz (200 s). This also proves the selected discretization time is sufficient and a further reduction of the discretization time is not required.

Each continuous transfer function is converted to the z-domain (see Equation (B.2)). The used discretization method is *zero-order hold*. Afterwards, the MIMO model is converted to discrete state-space and the balanced realization [33] is computed. The Hankel singular values are calculated an these lower than 0.01 are detected and removed from the balanced state-space model. As no clear step can be observed in the Hankel singular values plot, it is possible to select a model a different number of states, most likely less

states. However, to keep the dynamic behavior of the system close the the converted discrete state-space model, only state energies of the Hankel singular values lower than 0.01 are removed. Following this procedure, the model in Equation (B.1) with matrices (B.3) to (B.5) is obtained.

B.2 The employed Matlab code

The discrete MIMO transfer model from (B.2) has been converted to state-space with the following Matlab code [40]:

```
% kolomtf is the continues MIMO transfer model of the pilot-scale
    % distillation column

dkolomtf = c2d(kolomtf,Ts);   % Ts = discretization time
    % discretization by ZOH

dkolomtf = delay2z(dkolomtf); % Put all delays into transfer
    % functions

kolomss = ss(dkolomtf); % Convert z-domain model to state-space
    % form
kolomss = minreal(kolomss); % Remove any nonminimal modes

[sys,g] = balreal(kolomss)   % compute balanced realization
elim = (g<1e-2)          % small entries of g -> negligible
    % states

sysr = modred(sys,[14:20],'del') % remove negligible states
    % sysr is the reduced state-space model used as controller
    % model.
```

B.3 The employed state-Space model

The parameters A, B and C in the reduced state-space model with formulation

$$\mathbf{x} = A\mathbf{x} + B\mathbf{u} \tag{B.1}$$

$$\mathbf{y} = C\mathbf{x}$$

are presented in Equations (B.3) to (B.5) on page 135.

$$\begin{bmatrix} Tt \\ Tb \end{bmatrix} =$$

$$\begin{bmatrix} \frac{-0.01011z-0.008649}{z^2-1.618z+0.6265} & \frac{0.04158}{z-0.9216} & \frac{0.0372z+0.0285}{z^2-1.431z+0.4483} & z^{-2}\,\frac{-0.05972z-0.01615}{z-0.978} \\ \frac{-0.01949z-0.01693}{z^2-1.636z+0.6558} & z^{-6}\,\frac{0.02179z+0.003007}{z-0.9599} & z^{-1}\,\frac{0.02934z^2+0.07874z-3.497e-17}{z^2-0.9793z} & \frac{-0.006013z-0.005699}{z^2-1.846z+0.8511} \end{bmatrix} \begin{bmatrix} \mathrm{Fv} \\ \mathrm{Qv} \\ \mathrm{Qr} \\ \mathrm{Fr} \end{bmatrix} \tag{B.2}$$

$$A = \begin{bmatrix} 0.98 & 0.0027 & -0.012 & -0.032 & -0.0083 & 0.026 & -0.0035 & -0.0051 & 0.003 & -0.00032 & 0.00052 & -0.0014 & -0.0017 \\ 0.00035 & 1 & -0.019 & 0.011 & 0.018 & -0.0065 & 0.014 & -0.0097 & -0.0026 & -0.0013 & -0.00036 & 0.0003 & 0.00032 \\ 0.0097 & 0.012 & 0.95 & -0.095 & 0.067 & 0.047 & 0.053 & -0.057 & -0.00049 & -0.0097 & -0.0014 & -0.0046 & -0.0074 \\ 0.027 & -0.02 & 0.017 & 0.81 & -0.034 & 0.28 & -0.035 & -0.1 & 0.043 & -0.013 & 0.0059 & -0.028 & -0.036 \\ 0.0068 & -0.00039 & -0.043 & -0.061 & 0.86 & 0.11 & -0.23 & 0.12 & 0.05 & -0.0074 & 0.0093 & -0.017 & -0.03 \\ -0.028 & -0.0036 & 0.024 & 0.13 & 0.1 & 0.6 & 0.22 & 0.22 & -0.1 & 0.073 & 0.011 & 0.066 & 0.11 \\ 0.0057 & -0.0036 & -0.075 & -0.053 & -0.045 & -0.045 & 0.59 & 0.38 & 0.18 & -0.00072 & 0.028 & -0.11 & -0.15 \\ 0.0028 & 0.013 & -0.0091 & 0.028 & -0.13 & 0.2 & -0.017 & 0.21 & -0.18 & -0.57 & -0.18 & 0.053 & -0.14 \\ -0.003 & 0.0027 & 0.021 & 0.058 & 0.02 & -0.073 & 0.14 & -0.0035 & 0.84 & -0.19 & 0.017 & 0.14 & 0.17 \\ 0.0026 & 0.0034 & -0.016 & 0.0055 & 0.089 & 0.076 & 0.021 & 0.043 & 0.14 & 0.25 & -0.66 & 0.012 & -0.086 \\ -0.0005 & 0.0027 & 0.0039 & 0.014 & -0.055 & 0.011 & 0.039 & -0.078 & 0.034 & 0.59 & -0.081 & 0.24 & -0.38 \\ -0.00024 & -0.0013 & -0.0057 & -0.018 & -0.039 & 0.01 & -0.032 & -0.17 & 0.033 & -0.034 & -0.53 & -0.21 & 0.48 \\ 0.00067 & -0.00092 & -0.014 & -0.02 & -0.003 & 0.032 & -0.04 & -0.18 & 0.038 & -0.14 & 0.0018 & 0.81 & -0.048 \end{bmatrix} \tag{B.3}$$

$$B = \begin{bmatrix} 0.18 & -0.058 & -0.34 & 0.22 \\ -0.035 & 0.0015 & -0.048 & -0.064 \\ 0.13 & 0.038 & 0.1 & -0.11 \\ -0.16 & 0.0054 & 0.076 & -0.23 \\ 0.0059 & -0.16 & 0.086 & -0.041 \\ 0.14 & -0.022 & -0.23 & 0.048 \\ 0.099 & -0.004 & 0.044 & -0.13 \\ 0.032 & -0.049 & 0.11 & 0.078 \\ -0.0068 & -0.022 & -0.0072 & 0.05 \\ 0.0076 & 0.1 & 0.035 & 0.026 \\ 0.0078 & -0.05 & 0.018 & 0.018 \\ 0.01 & -0.031 & -0.0053 & -0.022 \\ 0.016 & 0.0095 & 0.002 & -0.022 \end{bmatrix} \tag{B.4}$$

$$C = \begin{bmatrix} -0.31 & 0.078 & 0.064 & -0.17 & -0.18 & 0.18 & -0.13 & 0.052 & 0.036 & 0.011 & 0.0067 & -0.01 & -0.011 \\ -0.32 & -0.039 & -0.2 & -0.23 & 0.064 & 0.17 & 0.073 & -0.12 & 0.006 & -0.016 & 0.00012 & -0.0099 & -0.013 \end{bmatrix} \tag{B.5}$$

Bibliography

[1] Akaike, H. A new look at the statistical model identification. *IEEE Transactions on Automatic Control 19* (1974), 716 – 723.

[2] Bauer, M., and Craig, I. K. Economic assessment of advanced process control - a survey and framework. *Journal of Process Control* (2008), 2 – 18.

[3] Bellemans, T., De Schutter, B., and De Moor, B. Model predictive control for ramp metering of motorway traffic: A case study. *Control Engineering Practice 14* (2006), 757 – 767.

[4] Bemporad, A., Morari, M., Dua, V., and Pistikopoulos, E. N. The explicit linear quadratic regulator for constrained systems. *Automatica 38* (2002), 3 – 20.

[5] Bemporad, A., Morari, M., and Ricker, L. *Model Predictive Control Toolbox User's Guide*. The MathWorks, Inc, Natick, 2009.

[6] Bolton, W. *Programmable Logic Controllers*, fifth ed. Newnes (Elsevier), 2009.

[7] Bonilla Alarcon, J. E. *Structure and Convexity Exploitation in Nonlinear Chemical Process Modeling and Estimation*. PhD thesis, Arenberg Doctoral School, 2011.

[8] Boyd, S., and Vandenberghe, L. *Convex Optimization*. Cambridge University Press, 2004.

[9] Burnham, K. P., and Anderson, D. R. *Model Selection and Multi-Model Inference*. Springer New York, 2002.

[10] Camacho, E. F., and Bordons, C. *Model Predictive Control*. Springer, 2003.

[11] Cutler, C., and Ramaker., B. Dynamic matrix control - a computer control algorithm. In *Proceedings of the Joint American Control Conference, San Francisco, 1980.* (1980), pp. Paper WP5–B.

[12] De Brabanter, K. *Least Squares Support Vector Regression with Applications to Large-ScaleData: a Statistical Approach.* PhD thesis, Arenberg Doctroal School, KU Leuven, 2011.

[13] De Brabanter, K., De Brabanter, J., Suykens, J., and De Moor, B. Approximate confidence and prediction intervals for least squares support vector regression. *IEEE Transactions on Neural Networks 22* (2011), 110 – 120.

[14] De Brabanter, K., Karsmakers, P., Ojeda, F., Alzate, C., De Brabanter, J., Pelckmans, K., De Moor, B., Vandewalle, J., and Suykens, J. LS-SVMlab toolbox user's guide version 1.7. Tech. rep., ESAT-SISTA, K.U.Leuven (Leuven, Belgium), 2010.

[15] Ferong, K. Ontwikkeling en implementatie van MPC voor de CIT-destillatiekolom. Master's thesis, KU Leuven: Master in de ingenieurswetenschappen - chemische technologie, 2010.

[16] Ferreau, H. J., Bock, H. G., and Diehl, M. An online active set strategy to overcome the limitations of explicit MPC. *International Journal of Robust and Nonlinear Control 18* (2008), 816 – 830.

[17] Fletcher, R. A numerically stable dual method for solving strictly convex quadratic programs. *IMA Journal of Applied Mathematics 7* (1971), 76 – 91.

[18] Garnier, H., and Wang, L., Eds. *Identification of Continuous-time Models from Sampled Data.* Springer, 2010.

[19] Glad, T., and Ljung, L. *Control Theory: Multivariable and Nonlinear Methods.* Taylor and Francis, London, 2000.

[20] Goldfarb, D., and Idnani, A. A numerically stable dual method for solving strictly convex quadratic programs. *Mathematical Programming 27* (1983), 1 – 33.

[21] Golub, G. H., and Van Loan, C. F. *Matrix Computations.* Johns Hopkins University Press, Baltimore, MD, USA, 1996.

[22] Hastie, T., Tibshirani, R., and Friedman, J. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.*, second ed. Springer, 2009.

[23] Hildreth, C. A quadratic programming procedure. *Naval Research Logistics Quarterly 4* (1957), 79 – 85.

[24] Holmström, K., Göran, A. O., and Edvall, M. M. Users guide for tomlab/sol. http://tomopt.com/docs/TOMLAB_SOL.pdf, February 2008.

[25] Hrovat, D., Di Cairano, S., Tseng, H., and Kolmanovsky, I. The development of model predictive control in automotive industry: A survey. In *IEEE International Conference on Control Applications (CCA)* (2012), pp. 295 –302.

[26] Huyck, B., De Brabanter, K., Logist, F., De Brabanter, J., Van Impe, J., and De Moor, B. Identification of a pilot scale distillation column: A kernel based approach. In *18th World Congress of the International Federation of AutomaticControl* (Milano, Italy, 2011), pp. 471 – 476.

[27] Huyck, B., Logist, F., De Brabanter, J., Van Impe, J., and De Moor, B. Constrained model predictive control on a programmable automation system exploiting code generation: practical considerations. In *18th World Congress of the International Federation of Automatic Control* (Milano, Italy, 2011), pp. 12207 – 12212.

[28] Iusem, A. N., and De Pierro, A. R. On the convergence properties of hildreth's quadratic programming algorithm. *Mathematical Programming 47* (1990), 37 – 51.

[29] Johansen, T. A., Grancharova, A., and Grancharova, R. Approximate explicit constrained linear model predictive control via orthogonal search tree. *IEEE Transactions on Automatic Control 48* (2003), 810 – 815.

[30] Johansen, T. A., Petersen, I., and Slupphaug, O. Explicit sub-optimal linear quadratic regulation with state and input constraints. *Automatica 38* (2002), 1099 – 1111.

[31] Kouro, S., Cortes, P., Vargas, R., Ammann, U., and Rodriguez, J. Model predictive control - a simple and powerful method to control power converters. *IEEE Transactions on Industrial Electronics 56* (2009), 1826 – 1838.

[32] Kvasnica, M., Rauova, I., and Fikar, M. Automatic code generation for real-time implementation of model predictive control. In *IEEE International Symposium on Computer-Aided Control System Design (CACSD)* (2010), pp. 993 – 998.

[33] Laub, A. J., Heath, M. T., Paige, C. C., and Ward, R. C. Computation of system balancing transformations and other applications of simultaneous diagonalization algorithms. *IEEE Transactions on Automatic Control 32* (1987), 115 – 122.

[34] Ljung, L. *System Identification: Theory for the User, Second Edition*. Prentice Hall, Upper Saddle River, New Jersey, 1999.

[35] Ljung, L. *System Identification Toolbox User's Guide*. The MathWorks, Inc, Natick, 2009.

[36] Logist, F., Huyck, B., Fabré, M., Verwerft, M., Pluymers, B., De Brabanter, J., De Moor, B., and Van Impe, J. Identification and control of a pilot scale binary distillation column. In *Proceedings of the European Control Conference* (2009), pp. 4659 – 4664.

[37] Luenberger, D. G. *Optimization by Vector Space Methods*. John Wiley & Sons, 1997.

[38] Maciejowski, J. *Predictive Control with Constraints*. Pearson Education Limited, 2002.

[39] Mathur, U., Rounding, R. D., Webb, D. R., and Conroy, R. J. Use model-predictive control to improve distillation operations. *Chemical Engineering Progress 104* (2008), 35 – 41.

[40] MathWorks. *Control System Toolbox User's Guide*. The MathWorks, Inc, Natick, 2009.

[41] Mattingley, J., and Boyd, S. CVXGEN: a code generator for embedded convex optimization. *Optimization and Engineering 13* (2012), 1 – 27.

[42] Mattingley, J., Wang, Y., and Boyd, S. Code generation for receding horizon control. In *Proceedings of the IEEE Multi-Conference on Systems and Control* (2010), pp. 985 – 992.

[43] Mayne, D., Rawlings, J., Rao, C., and Scokaert, P. Constrained model predictive control: Stability and optimality. *Automatica 36* (2000), 789 – 814.

[44] Beckhoff. CX1000 embedded pc hardware documentation, July 2006. version 1.1.

[45] Beckhoff. CX1010 embedded pc hardware documentation, July 2006. version 1.1.

[46] Beckhoff. CX50x0 - hardware documentation, May 2011. version 1.3.

[47] beckhoff. CX20x0 hardware-documentation, December 2012. version 1.0.

[48] Eaton Electric. Volume 7-logic control, operator interface and connectivity solutions. www.eaton.com, December 2012. CA08100008EV7-T4-1.

[49] GE Intelligent Platforms, Inc. Control systems solutions catalog, 2012. GFA-406H.

[50] National Instruments Corporation. Labview control design user manual, June 2009. 371057G-01.

[51] National Instruments Corporation. Operating instructions and specifications CompactRIO NI cRIO-9024, June 2010. 375233C-01.

[52] Phoenix Contact AG. Catalogue automation, February 2012. MnR.52006285/2011-06-01/00.

[53] Saia-Burgess Controls Ltd. Reference guide SAIA PCD, March 2001. English edition 26/733 E6.

[54] Siemens AG. S7-400 automation system, CPU specifications, August 2008. A5E00850746-10.

[55] Siemens AG. CPU 31xC and CPU 31x: Technical specifications manual, March 2011. A5E00105475-12.

[56] Siemens AG. Example projects for SIMATIC PID professional: Application examples, January 2012. A5E03806704-01.

[57] Siemens AG. Fluidized bed dryer - design of model predictive control with economical steady state optimization. Tech. rep., Siemens AG, 2012. MPC for Fluidized Bed Dryer V 1.0, Item-ID: 61926069.

[58] Siemens AG. S7-1200 programmable controller system manual, April 2012. A5E02486680-06.

[59] Siemens AG. SIMATIC controllers: The innovative solution for all automation tasks, November 2012.

[60] Siemens AG. S7-1500 cpu 1516-3 pn/dp, January 2013. A5E03461313-01.

[61] Mc Avoy, T., Arkun, Y., Chen, R., Robinson, D., and Schnelle, P. A new approach to defining a dynamic relative gain. *Control Enginering Practice 11* (2002), 907 – 914.

[62] Mercer, J. Functions of positive and negative type and their connection with the theory of integral equations. *Philosophical Transactions of the Royal Society 209* (1909), 415 – 446.

[63] Nelles, O. *Nonlinear System Identification: From Classical Approaches to NeuralNetworks and Fuzzy Models.* Springer, 2000.

[64] Pannocchia, G., Rawlings, J. B., and Wright, S. J. Fast, large-scale model predictive control by partial enumeration. *Automatica 43* (2007), 852 – 860.

[65] Pintelon, R., and Schoukens, J. *System Identification: A Frequency Domain Approach.* John Wiley & Sons, 2004.

[66] Pistikopoulos, E. N., Dua, V., Bozinis, N. A., Bemporad, A., and Morari, M. On-line optimization via off-line parametric optimization tools. *Computers & Chemical Engineering 26* (2002), 175 – 185.

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

[68] Rawlings, J., and Muske, K. The stability of constrained receding horizon control. *IEEE Transactions on Automatic Control 38* (1993), 1512 – 1516.

[69] Richalet, J., Rault, A., Testud, J., and Papon, J. Model predictive heuristic control: Applications to industrial processes. *Automatica 14* (1978), 413 – 428.

[70] Rossiter, J. *Model-based predictive control: a practical approach.* CRC, 2003.

[71] Saia. Saia system catalogue 2013/2014, February 2013. P+P 26/215 EN10 02.2013.

[72] Schmid, C., and Biegler, L. Quadratic programming methods for reduced hessian SQP. *Computers and Chemical Engineering 18* (1994), 817 – 832.

[73] Seader, J. D., and Henley, E. J. *Separation Process Principles.* John Wiley & Sons, Inc., 2006.

[74] Seron, M., De Dona, J., and Goodwin, G. Global analytical model predictive control with input constraints. In *Proceedings of the 39th IEEE Conference on Decision and Control* (2000), pp. 154 – 159.

[75] Shinskey, F. G. *Process Control Systems, Application, Design, and Tuning.* McGraw-Hill Publishing Company, New York, 1988.

[76] Skogestad, S. Dynamics and control of distillation columns: A tutorial introduction. *Chemical Engineering Research and Design 75* (1997), 539 – 562.

[77] Skogestad, S., and Morari, M. Implications of large rga elements on control performance. *Industrial & Engineering Chemistry Research 26* (1987), 2323 – 2330.

[78] Skogestad, S., Morari, M., and Doyle, J. Robust control of ill-conditioned plants: high-purity distillation. *IEEE Transactions on Automatic Control 33* (1988), 1092 – 1105.

[79] Söderström, T., and Stoica, P. *System Identification*. Prentice-Hall, London, United Kingdom, 1989.

[80] Sugiura, N. Further analysts of the data by Akaike's information criterion andthe finite correction. *Communications in Statistics - Theory and Methods 7* (1978), 13 – 26.

[81] Suykens, J. A., Van Gestel, T., De Brabanter, J., De Moor, B., and Vandewalle, J. *Least Squares Support Vector Machines*. World Scientific, Singapore, 2002.

[82] Valencia-Palomo, G., and Rossiter, J. Efficient suboptimal parametric solutions to predictive control for PLC applications. *Control Engineering Practice 19* (2011), 732 – 743.

[83] Van Overschee, P., and De Moor, B. *Subspace Identification of Linear Systems: Theory, Implementation, Applications*. Kluwer Academic Publishers, 1996.

[84] VanAntwerp, J., and Braatz, R. Fast model predictive control of sheet and film processes. *Control Systems Technology, IEEE Transactions on 8* (2000), 408 – 417.

[85] Vapnik, V. *Statistical Learning Theory*. John Wiley & Sons, Inc, 1999.

[86] Wächter, A., and Biegler, L. T. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming 106* (May 2006), 25 – 57.

[87] Wang, L. *Model Predictive Control System Design and Implementation Using MATLAB*. Springer-Verlag London Limited, 2009.

[88] Wang, Y., and Boyd, S. Fast model predictive control using online optimization. *IEEE Transactions on Control Systems Technology 18* (2010), 267 – 278.

[89] Wong, E. *Active-Set Methods for Quadratic Programming*. PhD thesis, UNIVERSITY OF CALIFORNIA, SAN DIEGO, 2011.

[90] Wright, J. R. The debate over which PLC programming language is the state-of-the-art. *Journal of Industrial Technology 15* (1999), 2–5.

[91] Wright, S. J. *Primal-Dual Interior-Point Methods*. SIAM, 1997.

Appendix C

Curriculum vitae



Bart Louis Julien Huyck was born on May 12, 1979 in Gent, Belgium. He went to the Sint-Jan Berchmanscollege in Sint-Amandsberg where he graduated in 1997. He received a degree of Master of Science of Industrial Engineering Electromechanics, option Automation (industrieel ingenieur elektromechanica, optie automatisering) in 2002. The master thesis was entitled *Softwareontwikkeling van een visiesysteem voor automatische sortering van Chamaedorea-planten* and won the 2002 Egemin price. In 2004 he received his Master degree in Biomedical and clinical engineering at the University of Ghent. After one year as co-worker in the Tetra Project *Elektrisch energiebeheer bij laagspanningsklanten* at KaHo St-Lieven in Ghent, he became an assistant at the same University College. Half of the allocated time was dedicated to PhD research, whereas the rest comprised educational tasks for industrial engineering students at KaHo St-Lieven. The PhD research started in April 2007 and the results of this research are summarized in this dissertation.

Appendix D

Publications by the author

Journal articles with full review

1. Bart Huyck, Hans Joachim Ferreau, Moritz Diehl, Jos De Brabanter, Jan Van Impe, Bart De Moor and Filip Logist. Towards online model predictive control on a programmable logic controller: practical considerations. *Mathematical Problems in Engineering*, 2012:20, 2012 (Article ID 912603, http://www.hindawi.com/journals/mpe/2012/912603/, doi:10.1155/2012/912603).

2. Diana Ugryumova, Gerd Vandersteen, Bart Huyck, Filip Logist, Jan Van Impe, and Bart De Moor. Identification of a non-insulated distillation column from transient response data. *IEEE Transactions on Instrumentation & Measurement*, 2012:10, 2012 (doi:10.1109/TIM.2012.2234577).

3. Julian Bonilla, Filip Logist, Bart Huyck, Jan Degrève, Bart De Moor and Jan Van Impe. A simultaneous approach for calibrating Rate Based Models of Packed Distillation columns based on multiple experiments. *Chemical Engineering Science* [Submitted].

4. Bart Huyck, Jos De Brabanter, Jan Van Impe, Bart De Moor and Filip Logist. Online model predictive control hosted by low level hardware: a pilot-scale distillation column case study. *Control Engineering Practice* [Submitted].

Conference articles with full review

1. Bart Huyck, Jos De Brabanter, Jan Van Impe, and Bart De Moor. Identification and modeling of a dynamical system. In *Proceedings of the Third European Conference on the Use of Modern Information and Communication Technologies*, pages 213–224, Gent, Belgium, March 13–14, 2008.

2. Filip Logist, Bart Huyck, Maarten Fabré, Maarten Verwerft, Bert Pluymers, Jos De Brabanter, Bart De Moor, and Jan Van Impe. Identification and control of a pilot scale binary distillation column. In *Proceedings of European Control Conference 2009 (ECC09)*, pages 4659–4664, Budapest, Hungary, August 23–26, 2009.

3. Bart Huyck, Kris De Brabanter, Filip Logist, Jos De Brabanter, Jan Van Impe, and Bart De Moor. Identification of a pilot scale distillation column: A kernel based approach. In *Proceedings of the 18th World Congress of the International Federation of Automatic Control*, pages 471–476, Milano, Italy, August 28–September 2, 2011.

4. Bart Huyck, Filip Logist, Jos De Brabanter, Jan Van Impe, and Bart De Moor. Constrained model predictive control on a programmable automation system exploiting code generation: practical considerations. In *Proceedings of the 18th World Congress of the International Federation of Automatic Control*, pages 12207–12212, Milano, Italy, August 28–September 2, 2011.

5. Diana Ugryumova, Gerd Vandersteen, Bart Huyck, Filip Logist, Jan Van Impe, and Bart De Moor. Identification and modeling of distillation columns from transient response data. In *Proceedings of the IEEE International Instrumentation and Measurement Technology Conference (I2MTC 2012)*, pages 2098–2103, Graz, Austria, May 13–16, 2012.

6. Bart Huyck, Filip Logist, Hans Joachim Ferreau, Moritz Diehl, Jos De Brabanter, Jan Van Impe, and Bart De Moor. Implementation and experimental validation of classical mpc on programmable logic controllers. In *Proceedings of the 20th Mediterranean Conference on Control and Automation. Barcelona (MED 2012)*, pages 679–684, Barcelona, Spain, July 3-6, 2012.

7. Bart Huyck, Jos De Brabanter, Bart De Moor, Jan Van Impe, and Filip Logist. Model predictive control of a pilot-scale destination column using a programmable automation controller In *Proceedings of European Control Conference 2013 (ECC13)*, pages 1053–1058, Zurich, Switzerland, July 17-19, 2013

Conference abstracts

1. Bart Huyck, Jos De Brabanter, Jan Van Impe, and Bart De Moor. Identification and modeling of a distillation column. In *27th Benelux Meeting on Systems and Control - Book of Abstracts*, page 17, Heeze, The Netherlands, March 18–20, 2008.

2. Bart Huyck, Filip Logist, Jos De Brabanter, Jan Van Impe, and Bart De Moor. Identification of a distillation column for plc control purposes. In *28th Benelux Meeting on Systems and Control - Book of Abstracts*, page 21, Spa, Belgium, March 16–18, 2009.

3. Bart Huyck, Filip Logist, Jos De Brabanter, Jan Van Impe, and Bart De Moor. Identification and control of a binairy distillation column in view of plc based control. In *29th Benelux Meeting on Systems and Control - Book of Abstracts*, page 33, Heeze, The Netherlands, March 30–April 1, 2010.

4. Bart Huyck, Filip Logist, Jos De Brabanter, Jan Van Impe, and Bart De Moor. Code generation for mpc controllers on industrial hardware. In *30th Benelux Meeting on Systems and Control - Book of Abstracts*, page 27, Lommel, Belgium, March 15–17, 2011.

5. Bart Huyck, Kris De Brabanter, Filip Logist, Jos De Brabanter, Jan Van Impe, and Bart De Moor. Ls-svm identification of a distillation column. In *30th Benelux Meeting on Systems and Control - Book of Abstracts*, page 139, Lommel, Belgium, March 15–17, 2011.

6. Diana Ugryumova, Bart Huyck, Gerd Vandersteen, Julian Bonilla, Filip Logist, Jan van Impe, and Rik Pintelon. Identification of distillation columns using multisine excitation. In *30th Benelux Meeting on Systems and Control - Book of Abstracts*, page 39, Lommel, Belgium, March 15–17, 2011.

7. Bart Huyck, Filip Logist, Jos De Brabanter, Jan Van Impe, and Bart De Moor. Online mpc on industrial programmable logic controllers. In *31th Benelux Meeting on Systems and Control - Book of Abstracts*, page 54, Heijden, netherlands, March 27–29, 2012.

8. Diana Ugryumova, Gerd Vandersteen, Bart Huyck, and Filip Logist. Identification and modeling of distillation columns from transient response data. In *31th Benelux Meeting on Systems and Control - Book of Abstracts*, page 172, Heijden, netherlands, March 27–29, 2012.

FACULTY OF ENGINEERING SCIENCE
DEPARTMENT OF ELECTRICAL ENGINEERING
ESAT-SCD
Kasteelpark Arenberg 10
B-3001 Heverlee