

# High-performance model predictive control for process industry

***Citation for published version (APA):***

Tyagounov, A. (2004). *High-performance model predictive control for process industry*. [Phd Thesis 1 (Research TU/e / Graduation TU/e), Electrical Engineering]. Technische Universiteit Eindhoven.  
<https://doi.org/10.6100/IR576761>

***DOI:***

[10.6100/IR576761](https://doi.org/10.6100/IR576761)

***Document status and date:***

Published: 01/01/2004

***Document Version:***

Publisher's PDF, also known as Version of Record (includes final page, issue and volume numbers)

***Please check the document version of this publication:***

- A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
- The final author version and the galley proof are versions of the publication after peer review.
- The final published version features the final layout of the paper including the volume, issue and page numbers.

[Link to publication](#)

***General rights***

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the "Taverne" license above, please follow below link for the End User Agreement:

[www.tue.nl/taverne](http://www.tue.nl/taverne)

***Take down policy***

If you believe that this document breaches copyright please contact us at:

[openaccess@tue.nl](mailto:openaccess@tue.nl)

providing details and we will investigate your claim.

# *High-Performance Model Predictive Control for Process Industry*

PROEFSCHRIFT

ter verkrijging van de graad van doctor aan de  
Technische Universiteit Eindhoven,  
op gezag van de Rector Magnificus, prof.dr. R.A. van Santen,  
voor een commissie aangewezen door het College voor  
Promoties in het openbaar te verdedigen op  
vrijdag 18 juni 2004 om 16.00 uur

door

Andrey Alexandrovich Tyagunov

geboren te Kaluga, Rusland

Dit proefschrift is goedgekeurd door de promotoren:

prof.dr.ir. A.C.P.M. Backx  
en  
prof.ir. O.H. Bosgra

Copromotor:  
dr. S. Weiland

CIP-DATA LIBRARY TECHNISCHE UNIVERSITEIT EINDHOVEN

Tyagunov, Andrey A.

High-performance model predictive control for process industry / by  
Andrey A. Tyagunov. – Eindhoven: Technische Universiteit Eindhoven, 2004.  
Proefschrift. – ISBN 90-386-1573-6  
NUR 959

Trefw.: procesregeling / regeltechniek ; proces-industrie / nietlineaire  
systemen / regelsystemen ; optimalisering / kwadratisch programmeren.  
Subject headings: process control / predictive control / nonlinear systems /  
optimal control / quadratic programming.



Eerste promotor:      prof.dr.ir. A.C.P.M. Backx

Tweede promotor:    prof.ir. O.H. Bosgra

Copromotor:          dr. S. Weiland

Kerncommissie:

prof.dr.ir. M. Steinbuch

prof.dr.ir. B.L.R. De Moor

The research has been funded by the European Commission under grant G1RD-CT-1999-00146 in the EU project “INtegration of process COntrol and plantwide OPtimization” (INCOOP).

The Ph.D. work forms a part of the research program of the Dutch Institute of Systems and Control (DISC).

## *Preface*

Process industry has confronted a major change in the market during the past decades. World wide competition has drastically increased and environmental legislation has been tightened severely on the consumption of natural resources. Market driven operation in process industry has in addition confronted further complicating factors related to hard operating constraints imposed upon production sites in terms of required reduction of consumption of energy and materials. The environmental constraints imposed by legislation has in addition resulted in a significant increase of process complexity and costs of production equipment. More advanced process support systems will therefore be required to exploit the freedom available in process operation. This thesis presents an example of a model based control technology that can be used to support process operation in the most flexible way, in accordance with market requirements.

The model predictive control technology is used to steer processes closer to their physical limits in order to obtain a better economic result. This is perhaps one of the most appealing and attractive approaches in industrial process control practice. Despite of its importance, various problems associated with this subject still remain unsolved, providing respectful challenges for researchers. There is a wide spectrum of essential issues to be investigated in the area of process control. This thesis represents an attempt to light up several aspects related to the model predictive control approach for nonlinear processes. It is my hope that this work will have some impact on further research in this field as well as the industrial practice.

There are many people who have contributed to my work in different ways and who deserve my deep gratitude. First of all I would like to thank my supervisor Prof. Ton Backx for his continuous support during these four years. I am grateful for his endless optimism and encouraging my collaboration with other colleagues and professors.

I want to express my gratitude to Dr. Siep Weiland who provided a unique support, not only during the entire work on this PhD thesis, but also during my MSc studies. My work has benefited very much from his valuable ideas, instructive comments and suggestions. I would also like to thank Prof. Okko Bosgra who kindly agreed to be my second promotor and provided a lot of suggestions for the final version of my thesis.

In particular, I would like to acknowledge my collaboration with Jeroen Buijs (KU Leuven) who provided the initial framework and tools for working on development of the structured interior-point method.

My fellow AIO's have also been very helpful. I would like to mention Mario, Aleksandar, Patricia, Leo, Bart and Mircea for creating a pleasant atmosphere

in the group. I am also grateful to my former roommate Hardy for many helpful discussions on the common issues. Besides, the help of Udo Bartzke is certainly appreciated for solving a lot of computer related problems.

I am especially grateful to my parents for their enormous support during all the years of my studies and research work. They always encouraged me to follow the way I chose and convinced me that I would actually finish this thesis.

Finally, I will mention the most important person in my life and who is always there for me. I would like to thank my fiancée Katya for her endless support and understanding. I would not have passed through the last two years without her.

Andrey Tyagunov  
Eindhoven, April 2004

# *Abstract*

Process industry requires now accurate, efficient and flexible operation of the plants. There is always a need for development of innovative technologies and integrated software tools for process modelling, dynamic trajectory optimization and high-performance industrial process control. Process dynamics tend to become too complex to be efficiently controlled by the current generation of control and optimization techniques. The main goal of research in this thesis was the development of advanced process control technology and its implementation in an integrated real-time software environment. The research was done as part of the international European project: “INtegration of process COntrol and plantwide OPTimization” (INCOOP).

The only advanced control technology which made a significant impact on industrial control engineering is model predictive control (MPC). Specifically, the research in this thesis is focused on MPC for nonlinear processes. Nonlinear MPC optimizations become computationally expensive to be solved in real-time. This thesis presents various MPC algorithms for nonlinear plants using successive linearizations. The prediction equation is computed via nonlinear integration. Local linear approximation of the state equation is used to develop an optimal prediction of the future states. The output prediction is made linear with respect to the undecided control input moves, which allows to reduce the MPC optimization to a quadratic programming problem (QP).

It is shown that the constrained QP problem can be solved in various ways. First of all, one can use the model equations to eliminate the states, thus reducing the number of variables in the optimization. However, this makes the problem formulation dense. Solving QPs with these methods typically requires a computational time that increases with the third power of the number of optimization variables. The constrained optimization programs tend to become too large to be solved in real-time when these standard QP solvers are used. Many industrial examples show that large-scale, usually stiff, nonlinear systems may require long prediction horizons to fulfill certain performance specifications. These requirements increase the number of variables in the optimization. Naive implementations of standard QP solvers could be inefficient for such MPC problems. A structured interior-point method (IPM) has been developed in this thesis to solve the MPC problem for large-scale nonlinear systems to reduce the computational complexity. The developed optimization algorithm explicitly takes the structure of the given problem into account such that the computational cost varies linearly with the number of optimization variables, compared with the cubic growth for the standard QP solvers. The algorithm also easily allows to introduce multiple linear models, thus making the control more flexible. The state elimination was not carried out and the structure given by the dynamics



of the plant reflected in the Karush-Kuhn-Tucker (KKT) equations that were used to solve the QP using an interior-point method. The structured IPM was implemented using primal-dual Mehrotra's algorithm including prediction, correction and centering steps. The optimization variables consist of the inputs and the states over the horizon, but the optimization problem becomes sparse to allow computational time reduction, which is an important issue for on-line implementations of MPC for nonlinear stiff systems.

In this thesis the effectiveness of the structured IPM based model predictive controller was demonstrated on several industrial chemical processes, e.g. a continuous stirred tank reactor and a stiff nonlinear batch reactor. These types of systems are usually represented by dynamics with time constants of different magnitude. A range of control problems, such as reference tracking, process start-up and disturbance rejection, has been efficiently solved in this thesis by the proposed high-performance MPC controller. The controller has also been successfully tested as part of the INCOOP's integrated process control and optimization software environment.

# Contents

<b>Preface</b>	<b>1</b>
<b>Abstract</b>	<b>3</b>
<b>1 Introduction</b>	<b>9</b>
1.1 Process industry perspective . . . . .	9
1.2 Model Predictive Control overview . . . . .	10
1.2.1 Introduction . . . . .	10
1.2.2 Nonlinear models . . . . .	11
1.2.3 Theoretical developments for nonlinear MPC . . . . .	13
1.2.4 Industrial implementations of nonlinear MPC . . . . .	18
1.2.5 Future needs for NMPC technology development . . . . .	24
1.3 Industrial technology . . . . .	25
1.3.1 Commercial predictive control schemes . . . . .	25
1.4 Receding horizon strategy . . . . .	26
1.4.1 Open-loop vs. closed-loop MPC . . . . .	28
1.5 Integration of process control and plantwide optimization . . . . .	28
1.5.1 Research motivation: role of MPC . . . . .	29
1.6 Objectives and main results of the thesis . . . . .	31
1.6.1 Research objectives . . . . .	31
1.6.2 Main results . . . . .	32
<b>2 Linear Model Predictive Controllers</b>	<b>35</b>
2.1 Basic properties of MPC . . . . .	35
2.2 Process models and prediction . . . . .	37
2.3 Optimal control . . . . .	38
2.3.1 Unconstrained linear case . . . . .	39
2.4 The finite horizon MPC problem . . . . .	40
2.4.1 Model . . . . .	41
2.4.2 Problem formulation . . . . .	42
2.4.3 Prediction: full state measurement . . . . .	44
2.4.4 Solving the MPC problem . . . . .	46
2.4.5 Stability issues . . . . .	49
2.4.6 Mathematical programming: optimality conditions . . . . .	50
2.4.7 Primal-dual interior-point methods . . . . .	53
2.5 Implementation and tuning . . . . .	56
2.5.1 Estimated state and observer dynamics . . . . .	56
2.5.2 Limitations of the current technology . . . . .	58

2.6	Design case studies . . . . .	59
2.6.1	Evaporation process . . . . .	59
2.6.2	High-purity binary distillation column . . . . .	64
2.6.3	Motivating example: Oil fractionator . . . . .	68
2.6.4	Concluding remarks . . . . .	75
<b>3</b>	<b>Model Predictive Control for Nonlinear Processes</b>	<b>77</b>
3.1	Nonlinear models and predictive control . . . . .	77
3.2	Algorithm overview . . . . .	78
3.3	MPC algorithm for nonlinear systems . . . . .	80
3.3.1	Model . . . . .	80
3.3.2	State estimation . . . . .	81
3.3.3	MPC problem formulation . . . . .	85
3.3.4	Approximation and prediction . . . . .	85
3.3.5	Optimizing control input in output prediction . . . . .	89
3.3.6	Algebraic problem formulation . . . . .	91
3.3.7	Constraint types and softening . . . . .	92
3.4	Structured interior-point method . . . . .	93
3.4.1	States as optimization variables in MPC . . . . .	94
3.4.2	Structured KKT equations . . . . .	96
3.4.3	Interior-point method based on SKKT equations . . . . .	97
3.5	Implementation of the structured interior-point algorithm . . . . .	103
3.5.1	IPM and calculation of the residuals . . . . .	104
3.5.2	The overall structured interior-point algorithm . . . . .	104
3.6	MPC application results . . . . .	105
3.6.1	Evaporation process . . . . .	105
3.6.2	High-purity binary distillation column . . . . .	106
3.6.3	MPC with reduced linear models . . . . .	108
<b>4</b>	<b>Chemical Process Applications</b>	<b>115</b>
4.1	MPC of a benchmark continuous stirred tank reactor . . . . .	115
4.1.1	Description of a CSTR . . . . .	115
4.1.2	Control problem at the point of optimal yield . . . . .	117
4.1.3	Process start-up . . . . .	120
4.1.4	Feed temperature disturbance attenuation . . . . .	122
4.1.5	Reference tracking problem . . . . .	123
4.2	A stiff nonlinear DAE test problem . . . . .	126
4.2.1	Batch reactor model . . . . .	127
4.2.2	Setpoint tracking problem . . . . .	130
4.3	Polymerization process . . . . .	130
4.3.1	Optimal load change problem . . . . .	133

---

<b>5</b>	<b>Conclusions and Outlook</b>	<b>139</b>
5.1	Review of the results . . . . .	139
5.2	Recommendations for further research . . . . .	141
<b>A</b>	<b>Dynamic Programming</b>	<b>143</b>
A.1	Interior-Point Methods . . . . .	143
A.1.1	Linear programming: primal-dual methods . . . . .	144
A.1.2	The central path . . . . .	146
A.1.3	A practical primal-dual algorithm . . . . .	148
A.1.4	Quadratic programming: IPM . . . . .	150
A.2	Active Set Methods . . . . .	152
A.2.1	Constrained QP . . . . .	153
A.2.2	Specification of ASM for convex QP . . . . .	155
	<b>Notation</b>	<b>159</b>
	<b>Bibliography</b>	<b>163</b>
	<b>Samenvatting</b>	<b>173</b>



# 1

## *Introduction*

---

1.1	Process industry perspective	1.5	Integration of process control and plantwide optimization
1.2	Model Predictive Control overview	1.6	Objectives and main results of the thesis
1.3	Industrial technology		
1.4	Receding horizon strategy		

---

### *1.1 Process industry perspective*

Process industry has confronted a major change in the market during the past decades. World wide competition has drastically increased and environmental legislation has been tightened severely on the consumption of natural resources. Market driven operation in process industry has in addition confronted further complicating factors related to hard operating constraints imposed upon production sites in terms of required reduction of consumption of energy and materials. The environmental constraints imposed by legislation has in addition resulted in a significant increase of process complexity and costs of production equipment. More advanced process support systems will therefore be required to exploit the freedom available in process operation. Many of the process industries are still operating their production facilities in a supply driven mode. This implies that no direct connection exists in these companies between actual market demand and actual production. Products are often produced cyclically in fixed sequences.

One of the major reasons for the changes is globalization of the market. Globalization is one of the results of the recent developments in the fields of telecommunication, transportation and advanced automation, which have emerged from the rapid developments in electronics, computer and information technology. As a consequence process industry has nowadays confronted a strongly competitive environment. The market has developed from a supplier driven market to a demand driven market. These changes have far reaching consequences for producers. The market requires producers to respond quickly and reliably to product demands. Products have to be delivered with short notice in strictly defined time windows at the right quality and in the requested

volume. The competition between product suppliers is heavy in a saturated market. If products meet imposed specifications, price will become the main discriminating factor to get a customer to decide to buy the product from a given supplier. After optimization of production, costs will level off at some minimum that can hardly be further reduced without a further improvement of the applied production processes.

In order to prepare for these drastic changes, tight control of the production processes over a broad operating range is needed. Process operation has to enable a completely predictable and reproducible operation with changeover between different operating points that correspond to the production of various product types under different economic objectives (minimize costs, maximize production rate, minimize stock, benefit from fluctuating prices, etc.). The strategy that results in the most profitable conditions has to be selected from a variety of potential operating scenarios to produce the desired product type. This decision is based on a thorough understanding of both process behavior and process operation. The freedom, available in process operation, must be used to predictably produce precisely what is required in terms of quality, volume and time with the best achievable business result.

This thesis presents an example of a model based control technology that can be used to support process operation in the most flexible way, in accordance with market requirements. The model predictive control technology is used to steer processes closer to their physical limits in order to obtain a better economic result.

## **1.2 Model Predictive Control overview**

### **1.2.1 Introduction**

The MPC research literature is large, but review papers have appeared at regular intervals. Theoretical and practical issues associated with MPC technology are summarized in several recent articles. The three MPC papers presented at the CPC conference in 1996 are an excellent starting point [45, 54, 73]. Qin and Badgwell [73] present a brief history of MPC technology and a survey of industrial MPC algorithms and applications that practitioners may find particularly useful. Meadows and Rawlings summarize theoretical properties of MPC algorithms in [59]. Morari and Lee discuss the past, present, and future of MPC technology in their recent review [64]. Kwon provides a very extensive list of references [42]. A more recent overview of MPC theory development can be found in [25]. Moreover, several excellent books have appeared recently [65, 86, 8]. The status of industrial MPC for nonlinear plants is covered in the proceedings of the 1998 conference [2].

The success of MPC technology as a process control paradigm can be attributed to three important factors. First and foremost is the incorporation

of an explicit process model into the control calculation. This allows the controller, in principle, to deal directly with all significant features of the process dynamics. Secondly the MPC algorithm considers plant behavior over a future horizon in time. This means that the effects of feedforward and feedback disturbances can be anticipated and removed, allowing the controller to drive the plant more closely along a desired future trajectory. Finally the MPC controller considers process input, state and output constraints directly in the control calculation. This means that constraint violations are far less likely, resulting in tighter control at the optimal constrained steady-state for the process. It is the inclusion of constraints that most clearly distinguishes MPC from other process control paradigms.

Though manufacturing processes are inherently nonlinear, the vast majority of MPC applications to date are based on linear dynamic models, the most common being step and impulse response models derived from the convolution integral. There are several potential reasons for this. Linear empirical models can be identified in a straightforward manner from process test data. In addition, most applications to date have been in refinery processing [73], where the goal is largely to maintain the process at a desired steady-state (regulator problem), rather than moving rapidly from one operating point to another (servo problem). A carefully identified linear model is sufficiently accurate in the neighborhood of a single operating point for such applications, especially if high quality feedback measurements are available. Finally, by using a linear model and a quadratic objective, the nominal MPC algorithm takes the form of a highly structured convex Quadratic Program (QP), for which reliable solution algorithms and software can easily be found [98]. This is important because the solution algorithm must converge reliably to the optimum in no more than a few tens of seconds to be useful in manufacturing applications. For these reasons, in many cases a linear model will provide the majority of the benefits possible with MPC technology.

Nevertheless, there are cases where nonlinear effects are significant enough to justify the use of NMPC technology. These include at least two broad categories of applications:

- Regulator control problems where the process is highly nonlinear and subject to large frequent disturbances (pH control, etc.).
- Servo control problems where the operating points change frequently and span a sufficiently wide range of nonlinear process dynamics (polymer manufacturing, ammonia synthesis, etc.).

### 1.2.2 Nonlinear models

Many processes are nonlinear with varying degrees of severity. Although in many situations the process will be operating in the neighborhood of a steady



state, and therefore a linear representation will be adequate, there are some very important situations where this does not occur. On one hand, there are processes for which the nonlinearities are so severe (even in the vicinity of steady states) and so crucial to the closed loop stability, that a linear model is not sufficient. On the other hand, there are processes that experience continuous transitions (start-ups, shutdowns, etc.) and spend a great deal of time away from a steady-state operating region, or never reach steady-state operation as is the case of batch processes where the whole operation is carried out in transient mode.

For these processes a linear control law will not be very effective, so nonlinear controllers will be essential for improved performance or simply for stable operation. There is nothing in the basic concepts of MPC against the use of a nonlinear model. Therefore, the extension of MPC ideas to nonlinear processes is straightforward at least conceptually. However, this is not a trivial matter, and there are many open issues (see [11]), such as:

- The availability of nonlinear models due to the lack of identification techniques for nonlinear processes.
- The computational complexities for solving the model predictive control of nonlinear processes.
- The lack of stability and robustness results for the case of nonlinear systems.

Some of these problems are partially solved and MPC, with the use of nonlinear models, is becoming a field of intense research and will become more common as users demand higher performance.

Developing adequate nonlinear empirical models may be very difficult and there is no model form that is clearly suitable to represent general nonlinear processes. Part of the success of standard MPC was due to the relative ease with which step and impulse responses or low order transfer functions could be obtained. Nonlinear models are much more difficult to construct, either from input/output data correlation or by the use of first principles from well known mass and energy conservation laws. A major mathematical obstacle to a complete theory of nonlinear processes is the lack of a superposition principle for nonlinear systems. Because of this, the determination of models from process input/output data becomes a very difficult task. The amount of plant tests required to identify a nonlinear plant is much higher than that for a linear plant.

A nonlinear model inside the MPC controller can be represented in state space form (see, for example [73, 76]):

$$\begin{aligned} \mathcal{P}x &= f(x, u) \\ y &= g(x) \end{aligned}$$

$$\begin{aligned} u &\in \mathcal{U} \\ x &\in \mathcal{X}, \end{aligned}$$

where  $\mathcal{P} = \frac{d}{dt}$  for continuous time and  $\mathcal{P}$  is the forward shift  $\mathcal{P}x(k) = x(k+1)$  for discrete time,  $u$  is input,  $x$  is state,  $y$  is output, all assuming (real) values in some finite dimensional vector space. If the model is nonlinear, there is no advantage in keeping the constraints as linear inequalities, so we consider the constraints as membership in more general regions  $\mathcal{U}, \mathcal{X}$ .

The use of nonlinear models in MPC is motivated by the possibility to improve control by improving the quality of the forecasting. The basic fundamentals in any process control problem—conservation of mass, momentum and energy, considerations of phase equilibria, relationships of chemical kinetics, and properties of final products—all introduce nonlinearity into the process description. In which settings use of nonlinear models for forecasting delivers improved control performance is an open issue, however. For continuous processes maintained at nominal operating conditions and subject to small disturbances, the potential improvement would appear small. For processes operated over large regions of the state space—semi-batch reactors, frequent product grade changes, processes subject to large disturbances, for example—the advantages of nonlinear models appear larger.

### 1.2.3 Theoretical developments for nonlinear MPC

In principle the NMPC method is limited to those problems for which a global optimal solution to the dynamic optimization can be found between one control execution and the next. With a linear model and a quadratic objective, the resulting optimization problem takes the form of a highly structured convex Quadratic Program (QP) for which there exists a unique optimal solution. Several reliable standard solution codes are available for this problem. Introduction of a nonlinear model leads, in the general case, to a loss of convexity. This means that it is much more difficult to find a solution, and once found, it cannot be guaranteed to be globally optimal. For both cases, recent research efforts are aimed at exploiting the structure to improve the efficiency and reliability of solution codes [98].

#### Stability

The early major contribution to receding horizon (model predictive) control for nonlinear systems was the demonstration by Keerthi and Gilbert [39] that, for *time-varying, constrained, nonlinear*, discrete-time systems, the addition of a terminal *stability constraint*  $x(k+N|k) = x_s$  to the open-loop optimal control problem ensures that, under mild conditions, the resultant receding horizon controller is stabilizing. This result is a significant generalization of the earlier linear results.

It is interesting to note that some of the very first MPC papers describe ways to address nonlinear process behavior while still retaining a linear dynamic model in the control algorithm. Richalet et al. [79], for example, describe how nonlinear behavior due to load changes in a steam power plant application was handled by executing their Identification and Command (IDCOM) algorithm at a variable frequency. Prett and Gillette [72] describe applying a Dynamic Matrix Control (DMC) algorithm to control a fluid catalytic cracking unit. Model gains were obtained at each control iteration by perturbing a detailed nonlinear steady-state model.

Recent research efforts on the problem of stability of NMPC with a perfect model (the so-called *nominal stability problem*) has produced three basic solutions (outlined in [1]). The first solution, proposed by Keerthi and Gilbert [39], involves adding a terminal state constraint to the NMPC algorithm of the form:  $x(k + N|k) = x_s$ . With such a constraint enforced, the objective function for the controller becomes a Lyapunov function for the closed loop system, leading to nominal stability. Unfortunately such a constraint may be quite difficult to satisfy in real time; exact satisfaction requires an infinite number of iterations for the numerical solution code. This motivated Michalska and Mayne [61] to seek a less stringent stability requirement. Their main idea is to define a neighborhood  $\mathcal{W}$  around the desired steady-state  $x_s$  within which the system can be steered to  $x_s$ , by a constant linear feedback controller. They add to the NMPC algorithm a constraint of the form:  $(x(k + N|k) - x_s) \in \mathcal{W}$ . If the current state  $x(k)$  lies outside this region then the NMPC algorithm is solved with the above constraint. Once inside the region  $\mathcal{W}$  the control switches to the previously determined constant linear feedback controller. Michalska and Mayne describe this as a *dual-mode* controller.

A third solution to the nominal stability problem, described by Meadows et al. [58], involves setting the prediction horizon and control horizons to infinity. For this case the objective function also serves as a suitable Lyapunov function, leading to nominal stability. They demonstrate that if the initial NMPC calculation has a feasible solution, then a feasible solution exists at each subsequent time step.

### ***Infinite horizon NMPC***

In linear MPC, infinite horizons (approximated by very large horizons) are in many cases a practical route to achieving stability as there exist very efficient ways to solve the corresponding huge quadratic programming problems. For nonlinear problems, on the other hand, the solution of such large optimization problems is extremely difficult if not impossible to obtain. Therefore finite horizons are indispensable in NMPC and infinite horizon NMPC only plays a role as a conceptual theoretical method.

In principle, it would be desirable to have a controller design procedure

that would allow to determine stabilizing prediction and control horizons for an NMPC setup based on the plant model and the chosen stage cost. This problem is, however, very difficult and has not yet been solved. There are even no analysis methods available that permit to analyze closed loop stability based on knowledge of the plant model, the objective functional and the horizon lengths.

Nevertheless, there are possibilities to achieve closed loop stability despite the fact that this property cannot be analyzed. The idea behind these approaches is to modify the NMPC setup such that stability of the closed loop can be guaranteed independent of the choice of the horizon length, independent of the choice of the stage cost and for any plant to be controlled. This is usually achieved by adding suitable equality or inequality constraints (and possibly suitable additional terms to the cost functional) to the setup. These additional constraints are not motivated by physical restrictions or desired performance requirements but have the sole purpose of enforcing stability of the closed loop. Therefore, they are usually termed *stability constraints* [54, 55]. NMPC formulations with a modified setup to achieve closed loop stability independent of the choice of the performance parameters in the cost functional are usually labelled NMPC approaches *with guaranteed stability*. A well-known control method with guaranteed stability is the standard linear quadratic regulator (LQR) that also achieves closed loop stability independent of the choice of the positive definite weighting matrices in the quadratic cost functional [37].

### ***NMPC with zero state terminal equality constraint***

The most widely suggested NMPC scheme with guaranteed stability utilizes a stability constraint in the form of a *zero state terminal equality constraint* [39, 40, 56]:  $x(k + N|k) = 0$ , that forces the state to be zero at the end of the finite horizon. Keerthi and Gilbert [39] were the first to show that feasibility of an NMPC formulation with zero state terminal equality constraint implies stabilization of a class of nonlinear constrained systems. Later, this result was further expanded in [77, 58]. Continuous time versions can be found in [56]. Under reasonable conditions, asymptotic stability for the closed-loop system can be proven.

Guaranteeing stability by imposing a zero state terminal equality constraint is by far the most popular technique at present. For one, this is certainly due to the clear theoretical framework, but also due to the fact that no on-line computation or tuning is needed. On the other hand, a terminal equality constraint is an artificial additional burden that may require significant extra on-line computation cost (see [17, 15] for a comparison to other approaches) and even more importantly, leads in many cases to a severely restricted region of operation due to feasibility problems.

### **Dual-mode NMPC**

From a computational point of view, an exact numerical satisfaction of a zero state terminal equality constraint is impossible. If only an approximated satisfaction of the terminal equality constrained is enforced, i.e. we only require the terminal state to lie in a small region around the origin, then the guaranteed stability is in general lost. In order to relax the equality stability constraint while not compromising the closed loop asymptotic stability of the origin, the so-called *dual-mode NMPC scheme* was introduced in [61]. The term dual-mode refers to two different controllers that are applied in different regions of the state space depending on the state being inside or outside of some terminal region that contains the origin. If the state is outside this terminal region, an NMPC controller with a variable horizon is applied. If the current state lies inside the terminal region, a linear state feedback  $u(k) = Kx(k)$  is applied. Closed-loop control with this scheme is implemented by switching between the two controllers.

Under the assumption that the Jacobian linearization of the nonlinear system is stabilizable, a simple procedure to determine a suitable terminal region and a linear feedback matrix  $K$  that satisfy certain requirements in the continuous time case is proposed in [56]. Under fairly weak conditions, it can be shown that starting from outside the terminal region, the nonlinear system with the predictive controller will reach the terminal region in a finite time. Closed-loop stability follows from the use of the stabilizing local linear feedback law thereafter. Discrete time versions of dual-mode NMPC with fixed horizons are described in [21, 83].

Computationally this approach is more attractive than the one imposing a terminal equality constraint, as inequality constraint can be handled more effectively during optimization than equality constraints. In addition, less feasibility problems have to be expected and hence the region of attraction will be larger. It was also shown in [61] that the feasibility of the optimization problem, and not necessarily the optimality, is needed for closed-loop stability. A drawback of this approach is the involved implementation due to the required switching between control strategies, the need to determine a local stabilizing state feedback gain and the terminal region, and the fact that the predicted open loop and the actual closed loop trajectories will in general be different.

### **Contractive NMPC**

In contractive NMPC as suggested in [100, 26], a stability constraint of the form  $\|x(k + N|k)\|^2 \leq \alpha^2 \|x(k)\|^2$  is added to the control problem. This constraint directly forces the magnitude of the state vector to contract by a pre-specified factor each time a new input is calculated. This constraint is referred to as *contraction stability constraint*. Different from standard NMPC, the entire input function  $u_k^N$  over the interval  $[k, k + N - 1]$  is applied to the nonlinear

system. The next optimization problem is only solved at time instance  $k + N$ . In the formulation suggested by [26], an additional second stability constraint is included in the formulation and, like in the dual-mode NMPC case, the horizon length  $N \leq N_{\max}$  is used as an additional minimizer.

The feasibility has to be assumed for all time instances for which an optimization is performed. Hence, this approach is not very attractive for applications, since the feasibility of the optimization problem at subsequent sampling instances is in general not guaranteed.

### ***Quasi-infinite horizon NMPC***

In the quasi-infinite horizon NMPC scheme [14, 17] an inequality stability constraint  $x(k + N|k) \in \Omega$  and a quadratic terminal penalty term  $\Phi(x(k + N|k)) = x(k + N|k)^\top P x(k + N|k)$  are added to the standard setup. The basic idea behind this scheme is that the terminal penalty term is not a performance specification that can be chosen freely, but rather that matrix  $P$  is determined off-line according to a specific procedure such that the objective function with  $\Phi$  chosen to approximate an infinite horizon prediction cost functional. This way closed-loop stability can be achieved, while only an optimization problem over a finite horizon must be solved numerically.

In this setting the optimal value of the finite horizon optimization problem bounds that of the corresponding infinite horizon optimization problem. In this sense, the prediction horizon in the nonlinear MPC scheme can be thought of as expanding quasi to infinity. Similar versions for discrete time nonlinear systems can be found in [28, 24]. Like in the dual-mode approach, the use of the terminal inequality constraint gives the quasi-infinite horizon nonlinear MPC scheme computational advantages. The implementation is simpler than the dual-mode approach, because no switching between control strategies is needed. Moreover, the additional terminal penalty term is introduced to approximate an infinite horizon objective functional and thus predicted open loop and actual closed loop trajectories will be at least similar and control performance can be adjusted via the stage cost. Like for dual-mode NMPC, it is not necessary to find optimal solutions of the control problem in order to guarantee stability. Feasibility also implies stability here (see [17]).

### ***Robust NMPC***

A number of results have been published for linear predictive control schemes (see for example [30, 101, 31, 103, 41, 5, 47, 20]). Even though the analysis of robustness properties in nonlinear NMPC must still be considered an unsolved problem in general, some preliminary results are available. Firstly, some of the schemes discussed above (including zero state terminal equality, dual-mode, contractive, quasi-infinite horizon NMPC) have some inherent robustness properties, or can be made robust by simple changes (for example

the use of conservative terminal inequality constraints in the dual-mode approach [61]). However, these results essentially only state that sufficiently “small” model uncertainties do not affect closed loop stability. But they do not permit to derive controllers that guarantee stability for a given uncertainty description with given bounds on the size of this uncertainty. Typically, for all these approaches no explicit quantitative nonlinear uncertainty model or only very simple ones, like for example gain and additive perturbations in [23], are used. This is not surprising as the definition of meaningful uncertainty descriptions for nonlinear systems is an open problem not only in the NMPC context, but also in other areas of control. First NMPC approaches that make use of quantitative uncertainty models are described in [32, 6, 19, 85, 51, 104]. More detailed discussions of this issue can be found in [54, 55, 16].

#### **1.2.4 Industrial implementations of nonlinear MPC**

While theoretical aspects of NMPC algorithms have been discussed quite effectively in several recent publications (see, for example, [54] and [59]), descriptions of industrial NMPC applications are much more difficult to find. This is probably due to the fact that industrial activity in NMPC applications has only begun to take off in the last few years.

From the practical side, industrial implementation of MPC with nonlinear models has already been reported, so it is certainly possible. But the implementations are largely without any established closed-loop properties, even nominal properties. A lack of supporting theory should not and does not, examining the historical record, discourage experiments in practice with promising new technologies. But if nonlinear MPC is to become widespread in the environment of applications, it must eventually become reasonably reliable, predictable, efficient and robust against on-line failure.

In the survey of MPC technology [73], over 2200 commercial applications were discovered. However, almost all of these were implemented with linear models and were clustered in refinery and petrochemical processes. Figure 1.1 (from [73]) shows a rough distribution of the number of MPC applications versus the degree of process nonlinearity. MPC technology has not yet penetrated deeply into areas where process nonlinearities are strong and market demands require frequent changes in operating conditions. It is these areas that provide the greatest opportunity for NMPC applications.

Table 1.1 (from [73]) lists the industrial products and the companies supplying them. The following sub-sections describe these aspects in more detail.

#### **State-space models**

The first issue encountered in NMPC implementation is the derivation of a dynamic nonlinear model suitable for model predictive control. In the gen-

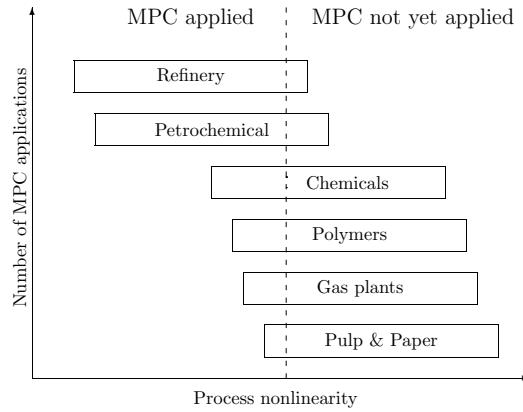


Figure 1.1: Distribution of MPC applications versus the degree of process nonlinearity.

Company	Product name
Adersa	Predictive Functional Control (PFC)
Aspen Technology	Aspen Target
Continental Controls	Multivariable Control (MVC)
DOT Products	NOVA Nonlinear Controller (NOVA-NLC)
Pavilion Technologies	Process Perfector

Table 1.1: NMPC companies and product names.

eral practice of linear MPC, the majority of dynamic models are derived from plant testing and system identification. For NMPC, however, the issue of plant testing and system identification becomes much more complicated.

A class of state-space models is adopted in the Aspen Target product, which has a linear dynamic state equation and a nonlinear output relation (see [73]). More specifically, the output nonlinearity is modelled with a linear relation superimposed with a nonlinear neural network. A difficult issue in nonlinear modelling is not the selection of a nonlinear relation, but rather the selection of a robust and reliable identification algorithm. The identification algorithm discussed in [102] builds one model for each output separately. Besides the identification of the state-space model, a *model confidence index* (MCI) is also calculated on-line. If the MCI indicates that the neural network prediction is unreliable, the neural net nonlinear map is gradually turned off and the model calculation relies on the linear model only. Another feature of this modelling algorithm is the use of extended Kalman filters (EKF) to correct for model-plant mismatch and unmeasured disturbances [102]. The EKF provides a bias and gain correction to the model on-line. This function replaces the constant



output error feedback scheme typically employed in MPC practice.

### ***Input-output models***

The MVC algorithm and the Process Perfecter use input-output models. To simplify the system identification task, both products use a static nonlinear model superimposed upon a linear dynamic model. Martin, et al. [52] describe the details of the Process Perfecter modelling approach. Their presentation is in single-input-single-output form, but the concept is applicable to multi-input-multi-output models. It is assumed that the process input and output can be decomposed into a steady-state portion which obeys a nonlinear static model and a deviation portion that follows a dynamic model. The identification of the linear dynamic model is based on plant test data from pulse tests, while the nonlinear static model is a neural network built from historical data. It is believed that the historical data contain rich steady-state information and plant test is needed only for the dynamic sub-model.

### ***First principles models***

Since empirical modelling approaches can be unreliable and require tremendous amount of experimental data, some products provide the option to use first principles models. These products usually ask the user to provide the first principles models with some kind of open equation editor, then the control algorithms can use the user-supplied models to calculate future control moves. The NOVA-NLC falls in this category. In both cases, model parameters must be estimated from plant data.

### ***Output feedback***

In the face of unmeasured disturbances and model errors, some form of feedback is required to remove steady-state offset. The most common method for incorporating feedback into MPC algorithms involves comparing the measured and predicted process outputs. The difference between the two is added to future output predictions to bias them in the direction of the measured output. This can be interpreted as assuming that an unmeasured step disturbance enters at the process output and remains constant for all future time. For the case of a linear model and no active constraints, Rawlings, et al. [77] have shown that this form of feedback leads to offset-free control. Many industrial NMPC algorithms provide the constant output feedback option.

When the process has a pure integrator, the constant output disturbance assumption will no longer lead to offset-free control. For this case it is common to assume that an integrating disturbance with a constraint ramp rate has entered at the output. The PFC, Aspen Target, and Process Perfecter algorithms provide this feedback option.

It is well known from linear control theory that additional knowledge about unmeasured disturbances can be exploited to provide better feedback by designing a Kalman Filter [38]. Muske and Rawlings demonstrate how this can be accomplished in the context of MPC [66]. It is interesting to note (see [73]) that a number of industrial NMPC algorithms provide options for output feedback based on a nonlinear generalization of the Kalman Filter known as the Extended Kalman Filter (EKF) [29]. Aspen Target provides an EKF to estimate both a bias and a feedback gain. NOVA-NLC uses an EKF to develop complete state and noise estimates.

### ***Steady-state and dynamic optimization***

The PFC, Aspen Target, MVC, and Process Perfecter controllers split the control calculation into a local steady-state optimization followed by a dynamic optimization. Optimal steady-state targets are computed for each input and output; these are then passed to a dynamic optimization to compute the optimal input sequence required to move toward these targets. These calculations involve optimizing a quadratic objective that includes input and output contributions. The exception is the NOVA-NLC controller that performs the dynamic and steady-state optimizations simultaneously. At the dynamic optimization level, an MPC controller must compute a set of MV adjustments that will drive the process to the steady-state operating point without violating constraints. Almost all the products (see [73]) mentioned above use the same type of dynamic objective and constant weight matrices in the objective function.

### ***Constraint formulation***

There are basically two types of constraints used in industrial MPC technology: hard and soft. Hard constraints are those which should never be violated. Soft constraints allow the possibility of a violation; the magnitude of the violation is generally subjected to a quadratic penalty in the objective function. All of the NMPC algorithms described here allow hard input maximum, minimum, and rate of change constraints to be defined. These are generally defined so as to keep the lower level MV controllers in a controllable range, and to prevent violent movement of the MV's at any single control execution. The PFC algorithm also accommodates maximum and minimum input acceleration constraints which are useful in mechanical servo control applications.

The Aspen Target, MVC, NOVA-NLC, and Process Perfecter algorithms perform rigorous optimizations subject to the hard input constraints. The PFC algorithm, however, enforces input hard constraints only after performing an unconstrained optimization. This is accomplished by clipping input values that exceed the input constraints. All control products enforce output constraints as part of the dynamic optimization. The Aspen Target, NOVA-NLC, and Process Perfecter products allow options for both hard and soft output

constraints. The PFC product allows only hard output constraints, while the MVC product allows only soft output constraints. The exclusive use of hard output constraints is generally avoided in MPC technology because a disturbance can cause such a controller to lose feasibility.

The Process Perfecter product applies soft constraints by using a *frustum method* that permits a larger control error in the beginning of the horizon than in the end, but no error is allowed outside the frustum. At the end of the horizon the frustum can have a non-zero zone, instead of merging to a single line, which is determined based on the accuracy of the process model to allow for model errors.

### ***Output trajectories***

Industrial MPC controllers use four basic options to specify future CV behavior: a setpoint, zone, reference trajectory or funnel [73]. All of the NMPC controllers described here provide the option to drive the CV's to a fixed setpoint, with deviations on both sides penalized in the objective function. In practice this type of specification is very aggressive and may lead to very large input adjustments, unless the controller is detuned in some fashion. This is particularly important when the model differs significantly from the true process. For this reason all of the controllers provide some way to detune the controller using either move suppression, a reference trajectory, or time-dependent weights.

All of the controllers also provide a CV zone control option, designed to keep the CV within a zone defined by upper and lower boundaries. A simple way to implement zone control is to define soft output constraints at the upper and lower boundaries. The PFC, Aspen Target, MVC, and NOVA-NLC algorithms provide a CV reference trajectory option, in which the CV is required to follow a smooth path from its current value to the setpoint. Typically a first order path is defined using an operator entered closed loop time constant. In the limit of a zero time constant the reference trajectory reverts back to a pure setpoint; for this case, however, the controller would be sensitive to model mismatch unless some other strategy such as move suppression is also being used. In general, as the reference trajectory time constant increases, the controller is able to tolerate larger model mismatch.

### ***Output horizon and input parameterization***

Industrial MPC controllers generally evaluate future CV behavior over a finite set of future time intervals called the *prediction horizon*. This finite output horizon formulation is used by all of the industrial algorithms. The length of the horizon is a basic tuning parameter for these controllers, and is generally set long enough to capture the steady-state effects of all computed future MV moves. This is an approximation of the infinite horizon solution for closed loop

stability, and may explain why none of the industrial NMPC algorithms include a terminal state constraint.

The PFC and Aspen Target controllers allow the option to simplify the calculation by considering only a subset of future points called *coincidence points*, so named because the desired and predicted future outputs are required to coincide at these points. A separate set of coincidence points can be defined for each output, which is useful when one output responds quickly relative to another.

Industrial MPC controllers use three different methods to parameterize the MV profile: a single move, multiple moves, and basis functions [73]. The MVC product computes a single future input value; the PFC controller also provides this option. The Aspen Target, NOVA-NLC, and Process Perfector controllers can compute a sequence of future moves spread over a finite control horizon. The length of the control horizon is another basic tuning parameter for these controllers. Better control performance is obtained as the control horizon increases, at the expense of additional computation.

The PFC controller parameterizes the input function using a set of polynomial basis functions. This allows a relatively complex input profile to be specified over a large (potentially infinite) control horizon, using a small number of unknown parameters. This may provide an advantage when controlling nonlinear systems. Choosing the family of basis functions establishes many of the features of the computed input profile; this is one way to ensure a smooth input signal, for example. If a polynomial basis is chosen then the order can be selected so as to follow a polynomial setpoint signal with no lag. This feature is important for mechanical servo control applications.

### ***Solution methods***

The PFC controller performs an unconstrained optimization using a nonlinear least-squares algorithm. The solution can be computed very rapidly, allowing the controller to be used for short sample time applications. Some performance loss may be expected, however, since input constraints are enforced by clipping.

The Aspen Target product uses a multi-step Newton-type algorithm developed by Oliveira and Biegler [70, 71], and makes use of analytical model derivatives. Due to the sparseness of the state space model in Aspen Target, the derivative computation is straightforward. The Newton algorithm makes use of the QPKWIK solver which has the advantage that intermediate solutions, although not optimal, are guaranteed feasible. This permits early termination of the optimization algorithm if the optimum is not found within the sampling time. Aspen Target uses the same QPKWIK engine for local steady-state optimization and the dynamic MV calculation.

The MVC and Process Perfector products use a generalized reduced gradient (GRG) code called GRG2 developed by Lasdon and Warren [44]. The

NOVA-NLC product uses the NOVA optimization package, a proprietary mixed complementarity nonlinear programming code developed by DOT Products.

### 1.2.5 *Future needs for NMPC technology development*

Although many academic research results are available for NMPC, many issues (see [73]) that affect industrial practice are as yet unresolved:

- *Modelling approaches.* There is no systematic approach for building nonlinear dynamic models for NMPC. The first difficult issue is how to perform plant tests. To capture any nonlinearity in the process, extensive testing using a multilevel design is desired. This will make the testing period much longer than that of a linear plant test. In the case of empirical approaches, guidelines for plant tests are needed to build a reliable model.
- *Control and optimization.* Because of the use of a nonlinear model, the NMPC calculation usually involves a non-convex nonlinear program, for which the numerical solution is very challenging. Speed and the assurance of a reliable solution in real-time are major limiting factors in existing applications.
- *Output feedback.* Most current NMPC implementations use the traditional bias correction to the model prediction based on current measurements. While this approach is meaningful for linear MPC because of the principle of superposition, it is questionable how general this approach is to nonlinear processes. Nonlinear state estimation may provide an optimal approach to this issue.
- *Justification of NMPC.* Because of the difficulties involved in NMPC implementations, the added benefit of applying NMPC has to be justified. To deal with this problem most products provide linear MPC as a backup. In the case that NMPC is not needed or difficult to implement, linear MPC is implemented instead. Criteria on where NMPC is needed are desirable but difficult to obtain. Benchmarks on the justification of NMPC are required on an array of industrial processes. Unfortunately, only one such activity has been reported so far [27].
- *Other issues.* Other issues that are applicable to linear MPC technology [74] should also be of the same level of concern for NMPC, if not more. These issues include multiple prioritized objective functions, determining controllable sub-processes, tuning, ill-conditioning, and fault tolerance.

### 1.3 Industrial technology

The main reasons for increasing acceptance of MPC technology by the process industry since 1985 are clear:

- MPC is a model based controller design procedure, which can handle processes with large time-delays, non-minimum phase, unstable and non-linear processes.
- It is an easy-to-tune method, in principle there are several basic parameters to be tuned.
- Industrial processes have their limitations in valve capacity, technological requirements and are supposed to deliver output products with some pre-specified quality specifications. MPC can handle these constraints in a systematic way during the design and implementation of the controller.
- Finally MPC can handle structural changes, such as sensor and actuator failures, changes in system parameters and system structure by adapting the control strategy on a sample-by-sample basis.

There is a number of names denoting particular variants of predictive control, usually with corresponding acronyms. Examples of these are:

- Dynamic Matrix Control (DMC),
- Extended Prediction Self-Adaptive Control (EPSAC),
- Generalized Predictive Control (GPC),
- Model Algorithmic Control (MAC),
- Predictive Functional Control (PFC),
- Quadratic Dynamic Matrix Control (QDMC),
- Sequential Open Loop Optimization (SOLO),

and so on. Generic names which have become widely used to denote the whole area of predictive control are Model Predictive Control (MPC) and Model-Based Predictive Control (MBPC).

#### 1.3.1 Commercial predictive control schemes

Although there are companies that make use of technology developed in-house, that is not offered externally, the ones listed below (some already mentioned before) can be considered representative of the current state of the art of Model Predictive Control technology. Their product names and acronyms are:

- Aspen Technology: Dynamic Matrix Control (DMC-Plus), Setpoint Multivariable Control Architecture (SMCA).
- Adersa: Identification and Command (IDCOM), Hierarchical Constraint Control (HIECON) and Predictive Functional Control (PFC).
- Honeywell Profimatics: Profit Control, Robust Model Predictive Control Technology (RMPCT) and Predictive Control Technology (PCT).
- Pavilion Technologies: Process Perfector.
- SCAP Europa: Adaptive Predictive Control System (APCS).
- IPCOS: IPCOS Novel Control Architecture (INCA).

Notice that each product is not the algorithm alone, but it is accompanied by additional packages, usually identification or plant test packages. Qin and Badgwell [74] present the results obtained from an industrial survey in 1997. The total number of applications reported in the paper is over 2200 and is quickly increasing. The majority of applications (67%) are in the area of refining, one of the original application fields of MPC, where it has a solid background. An important number of applications can be found in petrochemicals and chemicals. Significant growth areas include pulp and paper, food processing, aerospace and automotive industries. Other areas such as gas, utility, furnaces or mining and metallurgy also appear in the report. The DMC corporation reports the largest total number of applications (26%), while the other four vendors share the remaining applications almost equally.

The past three years have shown rapid progress in the development and application of industrial NMPC technology. While MPC applications are concentrated in refining [73], NMPC applications cover a much broader range of application areas. Areas with the largest number of NMPC applications include chemicals, polymers, air and gas processing. Although not shown in the table, it has been observed that the size and scope of NMPC applications are typically much smaller than that of linear MPC applications [53]. This is likely due to the computational complexity of NMPC algorithms.

## 1.4 *Receding horizon strategy*

Model predictive control is a control strategy developed around certain common key principles:

- Explicit on-line use of a process model to forecast the process output at future time instants.
- Calculation of an optimal control action based on the minimization of one or more cost functions, possibly including constraints on the process variables.

- Receding horizon implementation.

The various MPC algorithms differ mainly in the type of model used to represent the process and its disturbances, as well as the cost functions to be minimized, with or without constraints. Referring to Figure 1.2, the MPC principle is characterized by the following strategy:

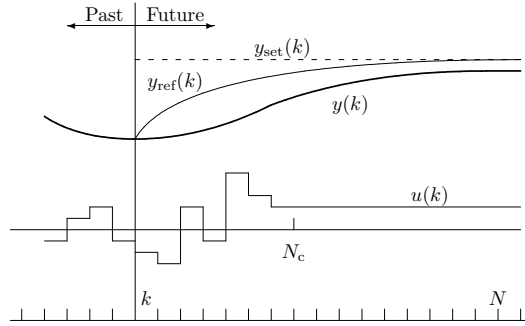


Figure 1.2: Receding horizon principle within MPC.

- at each current moment  $k$ , the process output  $y(k+j)$  is predicted over a finite time horizon  $j = 1, \dots, N$ . The predicted output values at time  $k$  are indicated by  $y(k+j|k)$  and the value  $N$  is called the *prediction horizon*. The prediction is done by means of a model of the process; it is assumed that this model is available. The forecast depends on the past inputs and outputs, but also on the future control scenario  $\{u(k+j|k), j = 0 \dots N_c - 1\}$  (i.e. the control actions that we intend to apply from the present moment  $k$  on);
- a *reference trajectory*  $\{y_{\text{ref}}(k+j|k), j = 1 \dots N\}$ , starting at  $y_{\text{ref}}(k|k) = y(k)$ , is defined over the prediction horizon, describing how we want to guide the process output so as to minimize the tracking error  $e(k+j|k) = y_{\text{ref}}(k+j|k) - y(k+j|k)$ ;
- an output measurement  $\hat{y}(k)$  available for feedback and state estimation;
- the *control sequence*  $\{u(k+j|k), j = 0 \dots N_c - 1\}$  is calculated on the basis of a measurement in order to minimize a specified cost function, depending on the predicted output errors  $\{y_{\text{ref}}(k+j|k) - y(k+j|k), j = 1 \dots N\}$ . Also, in most methods there is some structuring of the future control law  $\{u(k+j|k), j = 0 \dots N_c - 1\}$  and there might also be constraints on the process variables;
- the first element  $u(k|k)$  of the optimal control sequence  $\{u(k+j|k), j = 0 \dots N_c - 1\}$  is actually applied to the real process and defines the control



action when ranging over all  $k \in \mathbb{Z}_+$ . All other elements of the calculated control vector can be forgotten, because at the next sampling instant all time-sequences are shifted, a new output measurement  $y(k+1)$  is obtained and the whole procedure is repeated. This leads to a new control input  $u(k+1|k+1)$ , which is generally different from the previously calculated  $u(k+1|k)$ . This principle is called the “receding horizon” strategy.

### 1.4.1 Open-loop vs. closed-loop MPC

Assume that at time  $k$ , we measure the state  $x(k)$ . Let  $J_k(x(k), u)$  denote the (nonnegative) cost that input  $u$  occurs at time  $k$  when starting on initial condition  $x(k)$ . We may define two different general MPC problems:

#### Open-loop MPC

Given  $x(k)$ , find  $u_{\text{opt}} : T_N \rightarrow \mathcal{U}$  ( $T_N = [k, \dots, k+N]$ ) such that the performance objective function  $J_k(x(k), u)$  is minimized over all  $u : T_N \rightarrow \mathcal{U}$  that satisfies constraints and model equations.

#### Static state feedback

Given  $x(k)$ , find a sequence  $\pi_{\text{opt}} := (\pi_k)_{k=0}^N$  of state mappings  $\pi_k : \mathcal{X} \rightarrow \mathcal{U}$  such that the input

$$u(k+j) := \pi_{k+j}(x(k+j|k))$$

minimizes  $J_k(x(k), \pi)$  over all feedback strategies.

From receding horizon perspective the open-loop strategy produces an optimal control input  $u(k) = u_{\text{opt}}(k)$ ,  $k \in \mathbb{Z}_+$ , whereas the feedback strategy gives an optimal state law  $u(k) = \pi_k^{\text{opt}}(x(k))$ ,  $k \in \mathbb{Z}_+$ . Note that both depend on  $x(k)$  only, but the feedback strategies explicitly take the state on the prediction horizon into consideration to define the input. Feedback strategies have therefore the advantage that the effect of disturbances can be taken into account. However, the computation and synthesis of feedback strategies is much more difficult and involved than the synthesis of open-loop strategies. In this thesis we will mainly focus on open-loop strategies.

## 1.5 Integration of process control and plantwide optimization

The aim of the INCOOP project is to delivering high-performance process controllers and process optimization technology that enable given production processes to respond fast and reliably to market demand within permitted operating constraints of governed processes. The project had to set the first

important steps in realizing a fully integrated, dynamic and nonlinear process control and optimization system.

### 1.5.1 Research motivation: role of MPC

The research in the INCOOP project has been motivated by the need to improve MPC techniques, which is probably the most widely used multivariable control design in industry. Commercially available model predictive controllers vary in many details, but they are all based on finite time horizon optimization problems, based on one linear model at a time. The characteristic feature of model predictive controllers is that the control strategy is determined by the optimization of a performance function on a finite time interval. This interval stretches from the current time to a time instant, which is a fixed time slot ahead. The optimal control is calculated and implemented only until new measurements become available. Based on the new measurements, an update of the control strategy is determined by repeating the optimization of the performance function at the next time step. In this way, the control strategy depends on the measurements and could therefore be called of feedback type. The current generation of industrial MPC covers only a restricted range of the overall operating envelope of the process, due to the linear model used. For the INCOOP project the whole relevant operating range of the process is to be covered by the MPC. Hence a new MPC is needed, that fulfills this requirement.

The project architecture consists of several modules, such as dynamic real-time optimizer, state estimator and the MPC controller (see Figure 1.3). The

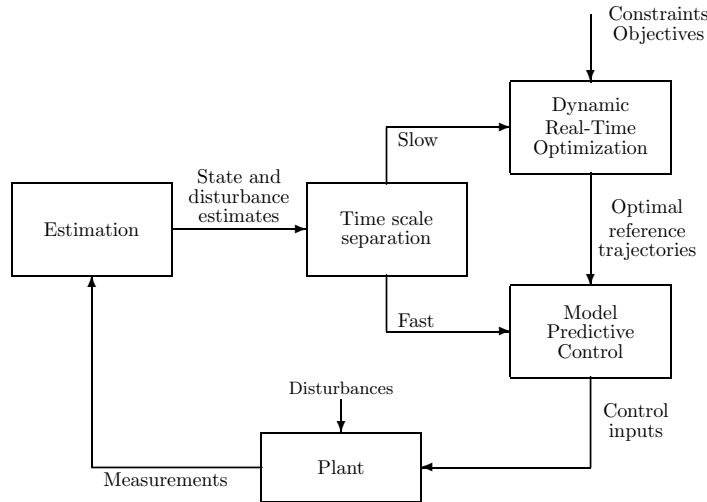


Figure 1.3: Role of MPC in the INCOOP project.

purpose of the MPC module is to achieve on-line accurate tracking of the trajectory delivered by the dynamic real-time optimizer.

The model predictive control component (MPC) solves a constrained optimization problem on-line and determines an optimal control input over a fixed future time horizon, based on the predicted future behavior of the process. Although more than one control move is generally calculated, only the first one is implemented. At the next sampling time, the optimization problem is reformulated and solved with new measurements obtained from the system. The optimal reference trajectories for the manipulated and the controlled variables are produced by the dynamic real-time trajectory optimizer and passed to the model predictive control module. The status of all process variables,

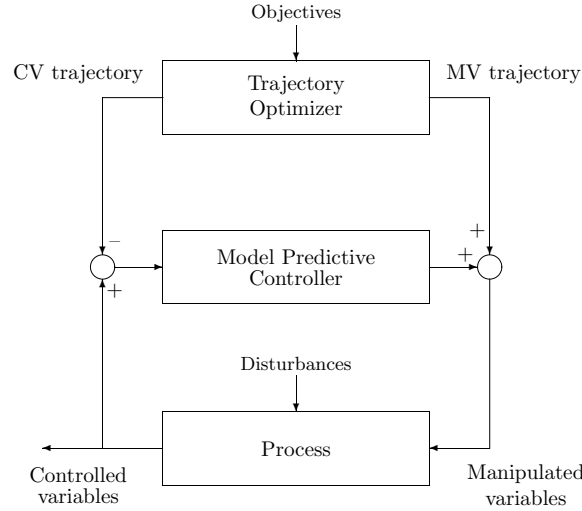


Figure 1.4: MPC delta mode.

including control variables, updates of the model parameters and estimates of disturbance signals are assumed to be available on-line and input to the MPC module. The current status of the real-time optimization is also transferred to the MPC block to ensure feasibility and proper functionality of all system components. Given the initial status of the process, estimates of disturbances and the reference trajectories, the optimizer in the MPC module produces the manipulated variables, such that input and output trajectories follow the reference trajectories as close as possible subject to the constraints imposed in the optimization. The time horizon for which the MPC block operates will be smaller than the time horizon of the trajectory optimizer module. This separation of time scales is motivated by the intuitive idea that MPC compensates for fast changes in the process behavior that are not taken into account in the trajectory optimizer module. The MPC module will be operating in a so-called

delta mode. This structure is shown in Figure 1.4. This means that the differences between actual trajectories and reference trajectories are treated, in a well defined sense within the algorithm.

Nonlinear high-performance MPC results in wide bandwidth control by enabling large prediction/control horizon. This technology makes it possible to control processes with large range of process dynamics and close tracking of optimum trajectories (e.g. transitions, recovery of process upsets). The optimization techniques developed in high-performance nonlinear MPC enables control of poorly conditioned processes (stiff systems).

The model predictive controller is one of the essential components of the control hierarchy in the project. One of the goals of the project is to develop a nonlinear multivariable predictive control system (MPC) that supports large bandwidth, high-performance quality control, accurate transition trajectory tracking control, dynamic constraint handling and performance based constraint pushing over a large operating envelope.

The aims of this thesis can be outlined as:

- investigate possibilities for development of efficient MPC technology for a broad class of systems, relevant for application in process industry.
- develop efficient numerical tools for the implementation of model predictive controllers, so as to allow large range of nonlinear system dynamics, flexible constraint handling and reduction of computational complexity.

## 1.6 Objectives and main results of the thesis

### 1.6.1 Research objectives

As it was already mentioned in Section 1.2 an important limitation of nonlinear MPC is the high computational load. The type of optimization problem that has to be solved online is dependent on the cost function and the applied nonlinear model. Due to the scale of industrial processes and high-performance requirements, the optimization problems that have to be solved online are large. The solution must be obtained in a limited amount of time because it is implemented in a receding horizon fashion.

The high computational complexity has restricted the use of this technology to relatively slow systems encountered in the chemical and petrochemical industry. There are several reasons why it is desirable to decrease the sampling time that is achievable with this technology. In process industry the motivation is obvious for those systems which mostly exhibit fast dynamics. Higher sampling rates are also necessary for fast systems such as mechanical systems. For these applications the current generation model predictive control is not feasible due to the high computational load, while efficient constraint handling can have important benefits in these industrial sectors.

For unconstrained systems, both linear and nonlinear, singular perturbation theory provides a firm theoretical basis for separating time scales. For constrained systems this theory is not applicable. Constraints are not related to a certain time scale. The constraints are usually incorporated in a controller running at a low sampling frequency due to purely practical reasons. The reason is that with the current state of technology it is not yet possible to run model predictive control at a high sampling frequency due to the large complexity of the online computation. If MPC is fast enough to run at the higher sampling rate, this situation can be avoided.

The specific properties of the processes encountered in industry make it difficult to obtain an accurate model of the behavior of the system. First of all, the systems are usually large. The large scale appears not only from the high number of inputs and outputs but also from the complex dynamics that are modelled. These circumstances give rise to solve large optimization problems. Secondly, the systems are usually sensitive for directional changes in the inputs (for example, distillation columns). In system theoretical terms this is denoted as a bad conditioning in space. Finally, the processes exhibit dynamical phenomena that take place at totally different time-scales: very fast phenomena and slow phenomena have to be accounted for. This is denoted with a stiff system or a system with bad conditioning in time. Numerical problems are likely to occur during simulation of such systems.

It is clear that a new high-performance MPC technology and more computationally efficient algorithms are required for large scale industrial systems. The research objectives in this thesis are outlined as follows:

- develop a computationally efficient MPC technology for a broad class of nonlinear systems.
- extend the MPC algorithm with capabilities for control of different system dynamics and for flexible constraint handling.
- improve computational efficiency in optimization to allow online MPC application.

### 1.6.2 Main results

The MPC module solves on-line a constrained optimization problem and determines an optimal control input over a fixed future time-horizon, based on the predicted future behavior of the process and on the desired reference trajectory. The MPC module respects the dynamic constraints of the process. A number of MPC schemes have been investigated, implemented and thoroughly tested.

Some specific features of the developed MPC controller are as follows:

- The predicted future process behavior has been represented as the sum of a non-linear prediction component and a component based on linear

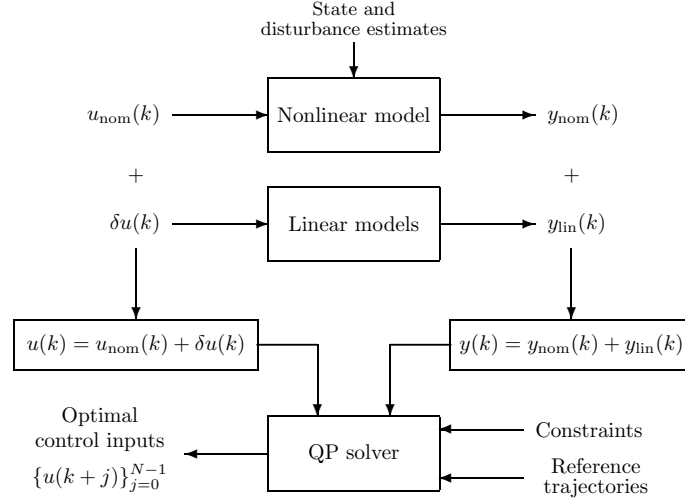


Figure 1.5: Proposed MPC concept.

time-varying models defined along the reference trajectory, which need to be tracked. The first component constitutes a future output prediction using non-linear simulation models, given past process inputs and measured disturbance history. The second component uses linearized models for prediction of future process outputs as required for calculation of optimum future process input manipulations (see Figure 1.5). We assume for that “small” inputs  $\delta u$  the superposition principle holds so that the output  $y = y_{\text{nom}} + y_{\text{lin}}$ .

- The end-point state of the transition process is controlled inside a region around a setpoint by means of a quadratic weighting of the final state.
- The constrained optimization problem leads to a quadratic programming problem, which is a convex optimization problem. A new routine has been developed for the efficient calculation of solutions of this problem. The computational cost of this optimization approach varies linearly with the number of optimization variables. This significantly improves some standard QP solvers for which the computational cost is cubic in the number of optimization variables (horizon  $N$ ).
- The state estimator produces at each sampling instant an optimal estimate of the initial state which is used as initialization of the MPC optimization.
- The status of the real-time optimization is transferred to the MPC module to ensure feasibility and proper functionality of all system components.

Given the initial status of the process, estimates of disturbances and the reference trajectories, the optimizer in the MPC module produces the manipulated variable such that input and output trajectories follow the reference trajectories as close as possible subject to the constraints imposed in the optimization. The MPC module is operating in delta-mode, which means that the differences between actual trajectories and reference trajectories are treated, in a well-defined sense, within the algorithm.

The need for high frequent solutions of a quadratic program and the repeated linearizations of the nonlinear model determine the main computational load in the MPC module. This load increases with increasing prediction horizons. A structured interior point method has been employed and implemented as a highly efficient numerical procedure for solving large scale quadratic programs. Its main merit is a feasible computational load for long prediction horizons. As such it allows to control the process over substantially larger bandwidths than standard QP solvers such as active set methods (ASM) or commercial algorithms (e.g. Mosek [3]).

## ***Linear Model Predictive Controllers***

---

2.1	Basic properties of MPC	2.4	The finite horizon MPC
2.2	Process models and prediction		problem
2.3	Optimal control	2.5	Implementation and tuning
		2.6	Design case studies

---

### ***2.1 Basic properties of MPC***

The flexibility of a receding horizon implementation has been useful in addressing various implementation issues that traditionally have been problematic. There are certain advantages which have led to the popularity of MPC in industrial applications. On the other hand, there are several difficulties associated with MPC, both inherent and imposed by the involved optimization, which make its analysis very hard.

#### ***Advantages of MPC***

- Constraints handling
- Straightforward application to MIMO case
- Handling systems with time-delays
- Compensation for actuator-sensor failures

From a practical viewpoint, an attractive feature of MPC is its ability to naturally and explicitly handle both multivariable input and output constraints by direct incorporation into the on-line optimization. Specifically, the repeated optimization performed at each time step is required to satisfy the imposed constraints. The most important, and also the most easily handled, are hard constraints or saturation of the control input, which makes MPC very useful in a variety of applications.



### **Limitations of MPC**

- *Stability/Robustness*

Theoretical aspects associated with stability and performance properties of MPC have proven to be a complicated and difficult issue. Stability tends to be problematic because of the following facts

1. The presence of constraints in the optimization problem results in a nonlinear closed-loop system, even if the model and plant dynamics are linear.
2. There is no explicit functional description of the control algorithm, as is required for most stability analyses.

In general, there is no stability guarantee when using MPC, and only recently theoretical results have emerged, pushing forward this topic. Still, concerns such as robust performance analysis and robust synthesis are difficult and remain generally unsolved.

- *Optimization*

The purely local nature of the underlying Euler-Lagrange treatment of optimal control remains one of the major challenges facing application of MPC to nonlinear systems, because the resulting nonlinear optimization problems rarely have exploitable convexity properties. For these reasons, an essential issue, both theoretical and practical, is whether the optimization can be successfully employed in MPC. Since in most cases, the optimal solution, even if there are no constraints, is unknown and uncomputable, it is not easy to evaluate results obtained by using MPC.

- *Noise handling*

Most of the existing MPC techniques are open-loop strategies. As it was already mentioned in the previous chapter, the feedback strategies have an advantage that the effect of noise and disturbances can be efficiently taken into account. However, the design of such feedback strategies is more difficult than conventional open-loop MPC control.

- *Performance assessment*

Achievable performance is usually unknown and generally impossible to assess or evaluate before the actual computation of MPC controllers.

### **Linear vs. Nonlinear MPC**

While for linear plants the MPC problem is usually reduced to simple linear or quadratic programs, for which efficient software exists, application of the MPC concept to nonlinear systems leads, in general, to involved nonlinear programming (NLP) problems. In general, the optimization problem is nonconvex

and leads to many difficulties impacting on implementation of MPC. These difficulties are related to feasibility and optimality, computation and stability aspects.

The important distinction in NLP is not linear versus nonlinear, but rather convex versus nonconvex. If the resulting nonlinear optimization problem is convex (e.g. for the linear system, convex cost function and convex I/O constraints), there exist methods which ensure convergence to a global minimum, which is unique if the performance criterion is strictly convex.

On the other hand, if the system to be controlled is nonlinear, even if the cost function and constraint sets are convex, the control problem will be, in general, a nonconvex nonlinear optimization problem. Therefore, finding a global optimum can be a difficult and computationally very demanding task, if possible at all. In other words, non-convexity makes the solution of the NLP uncertain.

## 2.2 Process models and prediction

An important difference between Model Predictive control (MPC) and PID-kind design methods is the explicit use of a model. This aspect is both the advantage and the disadvantage of MPC. The advantage is that the behavior of our controller can be studied in detail, simulations can be made and possible failures in plant or controller can be well detected. The disadvantage is that a detailed study of the plant behavior has to be done before the actual MPC-design can be started. About 80% of the work that has to be done, is in modelling and identification of the plant [79]. However, in the final result this effort and the investment to obtain a good model nearly always pay back in a short time.

The models applied in MPC serve two purposes:

- Prediction of expected future process output behavior on the basis of inputs and known disturbances applied to the process in the past.
- Calculation of the next process input signal that minimizes the control objective function.

The models required for these tasks do not necessarily have to be the same. The model applied for prediction may differ from the model applied for calculation of the next control action. In practice though both models are almost always chosen to be the same. As the models play such an important role in model predictive control the models are discussed in this chapter. The models applied are so called Input-Output (IO) models. These models describe the input-output behavior of the process.

The models applied in the controller are chosen to be linear models therefore. Two types of IO models are applied:

- Direct Input-Output models (IO models) in which the input signal  $u$  is directly applied to the models.
- Increment Input-Output models (IIO models) in which the increments  $\Delta u$  of the input signal are applied to the models instead of the input directly.

For applications of linear MPC, we consider models of the form

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k) \\ y(k) &= Cx(k) + Du(k) \end{aligned}$$

It is assumed that the sample time has been normalized to 1. The model is assumed to be time-invariant because of technical convenience only (time-variant models can be considered without serious loss of generality). Here we describe linear models, nonlinear aspects will be considered in the next chapter. Throughout the thesis we use only discrete-time models. These assumptions have to be validated against the actual process behavior as part of the process modelling or process identification phase. In this chapter only discrete time models will be considered. The control algorithm will always be implemented on a digital computer. The design of a discrete time controller is obvious. Further a linear time-invariant continuous time model can always be transformed into a linear time-invariant discrete time model using a zero-order hold z-transformation.

## 2.3 Optimal control

A general formulation of the optimal control problem, which is a dynamic optimization problem, amounts to a minimization of a cost function subject to a set of differential and algebraic constraint and some additional equality and inequality constraints:

$$\begin{aligned} \min_{u(t)} \quad & J(x_0, u(t)), \quad t \in [t_0, t_f] \\ \text{subject to} \quad & \dot{x} = f(x, u), \quad x(t_0) = x_0 \\ & h(x(t), u(t)) = 0 \\ & g(x(t), u(t)) \leq 0 \end{aligned}$$

Here,  $J$  is a (non-negative) cost function,  $f$  is assumed to be sufficiently smooth so that  $\dot{x} = f(x, u)$  has a unique (absolutely continuous) solution on  $[t_0, t_f]$  for any input  $u$ . Among admissible controls there exist either open-loop or feedback strategies. We will focus on open-loop strategies throughout this thesis. To solve this problem basically two possible solution strategies are available:

1. Dynamic programming. This strategy boils down to finding a solution to the Hamilton Jacobi Bellman partial differential equation. Which is,

for problems with a large number of states and inputs, a very large computational problem due to the problem of dimensionality in Bellman's method. A consequence of this approach is that the optimal input is a function of the state. In this respect the Hamilton-Jacobi-Bellman approach represents a closed-loop optimal strategy.

2. Classical Variational Approach. This strategy amounts to finding necessary conditions for optimality using the Euler-Lagrange equations. A general formulation of the necessary conditions for optimality is provided by Pontryagin Minimum Principle.

The dynamic programming approach has provided some important results in the specific case of unconstrained linear systems with a quadratic cost function. Kalman [37] showed that the HJB partial differential equation has a solution for this case and provided a method to compute this optimum mathematically using the Riccati difference equation (RDE) for the finite horizon case and the algebraic Riccati equation (ARE) for the infinite horizon case. This is the solution to the well known linear quadratic regulator (LQR) problem. For the discrete time case the unconstrained LQR problem and its solution are given next.

### 2.3.1 Unconstrained linear case

We consider the problem of having an initial state  $x(k)$  at time  $k$ , and finding a control sequence  $\{u(k+j)\}_{j=0}^{N-1}$ , which will minimize the finite horizon cost function

$$J_N(x(k), u) = \sum_{j=0}^{N-1} \left[ \|x(k+j)\|_{Q(j)}^2 + \|u(k+j)\|_{R(j)}^2 \right] + \|x(k+N)\|_{P_N}^2 \quad (2.1)$$

subject to  $x(k+1) = Ax(k) + Bu(k)$ , where  $Q(j) \geq 0, R(j) \geq 0, P_N \geq 0$  and  $\|x\|_Q^2$  means  $x^\top Qx = \langle x, Qx \rangle$  with  $\langle \cdot, \cdot \rangle$  the standard inner product. The optimal control is found by iterating backwards the Riccati difference equation [65, 8]:

$$P(j) = A^\top P(j+1)A - A^\top P(j+1)B(B^\top P(j+1)B + R(j))^{-1} B^\top P(j+1)A + Q(j)$$

for  $j = 0, \dots, N-1$  with  $P(N) = P_N$  and forming the state feedback gain matrix

$$K(j) = (B^\top P(j+1)B + R(j))^{-1} B^\top P(j+1)A.$$

The optimal control sequence is then given by

$$u(k+j) = -K(j)x(k+j)$$

and the optimal value of the cost (2.1) obtained in this way is  $J_N^{\text{opt}}(x(k)) = \|x(k)\|_{P(0)}^2$ . In [8, 65] it was shown that stability can sometimes be guaranteed with finite horizons, even when there is no explicit terminal constraint. The closed-loop asymptotic stability of the finite horizon model predictive control scheme with a varying weight on the end-point state was investigated in [97].

For the infinite horizon case we have the following cost function

$$J_\infty(x(k), u) = \lim_{N \rightarrow \infty} J_N(x(k), u)$$

and the weighting matrices are constant:  $Q(j) = Q, R(j) = R$  for all  $j \geq 0$ . If  $R > 0$ , the pair  $(A, B)$  is stabilizable, and the pair  $(A, Q^{1/2})$  is detectable, then the limit  $P = \lim_{j \rightarrow \infty} P(j)$  exists, is non-negative and is the unique non-negative definite solution of the algebraic Riccati equation

$$P = A^\top P A - A^\top P B (B^\top P B + R)^{-1} B^\top P A + Q.$$

Moreover,  $K = \lim_{j \rightarrow \infty} K(j)$  exists and is given by

$$K = (B^\top P B + R)^{-1} B^\top P A$$

and the optimal control sequence becomes the constant state feedback law

$$u(k+j) = -Kx(k+j).$$

It can be proved that this feedback law is stabilizing so that all the eigenvalues of the closed-loop state transition matrix  $A - BK$  lie strictly within the unit circle (if the stated conditions hold). The optimal cost is now  $J_\infty^{\text{opt}}(x(k)) = \|x(k)\|_P^2$ .

## 2.4 The finite horizon MPC problem

Continuous time optimal control needs the solution of a function optimization problem. This is an infinite dimensional optimization problem which is in many cases intractable. The main idea of model predictive control is to restrict the set of possible inputs such that only a finite dimensional optimization problem has to be solved. This is done by using a discrete time model and a finite horizon. Instead of calculating a closed form state or output feedback as was possible for the LQR problem, the receding horizon principle is used. This implies that the optimal input trajectory is calculated at each time instant on the basis of the last measurement of the output. In this way the optimal control solution can be utilized in a feedback strategy also if no closed form expression exists for the optimal feedback controller.

Model predictive control is applied in many settings in which e.g. the system description and the cost function differ. The description of model predictive control in this section is restricted to a linear model subject to linear input and output constraints and a quadratic cost function. In Figure 1.2 the general principle of model predictive control is graphically depicted.

### 2.4.1 Model

Suppose a linear, discrete-time, state-space model of the plant is given in the form

$$x(k+1) = Ax(k) + Bu(k) \quad (2.2)$$

$$y(k) = C_y x(k) \quad (2.3)$$

$$z(k) = C_z x(k) \quad (2.4)$$

where  $x$  is an  $n_x$ -dimensional state vector,  $u$  is an  $n_u$ -dimensional input vector,  $y$  is an  $n_y$ -dimensional vector of measured outputs and  $z$  is an  $n_z$ -dimensional vector of outputs which are to be controlled, either to particular set-points, or to satisfy some constraints, or both. The components in  $y$  and  $z$  may overlap, and may be the same – that is, all the controlled outputs could as well be measured. We will assume that  $y = z$ , and we will then use  $C$  to denote both  $C_y$  and  $C_z$ .

As usual, our plant model (2.2) expresses the plant state  $x$  in terms of the values of the input  $u$ . But the cost function will penalize rates of change of the input,  $(\Delta u)(k) = u(k) - u(k-1)$ , rather than the input values themselves. We shall see in the next section that the predictive control algorithm will in fact produce the rate of change  $\Delta u$  rather than  $u$ . It is therefore convenient for many purposes to regard the controller as producing the signal  $\Delta u$ , and the plant as having this signal as its input. That is, it is often convenient to regard the discrete-time integration from  $\Delta u$  to  $u$  as being included in the plant dynamics to ultimately obtain an offset-free tracking controller. The MPC controller produces the signal  $\Delta u$ , which is passed to the plant. There are several ways of including this “integration” in a state-space model. All of them involve augmenting the state vector. For example, one way is to define the state vector

$$\xi(k) = \begin{bmatrix} x(k) \\ u(k-1) \end{bmatrix}.$$

Then, assuming the linear model (2.2) holds for the real plant state, we have

$$\begin{bmatrix} x(k+1) \\ u(k) \end{bmatrix} = \begin{bmatrix} A & B \\ 0 & I \end{bmatrix} \begin{bmatrix} x(k) \\ u(k-1) \end{bmatrix} + \begin{bmatrix} B \\ I \end{bmatrix} \Delta u(k) \quad (2.5)$$

$$y(k) = \begin{bmatrix} C & 0 \end{bmatrix} \begin{bmatrix} x(k) \\ u(k-1) \end{bmatrix}. \quad (2.6)$$

The reason for using one of these standard forms is mainly that it connects well with the standard theory of linear systems and control. We may also generalize this model by including the effects of measured or unmeasured disturbances, and of measurement noise. We are going to assume that the sequence of actions at time step  $k$  is the following:

1. Obtain measurements  $y(k)$ .
2. Compute the required plant input sequence  $\{u(k+j|k)\}_{j=0}^N$ .
3. Apply  $u(k)$  to the plant.

This implies that there is always some delay between measuring  $y(k)$  and applying  $u(k)$ . For this reason there is no direct feed-through from  $u(k)$  to  $y(k)$  in the measured output equation (2.3), so that the model is strictly proper.

### 2.4.2 Problem formulation

For the basic formulation of predictive control we shall assume that the plant model is linear, that the cost function is quadratic, and that constraints are in the form of linear inequalities. Furthermore, we shall assume that the cost function does not penalize particular values of the input vector  $u(k)$ , but only changes of the input vector,  $\Delta u(k)$ , which are defined as before. This formulation coincides with that used in the majority of the predictive control literature.

To make the formulation useful in the real world, we shall not assume that the state variables can be measured, but that we can obtain an estimate  $\hat{x}(k|k)$  of the state  $x(k)$ , the notation indicating that this estimate is based on measurements up to time  $k$  – that is, on measurements of the outputs up to  $y(k)$ , and on knowledge of the inputs only up to  $u(k-1)$ , since the next input  $u(k)$  has not yet been determined. Signals  $u(k+j|k)$  will denote a future value (at time  $k+j$ ) of the input  $u$ , which is assumed at time  $k$ . Signals  $x(k+j|k)$  and  $y(k+j|k)$  will denote the predictions, made at time  $k$ , of the variables  $x$  and  $y$  at time  $k+j$ , on the assumption that some sequence of inputs  $u(k+i|k)$  ( $i = 0, 1, \dots, j-1$ ) has been applied. These predictions will be made consistently with the assumed linearized model (2.2)–(2.4). We will usually assume that the real plant is governed by the same equations as the model, although this is not really true in practice.

A cost function  $J$  penalizes deviations of the predicted controlled outputs  $y(k+j|k)$  from a (vector-valued) reference trajectory  $y_{\text{ref}}(k+j|k)$ . Again the notation indicates that this reference trajectory may depend on measurements made up to time  $k$ ; in particular, its initial point may be the output measurement  $y(k)$ . But it may also be a fixed set-point, or some other predetermined trajectory. We define the cost function to be

$$J_k(x(k), u) = \sum_{j=1}^N \|y(k+j|k) - y_{\text{ref}}(k+j|k)\|_{Q(j)}^2 + \sum_{j=0}^{N_c-1} \|\Delta u(k+j|k)\|_{R(j)}^2 \quad (2.7)$$

subject to the element-wise constraints

$$y_{\min} \leq y(k+j|k) \leq y_{\max} \quad j = 0, \dots, N \quad (2.8)$$

$$u_{\min} \leq u(k+j|k) \leq u_{\max} \quad j = 0, \dots, N_c - 1 \quad (2.9)$$

$$\Delta u_{\min} \leq \Delta u(k+j|k) \leq \Delta u_{\max} \quad j = 0, \dots, N_c - 1. \quad (2.10)$$

Although (2.8)–(2.10) are amplitude constraints, we may have more general polytopic type constraints, which can be easily incorporated in the MPC theory of this chapter.

There are several points to note here. The prediction horizon has length  $N$ , but we do not necessarily have to start penalizing deviations of  $y$  from  $y_{\text{ref}}$  immediately, because there may be some delay between applying an input and seeing any effect.  $N_c$  is the control horizon. We will always assume that  $N_c < N$ , and that  $\Delta u(k+j|k) = 0$  for  $j > N_c$ , so that  $u(k+j|k) = u(k+N_c-1|k)$  for all  $j > N_c$ , i.e. a zero order hold is applied on the input for  $j > N_c$ . The form of the cost function (2.7) implies that the error vector  $y(k+j|k) - y_{\text{ref}}(k+j|k)$  is penalized at every point in the prediction horizon if  $Q(j) > 0$ . This is indeed the most common situation in predictive control. But it is possible to penalize the error at only a few coincidence points, by setting  $Q(j) = 0$  for some values of  $j$ . It is also possible to have different coincidence points for different components of the error vector by setting appropriate elements of the weighting matrices  $Q(j)$  to 0. To allow for these possibilities, we do not insist on  $Q(j) > 0$ , but allow the weaker condition  $Q(j) \geq 0$ . (At least this condition is required, to ensure that  $J_k \geq 0$ .)

We also need  $R(j) \geq 0$  to ensure that  $J_k > 0$ . Again, we do not insist on the stronger condition that  $R(j) > 0$ , because there are cases in which the changes in the control signal are not penalized. The weighting matrices  $R(j)$  are sometimes called *move suppression factors*, since increasing them penalizes changes in the input vector more heavily.

The cost function (2.7) only penalizes rates of the input vector, not its value. In some cases an additional term of the form  $\sum \|u(k+j|k) - u_0\|_S^2$  is added, which penalizes deviations of the input vector from some ideal setpoint value. Usually this is done only if there are more inputs than variables which are to be controlled to setpoints. We shall not include such a term in our basic formulation.

The prediction and control horizons  $N$  and  $N_c$ , the weights  $Q(j)$  and  $R(j)$ , and the reference trajectory  $y_{\text{ref}}(k+j)$ , all affect the behavior of the closed-loop combination of plant and predictive controller. Some of these parameters, particularly the weights, may be dictated by the economic objectives of the control system, but usually they are in effect tuning parameters which are adjusted to give satisfactory dynamic performance.



### 2.4.3 Prediction: full state measurement

We will start with the simplest situation. Assume that the whole state vector is measured, so that  $\hat{x}(k|k) = x(k) = y(k)$  (so  $C = I$ ). Also assume that we know nothing about any disturbances or measurement noise. Then all we can do is to predict by iterating the model (2.2)–(2.3). So we get

$$\begin{aligned}
 x(k+1|k) &= Ax(k) + Bu(k|k) \\
 x(k+2|k) &= Ax(k+1|k) + Bu(k+1|k) \\
 &= A^2x(k) + ABu(k|k) + Bu(k+1|k) \\
 &\vdots \\
 x(k+N|k) &= Ax(k+N-1|k) + Bu(k+N-1|k) \\
 &= A^Nx(k) + A^{N-1}Bu(k|k) + \dots + Bu(k+N-1|k)
 \end{aligned}$$

which can be summarized as

$$x(k+j|k) = A^jx(k) + \begin{bmatrix} A^{j-1} & A^{j-2} & \dots & I \end{bmatrix} B \begin{bmatrix} u(k|k) \\ \vdots \\ u(k+j-1|k) \end{bmatrix}$$

for  $j = 1, \dots, N$ . In the first line we have used  $u(k|k)$  rather than  $u(k)$ , because at the time when we need to compute the predictions we do not yet know what  $u(k)$  will be.

Now recall that we have assumed that the input may only change at times  $k, k+1, \dots, k+N_c-1$ , and will remain constant after that. So we have  $u(k+j|k) = u(k+N_c-1)$  for  $N_c < j < N-1$ . In fact, we will later want to have the predictions expressed in terms of  $\Delta u(k+j|k)$  rather than  $u(k+j|k)$ , so let us do that now. Recall that  $\Delta u(k+j|k) = u(k+j|k) - u(k+j-1|k)$ , and that at time  $k$  we already know  $u(k-1)$ . So we have

$$\begin{aligned}
 u(k|k) &= \Delta u(k|k) + u(k-1) \\
 u(k+1|k) &= \Delta u(k+1|k) + \Delta u(k|k) + u(k-1) \\
 &\vdots \\
 u(k+N_c-1|k) &= \Delta u(k+N_c-1|k) + \dots + \Delta u(k|k) + u(k-1)
 \end{aligned}$$

then for  $1 \leq j \leq N_c$

$$\begin{aligned}
 x(k+1|k) &= Ax(k) + B[\Delta u(k|k) + u(k-1)] \\
 x(k+2|k) &= A^2x(k) + AB[\Delta u(k|k) + u(k-1)] \\
 &\quad + B \underbrace{[\Delta u(k+1|k) + \Delta u(k|k) + u(k-1)]}_{u(k+1|k)} \\
 &= A^2x(k) + (A+I)B\Delta u(k|k) + B\Delta u(k+1|k) \\
 &\quad + (A+I)Bu(k-1)
 \end{aligned}$$

till the end of the control horizon

$$\begin{aligned} x(k + N_c | k) &= A^{N_c} x(k) + (A^{N_c-1} + \dots + A + I) B \Delta u(k | k) \\ &\quad \dots + B \Delta u(k + N_c - 1 | k) \\ &\quad + (A^{N_c-1} + \dots + A + I) B u(k - 1) \end{aligned}$$

That is, we get

$$\begin{aligned} x(k + j | k) &= A^j x(k) \\ &\quad + \begin{bmatrix} \sum_{i=0}^{j-1} A^i B & \dots & B \end{bmatrix} \begin{bmatrix} u(k | k) \\ \vdots \\ u(k + j - 1 | k) \end{bmatrix} \\ &\quad + \sum_{i=0}^{j-1} A^i B u(k - 1) \end{aligned}$$

for  $j \leq N_c$ . Next

$$\begin{aligned} x(k + N_c + 1 | k) &= A^{N_c+1} x(k) + (A^{N_c} + \dots + A + I) B \Delta u(k | k) \\ &\quad \dots + (A + I) B \Delta u(k + N_c - 1 | k) \\ &\quad + (A^{N_c} + \dots + A + I) B u(k - 1), \\ &\quad \vdots \\ x(k + N | k) &= A^N x(k) + (A^{N-1} + \dots + A + I) B \Delta u(k | k) \\ &\quad \dots + (A^{N-N_c} + \dots + A + I) B \Delta u(k + N_c - 1 | k) \\ &\quad + (A^{N-1} + \dots + A + I) B u(k - 1), \end{aligned}$$

which is summarized as

$$\begin{aligned} x(k + j | k) &= A^j x(k) \\ &\quad + \begin{bmatrix} \sum_{i=0}^{j-1} A^i B & \dots & \sum_{i=0}^{j-N_c} A^i B \end{bmatrix} \begin{bmatrix} u(k | k) \\ \vdots \\ u(k + N_c - 1 | k) \end{bmatrix} \\ &\quad + \sum_{i=0}^{j-1} A^i B u(k - 1) \end{aligned}$$

for  $N_c < j \leq N$ .

Notice the adjustments we had to make after introducing the control horizon to restrict the interval where input signal may change. We are now ready to combine all state predictions in one expression. Finally we can write this in

matrix-vector form:

$$\begin{bmatrix} x(k+1|k) \\ \vdots \\ x(k+N_c|k) \\ x(k+N_c+1|k) \\ \vdots \\ x(k+N|k) \end{bmatrix} = \underbrace{\begin{bmatrix} A \\ \vdots \\ A^{N_c} \\ A^{N_c+1} \\ \vdots \\ A^N \end{bmatrix}}_{\Phi} x(k) + \underbrace{\begin{bmatrix} B \\ \vdots \\ \sum_{i=0}^{N_c-1} A^i B \\ \sum_{i=0}^{N_c} A^i B \\ \vdots \\ \sum_{i=0}^{N-1} A^i B \end{bmatrix}}_{\Gamma} u(k-1) + \underbrace{\begin{bmatrix} B & \cdots & 0 \\ AB+B & \cdots & 0 \\ \vdots & \ddots & \vdots \\ \sum_{i=0}^{N_c-1} A^i B & \cdots & B \\ \sum_{i=0}^{N_c} A^i B & \cdots & AB+B \\ \vdots & \vdots & \vdots \\ \sum_{i=0}^{N-1} A^i B & \cdots & \sum_{i=0}^{N-N_c} A^i B \end{bmatrix}}_{G_y} \begin{bmatrix} \Delta u(k|k) \\ \vdots \\ \Delta u(k+N_c-1|k) \end{bmatrix}. \quad (2.11)$$

The prediction of  $y$  is now obtained as

$$y(k+j|k) = Cx(k+j|k) \quad (2.12)$$

for  $j = 1, \dots, N$ .

#### 2.4.4 Solving the MPC problem

We can rewrite the objective function (2.7) as

$$J_k = \|Y(k) - Y_{\text{ref}}(k)\|_{\mathcal{Q}}^2 + \|\Delta U(k)\|_{\mathcal{R}}^2, \quad (2.13)$$

where

$$\begin{aligned} Y(k) &= \begin{bmatrix} y(k+1|k) \\ \vdots \\ y(k+N|k) \end{bmatrix} & Y_{\text{ref}}(k) &= \begin{bmatrix} y_{\text{ref}}(k+1|k) \\ \vdots \\ y_{\text{ref}}(k+N|k) \end{bmatrix} \\ \Delta U(k) &= \begin{bmatrix} \Delta u(k|k) \\ \vdots \\ \Delta u(k+N_c-1|k) \end{bmatrix} \end{aligned}$$

and the weighting matrices  $\mathcal{Q}$  and  $\mathcal{R}$  are given by

$$\mathcal{Q} = \begin{bmatrix} Q(1) & 0 & \dots & 0 \\ 0 & Q(2) & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & Q(N) \end{bmatrix}$$

$$\mathcal{R} = \begin{bmatrix} R(0) & 0 & \dots & 0 \\ 0 & R(1) & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & R(N_c - 1) \end{bmatrix}$$

From (2.11)–(2.12) we see that  $Y(k)$  has form

$$Y(k) = \Phi x(k) + \Gamma u(k-1) + G_y \Delta U(k) \quad (2.14)$$

for suitable matrices  $\Phi, \Gamma$  and  $G_y$ . Define

$$E(k) = Y_{\text{ref}}(k) - \Phi x(k) - \Gamma u(k-1). \quad (2.15)$$

This vector can be thought of as a “tracking error”, in the sense that it is the difference between the future target trajectory and the “free response” of the system, namely the response that would occur over the prediction horizon if no input changes were made – that is, if we set  $\Delta U(k) = 0$ . If  $E(k)$  really were 0, then it would indeed be correct to set  $\Delta U(k) = 0$ . Now we can write

$$\begin{aligned} J_k &= \|G_y \Delta U(k) - E(k)\|_{\mathcal{Q}}^2 + \|\Delta U(k)\|_{\mathcal{R}}^2 \\ &= [\Delta U^\top(k) G_y^\top - E^\top(k)] \mathcal{Q} [G_y \Delta U(k) - E(k)] + \Delta U^\top(k) \mathcal{R} \Delta U(k) \\ &= \Delta U^\top(k) [G_y^\top \mathcal{Q} G_y + \mathcal{R}] \Delta U(k) - 2E^\top(k) \mathcal{Q} G_y \Delta U(k) + E^\top(k) \mathcal{Q} E(k) \end{aligned} \quad (2.16)$$

This has the form

$$J_k = \frac{1}{2} \Delta U^\top(k) H \Delta U(k) + f^\top \Delta U(k) + \text{const}, \quad (2.17)$$

where

$$H = 2(G_y^\top \mathcal{Q} G_y + \mathcal{R})$$

and

$$f = -2G_y^\top \mathcal{Q} E(k)$$

and neither  $H$  nor  $f$  depends on  $\Delta U(k)$ .

Recall that a simple relationship exists between the input increments  $\Delta u$  and control input  $u$ :

$$\begin{bmatrix} u(k|k) \\ u(k+1|k) \\ \vdots \\ u(k+N_c-1|k) \end{bmatrix} = M \begin{bmatrix} \Delta u(k|k) \\ \Delta u(k+1|k) \\ \vdots \\ \Delta u(k+N_c-1|k) \end{bmatrix} + F u(k-1),$$

where

$$M = \begin{bmatrix} I & 0 & \cdots & 0 \\ I & I & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ I & I & \cdots & I \end{bmatrix}, \quad F = \begin{bmatrix} I \\ I \\ \vdots \\ I \end{bmatrix}.$$

The constraints (2.8)–(2.10) can be rewritten as a single inequality

$$\begin{bmatrix} I \\ -I \\ M \\ -M \\ G_y \\ -G_y \end{bmatrix} \Delta U(k) \leq \begin{bmatrix} b_1 \\ -b_2 \\ d_1 - Fu(k-1) \\ -d_2 + Fu(k-1) \\ y_1 - \Phi x(k) - \Gamma u(k-1) \\ -y_2 + \Phi x(k) + \Gamma u(k-1) \end{bmatrix}, \quad (2.18)$$

where  $b_1$ ,  $b_2$ ,  $d_1$  and  $d_2$  are of dimension  $N_c n_u$  and consist of  $N_c$  copies of  $\Delta u_{\max}$ ,  $\Delta u_{\min}$ ,  $u_{\max}$  and  $u_{\min}$  respectively. In the same way, vectors  $y_1$  and  $y_2$  are of dimension  $N n_y$  and consist of  $N$  copies of  $y_{\max}$ ,  $y_{\min}$ .

So, from (2.17) we see that we have to solve the following constrained optimization problem

$$\min_{\Delta U(k)} \frac{1}{2} \Delta U^\top(k) H \Delta U(k) + f^\top \Delta U(k) \quad (2.19)$$

subject to the inequality constraint (2.18). This is a standard optimization problem known as the *Quadratic Programming* (QP) problem, and standard algorithms are available for its solution (see Appendix A).

Remember that we use only the part of this solution corresponding to the first step, in accordance with the receding horizon strategy. So if the number of plant inputs is  $n_u$  then we just use the first  $n_u$  rows of the vector  $\Delta U_{\text{opt}}(k)$ . We can represent this as

$$\Delta u_{\text{opt}}(k) = \begin{bmatrix} I_{n_u} & \underbrace{0 \cdots 0}_{(N_c-1) \text{ times}} \end{bmatrix} \Delta U_{\text{opt}}(k).$$

We assume that  $\mathcal{Q} \geq 0$  and  $\mathcal{R} > 0$ , which will be the case if  $R(j) > 0$  for each  $j$ . This ensures that the Hessian  $G_y^\top \mathcal{Q} G_y + \mathcal{R} > 0$ . In this case the QP problem which we have to solve is strictly convex. Because of the strict convexity a unique solution exists and we can guarantee termination of the optimization problem. Because of the additional structure of the QP problem, we can estimate how long it will take to solve. This is an extremely desirable property for an algorithm which has to be used on-line, so as to keep up with the real-time operation of the plant.

An other important issue is feasibility. We call a state  $x \in \mathbb{R}^{n_x}$  feasible if there exists  $u : T_N \rightarrow \mathbb{R}^{n_u}$  ( $T_N = [k, \dots, k+N]$ ) such that (2.7)–(2.10) hold

together with (2.2) that achieves finite cost  $J(x, u)$ . If  $x$  is not feasible then it is called infeasible. Suppose that for a feasible problem  $J(x(k), u)$  at time  $k$  there exists  $u_{\text{opt}}$  such that  $J(x(k), u_{\text{opt}})$  is minimal. Let  $x_{\text{opt}}$  denote the corresponding state trajectory. Then it can be shown that for  $x_{\text{opt}}(k+1) = Ax(k) + Bu_{\text{opt}}(k)$  the problem  $J(x_{\text{opt}}(k+1), u)$  at time  $k+1$  is also feasible if all eigenvalues of matrix  $A$  lie strictly inside the unit circle ( $|\lambda(A)| < 1$ ). This can be proved by noting that at time  $k+1$  the input signal

$$u(k+1+j|k) = \begin{cases} u_{\text{opt}}(k+1+j|k) & j = 0, \dots, N-2 \\ u_{\text{opt}}(k+N|k) & j = N-1 \end{cases}$$

satisfies the constraints (2.9), because  $u_{\text{opt}}$  satisfies the constraints and  $\|y(k+N+1)\| \leq \|y(k+N)\|$  (as the system is stable and strictly proper).

A major problem which can occur with constrained optimization is that the problem may be infeasible. Standard QP solvers just stop in such cases. This is obviously unacceptable as a substitute for a control signal which must be provided to the plant. So when implementing predictive control it is essential to take steps either to avoid infeasibility, or to have a “back-up” method of computing the control signal. Various approaches to this have been suggested, including:

- Avoid “hard” constraints on  $y$ .
- Actively manage the constraint definition at each  $k$ .
- Actively manage the horizons at each  $k$ .
- Use non-standard solution algorithms.

We will discuss some of these options in later chapters.

### 2.4.5 Stability issues

In the early eighties it became clear that model predictive control did not provide a stable closed loop system for all values of the tuning variables. This is a property that for example LQ optimal control did possess. It is a favorable property because no tuning is necessary to obtain stability and the tuning can be adjusted on-line without the danger of running unstable. Especially the academic community tried to gain insight in the stability problem and developed numerous approaches to obtain a closed loop system with guaranteed stability.

In [100] it is shown how a contraction constraint can be used to force the state to decrease with time and stability follows independent of the various parameters in the objective function.

Another approach is to use the value function  $J(x) = \inf_u J(x, u)$  as a Lyapunov function for the closed loop. One wishes to show that

$$J(x(k)) - J(x(k+1)) > 0 \quad (2.20)$$

along trajectories  $x$  of the closed loop system, because a Lyapunov function must be decreasing. There are several ways to assure that this is the case.

When a constraint for the final state  $x(k+N) = 0$  is added to, it can be shown [43] that the value function is a Lyapunov function for the closed loop. A property of this approach is that the additional state constraint may lead to infeasibility. This means that there is no input trajectory that satisfies all the constraints. Due to this infeasibility problem this approach may not be appropriate for industrial systems.

An other very elegant approach is given in [78]. There it is shown that the requirement (2.20) is fulfilled if the output horizon is set to infinity  $N = \infty$ . They show how this infinite dimensional optimization problem is solvable with finite dimensional quadratic programming.

#### 2.4.6 Mathematical programming: optimality conditions

In control a signal has to be searched for that is optimal in some sense. Clearly, optimization theory plays an important role in this field. In this section a brief review is given of some basic concepts of optimization theory that are used in this thesis.

In optimization theory two classes of problems can be distinguished:

1. Mathematical programming problems. In this class only algebraic relations occur.
2. Dynamic optimization problems. In this class apart from algebraic relations also differential equations occur.

The second class is usually considered in optimal control problems. The first class will be briefly discussed in the sequel of this section. In this thesis one convex optimization problem gets special attention, namely the quadratic programming problem.

A general formulation of a mathematical programming problem is the following

$$\begin{array}{ll} \min_{x \in \mathbb{R}^n} & f(x) \\ \text{subject to} & h(x) = 0 \\ & g(x) \leq 0 \end{array}$$

where the objective function  $f(x)$  is at least twice differentiable, the set of equality and inequality constraints  $h(x) = (h_1(x), \dots, h_m(x))$ ,  $g(x) = (g_1(x), \dots, g_p(x))$

are vector valued functions, where the functions  $h_i, g_i$  are assumed to be continuous and twice differentiable. The gradient of a function and set of functions is denoted as

$$\partial_x f(x) = \left[ \frac{\partial f}{\partial x_1}, \dots, \frac{\partial f}{\partial x_n} \right], \quad \partial_x h(x) = \begin{bmatrix} \frac{\partial h_1}{\partial x_1} & \dots & \frac{\partial h_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial h_m}{\partial x_1} & \dots & \frac{\partial h_m}{\partial x_n} \end{bmatrix},$$

where the gradient is defined as a row vector. Several strategies are available to solve this type of problem (see [67]). We will give some basic results of optimization theory that play a role in this thesis.

An important notion that is explained first is the notion of feasible point and feasible direction. A feasible point  $x$  is for unconstrained optimization simply a point in  $\mathbb{R}^n$  and for constrained optimization a point that satisfies  $h(x) = 0, g(x) \leq 0$ . All points that are feasible are denoted with the feasible region. A vector  $d \in \mathbb{R}^n$  is a feasible direction at  $x$  if there is an  $\bar{\alpha} > 0$  such that all points  $\{x + \alpha d \mid 0 \leq \alpha \leq \bar{\alpha}\}$  are feasible points. With respect to the optimization problem stated above, two types of optimal points can be distinguished: local and global minima.

**Definition 2.4.1** A point  $x^* \in \mathbb{R}^n$  is said to be a *local minimum* of  $f$  if there is an  $\varepsilon$  such that  $f(x) \geq f(x^*)$  for all feasible  $|x - x^*| < \varepsilon$  and a *global minimum* of  $f$  if  $f(x) \geq f(x^*)$  for all feasible  $x$ .

Hence, a global optimum is to be preferred over a local one as no better value of the cost function can be obtained. The first order necessary condition of unconstrained optimality is given next.

**Theorem 2.4.2 (First-order necessary condition [69, 67])** If  $x^*$  is a local minimum point of  $f$ , then for any  $d \in \mathbb{R}^n$  that is a feasible direction we have  $\partial_x f(x^*)d \geq 0$ .

This reduces to the well known requirement that the gradient is zero,  $\partial f(x^*) = 0$  if  $x^*$  is an interior point of the feasible region.

**Theorem 2.4.3 (Second-order necessary condition [69, 67])** If  $x^*$  is a local minimum point of  $f$ , then for any  $d \in \mathbb{R}^n$  that is a feasible direction we have (1)  $\partial_x f(x^*)d \geq 0$  and (2) if  $\partial_x f(x^*)d = 0$ , then  $d^\top \partial_x^2 f(x^*)d \geq 0$ .

This reduces to the well-known requirement of the second derivative to be positive (semi)definite if  $x^*$  is a point in the interior of the feasible region.

These conditions have a straightforward extension if constraints are present in the problem. The first order conditions are also denoted more commonly as Karush-Kuhn-Tucker (KKT) conditions.



**Theorem 2.4.4 (First-order necessary conditions: constrained case)**

Let  $x^*$  be a local minimum of  $f$  satisfying the constraints  $h(x^*) = 0, g(x^*) \leq 0$ . Then there are Lagrange multipliers  $\lambda \in \mathbb{R}^m$  and  $\mu \in \mathbb{R}^p$  with  $\mu \geq 0$  such that

$$\begin{aligned}\partial_x f(x^*) + \lambda^\top \partial_x h(x^*) + \mu^\top \partial_x g(x^*) &= 0 \\ \mu^\top g(x^*) &= 0.\end{aligned}$$

The second-order conditions are given next, where the following notation is adopted for the second derivative of a function

$$[F(x)]_{ij} = \frac{\partial^2 f}{\partial x_i \partial x_j}(x).$$

The notation for the second derivative of vectors of functions  $h(x)$  is given by the tensor  $H(x)$  that always occurs in a product with a vector  $\lambda^\top = [\lambda_1 \dots \lambda_m]$  such that the tensor is reduced to the matrix

$$\lambda^\top H(x) = \sum_{k=1}^m \lambda_k H_k(x),$$

where the second derivative of the consecutive functions is given by

$$[H_k(x)]_{ij} = \frac{\partial^2 h_k}{\partial x_i \partial x_j}(x)$$

for  $k = 1, \dots, m$  and  $i, j = 1, \dots, n$ . This definition also applies to  $g(x)$ .

**Theorem 2.4.5 (Second-order necessary conditions: constrained case)**

Let  $x^*$  be a local minimum of  $f$  satisfying the constraints  $h(x^*) = 0, g(x^*) \leq 0$ . If we denote by  $M$  the tangent plane

$$M = \{y \mid \partial_x h(x^*)y = 0, \partial_x g_i(x^*)y = 0 \text{ for } i \in \mathcal{J}\}$$

with  $\mathcal{J} = \{i \mid g_i(x^*) = 0\}$  being a finite set of indices, the active inequality constraints, then there is a  $\lambda \in \mathbb{R}^m$ ,  $\mu \in \mathbb{R}^p$ ,  $\mu \geq 0$  such that the matrix

$$L(x^*) = F(x^*) + \lambda^\top H(x^*) + \mu^\top G(x^*)$$

is positive semidefinite on  $M$ , that is  $y^\top L(x^*)y \geq 0$  for all  $y \in M$ .

The optimality conditions for unconstrained and equality constrained problems are practically equal. Therefore the complexity of algorithms for these cases are generally of the same order. For inequality constrained problems the first-order conditions are much more complicated than for the other classes of problems, which expresses itself in the high complexity and computational load of algorithms.

### 2.4.7 Primal-dual interior-point methods

As it was mentioned in the previous sections, in linear model predictive control the optimal inputs are achieved by solving quadratic programs (QPs). In the next chapter, we present a method for solving these quadratic programs efficiently. To solve the quadratic problem that arises in model predictive control, we employ a primal-dual interior-point method (IPM). In this chapter, we present the details of a primal-dual interior point method for the solution of quadratic programs of the structure found in our specific applications. For more details on primal-dual interior point methods, see [99]. The basic framework for the development of this method is similar to the linear MPC problem formulation found in [75], the highlight of which is the property that the QP solution time is a linear function of the prediction horizon length, rather than a cubic relationship. These details will be discussed in the next chapter.

Consider a convex quadratic program similar to (2.19):

$$\begin{aligned} \min_w \quad & \frac{1}{2}w^\top Hw + f^\top w, \\ \text{subject to} \quad & A_{\text{eq}}w = b_{\text{eq}}, \\ & A_{\text{in}}w \leq b_{\text{in}}, \end{aligned} \tag{2.21}$$

where matrix  $H$  is positive semidefinite. We have also introduced an equality constraint in this formulation. The Karush-Kuhn-Tucker (KKT) conditions for optimality are that there exist Lagrange multipliers  $p$  and  $\lambda$  such that the following conditions hold:

$$\begin{aligned} Hw + A_{\text{eq}}^\top p + A_{\text{in}}^\top \lambda + f &= 0, \\ -A_{\text{eq}}w + b_{\text{eq}} &= 0, \\ (-A_{\text{in}}w + b_{\text{in}})\lambda &= 0, \\ -A_{\text{in}}w + b_{\text{in}} &\geq 0, \\ \lambda &\geq 0. \end{aligned}$$

The convex nature of the objective guarantees that these conditions are necessary and sufficient for optimality (see [69, 67]). It is convenient to replace the last two conditions with the equivalent, but easier to handle, system

$$\begin{aligned} \begin{bmatrix} Hw + A_{\text{eq}}^\top p + A_{\text{in}}^\top \lambda + f \\ -A_{\text{eq}}w + b_{\text{eq}} \\ -A_{\text{in}}w + b_{\text{in}} - t \\ T\Lambda e \end{bmatrix} &= 0, \\ (\lambda, t) &\geq 0, \end{aligned} \tag{2.22}$$

in which  $t$  is the vector of slack variables,  $\Lambda$  and  $T$  are matrices with  $\lambda$  and  $t$  on the diagonals, respectively, and  $e$  is a vector of ones.

A very common IPM technique is the primal-dual Mehrotra's prediction-corrector algorithm [60]. Primal-dual interior-point methods solve the quadratic program by iterating from an initial guess to the optimal one. Using the notation in (2.22), the iterates are determined by solving the linear system

$$\begin{bmatrix} H & A_{\text{eq}}^\top & A_{\text{in}}^\top & 0 \\ -A_{\text{eq}} & 0 & 0 & 0 \\ -A_{\text{in}} & 0 & 0 & -I \\ 0 & 0 & T & \Lambda \end{bmatrix} \begin{bmatrix} \Delta w \\ \Delta p \\ \Delta \lambda \\ \Delta t \end{bmatrix} = \begin{bmatrix} r_H \\ r_{A_{\text{eq}}} \\ r_{A_{\text{in}}} \\ r_t \end{bmatrix}, \quad (2.23)$$

for specific choices of the right hand side variables. The set of linearized KKT equations is solved three times in each iteration. First a classical *Newton step* is calculated in the primal-dual space to approach the optimum of the non-linear KKT equations. This first step is also called the predictor step. However, if only this Newton step is considered, the convergence will become very slow once one approaches the constraint boundaries in the primal-dual space (given by  $t^\top \lambda = 0$ ) and therefore a *centering step* is calculated that brings the current iterate back to the interior region in the primal-dual space. Indeed, the best convergence turns out to be reached when the so-called "central path" is followed in the primal-dual space. This central path can be parameterized by  $\tau$  and is defined as  $\tau = t^\top \lambda$ . Finally, a *corrector step* is added that uses second-order information from the solution of the Newton step (i.e. information about the curvature of the central path) to approach this central path even closer. The centering and the correction can be carried out together.

Thus, in the Mehrotra predictor-corrector algorithm, we solve (2.23) twice; the first phase is the predictor or affine-scaling step, while the second phase is the centering-correction direction. The value of  $\tau$  is mostly written as the product of a *centering parameter*  $\sigma \in [0, 1]$  and the so-called *duality gap*  $\mu$ , defined as the average value of the product of  $t$  and  $\lambda$ . So, the first step for the solution of the QP is to determine the duality gap

$$\mu = \frac{t^\top \lambda}{n_{\text{in}}},$$

in which  $n_{\text{in}}$  is the number of rows in  $A_{\text{in}}$ . The duality gap is a measure of the feasibility of the solution and it makes sure that we take a step which centers all pair-wise  $t, \lambda$ -products at the same rate. As the algorithm proceeds, the duality gap converges to zero, as required by the fourth equation of (2.22). The second step is to solve (2.23) with the affine-scaling step with the right-hand side

$$\begin{bmatrix} r_H \\ r_{A_{\text{eq}}} \\ r_{A_{\text{in}}} \\ r_t \end{bmatrix} = \begin{bmatrix} -Hw - A_{\text{eq}}^\top p - A_{\text{in}}^\top \lambda - f \\ A_{\text{eq}}w - b_{\text{eq}} \\ A_{\text{in}}w - b_{\text{in}} + t \\ -T\Lambda e \end{bmatrix}. \quad (2.24)$$

We denote the solution to the predictor step by  $(\Delta w_{\text{aff}}, \Delta p_{\text{aff}}, \Delta \lambda_{\text{aff}}, \Delta t_{\text{aff}})$ . The next step is to determine the maximum affine-scaling step length by

$$\alpha_{\text{aff}} = \arg \max \{ \alpha \in [0, 1] | (\lambda, t) + \alpha(\Delta \lambda_{\text{aff}}, \Delta t_{\text{aff}}) \geq 0 \}. \quad (2.25)$$

The scalar  $\alpha_{\text{aff}}$  determines the duality gap for the full step to the boundary

$$\mu_{\text{aff}} = \frac{(\lambda + \alpha_{\text{aff}} \Delta \lambda_{\text{aff}})^\top (\lambda + \alpha_{\text{aff}} \Delta \lambda_{\text{aff}})}{n_{\text{in}}}.$$

Using this value, we define

$$\sigma = (\mu_{\text{aff}} / \mu)^3.$$

The next step is to solve for the corrector step. We determine

$$(\Delta w_{\text{cc}}, \Delta p_{\text{cc}}, \Delta \lambda_{\text{cc}}, \Delta t_{\text{cc}})$$

by solving (2.23) with

$$\begin{bmatrix} r_H \\ r_{A_{\text{eq}}} \\ r_{A_{\text{in}}} \\ r_t \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ -\Delta T_{\text{aff}} \Delta \lambda_{\text{aff}} e + \sigma \mu e \end{bmatrix}. \quad (2.26)$$

The overall search direction is then defined by

$$(\Delta w, \Delta p, \Delta \lambda, \Delta t) = (\Delta w_{\text{aff}}, \Delta p_{\text{aff}}, \Delta \lambda_{\text{aff}}, \Delta t_{\text{aff}}) + (\Delta w_{\text{cc}}, \Delta p_{\text{cc}}, \Delta \lambda_{\text{cc}}, \Delta t_{\text{cc}}).$$

To keep  $(\lambda, t)$  strictly nonnegative, we determine

$$\alpha_{\text{max}} = \arg \max \{ \alpha \in [0, 1] | (\lambda, t) + \alpha(\Delta \lambda, \Delta t) \geq 0 \} \quad (2.27)$$

and define the new iterate as

$$(w, p, \lambda, t)^+ = (w, p, \lambda, t) + \alpha_{\text{max}} \gamma (\Delta w, \Delta p, \Delta \lambda, \Delta t),$$

in which the heuristic factor  $\gamma$  is chosen to be close to 1. The process is repeated until convergence to within a specified tolerance is achieved.

In our algorithm, we perform block elimination on (2.23) and solve the more condensed system

$$\begin{bmatrix} H & A_{\text{eq}}^\top & A_{\text{in}}^\top \\ -A_{\text{eq}} & 0 & 0 \\ -A_{\text{in}} & 0 & \Lambda^{-1} T \end{bmatrix} \begin{bmatrix} \Delta w \\ \Delta p \\ \Delta \lambda \end{bmatrix} = \begin{bmatrix} \hat{r}_H \\ \hat{r}_{A_{\text{eq}}} \\ \hat{r}_{A_{\text{in}}} \end{bmatrix}, \quad (2.28)$$

noting that

$$\Delta t = \Lambda^{-1} (r_t - T \Delta \lambda). \quad (2.29)$$

In practice, the linear programs in (2.25) and (2.27) are not solved by a linear programming method. Instead, we present the more convenient solution

$$\alpha = \min \left\{ \frac{-1}{\min\{(\Delta\lambda, \Delta t)/(\lambda, t)\}}, 1 \right\}, \quad (2.30)$$

which is valid unless all elements of  $(\Delta\lambda, \Delta t)$  are nonnegative, in which case  $\alpha = 1$ .

## 2.5 Implementation and tuning

### 2.5.1 Estimated state and observer dynamics

We need to address the more realistic case, when we do not have measurements of the whole state vector, and must use an observer. We will see that the solution is very similar to the solution in the previous case. The only difference is that we use an observer, and use the state estimate  $\hat{x}(k|k)$  to replace the measured state  $x(k)$ . Now we have more dynamic complexity in the controller, because the state vector includes the observer state. We will obtain an optimal estimate by solving a stochastic linear quadratic problem, where we have Gaussian noises acting on the states and outputs, and the observer gain is obtained using Kalman filtering theory (see, for example [4, 9, 8]).

In order to use this theory, we put our plant and assumptions into standard form. Instead of the model (2.2)–(2.3) which we used before, we will now use the model

$$x(k+1) = Ax(k) + Bu(k) + B_d d(k) \quad (2.31)$$

$$y(k) = Cx(k) + v(k), \quad (2.32)$$

where the unmeasured disturbance  $d$  is modelled as a stochastic process, which is assumed to be generated through the following difference equation

$$x_w(k+1) = A_w x_w(k) + B_w w(k)$$

$$d(k) = C_w x_w(k),$$

where  $w(k)$  is a discrete-time white noise with covariance  $R_w \geq 0$ . The measurements of  $y(k)$  are corrupted by white noise  $v(k)$  with covariance  $R_v > 0$ . We also assume  $w$  and  $v$  to be uncorrelated, i.e.  $E\{wv^\top\} = 0$ . Combining the equations gives

$$\begin{aligned} \tilde{x}(k+1) &= \begin{bmatrix} A & B_d C_w \\ 0 & A_w \end{bmatrix} \tilde{x}(k) + \begin{bmatrix} B \\ 0 \end{bmatrix} u(k) + \begin{bmatrix} 0 \\ B_w \end{bmatrix} w(k) \\ y(k) &= \begin{bmatrix} C & 0 \end{bmatrix} \tilde{x}(k) + v(k), \end{aligned}$$

where  $\tilde{x}(k) = \begin{bmatrix} x^\top(k) & x_w^\top(k) \end{bmatrix}^\top$ .

Let this state-space representation be denoted by  $(\tilde{A}, \tilde{B}, \tilde{B}_w, \tilde{C})$  and  $\hat{x}(k|k)$  be the estimate of the state  $\tilde{x}(k)$ . Assume that  $\tilde{x}(0) \in \mathcal{N}(\bar{x}, P_0)$  is independent of  $w$  and  $v$ . The estimation error  $e(k) = \tilde{x}(k) - \hat{x}(k|k)$  satisfies  $\mathbb{E}e(k) = 0$  provided  $\hat{x}(0) = \bar{x}$ . An optimal (minimum error variance) estimate can then be obtained by iterating the following equations which define the Kalman filter

$$\begin{aligned} \text{Correction:} \quad & \hat{x}(k|k) = \hat{x}(k|k-1) + L'(k) \left[ y(k) - \tilde{C}\hat{x}(k|k-1) \right] \\ \text{Prediction:} \quad & \hat{x}(k+1|k) = \tilde{A}\hat{x}(k|k) + \tilde{B}u(k), \end{aligned}$$

where the Kalman filter gain  $L'(k)$  is given by

$$L'(k) = P(k)\tilde{C}^\top \left[ \tilde{C}P(k)\tilde{C}^\top + R_v \right]^{-1}$$

and where the covariance  $P(k) = \mathbb{E}e(k)e^\top(k)$  satisfies

$$P(k+1) = \tilde{A}P(k)\tilde{A}^\top - \tilde{A}P(k)\tilde{C}^\top \left[ \tilde{C}P(k)\tilde{C}^\top + R_v \right]^{-1} \tilde{C}P(k)\tilde{A}^\top + \tilde{B}_w R_w \tilde{B}_w^\top,$$

where  $P(0) = P_0$  represents  $\mathbb{E}\tilde{x}(0)\tilde{x}^\top(0)$ . The limit  $P_\infty = \lim_{k \rightarrow \infty} P(k)$  exists, which is a solution to the (filtering) algebraic Riccati equation

$$P_\infty = \tilde{A}P_\infty\tilde{A}^\top - \tilde{A}P_\infty\tilde{C}^\top \left[ \tilde{C}P_\infty\tilde{C}^\top + R_v \right]^{-1} \tilde{C}P_\infty\tilde{A}^\top + \tilde{B}_w R_w \tilde{B}_w^\top.$$

The stationary Kalman filter gain is obtained as

$$L'_\infty = P_\infty\tilde{C}^\top \left[ \tilde{C}P_\infty\tilde{C}^\top + R_v \right]^{-1}.$$

Note that the Kalman filter equations have the form of an observer, with a special choice of observer gain. If the pair  $(\tilde{A}, \tilde{B}_w R_w^{1/2})$  is stabilizable and the pair  $(\tilde{C}, \tilde{A})$  is detectable, then state-transition matrix  $\tilde{A}(I - L'_\infty\tilde{C})$  has all its eigenvalues inside the unit circle. Also, note that  $R_v \geq 0$  or even  $R_v = 0$  is allowed, provided that  $\tilde{C}P_\infty\tilde{C}^\top > 0$  or  $\tilde{C}P(k)\tilde{C}^\top > 0 \forall k$ .

To obtain the vector of predicted controlled outputs,  $Y(k)$ , we go back to equation (2.14), and simply replace  $x(k)$  by the best estimate of it available to us, namely  $\begin{bmatrix} I_{n_x} & 0 \end{bmatrix} \hat{x}(k|k)$ . So now we define

$$Y(k) = \Phi \begin{bmatrix} I_{n_x} & 0 \end{bmatrix} \hat{x}(k|k) + \Gamma u(k-1) + G_y \Delta U(k).$$

The controller structure in this case is shown in Figure 2.1. Once these changes have been made, the derivation of the optimal control  $\Delta u_{\text{opt}}(k)$  is exactly the same as before. That is, we solve  $J_k(\begin{bmatrix} I_{n_x} & 0 \end{bmatrix} \hat{x}(k|k), u)$  subject to (2.2) and constraints.

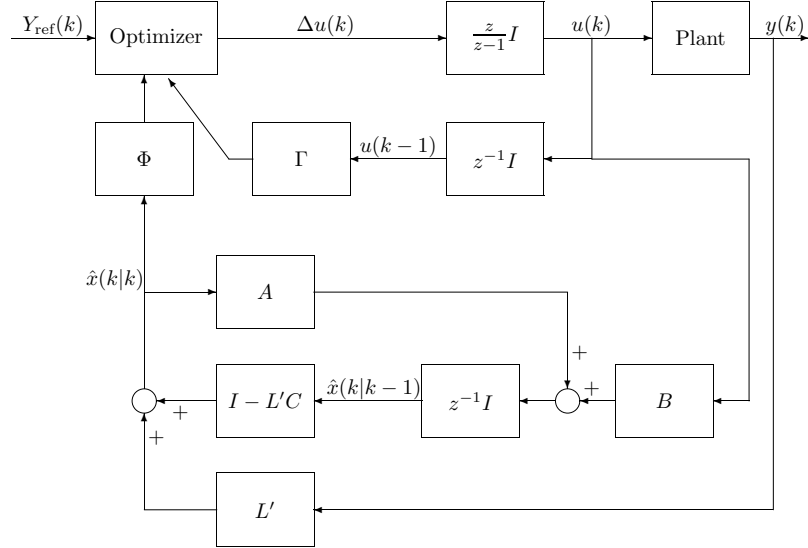


Figure 2.1: MPC controller structure with state estimator.

### 2.5.2 Limitations of the current technology

The existing industrial MPC technology has several limitations, as pointed out in [66]. The most relevant ones are:

- *Over-parameterized models*: most of the commercial products use a step or impulse response model of the plant. For instance, a first order process can be described by a transfer function model using only three parameters (gain, time constant and dead-time) while a step response model may require an infinite number of coefficients to describe the same dynamics. Besides, these models can not be inferred from observations of unstable processes. These problems can be overcome by using an auto-regressive parametric model.
- *Tuning*: the tuning procedure is not clearly defined since the trade-off between tuning parameters and closed loop behavior is generally not very clear. Tuning in the presence of constraints may be even more difficult, and even for the nominal case, it is not easy to guarantee closed loop stability; that is why so much effort must be spent on prior simulations. The feasibility of the problem is one of the most challenging topics of MPC nowadays.
- *Sub-optimality of the dynamic optimization*: several packages provide sub-optimal solutions to the minimization of the cost function in order to

speed up the solution time. It can be accepted in high speed applications (tracking systems) where solving the problem at every sampling time may not be feasible, but it is difficult to justify for process control applications unless it can be shown that the sub-optimal solution is always very nearly optimal.

- *Model uncertainty*: although model identification packages provide estimates of model uncertainty, only one product (RMPCT) uses this information in the control design. All other controllers can be detuned in order to improve robustness, although the relation between performance and robustness is not very clear.
- *Constant disturbance assumption*: although perhaps the most reasonable assumption is to consider that the output disturbance will remain constant in the future, better feedback would be possible if the distribution of the disturbance could be characterized more carefully.
- *Analysis*: a systematic analysis of stability and robustness properties of MPC is not possible in its original finite horizon formulation. The control law is in general time-varying and cannot be represented in the standard closed loop form, especially in the constrained case.

The technology is continually evolving and the next generation will have to face new challenges in open topics such as model identification, unmeasured disturbance estimation and prediction, systematic treatment of modelling error and uncertainty or such an open field as nonlinear model predictive control.

## 2.6 Design case studies

### 2.6.1 Evaporation process

The first nonlinear process is based on the forced-circulation evaporator described in [68], and shown in Figure 2.2. A feed stream enters the process at concentration  $X_1$  and temperature  $T_1$ , with flow rate  $F_1$ . It is mixed with a recirculating liquor, which is pumped through the evaporator at flow rate  $F_3$ . The evaporator itself is a heat exchanger, which is heated by steam flowing at a rate  $F_{100}$ , with entry temperature  $T_{100}$  and pressure  $P_{100}$ . The mixture of feed and recirculating liquor boils inside the heat exchanger, and the resulting mixture of vapor and liquid enters a separator, in which the liquid level is  $L_2$ . The operating pressure inside the evaporator is  $P_2$ . Most of the liquid from the separator becomes the recirculating liquor. A small portion of it is drawn off as product, with concentration  $X_2$ , at a flow rate  $F_2$  and temperature  $T_2$ . The vapor from the separator flows to a condenser at flow rate  $F_4$  and temperature  $T_3$ , where it is condensed by being cooled with water flowing at a rate  $F_{200}$ ,



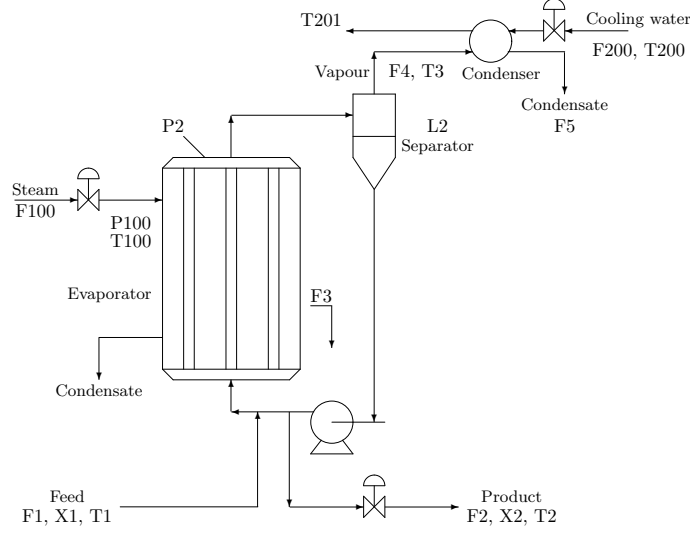


Figure 2.2: Evaporation process.

with entry temperature  $T_{200}$  and exit temperature  $T_{201}$ . The details of the process model are given below.

### Process liquid mass balance

A mass balance on the total process liquid (solvent and solute) in the system yields

$$\rho A \frac{dL_2}{dt} = F_1 - F_4 - F_2,$$

where  $\rho$  is the liquid density and  $A$  is the cross-sectional area of the separator. (The product  $\rho A$  is assumed to be constant at 20 kg/m.)

### Process liquid solute mass balance

A mass balance on the solute in the process liquid phase yields

$$M \frac{dX_2}{dt} = F_1 X_1 - F_2 X_2,$$

where  $M$  is the amount of liquid in the separator and is assumed to be constant at 20 kg.

**Process vapor mass balance**

A mass balance on the process vapor in the evaporator will express the total mass of the water vapor in terms of the pressure that exists in the system

$$C \frac{dP_2}{dt} = F_4 - F_5,$$

where  $C$  is a constant that converts the mass of vapor into an equivalent pressure and is assumed to have a value of 4 kg/kPa. This constant can be derived from the ideal gas law.

**Process liquid energy balance**

The process liquid is assumed to always exist at its boiling point and to be perfectly mixed (assisted by the high circulation rate). The liquid temperature is

$$T_2 = 0.5616 P_2 + 0.3126 X_2 + 48.43,$$

which is a linearization of the saturated liquid line for water about the standard steady-state value and includes a term to approximate boiling point elevation due to the presence of the solute. The vapor temperature is

$$T_3 = 0.507 P_2 + 55.0,$$

which is a linearization of the saturated liquid line for water about the standard steady-state value.

The dynamics of the energy balance are assumed to be very fast so that

$$F_4 = (Q_{100} - F_1 C_P (T_2 - T_1)) / \lambda,$$

where  $C_P$  is the heat capacity of the liquor and is assumed constant at a value of 0.07 kW/K(kg/min) and  $\lambda$  is the latent heat of vaporization of the liquor and is assumed to have a constant value of 38.5 kW/(kg/min).

The sensible heat change between  $T_2$  and  $T_3$  for  $T_4$  is considered small compared to the latent heat. It is assumed that there are no heat losses to the environment or heat gains from the energy input of the pump.

**Heater steam jacket**

Steam pressure  $P_{100}$  is a manipulated variable which determines steam temperature under assumed saturated conditions. An equation relating steam temperature to steam pressure can be obtained by approximating the saturated steam temperature-pressure relationship by local linearization about the steady-state value

$$T_{100} = 0.1538 P_{100} + 90.$$

The rate of heat transfer to the boiling process liquid is given by

$$Q_{100} = UA_1(T_{100} - T_2),$$

where  $UA_1$  is the overall heat transfer coefficient times the heat transfer area and is a function of the total flow-rate through the tubes in the evaporator

$$UA_1 = 0.16 (F_1 + F_3).$$

The steam flow-rate is calculated from

$$F_{100} = Q_{100}/\lambda_s,$$

where  $\lambda_s$  is the latent heat of steam at the saturated conditions, assumed constant at a value of 36.6 kW/(kg/min). The dynamics within the steam jacket have been assumed to be very fast.

### **Condenser**

The cooling water flow-rate  $F_{200}$  is a manipulated variable and the inlet temperature  $T_{200}$  is a disturbance variable. A cooling water energy balance yields

$$Q_{200} = F_{200}C_P(T_{201} - T_{200}),$$

where  $C_P$  is the heat capacity of the cooling water assumed constant at 0.07 kW/(kg/min). The heat transfer rate equation is approximated by

$$Q_{200} = UA_2(T_3 - 0.5 (T_{200} + T_{201})),$$

where  $UA_2$  is the overall heat transfer coefficient times the heat transfer area, which is assumed constant with a value of 6.84 kW/K.

These two equations can be combined to eliminate  $T_{201}$  to give explicitly

$$Q_{200} = \frac{UA_2(T_3 - T_{200})}{1 + UA_2/(2C_P F_{200})}.$$

It follows that

$$T_{201} = T_{200} + Q_{200}/(F_{200}C_P).$$

The condensate flow-rate is

$$F_5 = Q_{200}/\lambda,$$

where  $\lambda$  is the latent heat of vaporization of water assumed constant at 38.5 kW/K(kg/min). The dynamics within the condenser have been assumed to be very fast.

State variables  $L_2$ ,  $X_2$  and  $P_2$  are the controlled outputs for this process. These outputs, together with their initial equilibrium (steady-state) values and

Output variable		Equilibrium	Lower limit	Upper limit
Separator level	$L_2$	1 m	0	2
Product composition	$X_2$	25 %	0	50
Operating pressure	$P_2$	50.5 kPa	0	100

Table 2.1: Output variables.

Input variable		Equilibrium	Lower limit	Upper limit
Product flow rate	$F_2$	2.0 kg/min	0	4
Steam pressure	$P_{100}$	194.7 kPa	0	400
Cooling water flow rate	$F_{200}$	208.0 kg/min	0	400

Table 2.2: Input variables.

Disturbance		Equilibrium
Circulating flow rate	$F_3$	50.0 kg/min
Feed flow rate	$F_1$	10.0 kg/min
Feed concentration	$X_1$	5.0 %
Feed temperature	$T_1$	40.0 °C
Cooling water entry temperature	$T_{200}$	25.0 °C

Table 2.3: Disturbance variables.

constraints, are given in Table 2.1. The manipulated variables are chosen to be  $F_2$ ,  $P_{100}$  and  $F_{200}$  (see Table 2.2). There are five disturbance signals, namely  $F_3$ ,  $F_1$ ,  $X_1$ ,  $T_1$  and  $T_{200}$ , which are left fixed at their equilibrium values (Table 2.3).

Input and output constraints are imposed in the optimization. The level  $L_2$  is kept at the value of 1 m. The setpoint for  $X_2$  is ramped down linearly from 25% to 15% over a period of 20 minutes, and the operating pressure  $P_2$  is simultaneously ramped up from 50.5 kPa to 70 kPa. The process is sampled with  $T = 1$  min. The tuning parameters include prediction horizon  $N = 30$ , control horizon  $N_c = 3$  and the following constant weighting matrices:

$$Q = \begin{bmatrix} 10^3 & 0 & 0 \\ 0 & 10^2 & 0 \\ 0 & 0 & 10^2 \end{bmatrix}, \quad R = I_3.$$

The result of linear MPC application with a single model obtained at the initial equilibrium condition is given in Figure 2.3 (dashed line). We see that control is initially satisfactory and the setpoints are held quite closely, even when  $X_2$  and  $P_2$  start changing as they flow their ramp setpoints. But about half-way through the ramp the separator level  $L_2$  drifts considerably off its setpoint and does not return to it. Although the product concentration  $X_2$  tracks the

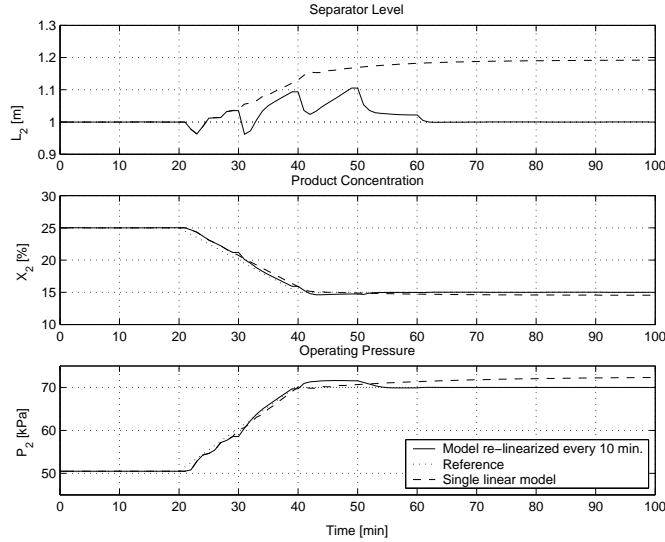


Figure 2.3: Outputs with linear MPC based on a single linear model (dashed) and model re-linearized every 10 min (solid).

setpoint quite closely and settles correctly at the required steady-state level, the operating pressure  $P_2$  drifts away from its setpoint. The deterioration of the closed-loop behavior is due to the fact that the linearized model used by the MPC has become a very inaccurate representation of the small-deviation behavior of the evaporator at the new operating conditions.

The most common way of dealing with plant nonlinearities in practice is to re-linearize or adapt the linear internal model. Figure 2.3 also shows the result of linear MPC when the internal model is re-linearized every 10 minutes. It remains a reasonably good representation of the plant behavior throughout the change of operating conditions, and it is clearly seen that the tracking control is much better. The product concentration and operating pressure are both held close to their setpoints, and the separator level, although still undergoing considerable deviations from its setpoint, is now controlled much better than before.

### 2.6.2 High-purity binary distillation column

The second case study is concerned with MPC controller design for a high purity distillation column. Figure 2.4 presents the distillation process, its 82-state nonlinear model can be found in [84]. The column contains 41 trays that are located along its length. The raw material enters the column at a flow rate of  $F$  and with composition  $z_f$ . The top product, the distillate, is condensed and

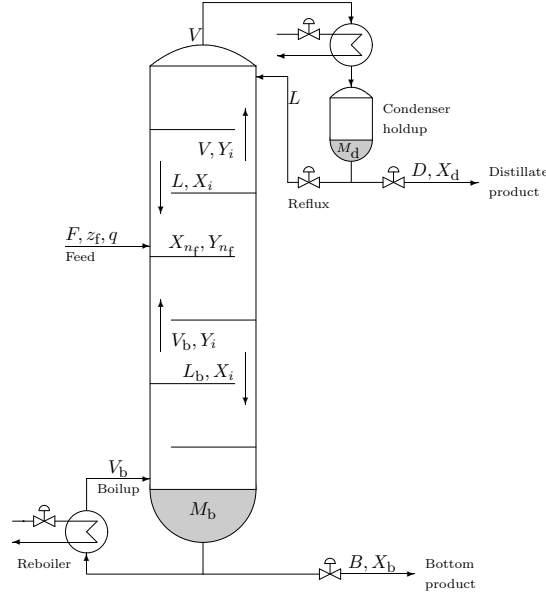


Figure 2.4: Binary high-purity distillation column.

removed at a flow rate of  $D$  and with composition  $X_d$ . The bottom product is removed as a liquid at a flow rate of  $B$  and with composition  $X_b$ . The operation of the column requires that some of the bottom product is reboiled at a rate of  $V_b$  to ensure the continuity of the vapor flow. In the same way, some of the distillate is refluxed to the top tray at a rate of  $L$  to ensure the continuity of the liquid flow. The vapor boilup  $V_b$  and the reflux flow  $L$  are the manipulated variables. Feed flow rate  $F$  with composition  $z_f$  act as disturbances.

The distillation column is known to be an ill-conditioned process. This can be shown by considering the following steady-state model of the column

$$G = \begin{bmatrix} 87.8 & -86.4 \\ 108.2 & -109.6 \end{bmatrix} \quad (2.33)$$

for scaled output variables. Thus, since the elements are much larger than 1 in magnitude this suggests that there will be no problems with input constraints. However, this is somewhat misleading as the gain in the low-gain direction (corresponding to the smallest singular value) is actually only just above 1. To see this we consider the SVD of  $G$ :

$$G = \begin{bmatrix} 0.625 & -0.781 \\ 0.781 & 0.625 \end{bmatrix} \begin{bmatrix} 197.2 & 0 \\ 0 & 1.39 \end{bmatrix} \begin{bmatrix} 0.707 & -0.708 \\ -0.708 & -0.707 \end{bmatrix}^T.$$

From the first input singular vector,  $\bar{u} = \begin{bmatrix} 0.707 & -0.708 \end{bmatrix}^T$  we see that the

gain is 197.2 when we increase one input and decrease the other input by a similar amount. On the other hand, from the second input singular vector,  $\underline{u} = \begin{bmatrix} -0.708 & -0.707 \end{bmatrix}^\top$ , we see that if we increase both inputs by the same amount then the gain is only 1.39. The reason for this is that the plant is such that the two inputs counteract each other. Thus, the distillation process is ill-conditioned, at least at steady-state, and the condition number is  $197.2/1.39 = 141.7$ .

The model in (2.33) represents two point (dual) composition control of a distillation column, where the top composition is to be controlled at  $X_d = 0.99$  (output  $y_1$ ) and the bottom composition at  $X_b = 0.01$  (output  $y_2$ ), using reflux  $L$  (input  $u_1$ ) and boilup  $V_b$  (input  $u_2$ ) as manipulated inputs. The upper left element of the gain matrix  $G$  is 87.8. Thus an increase in  $u_1$  by 1 (with  $u_2$  constant) yields a large steady-state change in  $y_1$  of 87.8, that is, the outputs are very sensitive to changes in  $u_1$ . Similarly, an increase in  $u_2$  by 1 (with  $u_1$  constant) yields  $y_1 = -86.4$ . Again, this is a very large change, but in the opposite direction of that for the increase in  $u_1$ . We therefore see that changes in  $u_1$  and  $u_2$  counteract each other, and if we increase  $u_1$  and  $u_2$  simultaneously by 1, then the overall steady-state change in  $y_1$  is only  $87.8 - 86.4 = 1.4$ .

Physically, the reason for this small change is that the compositions in the distillation column are only weakly dependent on changes in the internal flows (i.e. simultaneous changes in the internal flows  $L$  and  $V_b$ ). This can also be seen from the smallest singular value,  $\sigma(G) = 1.39$ , which is obtained for inputs in the direction  $\underline{u} = \begin{bmatrix} -0.708 & -0.707 \end{bmatrix}^\top$ . From the output singular vector  $\underline{y} = \begin{bmatrix} -0.781 & 0.625 \end{bmatrix}^\top$  we see that the effect is to move the outputs in different directions, that is, to change  $y_1 - y_2$ . Therefore, it takes a large control action to move the compositions in different directions, that is, to make both products purer simultaneously. This makes sense from a physical point of view.

On the other hand, the distillation column is very sensitive to changes in external flows (i.e. increase  $u_1 - u_2 = L - V_b$ ). This can be seen from the input singular vector  $\underline{u} = \begin{bmatrix} 0.707 & -0.708 \end{bmatrix}^\top$  associated with the largest singular value, and is a general property of distillation columns where both products are of high purity. The reason for this is that the external distillate flow (which varies as  $V_b - L$ ) has to be about equal to the amount of light component in the feed, and even a small imbalance leads to large changes in the product compositions.

The MPC objective would be to bring the controlled outputs  $X_d$  and  $X_b$  to the setpoints of 0.99 and 0.01, respectively. Figure 2.5 shows the outputs of the nonlinear distillation column controlled by a linear MPC. The linear controller used only one linear model obtained at the setpoint values. We see that it is quite difficult to efficiently tune the linear MPC controller and make the outputs smoothly approach the given setpoints.

As in the case of a distillation column, the process encountered by the con-

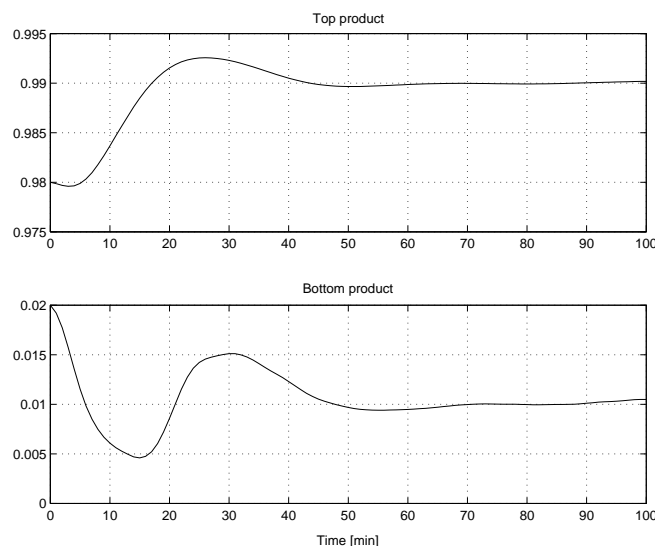


Figure 2.5: Top and bottom product compositions simulated with linear MPC.

troller may require excessive input movement in order to control the outputs independently. This problem arises if two outputs respond in an almost identical way to the available inputs. It is important to note that this is a feature of the process to be controlled, any algorithm which attempts to control an ill-conditioned process must address this problem. For a process with gain matrix  $G$ , the condition number of  $G^T G$  provides a measure of process ill-conditioning, a high condition number means that small changes in the future error vector will lead to large MV moves.

Three active strategies are currently used by MPC controllers to deal with ill-conditioned processes: singular value thresholding, controlled variable ranking and LP slack variables. Input move suppression also improves the condition number of the process internal model, this can be thought of as a passive strategy for dealing with ill-conditioning.

The Singular Value Thresholding (SVT) method involves decomposing the process model using a singular value decomposition. Singular values below a threshold magnitude are discarded, and a process model with a much lower condition number is then reassembled and used for control. The neglected singular values represent the direction along which the process hardly moves even if a large MV change is applied, the SVT method gives up this direction to avoid erratic MV changes. This method solves the ill-conditioning problem at the expense of neglecting the smallest singular values. If the magnitude of these singular values is small comparing to model uncertainty, it may be better to



neglect them anyway. After thresholding, the collinear CV's are approximated with the principal singular direction. In the case of two collinear CV's, for example, this principal direction is a weighted average of the two CV's. Note that the SVT approach is sensitive to output weighting. If one CV is weighted much more heavily than another, this CV will represent the principal singular direction.

Another way to address this issue is to use a user-defined set of CV controllability ranks. When a high condition number is detected, the controller drops low priority CV's until a well-conditioned sub-process remains. The sub-process will be controlled without erratic input movement but the low priority CV's will be uncontrolled. Note, however, that if a low priority CV is dropped because its open loop response is close to that of a high priority output, it will follow the high priority CV and will therefore still be controlled in a loose sense. In the case of two collinear CV's having no differentiable priority, it may be desirable to use an weighted average of the two.

The DMC algorithm handles steady-state ill-conditioning through the use of slack variable weights in the steady-state LP. If two outputs are nearly collinear, the LP will select the one with the largest slack variable weight for control. Controllers that use input move suppression, such as DMC algorithm, provide an alternative strategy for dealing with ill-conditioning. Input move suppression factors increase the magnitude of the diagonal elements of the matrix to be inverted in the least squares solution, directly lowering the condition number. The move suppression values can be adjusted to the point that erratic input movement is avoided for the commonly encountered sub-processes. In the limit of infinite move suppression the condition number becomes one for all sub-processes. There probably exists a set of finite move suppression factors which guarantee that all sub-processes have a condition number greater than a desired threshold value. In the case of two collinear CV's, the move suppression approach gives up a little bit on moving each CV towards its target. The move suppression solution is similar to that of the SVT solution in the sense that it tends to minimize the norm of the MV moves.

In the next chapter we present an MPC algorithm for nonlinear systems, which allows the direct use of multiple linear models on the prediction horizon. We will also investigate its performance on the distillation column in case of high purity components.

### 2.6.3 Motivating example: Oil fractionator

In this section we shall consider the well-known heavy oil fractionator (distillation column) problem, as defined in [50]. This is a linear model which does not represent a real, existing process, but was devised to exhibit the most significant control engineering features faced on real oil fractionators. It was intended to serve as a standard problem on which various control solutions

could be demonstrated in simulation. The fractionator is shown in Figure 2.6.

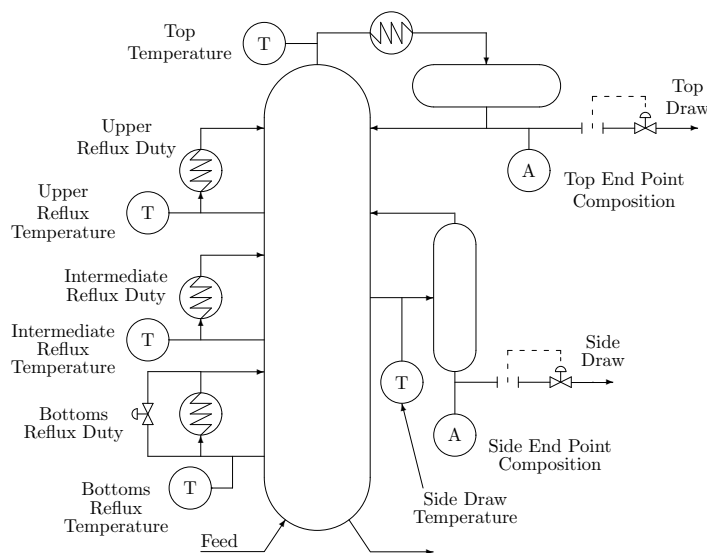


Figure 2.6: Heavy oil fractionator.

A gaseous feed enters at the bottom of the fractionator. Three product streams are drawn off, at the top, side and bottom of the fractionator (shown on the right-hand side in the Figure 2.6). There are also three circulating (reflux) loops at the top, middle (subsequently referred to as intermediate) and bottom of the fractionator, which are shown on the left-hand side of the Figure 2.6. Heat is carried into the fractionator by the feed, and these circulating reflux loops are used to remove heat from the process, by means of heat exchangers. The heat removed in this way is used to supply heat to other processes, such as other fractionators. The amount of heat removed by each reflux loop is expressed as heat duty; the larger the duty, the more heat is recirculated back into the fractionator, and thus the smaller the amount of heat removed. The gains from heat duties to temperatures are therefore positive. The heat removed in the top two reflux loops is determined by the requirements of other processes, and these therefore act as disturbance inputs to the fractionator. An important difference between them is that the Intermediate Reflux Duty is considered to be measured, and is therefore available for feed-forward control, whereas the Upper Reflux Duty is an unmeasured disturbance. The Bottoms Reflux Duty, by contrast, is a manipulated variable, which can be used to control the process. The Bottoms Reflux is used to generate steam for use on other units; so, although the Bottoms Reflux Duty can be used to control the oil fractionator, keeping it as low as possible corresponds to generating as much

steam as possible, and is therefore economically advantageous. There are two additional manipulated variables (that is, control inputs): the Top Draw and the Side Draw, namely the flow rates of the products drawn off at the top and side of the fractionator. Again, these are defined in such a way that a positive change leads to an increase of all temperatures in the fractionator. Altogether, then, there are five inputs to the fractionator, of which three are manipulated variables (control inputs) and two are disturbances.

There are seven measured outputs from the fractionator. The first two outputs are the compositions of the Top End Point product and of the Side End Point product. These are measured by analyzers, and the composition is represented by a scalar variable in each case. The remaining five outputs are all temperatures: the Top Temperature, the Upper Reflux Temperature, the Side Draw Temperature, the Intermediate Reflux Temperature, and the Bottoms Reflux Temperature. Only three of these outputs are to be controlled: the Top End Point and Side End Point compositions, and the Bottoms Reflux temperature. The remaining four output measurements are available for use by the controller. (In the original problem definition the possibility is raised that the two analyzers may be unreliable, in which case these additional output measurements play an important role in allowing control to continue without having the analyzer measurements available.)

The transfer function from each input to output is modelled as a first-order lag with time delay. We shall order the inputs and outputs as shown in Table 2.4. The nominal transfer functions from the control and disturbance inputs

Variable	Function	Symbol
Top Draw	Control input	$u_1$
Side Draw	Control input	$u_2$
Bottoms Reflux Duty	Control input	$u_3, z_4$
Intermediate Reflux Duty	Measured disturbance	$d_m$
Upper Reflux Duty	Unmeasured disturbance	$d_u$
Top End Point	Controlled and measured output	$y_1, z_1$
Side End Point	Controlled and measured output	$y_2, z_2$
Top Temperature	Measured output	$y_3$
Upper Reflux Temp.	Measured output	$y_4$
Side Draw Temp.	Measured output	$y_5$
Intermediate Reflux Temp.	Measured output	$y_6$
Bottoms Reflux Temp.	Controlled and measured output	$y_7, z_3$

Table 2.4: Oil fractionator variables.

to all the outputs are given in Table 2.5. The step response of the fractionator confirms that the nominal plant is asymptotically stable, and that the settling times range from about 10 minutes (from the Intermediate Reflux Duty to the Side Draw Temperature) to about 250 minutes (from the Side Draw to the Side

End Point).

	$u_1$	$u_2$	$u_3$	$d_m$	$d_u$
$y_1, z_1$	$4.05 \frac{e^{-27s}}{50s+1}$	$1.77 \frac{e^{-28s}}{60s+1}$	$5.88 \frac{e^{-27s}}{50s+1}$	$1.20 \frac{e^{-27s}}{45s+1}$	$1.44 \frac{e^{-27s}}{40s+1}$
$y_2, z_2$	$5.39 \frac{e^{-18s}}{50s+1}$	$5.72 \frac{e^{-14s}}{60s+1}$	$6.90 \frac{e^{-15s}}{40s+1}$	$1.52 \frac{e^{-15s}}{25s+1}$	$1.83 \frac{e^{-15s}}{20s+1}$
$y_3$	$3.66 \frac{e^{-2s}}{9s+1}$	$1.65 \frac{e^{-20s}}{30s+1}$	$5.53 \frac{e^{-2s}}{40s+1}$	$1.16 \frac{1}{11s+1}$	$1.27 \frac{1}{6s+1}$
$y_4$	$5.92 \frac{e^{-12s}}{12s+1}$	$2.54 \frac{e^{-12s}}{27s+1}$	$8.10 \frac{e^{-2s}}{20s+1}$	$1.73 \frac{1}{5s+1}$	$1.79 \frac{1}{19s+1}$
$y_5$	$4.13 \frac{e^{-5s}}{8s+1}$	$2.38 \frac{e^{-7s}}{19s+1}$	$6.23 \frac{e^{-2s}}{10s+1}$	$1.31 \frac{1}{2s+1}$	$1.26 \frac{1}{22s+1}$
$y_6$	$4.06 \frac{e^{-20s}}{13s+1}$	$4.18 \frac{e^{-22s}}{33s+1}$	$6.53 \frac{e^{-s}}{9s+1}$	$1.19 \frac{1}{19s+1}$	$1.17 \frac{1}{24s+1}$
$y_7, z_3$	$4.38 \frac{e^{-20s}}{33s+1}$	$4.42 \frac{e^{-22s}}{44s+1}$	$7.20 \frac{1}{19s+1}$	$1.14 \frac{1}{27s+1}$	$1.26 \frac{1}{32s+1}$

Table 2.5: Oil fractionator model.

The focus of the control performance specifications is on rejecting disturbances, while minimizing the Bottoms Reflux Duty ( $u_3$ ) and satisfying constraints. There are setpoint specifications on the Top End Point and Side End Point compositions. Since we have a linearized model, we may as well assume that the set-points are at 0. There is a requirement that at steady state these two outputs should be within  $\pm 0.005$  of the set-point. The dynamic response requirement is more ambiguous, being that closed-loop speed of response should be between 0.8 and 1.25 of the open-loop speed. We shall assume that this refers to set-point response, rather than disturbance response; that removes most of the ambiguity because the open-loop response speed of the Top End Point and the Side End Point compositions to each of the three control inputs is nearly the same, with similar time delays and time constants. For disturbance rejection, the requirement is to return the Top and Side End Point compositions to their setpoints as quickly as possible, subject to satisfying constraints. The trade-off between returning these compositions to their set-points and minimizing the Bottoms Reflux Duty, should these conflict, is not specified. In addition there are constraints on inputs and outputs as shown in Table 2.6 ( $T$  is the sampling time). The variables have been scaled so that

Output constraints	$ z_i  \leq 0.5$ for $i = 1, 2, 3$
Input constraints	$ u_i  \leq 0.5$ for $i = 1, 2, 3$
Input move constraints	$ \Delta u_i  \leq 0.05T$ for $i = 1, 2, 3$

Table 2.6: Input and output constraints.

they all have similar constraints, and the input move constraints, which are 0.05 per minute for each input, have been expressed as limits on  $|\Delta u_j|$ , consistently with our earlier notation, in terms of the sampling and control update

interval  $T$ . Note that the Bottoms Reflux Temperature has constraints but no set-point, so it has a zone objective. The two disturbance inputs are known to take values in the range  $\pm 0.5$ , but they are otherwise unspecified.

One of the objectives is to minimize the Bottoms Reflux Duty, which is one of the control inputs. The most straightforward way to do this is to make this input also an output. Since we require the model to be strictly proper (no direct feed-through from input to output) we define a new controlled output as  $z_4(k) = u_3(k-1)$  for all  $k$ . It can be shown that a suitable setpoint for this output would be  $-0.32$ .

The impact of the sampling time on computational requirements is more significant. Closed-loop settling times in response to set-point changes are required to be in the region of 120-250 minutes (the relevant open-loop time constants range from 40 to 60 minutes; closed-loop time constants are to be between 80% and 125% of these values; take the settling time to be  $3 \times$  time constant and add the time delay of between 14 and 28 minutes). We will try a relatively long prediction horizon. Since constraints are to be enforced at all times, the option of choosing a small number of coincidence points is not available, at least for constraint enforcement – it remains a possibility for penalizing tracking errors in the cost function. Choosing a small sampling interval will therefore lead to a large prediction horizon, which will be computationally demanding. On the other hand, the effect of the Intermediate Reflux Duty, which is a measured disturbance, on the Bottoms Reflux Temperature, is relatively fast, with no time delay and a time constant of 27 minutes. Therefore choosing a sampling interval as large as 10 minutes may not give enough opportunity for effective feed-forward action. We shall therefore choose  $T = 4$  minutes.

If only the Top and Side End Point compositions and the Bottoms Reflux Temperature have been retained as outputs, and the Bottoms Reflux Duty is brought out as an additional output (with a one-step delay), then a minimal state-space realization of a discrete-time model, with  $T = 4$  minutes, has 80 states. Reducing the number of states would speed up the controller computations and reduce the risks of hitting numerical problems. We will use a reduced order approximate model with 20 states for MPC design, as proposed in [50], where balanced model reduction techniques were used. In order to predict beyond the expected settling time up to 300 minutes, the choice of sampling interval  $T = 4$  minutes will require a prediction horizon  $N = 75$ .

The choice of control horizon will have a large impact on the computational load on the controller algorithm. Since there is a large spread of slow and fast dynamics in the various transfer functions of the fractionator, the use of *blocking*, namely non-uniform intervals between control decisions seems appropriate here, as that allows control adjustments to be made throughout the predicted transient period without having too many decision variables in the optimization problem. Guided by precess step responses, we guess that it may be appropriate to keep  $u(k|k)$ ,  $u(k+1|k)$ ,  $u(k+3|k)$ ,  $u(k+7|k)$  and  $u(k+15|k)$

as the decision variables, so that the blocking pattern is as follows, each block being twice as long as the previous one:

$$\begin{aligned}
 &u(k|k) \\
 &u(k+1|k) = u(k+2|k) \\
 &u(k+3|k) = u(k+4|k) = u(k+5|k) = u(k+6|k) \\
 &u(k+7|k) = u(k+j|k) \quad \text{for } j = 8, 9, \dots, 14 \\
 &u(k+15|k) = u(k+j|k) \quad \text{for } j = 16, 17, \dots
 \end{aligned}$$

In this way we have only 15 decision variables (3 control inputs  $\times$  5 decision times).

Next we must choose weights for the cost function. Since the Bottoms Reflux Temperature ( $z_3$ ) has a zone objective only, the tracking error weight for it is 0. We assume that the variables have already been scaled such that equal errors on each of these are equally important. We will also keep the weights constant over the prediction horizon, except that we will not weight the Top End Point errors during the first seven steps, or the Side End Point errors during the first four steps, because of the long time delays in the transfer functions involved. The weights may be adjusted in order to change the trade-off between tracking the Top End Point and Side End Point compositions, and minimizing the Bottoms Reflux Duty. After several experiments we come up with

$$\begin{aligned}
 Q(j) &= \text{diag}(0, 0, 0, 1) \quad \text{for } j \leq 4 \\
 Q(j) &= \text{diag}(0, 10, 0, 1) \quad \text{for } 5 \leq j \leq 7 \\
 Q(j) &= \text{diag}(20, 10, 0, 1) \quad \text{for } j \geq 8.
 \end{aligned}$$

We impose the penalty weighting  $R(j) = I_3$  on control moves.

Figure 2.7 shows the response with these tuning parameters, when the set-points for  $z_1$  and  $z_2$  are both 0, the set-point for  $u_3$  is  $-0.32$ . Both the plant and the internal model had zero initial state vectors. The steady-state errors on  $z_1$  and  $z_2$  are  $+0.001$  and  $-0.001$ , respectively, which is within specification.

Next we examine the effect of a disturbance on the Intermediate Reflux Duty ( $d_m$ ), which is a measured disturbance, and can therefore be countered by feed-forward action. Figure 2.8 shows the fractionator initially at the same steady state as reached at the end of the run shown in Figure 2.7. A disturbance is expressed in steps, each step lasting 4 minutes:

$$d_m(k) = \begin{cases} 0 & \text{for } k \leq 12 \\ +0.5 & \text{for } 12 < k \leq 37 \\ 0 & \text{for } 37 < k \leq 62 \\ -0.5 & \text{for } 62 < k \leq 87 \\ 0 & \text{for } k > 87. \end{cases}$$

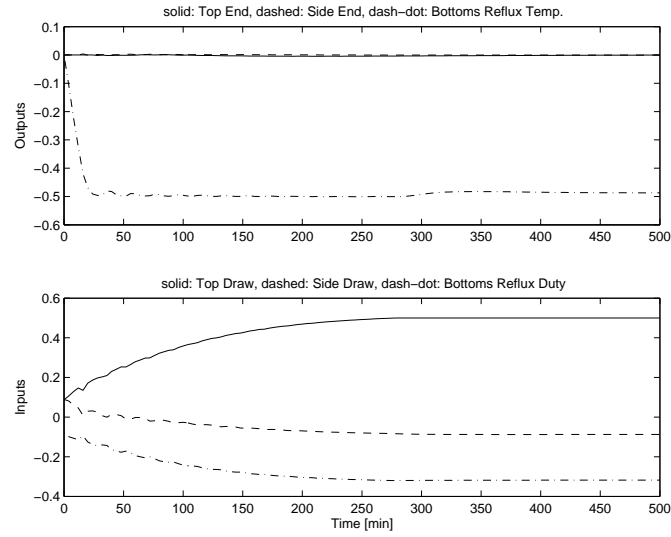


Figure 2.7: Setpoint tracking response simulated with linear MPC.

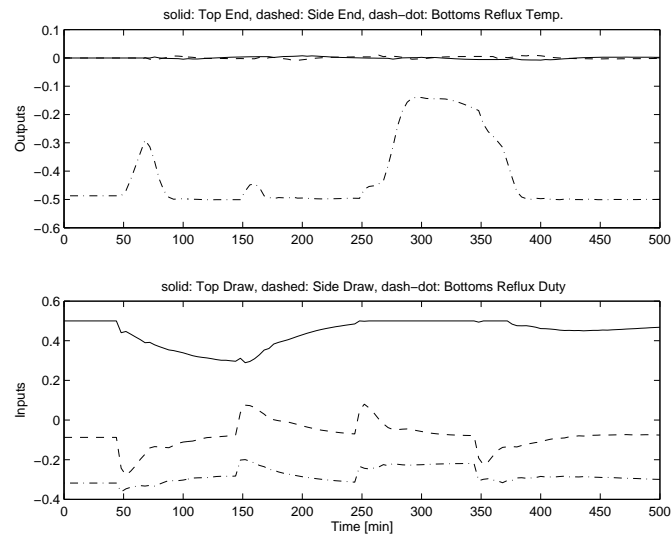


Figure 2.8: Response to a measured disturbance simulated with linear MPC.

Recall that an increase in the Intermediate Reflux Duty corresponds to less heat being taken away from the fractionator. Thus the initial disturbance of  $+0.5$  allows the Bottoms Reflux Duty ( $u_3$ ) to be reduced at first. This, however,

leads to the Bottoms Reflux Temperature ( $z_3$ ) falling below its constraint, so  $u_3$  increases again, this being compensated by a reduction of  $u_1$  away from its constraint;  $u_2$  follows a similar pattern to  $u_3$ . When the disturbance ceases, there is a transient lasting about 100 minutes, by the end of which all the variables have returned to their initial values. Then the disturbance  $d_m = -0.5$  arrives;  $u_1$  cannot increase now, since it is already at its upper constraint, so  $u_3$  has to increase (from  $-0.32$  to  $-0.22$ ) for as long as the disturbance lasts. Note that the Bottoms Reflux Temperature ( $z_3$ ) now moves away from its lower constraint, since it is a zone rather than a setpoint variable. Finally (for  $k > 87$ , namely after about 350 minutes) the disturbance ceases again, and all the variables return to their initial values.

#### 2.6.4 Concluding remarks

In the previous example it was shown that the performance of industrial model predictive control is limited by the large computational complexity of the control strategy. We may need to reduce the degrees of freedom in the optimization by precisely defining the points where the input signal may change. This prohibits the use of sampling times smaller than several minutes. Even for systems with relatively fast dynamics, so called stiff systems, large sampling times have to be used and thereby neglecting the fast dynamics. Therefore, with the current generation of MPC technology it is not possible to increase the sampling frequency and achieve good performance results.

New technology is needed to deal with high-performance model predictive control of large-scale, fast sampled systems. It is clear from the previous example that the on-line computational load of linear MPC is almost exclusively determined by a quadratic programming problem. Several optimization algorithms are available to solve a quadratic programming problem (see Appendix A). Two strategies are most widely used: active set methods and interior-point methods.

It is investigated in this thesis how the degrees of freedom in the optimization algorithms can be chosen carefully, such that a good trade-off between performance and computational complexity is obtained. A structured interior-point method (SIPM) will be proposed in the next chapter for high-performance model predictive control of large-scale stiff systems. The computational load can be then reduced by choosing effective optimization techniques.





## *Model Predictive Control for Nonlinear Processes*

---

3.1	Nonlinear models and predictive control	3.4	Structured interior-point method
3.2	Algorithm overview	3.5	Implementation of the structured interior-point algorithm
3.3	MPC algorithm for nonlinear systems	3.6	MPC application results

---

### **3.1 *Nonlinear models and predictive control***

Model predictive control (MPC), also referred to as receding horizon control, is a method that applies on-line optimization to a model of a system, with the aim of steering the system to a desired target state. In recent years, MPC has become a prominent advanced control technique, especially in the petrochemical process industry. However, for computational reasons, MPC applications largely have been limited to linear models; that is, those in which the dynamics of the system model are linear. Such models often do not capture the dynamics of the system adequately, especially in regions that are not close to the target state. In these cases, nonlinear models are necessary to describe accurately the behavior of physical systems. Different models may have static gain or dynamic type nonlinearities. From an algorithmic point of view, nonlinear model predictive control requires the repeated solution of nonlinear optimal control problems. At certain times during the control period, the state of the system is estimated, and an optimal control problem is solved over a finite time horizon (commencing at the present time), using this state estimate as the initial state. The control component at the current time is used as the input to the system. Algorithms for nonlinear optimal control, which are often specialized nonlinear programming algorithms, can therefore be used in the context of nonlinear model predictive control, with the additional imperatives that the problem must be solved in real time, and that good estimates of the solution may be available from the state and control profiles obtained at the

previous time-point. The linear MPC problem is well studied from an optimization standpoint. It gives rise to a sequence of optimal control problems with quadratic objectives and linear dynamics, which can be viewed as structured convex quadratic programming problems. These problems can be solved efficiently by algorithms that exploit the structure. For example, an interior-point method that uses a recursive relation to solve the linear systems at each iteration has been described by Rao, Wright, and Rawlings [75]. The nonlinear MPC problem has been less widely studied, and is a topic of recent interest.

Model Predictive Control (MPC), also known as moving or receding horizon control, has originated in industry as a real-time computer control algorithm to solve linear multi-variable problems that have constraints and time delays. MPC has received a great deal of attention and receives an ever growing interest for applications in industrial process control. Various MPC algorithms differ mainly in the type of model that is used to represent the process and its disturbances, as well as the cost functions to be minimized subject to constraints. During the last decade, many formulations have been developed for linear and nonlinear, stable and unstable plants [57, 66, 73, 78]. In MPC the controller predicts the behavior of a plant over a prediction horizon using the model and measurements, and determines a manipulated variable sequence that optimizes some open-loop performance objective over the horizon. This manipulated variable sequence is implemented until the next measurement becomes available. Then the optimization problem is solved again.

### 3.2 Algorithm overview

Due to computational limitations, nonlinear MPC technology has not yet been practiced at a large industrial scale. As an alternative to a fully nonlinear MPC, we introduce a different MPC algorithm that uses linear time-varying prediction models, obtained by applying a local linearization along the nominal input and state trajectory. In this way, the problematic setup of the standard NMPC approach is avoided, performance analysis may be simplified, and computational efficiency is significantly improved. Local linear approximation of the state equation was used to develop an optimal prediction of the future states. The output prediction was made linear with respect to the undecided control input moves, which allowed to reduce the MPC optimization to a quadratic programming problem (QP).

The block diagram of the proposed MPC algorithm for nonlinear systems is presented in Figure 3.1 giving the structure of the whole chapter. The process model is given in Section 3.3.1 and the problem itself is formulated in Section 3.3.3. Having selected the nominal input trajectory we construct output prediction via linearization of the process dynamics (Sections 3.3.4 and 3.3.5). Finally the MPC optimization problem is given in an algebraic form (see Section 3.3.6). We propose to use a structured interior-point method (SIPM) to

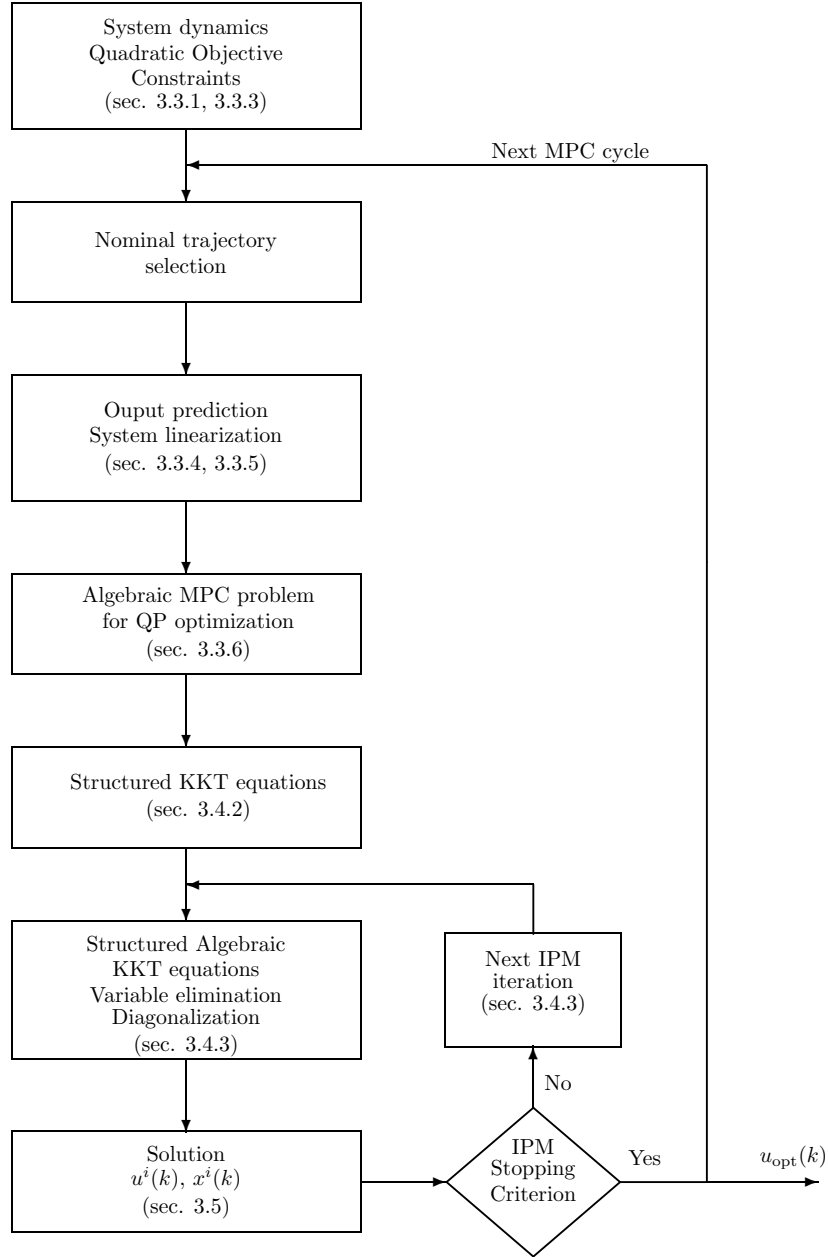


Figure 3.1: Algorithm overview.

reduce the computational complexity of the QP optimization. This optimization algorithm is given in Sections 3.4 and 3.5.

### 3.3 MPC algorithm for nonlinear systems

Nonlinear MPC optimizations tend to become too large to be solved in real-time. To reduce computational complexity, we propose to convert the NMPC optimization problem to either a linear program or a quadratic program (QP). In this paper we consider a nonlinear MPC scheme where the predicted future process behaviour is represented as a cumulative effect of a nonlinear prediction component and a component based on linear models defined along the predicted trajectory [22, 46, 82]. The first component constitutes a future output prediction using nonlinear simulation models, given past process inputs and measured disturbance signals. The second component uses linearized models for prediction of future process outputs as required for calculation of optimum future process inputs that bring the process behavior closest to the desired behavior. MPC controller solves online a constrained optimization problem and determines optimal control inputs over a fixed future time-horizon, based on the predicted future behaviour of the process, and based on the desired reference trajectory. The optimization programs (typically constrained QPs) tend to become too large to be solved in real-time when standard QP solvers are used. To reduce computational complexity we propose to solve QP problem using a structured interior-point method [75, 10]. The cost of this approach is linear in the horizon length, compared with the cubic growth for the standard approach. The effectiveness of the proposed NMPC algorithm will be demonstrated in the next chapter on two industrial chemical processes, namely a continuous stirred tank reactor and a stiff nonlinear batch reactor.

#### 3.3.1 Model

Consider the system equations of a nonlinear process model

$$\dot{x} = f(x, u, d) \quad (3.1)$$

$$y = g(x, d), \quad (3.2)$$

where  $x(t) \in \mathbb{R}^{n_x}$  is the state vector,  $u(t) \in \mathbb{R}^{n_u}$  denotes the input signal,  $y(t) \in \mathbb{R}^{n_y}$  is the measured output and  $d(t) \in \mathbb{R}^{n_d}$  represents unmeasured disturbance. For digital MPC controller design the control signal can be assumed to be constant between the sampling intervals  $t \in [kT, (k+1)T]$ . We can express a discrete version of the model (3.1) and (3.2) as follows:

$$x(k) = F_T(x(k-1), u(k-1), d(k-1)) \quad (3.3)$$

$$y(k) = g(x(k), d(k)), \quad (3.4)$$

where  $F_T(x(k-1), u(k-1), d(k-1))$  denotes the terminal state vector obtained by integrating (3.1) for one sample interval  $T$  with the initial condition  $x(k-1)$  and constant inputs  $u(k-1)$  and  $d(k-1)$ .

For the purpose of state estimation, it is common to express the unmeasured signal  $d$  as a stochastic process. Without loss of generality, we assume that  $d$  is generated through the following stochastic difference equation:

$$x_w(k) = A_w x_w(k-1) + B_w w(k-1) \quad (3.5)$$

$$d(k) = C_w x_w(k), \quad (3.6)$$

where  $w(k)$  is a discrete-time white noise with covariance  $R_w$ .

It is also possible that the measurements of  $y(k)$  are corrupted by measurement noise  $v(k)$  as follows:

$$\hat{y}(k) = g(x(k), d(k)) + v(k). \quad (3.7)$$

We assume that  $v(k)$  is white noise with covariance  $R_v$ .

Combining (3.3) and (3.4) with (3.5)–(3.7), we arrive at the following augmented model:

$$\begin{bmatrix} x(k) \\ x_w(k) \end{bmatrix} = \begin{bmatrix} F_T(x(k-1), u(k-1), C_w x_w(k-1)) \\ A_w x_w(k-1) \end{bmatrix} + \begin{bmatrix} 0 \\ B_w \end{bmatrix} w(k-1) \quad (3.8)$$

$$\hat{y}(k) = g(x(k), C_w x_w(k)) + v(k). \quad (3.9)$$

From this point on, our discussion will be based on the augmented form of the model.

### 3.3.2 State estimation

A straightforward extension of the optimal linear filter (“Kalman filter”) is the extended Kalman filter. The basic idea of EKF is to perform linearization at each time step to approximate the nonlinear system as a time-varying system affine in the variables to be estimated, and to apply the linear filtering theory to it.

Let us consider the model of (3.8) and (3.9). We will use the notation  $x(k|l)$  and  $x_w(k|l)$  to denote the optimal estimates (i.e., minimum variance estimates) for  $x(k)$  and  $x_w(k)$  based on the measurements up to time  $l$ . In probabilistic terms, they represent the conditional expectation of the Gaussian variables  $x(k)$  and  $x_w(k)$  with the conditions given by the measurements  $\hat{y}_1, \dots, \hat{y}_l$ . We may also express the confidence in these estimates through the conditional covariance matrix  $\Sigma_{k|l}$ , i.e.,

$$\Sigma_{k|l} = \mathbb{E} \left\{ \begin{bmatrix} x(k) - x(k|l) \\ x_w(k) - x_w(k|l) \end{bmatrix} \begin{bmatrix} x(k) - x(k|l) \\ x_w(k) - x_w(k|l) \end{bmatrix}^\top \right\}.$$

The problem of recursive state estimation can be viewed as computing the new estimates  $x(k|k)$  and  $x_w(k|k)$  and covariance matrix  $\Sigma_{k|k}$  based on the previous estimates  $x(k-1|k-1)$  and  $x_w(k-1|k-1)$ , their covariance matrix  $\Sigma_{k-1|k-1}$ , and new measurement  $\hat{y}(k)$ . It is instructive to view this as two sub-problems: “model prediction” and “measurement correction”.

### Model prediction

In the model prediction step of state estimation, the objective is to compute the estimates of the new states without using the new information of  $\hat{y}(k)$ , i.e. compute  $x(k|k-1)$ ,  $x_w(k|k-1)$  and  $\Sigma_{k|k-1}$  from  $x(k-1|k-1)$ ,  $x_w(k-1|k-1)$  and  $\Sigma_{k-1|k-1}$ .

This is in general a very difficult problem as the assumed normal distributions of the initial state variables are destroyed by propagation through nonlinear dynamics. Extended Kalman filter simplifies the problem significantly by making the following linear approximation of (3.8) with respect to  $x(k-1) = x(k-1|k-1)$  and  $x_w(k-1) = x_w(k-1|k-1)$ :

$$\begin{bmatrix} x(k) \\ x_w(k) \end{bmatrix} \approx \begin{bmatrix} F_T(x(k-1|k-1), u(k-1), C_w x_w(k-1|k-1)) \\ A_w x_w(k-1|k-1) \end{bmatrix} + \Phi_{k-1} \begin{bmatrix} x(k-1) - x(k-1|k-1) \\ x_w(k-1) - x_w(k-1|k-1) \end{bmatrix} + \Gamma_w w(k-1), \quad (3.10)$$

where

$$\Gamma_w = \begin{bmatrix} 0 \\ B_w \end{bmatrix}$$

and  $\Phi_k$  is calculated using the following formula:

$$\Phi_{k-1} = \begin{bmatrix} A_{k-1} & B_{k-1}^d C_w \\ 0 & A_w \end{bmatrix}$$

where  $A_{k-1}$  and  $B_{k-1}^d$  are obtained via zero-order-hold discretization of the following Jacobians:

$$\begin{aligned} \tilde{A}_{k-1} &= \left. \frac{\partial f}{\partial x} \right|_{x(k-1|k-1), u(k-1), C_w x_w(k-1|k-1)} \\ \tilde{B}_{k-1}^d &= \left. \frac{\partial f}{\partial d} \right|_{x(k-1|k-1), u(k-1), C_w x_w(k-1|k-1)}. \end{aligned}$$

For example,  $\tilde{A}_{k-1}$  is the Jacobian matrix for  $f(x, u, d)$  with respect to  $x$  evaluated at  $x = x(k-1|k-1)$ ,  $u = u(k-1)$  and  $d = C_w x_w(k-1|k-1)$ . We also assume that these Jacobian matrices remain constant throughout the time period between  $t = k-1$  and  $t = k$ . Note that  $x(k)$  and  $x_w(k)$  are Gaussian variables because of the linear approximation of (3.10).

Let  $e(k-1)$  and  $e_w(k-1)$  represent  $x(k-1) - x(k-1|k-1)$  and  $x_w(k-1) - x_w(k-1|k-1)$ , respectively. Then with  $e(0) = 0$  we have

$$\mathbb{E}e(k-1|k-1) = 0, \quad \mathbb{E}e_w(k-1|k-1) = 0, \quad \mathbb{E}w(k-1|k-1) = 0$$

since  $x(k-1|k-1)$  and  $x_w(k-1|k-1)$  represent the optimal estimates (conditional means of  $x(k-1)$  and  $x_w(k-1)$ ) and  $w(k-1)$  is zero-mean white noise, the effect of which does not appear in the measurements up to  $\hat{y}(k-1)$ .

Hence, for the approximate system (3.10),

$$\begin{bmatrix} x(k|k-1) \\ x_w(k|k-1) \end{bmatrix} = \begin{bmatrix} F_T(x(k-1|k-1), u(k-1), C_w x_w(k-1|k-1)) \\ A_w x_w(k-1|k-1) \end{bmatrix}.$$

In addition, since (3.10) can be rewritten as

$$\begin{aligned} \begin{bmatrix} x(k) \\ x_w(k) \end{bmatrix} &\approx \begin{bmatrix} x(k|k-1) \\ x_w(k|k-1) \end{bmatrix} \\ &+ \Phi_{k-1} \begin{bmatrix} x(k-1) - x(k-1|k-1) \\ x_w(k-1) - x_w(k-1|k-1) \end{bmatrix} + \Gamma_w w(k-1), \end{aligned}$$

$\Sigma_{k|k-1}$ , the covariance matrix for the error

$$\begin{bmatrix} x(k|k-1) - x(k) \\ x_w(k|k-1) - x_w(k) \end{bmatrix}$$

can be easily computed from  $\Sigma_{k-1|k-1}$  as follows:

$$\Sigma_{k|k-1} = \Phi_{k-1} \Sigma_{k-1|k-1} \Phi_{k-1}^\top + \Gamma_w R_w \Gamma_w^\top.$$

### Measurement correction

In the measurement correction stage of state estimation, the new information  $\hat{y}(k)$  is used to improve the state estimates. The problem can be formally stated as compute  $x(k|k)$ ,  $x_w(k|k)$  and  $\Sigma_{k|k}$  from  $x(k|k-1)$ ,  $x_w(k|k-1)$ ,  $\Sigma_{k|k-1}$  and measurement  $\hat{y}(k)$ . Because  $\hat{y}(k)$  is a nonlinear function of the states, the above is a nonlinear estimation problem that does not yield an analytical solution in general. In the EKF, the problem is again simplified by linearizing the output equation (3.9) at  $x(k|k-1)$  and  $x_w(k|k-1)$ :

$$\begin{aligned} \hat{y}(k) &\approx g(x(k|k-1), C_w x_w(k|k-1)) \\ &+ \Xi_k \begin{bmatrix} x(k) - x(k|k-1) \\ x_w(k) - x_w(k|k-1) \end{bmatrix} + v(k), \end{aligned} \quad (3.11)$$

where

$$\Xi_k = \begin{bmatrix} C_k & C_k^d C_w \end{bmatrix}$$



and  $C_k$  and  $C_k^d$  are Jacobian matrices defined as follows:

$$C_k = \frac{\partial g}{\partial x} \Big|_{x(k|k-1), C_w x_w(k|k-1)}$$

$$C_k^d = \frac{\partial g}{\partial d} \Big|_{x(k|k-1), C_w x_w(k|k-1)}.$$

Since we are given the means and covariances of the stochastic variables  $x(k) - x(k|k-1)$ ,  $x_w(k) - x_w(k|k-1)$ , and  $v(k)$ , the linear filtering theory can be applied to find the conditional means  $x(k|k)$  and  $x_w(k|k)$  with the measurement condition (3.11):

$$\begin{bmatrix} x(k|k) \\ x_w(k|k) \end{bmatrix} \approx \begin{bmatrix} x(k|k-1) \\ x_w(k|k-1) \end{bmatrix} + L_k(\hat{y}(k) - g(x(k|k-1), C_w x_w(k|k-1))),$$

where

$$L_k = \Sigma_{k|k-1} \Xi_k^\top (\Xi_k \Sigma_{k|k-1} \Xi_k^\top + R_v)^{-1}.$$

In addition, the conditional covariance matrix  $\Sigma_{k|k}$  expressing the confidence of the corrected state estimates is

$$\Sigma_{k|k} = (I - L_k \Xi_k) \Sigma_{k|k-1}.$$

### Implementation

In summary, under the assumption that the nonlinear system (3.8) and (3.9) is well approximated by the affine system of (3.10) and (3.11) obtained via local linearization, the following Kalman filter provides the optimal estimates:

Model Prediction:

$$\begin{bmatrix} x(k|k-1) \\ x_w(k|k-1) \end{bmatrix} = \begin{bmatrix} F_T(x(k-1|k-1), u(k-1), C_w x_w(k-1|k-1)) \\ A_w x_w(k-1|k-1) \end{bmatrix} \quad (3.12)$$

Measurement Correction:

$$\begin{bmatrix} x(k|k) \\ x_w(k|k) \end{bmatrix} \approx \begin{bmatrix} x(k|k-1) \\ x_w(k|k-1) \end{bmatrix} + L_k(\hat{y}(k) - g(x(k|k-1), C_w x_w(k|k-1))), \quad (3.13)$$

where

$$L_k = \Sigma_{k|k-1} \Xi_k^\top (\Xi_k \Sigma_{k|k-1} \Xi_k^\top + R_v)^{-1},$$

$$\Sigma_{k|k-1} = \Phi_{k-1} \Sigma_{k-1|k-1} \Phi_{k-1}^\top + \Gamma_w R_w \Gamma_w^\top,$$

$$\Sigma_{k|k} = (I - L_k \Xi_k) \Sigma_{k|k-1}.$$

Note that the model update equation (3.12) requires nonlinear integration of ODE (3.1) with known initial condition and constant inputs.

For control applications with relatively short sample intervals, the required computation time for (3.13) and control move computation may be comparable to the sampling time. Then,  $\hat{y}(k)$  may not be used for computing the estimate of  $x(k)$ , under closed-loop conditions. In this case, the EKF can be implemented as an estimator where the measurement correction step precedes the model prediction step.

### 3.3.3 MPC problem formulation

At time instant  $t_k := kT$  we consider an MPC problem which amounts to finding an optimal control sequence  $\{u(k+j)\}_{j=0}^{N-1}$  minimizing the quadratic objective function

$$J_k(x(k|k), u) = \sum_{j=1}^N \left[ \|y(k+j) - y_{\text{ref}}(k+j)\|_{Q(j)}^2 + \|u(k+j-1) - u_{\text{ref}}(k+j-1)\|_{R(j)}^2 \right] + x^\top(k+N)\bar{Q}x(k+N) \quad (3.14)$$

subject to the element-wise constraints

$$y_{\min} \leq y(k+j) \leq y_{\max} \quad (3.15)$$

$$u_{\min} \leq u(k+j) \leq u_{\max} \quad (3.16)$$

$$\Delta u_{\min} \leq \Delta u(k+j) \leq \Delta u_{\max} \quad (3.17)$$

for  $j = 0, \dots, N-1$ , where  $\Delta u(k) := u(k) - u(k-1)$  and  $N > 0$  is the prediction horizon. State constraints can be easily formulated using (3.15). Note that the objective function (3.14) can also use  $\Delta u(k+j)$  instead of  $u(k+j)$  itself. Here,  $\|x\|_Q^2$  will mean  $x^\top Qx$ . In this formulation it is assumed that a full state estimate is available at time  $t_k = kT$ . The MPC is trying to make inputs and outputs follow its reference trajectories  $u_{\text{ref}}(k+j)$  and  $y_{\text{ref}}(k+j)$ . The prediction horizon  $N$  is a design parameter and the tuning parameters include positive semi-definite weighting matrices  $Q(j) \in \mathbb{R}^{n_y \times n_y}$ ,  $\bar{Q} \in \mathbb{R}^{n_x \times n_x}$  and  $R(j) \in \mathbb{R}^{n_u \times n_u}$ . Note that the problem formulation can also use any convex objective function  $J_k(u)$ .

### 3.3.4 Approximation and prediction

In order to implement a predictive control algorithm, long-term prediction of the key states is required. Clearly, since the underlying system is nonlinear, the future states (and hence the outputs) are related to the current states and current/future inputs in a nonlinear fashion. This makes the problem of finding the optimal input sequence a complex nonlinear optimization problem.

We propose to make the relationship linear via local linearization, which is dual to the local linearization used for deriving the extended Kalman filter.

We define the one-step ahead prediction of the states as

$$\begin{bmatrix} x(k+1|k) \\ x_w(k+1|k) \end{bmatrix} = \begin{bmatrix} F_T(x(k|k), u(k), C_w x_w(k|k)) \\ A_w x_w(k|k) \end{bmatrix}, \quad (3.18)$$

where  $F_T(x(k|k), u(k), C_w x_w(k|k))$  denotes the terminal state vector obtained by integrating (3.1) for one sample interval  $T$  with the initial condition  $x(k|k)$  at time  $kT$  and constant input  $u(k)$  between  $kT$  and  $(k+1)T$ .

More generally we define multi-step prediction

$$x(k+j|k) := F_{jT}(x(k|k), \{u(k+i)\}_{i=0}^{j-1}, \{d(k+i)\}_{i=0}^{j-1}) = F_T(\dots(F_T(x(k|k), u(k), d(k)), \dots), u(k+j-1), d(k+j-1))$$

as  $j$  compositions of  $F_T$  to represent the terminal states obtained by integrating (3.1) for  $j$  sampling intervals with initial condition  $x(k|k)$  and piecewise constant input. Note from (3.18) that the state  $x(k+1|k)$  is related to the undecided manipulated variable  $u(k)$  through nonlinear integration. This makes the optimization required for the input move computation a nonlinear problem. To prevent this we further approximate the equation by linearizing  $F_T(x(k|k), u(k))$  at some nominal input value  $u_{\text{nom}}(k)$  (its choice will be explained later):

$$\begin{bmatrix} x(k+1|k) \\ x_w(k+1|k) \end{bmatrix} \approx \begin{bmatrix} F_T(x(k|k), u_{\text{nom}}(k), C_w x_w(k|k)) \\ A_w x_w(k|k) \end{bmatrix} + \begin{bmatrix} B_{k|k} \\ 0 \end{bmatrix} (u(k) - u_{\text{nom}}(k)), \quad (3.19)$$

where  $B_{k|k}$  is obtained via zero-order-hold discretization of one of the following Jacobians:

$$A_{k|k}^c = \frac{\partial f}{\partial x} \Big|_{x(k|k), u_{\text{nom}}(k), C_w x_w(k|k)}$$

$$B_{k|k}^c = \frac{\partial f}{\partial u} \Big|_{x(k|k), u_{\text{nom}}(k), C_w x_w(k|k)}.$$

*Remark 1* The Jacobians can be calculated numerically. Suppose that for a linearization point  $(x^*, u^*)$  and a small value  $\delta$  we define a state perturbation vector as  $x_{\text{per}}^* := \delta(1 + 10^{-3}|x^*|)$ . For example, the  $j$ -th column of the matrix  $A$  becomes

$$A(:, j) = \left[ f((x^*(1), \dots, x^*(j) + x_{\text{per}}^*(j), \dots, x^*(n_x))^{\top}, u^*) - f((x^*(1), \dots, x^*(j) - x_{\text{per}}^*(j), \dots, x^*(n_x))^{\top}, u^*) \right] / 2x_{\text{per}}^*(j). \quad (3.20)$$

The other Jacobians are calculated in a similar way. The procedure may also require further refinement of the matrices.

We can generalize the idea to develop multi-step predictions. Note from (3.18) that

$$\begin{bmatrix} x(k+2|k) \\ x_w(k+2|k) \end{bmatrix} = \begin{bmatrix} F_T(x(k+1|k), u(k+1), C_w x_w(k+1|k)) \\ A_w x_w(k+1|k) \end{bmatrix}, \quad (3.21)$$

where  $x(k+2|k)$  is related in a nonlinear fashion not only to  $u(k+1)$ , but also to  $u(k)$  appearing in expression (3.18) for  $x(k+1|k)$ . By appropriate linearization, we would like to derive an approximation that is linear with respect to the undecided inputs  $u(k)$  and  $u(k+1)$ . The linear relationship that approximates the local behavior can be obtained by linearizing the expression  $F_T(x(k+1|k), u(k+1), C_w x_w(k+1|k))$  with respect to  $x(k+1|k) = F_T(x(k|k), u_{\text{nom}}(k), C_w x_w(k|k))$  and  $u(k+1) = u_{\text{nom}}(k+1)$  as follows

$$\begin{aligned} F_T(x(k+1|k), u(k+1), C_w x_w(k+1|k)) &\approx F_{2T}(x(k|k), u_{\text{nom}}, d) \\ &+ A_{k+1|k}(x(k+1|k) - F_T(x(k|k), u_{\text{nom}}(k), C_w x_w(k|k))) \\ &+ B_{k+1|k}(u(k+1) - u_{\text{nom}}(k+1)), \end{aligned} \quad (3.22)$$

where  $A_{k+1|k}$  and  $B_{k+1|k}$  are obtained via zero-order-hold discretization of the following Jacobians:

$$\begin{aligned} A_{k+1|k}^c &= \frac{\partial f}{\partial x} \bigg|_{F_T(x(k|k), u_{\text{nom}}(k), C_w x_w(k|k)), u_{\text{nom}}(k+1), C_w x_w(k+1|k)} \\ B_{k+1|k}^c &= \frac{\partial f}{\partial u} \bigg|_{F_T(x(k|k), u_{\text{nom}}(k), C_w x_w(k|k)), u_{\text{nom}}(k+1), C_w x_w(k+1|k)}. \end{aligned}$$

Note that  $u_{\text{nom}}$  is a piecewise constant input taking, for instance, values of  $\{u_{\text{nom}}(k), u_{\text{nom}}(k+1)\}$  at the time interval  $[k, k+2]$ . Signal  $d$  is a piecewise constant input taking the value of  $C_w(A_w)^i x_w(k|k)$  during the time interval  $[k+i, k+i+1]$ .

Note from (3.19) that

$$x(k+1|k) - F_T(x(k|k), u_{\text{nom}}(k), C_w x_w(k|k)) \approx B_{k|k}(u(k) - u_{\text{nom}}(k)).$$

Substitute the affine approximation (3.22) into (3.21) to obtain

$$\begin{aligned} \begin{bmatrix} x(k+2|k) \\ x_w(k+2|k) \end{bmatrix} &\approx \begin{bmatrix} F_{2T}(x(k|k), u_{\text{nom}}, d) \\ (A_w)^2 x_w(k|k) \end{bmatrix} \\ &+ \begin{bmatrix} A_{k+1|k} B_{k|k} & B_{k+1|k} \\ 0 & 0 \end{bmatrix} \begin{bmatrix} u(k) - u_{\text{nom}}(k) \\ u(k+1) - u_{\text{nom}}(k+1) \end{bmatrix}. \end{aligned}$$

Carrying out the same derivations for  $x(k+j|k), j = 1, \dots, N$ , we obtain

$$\begin{aligned} \begin{bmatrix} x(k+j|k) \\ x_w(k+j|k) \end{bmatrix} &\approx \begin{bmatrix} F_{jT}(x(k|k), u_{\text{nom}}, d) \\ (A_w)^j x_w(k|k) \end{bmatrix} \\ &+ \begin{bmatrix} \prod_{i=1}^{j-1} A_{k+i|k} B_{k|k} & \prod_{i=2}^{j-1} A_{k+i|k} B_{k+1|k} & \cdots & B_{k+j-1|k} \\ 0 & 0 & \cdots & 0 \end{bmatrix} \\ &\times \begin{bmatrix} u(k) - u_{\text{nom}}(k) \\ u(k+1) - u_{\text{nom}}(k+1) \\ \vdots \\ u(k+j-1) - u_{\text{nom}}(k+j-1) \end{bmatrix}, \end{aligned} \quad (3.23)$$

where  $A_{k+i|k}$  and  $B_{k+i|k}$  are obtained via zero-order-hold discretization of the following Jacobians:

$$\begin{aligned} A_{k+i|k}^c &= \frac{\partial f}{\partial x} \bigg|_{F_{iT}(x(k|k), u_{\text{nom}}, d), u_{\text{nom}}(k+i), C_w x_w(k+i|k)} \\ B_{k+i|k}^c &= \frac{\partial f}{\partial u} \bigg|_{F_{iT}(x(k|k), u_{\text{nom}}, d), u_{\text{nom}}(k+i), C_w x_w(k+i|k)}. \end{aligned} \quad (3.24)$$

Note that the expression (3.23) requires integration of the ODE (3.1) for  $j$  sample time steps into the future and computation of the matrices for each of the  $j$  sample times (i.e.,  $A_{k+i|k}$  and  $B_{k+i|k}$  for  $i = 0, \dots, j-1$ ).

To reduce the computational complexity, the matrices  $A_{k+i|k}$  and  $B_{k+i|k}$  can be kept constant at the initial values of  $A_{k|k}$  and  $B_{k|k}$  throughout the prediction horizon. To avoid the computational complexity, we will adopt this simplification. Hence, (3.23) simplifies to

$$\begin{aligned} \begin{bmatrix} x(k+j|k) \\ x_w(k+j|k) \end{bmatrix} &\approx \begin{bmatrix} F_{jT}(x(k|k), u_{\text{nom}}, d) \\ (A_w)^j x_w(k|k) \end{bmatrix} \\ &+ \begin{bmatrix} A_{k|k}^{j-1} B_{k|k} & A_{k|k}^{j-2} B_{k|k} & \cdots & B_{k|k} \\ 0 & 0 & \cdots & 0 \end{bmatrix} \\ &\times \begin{bmatrix} u(k) - u_{\text{nom}}(k) \\ u(k+1) - u_{\text{nom}}(k+1) \\ \vdots \\ u(k+j-1) - u_{\text{nom}}(k+j-1) \end{bmatrix}. \end{aligned} \quad (3.25)$$

Similar techniques as described above can be found in, for example [46, 80, 89].

In order to develop a prediction for the output that is linear with respect to the undecided input moves, we linearize equation (3.2) with respect to  $x(k|k)$ :

$$y(k) \approx g(x(k|k), C_w x_w(k|k)) + \begin{bmatrix} C_{k|k} & C_{k|k}^d C_w \end{bmatrix} \begin{bmatrix} x(k) - x(k|k) \\ x_w(k) - x_w(k|k) \end{bmatrix},$$

where  $C_{k|k}$  and  $C_{k|k}^d$  are Jacobian matrices defined as

$$C_{k|k} = \left. \frac{\partial g}{\partial x} \right|_{x(k|k), C_w x_w(k|k)}$$

$$C_{k|k}^d = \left. \frac{\partial g}{\partial d} \right|_{x(k|k), C_w x_w(k|k)}.$$

Carrying out the same idea,

$$\begin{aligned} y(k+j|k) &= g(x(k+j|k), C_w x_w(k+j|k)) \\ &\approx g(x(k|k), C_w x_w(k|k)) \\ &+ \begin{bmatrix} C_{k|k} & C_{k|k}^d C_w \end{bmatrix} \begin{bmatrix} x(k+j|k) - x(k|k) \\ x_w(k+j|k) - x_w(k|k) \end{bmatrix} \end{aligned} \quad (3.26)$$

we obtain the output prediction.

### 3.3.5 Optimizing control input in output prediction

Define the so-called optimizing control input

$$\delta u(k+j|k) = u(k+j|k) - u_{\text{nom}}(k+j|k)$$

as a difference between the actual input signal and the selected nominal trajectory (see Figure 3.2). Every MPC cycle will try to refine the nominal trajectory

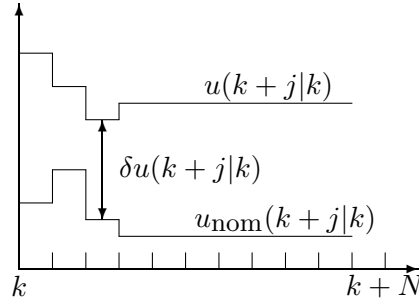


Figure 3.2: Optimizing control input.

by means of  $\delta u(k+j|k)$  to obtain the optimal input for the given nonlinear model.

*Remark 2* The local linearization in (3.23) makes sense only when the computed inputs  $\{u(k+i)\}_{i=0}^{j-1}$  do not deviate much from  $u_{\text{nom}}$ . For nonlinear models this can be achieved by finding a nominal trajectory  $u_{\text{nom}}(k+j|k)$  which is as close

as possible to the optimal strategy  $u_{\text{opt}}(k+j|k)$ . A simple but effective choice is to start with  $u_{\text{nom}}(k+j|k) = u_{\text{opt}}(k+j|k-1)$ , i.e. the optimal control policy derived at the previous sample. Naturally, the input trajectories computed in the subsequent optimization is likely to be a better approximation of the actual future input sequence [22, 82].

Combine (3.26) with the optimal multi-step prediction equation (3.25) for  $x(k+j|k)$  to obtain the complete output prediction

$$\begin{aligned}
 & \begin{bmatrix} y(k+1|k) \\ y(k+2|k) \\ \vdots \\ y(k+N|k) \end{bmatrix} = \\
 & = \begin{bmatrix} I \\ I \\ \vdots \\ I \end{bmatrix} (g(x(k|k), C_w x_w(k|k)) - C_{k|k} x(k|k) - C_{k|k}^d C_w x_w(k|k)) \\
 & + \begin{bmatrix} C_{k|k} F_T(x(k|k), u_{\text{nom}}, d) \\ C_{k|k} F_{2T}(x(k|k), u_{\text{nom}}, d) \\ \vdots \\ C_{k|k} F_{NT}(x(k|k), u_{\text{nom}}, d) \end{bmatrix} + \begin{bmatrix} C_{k|k}^d C_w A_w \\ C_{k|k}^d C_w (A_w)^2 \\ \vdots \\ C_{k|k}^d C_w (A_w)^N \end{bmatrix} x_w(k|k) \\
 & + \begin{bmatrix} g_1 & 0 & \cdots & 0 \\ g_2 & g_1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ g_N & g_{N-1} & \cdots & g_1 \end{bmatrix} \begin{bmatrix} \delta u(k) \\ \delta u(k+1) \\ \vdots \\ \delta u(k+N-1) \end{bmatrix},
 \end{aligned} \tag{3.27}$$

where  $g_j = C_{k|k} A_{k|k}^{j-1} B_{k|k}$ ,  $j = 1, \dots, N$  represent the Markov parameters. Note that  $F_{NT}(x(k|k), u_{\text{nom}}, d)$  can be computed recursively since

$$\begin{aligned}
 & F_{NT}(x(k|k), \{u_{\text{nom}}(k+i)\}_{i=0}^{N-1}, \{d(k+i)\}_{i=0}^{N-1}) = \\
 & F_T(F_{(N-1)T}(x(k|k), \{u_{\text{nom}}(k+i)\}_{i=0}^{N-2}, \{d(k+i)\}_{i=0}^{N-2}), \\
 & u_{\text{nom}}(k+N-1), d(k+N-1)).
 \end{aligned}$$

Note also that the first term of the right-hand side of (3.27) drops out if the output vector consists of linear combinations of the state (i.e.,  $y(k) = C_{k|k} x(k)$ ). In order to keep the notation simple, we will denote (3.27) in the matrix notation as

$$Y = Y_{\text{nom}} + G_y \delta U,$$

where  $Y_{\text{nom}}$  is a vector notation for the first three components in the right hand side of (3.27), and it can be computed from the state estimate  $x(k|k)$  by performing an integration of the nonlinear ODE. Matrix  $G_y$  must be recomputed at each time step based on the updated Jacobian matrices.

### 3.3.6 Algebraic problem formulation

Note that a simple relationship exists between the control actions  $\Delta u$  and  $\delta u$ :

$$\begin{aligned}\Delta u(k) &= u(k) - u(k-1) \\ &= u_{\text{nom}}(k) + \delta u(k) - u(k-1) \\ \Delta u(k+1) &= u(k+1) - u(k) \\ &= u_{\text{nom}}(k+1) + \delta u(k+1) \\ &\quad - u_{\text{nom}}(k) - \delta u(k) \\ &\text{etc.}\end{aligned}$$

Then

$$\begin{bmatrix} \Delta u(k) \\ \Delta u(k+1) \\ \vdots \\ \Delta u(k+N-2) \\ \Delta u(k+N-1) \end{bmatrix} = E \begin{bmatrix} \delta u(k) \\ \delta u(k+1) \\ \vdots \\ \delta u(k+N-2) \\ \delta u(k+N-1) \end{bmatrix} + F,$$

with

$$E = \begin{bmatrix} I & 0 & \cdots & 0 & 0 \\ -I & I & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & I & 0 \\ 0 & 0 & \cdots & -I & I \end{bmatrix}, F = EU_{\text{nom}} - \begin{bmatrix} u(k-1) \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix},$$

where

$$U_{\text{nom}} = (u_{\text{nom}}^\top(k), u_{\text{nom}}^\top(k+1), \dots, u_{\text{nom}}^\top(k+N-2), u_{\text{nom}}^\top(k+N-1))^\top.$$

The latter relationship will be used to formalize constraints on input increments. Once  $U_{\text{nom}}$  has been specified and using (3.25), the objective function (3.14) becomes a quadratic form in  $\delta U$ :

$$\begin{aligned}J(\delta U) &= \\ & (Y_{\text{nom}} + G_y \delta U - Y_{\text{ref}})^\top \mathcal{Q} (Y_{\text{nom}} + G_y \delta U - Y_{\text{ref}}) \\ & + (U_{\text{nom}} + \delta U - U_{\text{ref}})^\top \mathcal{R} (U_{\text{nom}} + \delta U - U_{\text{ref}}) \\ & + (F_{NT}(x(k|k), u_{\text{nom}}, d) + G_{x(N)} \delta U)^\top \bar{Q} (F_{NT}(x(k|k), u_{\text{nom}}, d) + G_{x(N)} \delta U),\end{aligned}$$

where  $G_y$  is introduced in (3.27) and the state prediction defines

$$G_{x(N)} = \begin{bmatrix} A_{k|k}^{N-1} B_{k|k} & A_{k|k}^{N-2} B_{k|k} & \cdots & B_{k|k} \end{bmatrix}.$$

The block diagonal matrices are constructed as  $\mathcal{Q} = \text{diag}(Q(1), \dots, Q(N)) \in \mathbb{R}^{Nn_y \times Nn_y}$  and  $\mathcal{R} = \text{diag}(R(1), \dots, R(N)) \in \mathbb{R}^{Nn_u \times Nn_u}$ .



The end point weighting  $\bar{Q}$  can be obtained for example by solving the discrete time algebraic Riccati equation. This choice is motivated by recent developments in infinite horizon MPC or the LQR problem. This choice of the end point weighting gives the minimal addition to the quadratic objective function if the horizon length is extended to infinity. The solution to the Riccati equation uses a local linearization of the system at the end of the prediction horizon and the weighting matrices  $Q(N)$  and  $R(N)$ :

$$\begin{aligned}\bar{Q} &= A_{k+N|k}^\top \bar{Q} A_{k+N|k} \\ &+ A_{k+N|k}^\top \bar{Q} B_{k+N|k} (B_{k+N|k}^\top \bar{Q} B_{k+N|k} + R(N)) B_{k+N|k}^\top \bar{Q} A_{k+N|k} \\ &+ C_{k+N|k}^\top Q(N) C_{k+N|k}.\end{aligned}$$

The objective function can easily be transformed into the standard quadratic cost index

$$J(\delta U) = \frac{1}{2} \delta U^\top H \delta U + f^\top \delta U + c, \quad (3.28)$$

where

$$H = 2(G_y^\top Q G_y + G_{x(N)}^\top \bar{Q} G_{x(N)} + \mathcal{R}) \quad (3.29)$$

$$\begin{aligned}f &= 2G_y^\top Q(Y_{\text{nom}} - Y_{\text{ref}}) + 2\mathcal{R}(U_{\text{nom}} - U_{\text{ref}}) \\ &+ 2G_{x(N)}^\top \bar{Q} F_{NT}(x(k|k), u_{\text{nom}}, d)\end{aligned} \quad (3.30)$$

$$\begin{aligned}c &= (Y_{\text{nom}} - Y_{\text{ref}})^\top Q(Y_{\text{nom}} - Y_{\text{ref}}) \\ &+ (U_{\text{nom}} - U_{\text{ref}})^\top \mathcal{R}(U_{\text{nom}} - U_{\text{ref}}) \\ &+ F_{NT}^\top(x(k|k), u_{\text{nom}}, d) \bar{Q} F_{NT}(x(k|k), u_{\text{nom}}, d).\end{aligned} \quad (3.31)$$

Then the optimization problem amounts to solving a quadratic programming problem with the objective function (3.28) with (3.29)–(3.31) subject to the constraints (3.15)–(3.17). The constraints can be transformed into the following form

$$\begin{bmatrix} E \\ -E \\ I \\ -I \\ G_y \\ -G_y \end{bmatrix} \delta U \leq \begin{bmatrix} b_1 - F \\ -b_2 + F \\ d_1 - U_{\text{nom}} \\ -d_2 + U_{\text{nom}} \\ y_1 - Y_{\text{nom}} \\ -y_2 + Y_{\text{nom}} \end{bmatrix}, \quad (3.32)$$

where  $b_1$ ,  $b_2$ ,  $d_1$  and  $d_2$  are of dimension  $Nn_u$  and consist of  $N$  copies of  $\Delta u_{\text{max}}$ ,  $\Delta u_{\text{min}}$ ,  $u_{\text{max}}$  and  $u_{\text{min}}$  respectively. In the same way, vectors  $y_1$  and  $y_2$  are of dimension  $Nn_y$  and consist of  $N$  copies of  $y_{\text{max}}$ ,  $y_{\text{min}}$ .

### 3.3.7 Constraint types and softening

A problem may occur when an optimizer is faced with an infeasible problem. This can happen because an unexpectedly large disturbance has occurred or

the real plant behaves differently from the model. There is really no way in which the plant can be kept within the specified constraints in that case, except moving on the boundary of the constraint. Although there will be no such circumstances for our design examples, it is still important to have a strategy for dealing with infeasibility to prevent the online MPC optimizer from producing an infeasible solution. One systematic way is to allow the constraints to be crossed occasionally, if necessary, rather than keeping them as “hard” boundaries which can never be crossed. There is an important distinction between input and output constraints. Usually input constraints can never be softened, because of the actuators having limited ranges of action.

One way to soften output constraints is to add new variables, so-called “slack-variables”, which are defined in such a way that they are non-zero only if the constraints are violated and they are very heavily penalized in the cost function. We propose to penalize 1-norm of the constraint violations, then the quadratic cost index (3.28) and the constraints (3.32) can be modified as

$$\frac{1}{2}\delta U^\top H \delta U + f^\top \delta U + W^\top \epsilon \quad (3.33)$$

subject to

$$\begin{bmatrix} E \\ -E \\ I \\ -I \\ G_y \\ -G_y \end{bmatrix} \delta U \leq \begin{bmatrix} b_1 - F \\ -b_2 + F \\ d_1 - U_{\text{nom}} \\ -d_2 + U_{\text{nom}} \\ y_1 - Y_{\text{nom}} \\ -y_2 + Y_{\text{nom}} \end{bmatrix} + \epsilon, \quad \epsilon \geq 0, \quad (3.34)$$

where  $\epsilon$  is a nonnegative vector of dimension equal to the number of constraints (3.32),  $W$  is a penalizing column vector. This is still a QP problem, though with a larger number of variables. Some of the constraints can be retained as hard constraints if the corresponding elements in  $W$  are assumed to tend to infinity. Thus, penalizing the 1-norm of constraint violations requires a separate slack variable for every constraint, at every point of the prediction horizon for which the constraint is enforced. With elements in  $W$  large enough, the choice for 1-norm penalty on constraint violations gives an “exact penalty” method, which means that constraint violations will not occur unless there is no feasible solution to the original “hard” problem. That is, the same solution will be obtained as with an original formulation, if a feasible solution exists.

### 3.4 Structured interior-point method

The constrained quadratic optimization problem (3.14) can be solved in various ways. First of all, one can use the model equations to eliminate the states, thus reducing the number of variables in the optimization, while, however,

making the problem formulation dense. If the states are not eliminated, the optimization variables consist of the inputs and the states over the horizon, but the optimization problem is sparse and structured and these properties can be exploited to reduce the cost to solve the QP. Therefore it depends on the number of inputs, states and the length of the horizon which method requires minimal computational effort, which is an important issue for on-line implementations of MPC for large, stiff systems.

If the states are eliminated, one can either use a standard active set method or a standard interior-point method [67, 69]. Active set methods (ASM) try iteratively to find the set of constraints that are active at the optimum. To obtain that goal they solve an equality constrained problem at each time step to determine a new search direction. This means that in each iteration a dense set of equations in  $Nn_u$  variables has to be solved. Moreover, the total number of iterations will increase with the number of active constraints since typically in each iteration only one or some constraints will become active. This leads to the fact that ASMs, though still widely used as the standard methods for solving the QPs in the MPC algorithms, may lead to very computationally expensive procedures if applied on large MPC problems with many constraints.

Interior-point methods (IPM) try to solve the nonlinear set of Karush-Kuhn-Tucker equations iteratively by making linear approximations to this set. An advantage over ASMs is that interior-point methods can efficiently make use of sparsity in the constraints, and will therefore often be faster for solving MPC related problems, since bound or rate of change constraints on the control inputs give rise to sparse constraint matrices. Also the number of iterations to reach a point close to the optimum is typically independent of the number of constraints and will be smaller as for ASMs.

Solving QPs with these standard methods typically requires a computational time that increases with the *third power* of the number of variables  $Nn_u$ .

For problems with large horizons, methods that do not eliminate the states, but that exploit the structure of the given MPC problem have been developed of which the cost varies linearly with the horizon length. In this chapter we discuss such a method, based on [75, 10] that will be suited for the problems discussed in this thesis.

### 3.4.1 States as optimization variables in MPC

We will consider a general linear MPC optimization problem with quadratic cost function based on (3.14) with inclusion of a terminal state cost,

$$\min_{\{u(j)\}_{j=0}^{N-1}} \sum_{j=0}^{N-1} [(y_{\text{nom}}(j) + y(j) - y_{\text{ref}}(j))^{\top} Q_y(j) (y_{\text{nom}}(j) + y(j) - y_{\text{ref}}(j)) + (u_{\text{nom}}(j) + \delta u(j) - u_{\text{ref}}(j))^{\top} R(j) (u_{\text{nom}}(j) + \delta u(j) - u_{\text{ref}}(j))] + x^{\top}(N) \bar{Q} x(N) \quad (3.35)$$

subject to

$$C_u(j)\delta u(j) + C_x(j)x(j) \leq C_c(j)$$

for  $j = 0, \dots, N$ , where  $C_u(N) = 0$ . Next

$$\begin{aligned} x(j+1) &= A_j x(j) + B_j \delta u(j), \\ y(j) &= C_j x(j) \end{aligned}$$

for  $j = 0, \dots, N-1$ , where  $x(j)$  are the states of the linearized system here. Matrices  $C_u(j)$ ,  $C_x(j)$  and  $C_c(j)$  are defined as real matrix valued mappings on  $\mathbb{T} \subseteq \mathbb{Z}$ . To simplify the notation we omitted index  $k$  in the above formulation and will use just  $u(j)$  instead of the optimizing control input  $\delta u(j)$  from now on. We also introduce the following signals

$$\begin{aligned} y_{\text{sig}}(j) &= y_{\text{ref}}(j) - y_{\text{nom}}(j) \\ u_{\text{sig}}(j) &= u_{\text{ref}}(j) - u_{\text{nom}}(j), \end{aligned}$$

so that the quadratic objective function becomes

$$\begin{aligned} \min_{\{u(j)\}_{j=0}^{N-1}} & \sum_{j=0}^{N-1} [(y(j) - y_{\text{sig}}(j))^{\top} Q_y(j)(y(j) - y_{\text{sig}}(j)) + \\ & (u(j) - u_{\text{sig}}(j))^{\top} R(j)(u(j) - u_{\text{sig}}(j))] + x^{\top}(N)\bar{Q}x(N) \end{aligned}$$

Note that in this formulation we may use different weighting  $Q_y(j)$  and  $R(j)$  at all points on the horizon. Moreover, the algorithm will also allow easily to introduce multiple linear models, as derived in (3.23). Indeed, with constant linear models, the Hessian of the optimization problem with eliminated states remains constant, but when using multiple linear models, this Hessian will vary, thus introducing an extra cost for the optimization to build this Hessian for each time step.

As mentioned before, if this state elimination is not carried out, the structure given by the dynamics of the plant (i.e. states and inputs at time step  $j$  only interact with states and inputs of the nearest time steps) will reflect in the optimization problem and thus also in the Karush-Kuhn-Tucker(KKT) equations that can be used to solve the QP using an interior-point method.

Notice that in the optimization problem formulation, bound constraints on inputs and states are included. Input bounds for instance, are obtained by taking (for  $j = 0, \dots, N-1$ ),

$$C_u(j) = \begin{bmatrix} I \\ -I \end{bmatrix}, \quad C_x(j) = \begin{bmatrix} 0 \\ 0 \end{bmatrix}.$$

However all types of constraints interconnecting variables of different time steps, like rate of change constraints, are not included in the above formulation. Also notice that constraints on the states can be defined if necessary. For sake of simplicity these cases are omitted here, although methods exist to include these types of constraints, see e.g. [75].

### 3.4.2 Structured KKT equations

To obtain the KKT equation in the inputs and the states, the outputs  $y(j)$  are eliminated using  $y(j) = C_j x(j)$ . Defining  $Q(j) = C^\top Q_y(j)C$ ,  $Q_{\text{ref}}(j) = C^\top Q_y(j)$  and the slacks variables  $t_j$  for the inequalities as

$$t_j := C_c(j) - C_u(j)u(j) - C_x(j)x(j), \quad j = 0, \dots, N,$$

the problem can be written as

$$\begin{aligned} \min_{\{u(j)\}_{j=0}^{N-1}} & \sum_{j=0}^{N-1} (x^\top(j)Q(j)x(j) + u^\top(j)R(j)u(j)) - \\ & \sum_{j=0}^{N-1} (y_{\text{sig}}^\top(j)Q_{\text{ref}}^\top(j)x(j) + u_{\text{sig}}^\top(j)R(j)u(j)) + x^\top(N)\bar{Q}x(N) \end{aligned} \quad (3.36)$$

subject to

$$C_u(j)u(j) + C_x(j)x(j) + t_j = C_c(j), \quad j = 0, \dots, N$$

and

$$\begin{aligned} x(j+1) &= A_j x(j) + B_j u(j), \quad j = 0, \dots, N-1 \\ t_j &\geq 0, \quad j = 0, \dots, N. \end{aligned}$$

Notice that the future values of  $y_{\text{sig}}$  and  $u_{\text{sig}}$  are already specified at this point and the optimal control sequence will not depend casually on these signals. Introducing Langrange multipliers  $p_j$  for the state space model equality constraints and  $\lambda_j$  for the inequality constraints, and taking into account the given state at the current time step  $x(0)$ , the following set of KKT equations is obtained, where constant parts are written on the right hand side

$$\begin{aligned} R(j)u(j) + B_j^\top p_j + C_u^\top(j)\lambda_j &= R(j)u_{\text{sig}}(j) & j = 0, \dots, N-1 \\ Q(j)x(j) + A_j^\top p_j - p_{j-1} + C_x^\top(j)\lambda_j &= Q_{\text{ref}}(j)y_{\text{sig}}(j) & j = 1, \dots, N-1 \\ \bar{Q}x(N) - p_{N-1} + C_x^\top(N)\lambda_N &= 0 \\ -x(j+1) + A_j x(j) + B_j u(j) &= 0 & j = 0, \dots, N-1 \\ C_u(j)u(j) + C_x(j)x(j) + t_j &= C_c(j) & j = 0, \dots, N-1 \\ t_j^\top \lambda_j &= 0 & j = 0, \dots, N \\ \lambda_j &\geq 0 & j = 0, \dots, N \\ t_j &\geq 0 & j = 0, \dots, N \end{aligned} \quad (3.37)$$

This set of equations will be referred to as the structured Karush-Kuhn-Tucker equations or SKKT equations. The next step is to solve these SKKT equations by applying an interior point method that exploits the given structure.

### 3.4.3 Interior-point method based on SKKT equations

An interior point method to solve the above KKT equations will be an iterative method where a step direction is obtained by solving a set of linear algebraic KKT equations in each iteration. This set of equations will exploit the structure of the given KKT equations by ordering them according to the time variable  $j$ . These equations will be referred to as structured algebraic KKT (SAKKT) equations. Introducing a vector of variables

$$w_j = \text{col}(u(j), \lambda_j, t_j, p_j, x(j+1))$$

for  $j = 0, \dots, N-1$ , define vector  $w$  of optimization variables in the following way

$$w = \text{col}(w_0, \dots, w_{N-1}, \lambda_N, t_N) \quad (3.38)$$

so that the resulting system has a particular block-wise diagonal structure. The variables are grouped according to the prediction stage, so that the resulting matrix (2.23) is sparse. The equations are rearranged such that the overall matrix is overlapping block diagonal. This structure enables the efficient solution of the primal-dual interior-point step.

Let  $w^0$  be a chosen starting point and let  $w^i$  be the result of the  $i$ -th iteration. Define  $\Delta w^i$  for  $i = 1, \dots, n_i$  with  $n_i$  the number of iterations as the step direction to be calculated in the  $i$ -th iteration to obtain the next iterate as

$$w^i = w^{i-1} + \alpha^i \Delta w^i, \quad (3.39)$$

where  $\alpha^i$  is the step length. This direction is, according to [69], calculated by solving the linear set of SAKKT equations

$$W^i \Delta w^i = r^i, \quad (3.40)$$

where  $W^i$  is obtained from the linearization of (3.37) and thus a function of  $w^{i-1}$ , and  $r^i$  is the residual at the  $i$ -th iteration.

Since the algebraic KKT equations will be ordered according to  $w^i$ , i.e. ordered according to  $j$ , the first part will contain the algebraic KKT equations for  $j = 0$ . From linearizing the equations in (3.37) for time step  $j = 0$  the following set of equations is obtained for  $j = 0$  at the  $i$ -th iteration

$$\begin{bmatrix} R(0) & C_u^\top(0) & 0 & B_0^\top & 0 \\ C_u(0) & 0 & I_{n_c} & 0 & 0 \\ 0 & T_0^{i-1} & \Lambda_0^{i-1} & 0 & 0 \\ B_0 & 0 & 0 & 0 & -I_{n_x} \end{bmatrix} \begin{bmatrix} \Delta u^i(0) \\ \Delta \lambda_0^i \\ \Delta t_0^i \\ \Delta p_0^i \\ \Delta x^i(1) \end{bmatrix} = r_0^i, \quad (3.41)$$

where

$$\begin{aligned} T_0^{i-1} &= \text{diag}(t_0^{i-1}) \\ \Lambda_0^{i-1} &= \text{diag}(\lambda_0^{i-1}) \end{aligned} \quad (3.42)$$

and

$$r_0^i = \begin{bmatrix} r_{u(0)}^i \\ r_{\lambda_0}^i \\ r_{t_0}^i \\ r_{p_0}^i \end{bmatrix}. \quad (3.43)$$

These residuals will be discussed later on.

For  $j = 1, \dots, N - 1$ , the linearization gives

$$\begin{bmatrix} -I_{n_x} & Q(j) & 0 & C_x^\top(j) & 0 & A_j^\top & 0 \\ 0 & 0 & R(j) & C_u^\top(j) & 0 & B_j^\top & 0 \\ 0 & C_x(j) & C_u(j) & 0 & I_{n_c} & 0 & 0 \\ 0 & 0 & 0 & T_j^{i-1} & \Lambda_j^{i-1} & 0 & 0 \\ 0 & A_j & B_j & 0 & 0 & 0 & -I_{n_x} \end{bmatrix} \begin{bmatrix} \Delta p_{j-1}^i \\ \Delta x^i(j) \\ \Delta u^i(j) \\ \Delta \lambda_j^i \\ \Delta t_j^i \\ \Delta p_j^i \\ \Delta x^i(j+1) \end{bmatrix} = r_j^i \quad (3.44)$$

with, analogous to the above definitions for  $j = 0$ ,

$$\begin{aligned} T_j^{i-1} &= \text{diag}(t_j^{i-1}) \\ \Lambda_j^{i-1} &= \text{diag}(\lambda_j^{i-1}) \end{aligned} \quad (3.45)$$

and

$$r_j^i = \begin{bmatrix} r_{x(j)}^i \\ r_{u(j)}^i \\ r_{\lambda_j}^i \\ r_{t_j}^i \\ r_{p_j}^i \end{bmatrix}. \quad (3.46)$$

Finally, for  $j = N$  we obtain

$$\begin{bmatrix} -I_{n_x} & \bar{Q} & C_x^\top(N) & 0 \\ 0 & C_x(N) & 0 & I_{n_c} \\ 0 & 0 & T_N^{i-1} & \Lambda_N^{i-1} \end{bmatrix} \begin{bmatrix} \Delta p_{N-1}^i \\ \Delta x^i(N) \\ \Delta \lambda_N^i \\ \Delta t_N^i \end{bmatrix} = r_N^i \quad (3.47)$$

with

$$\begin{aligned} T_N^{i-1} &= \text{diag}(t_N^{i-1}) \\ \Lambda_N^{i-1} &= \text{diag}(\lambda_N^{i-1}) \end{aligned} \quad (3.48)$$

and

$$r_N^i = \begin{bmatrix} r_{x(N)}^i \\ r_{\lambda_N}^i \\ r_{t_N}^i \end{bmatrix}. \quad (3.49)$$

Notice that the structure shows itself by the fact that in the SAKKT equations for a given time instant  $j$ , there is only a very small overlap with other time

instants in the sense that the step directions for the previous Lagrange multiplier of the state space equations and for the next state,  $\Delta p_{j-1}^i$  and  $\Delta x^i(j+1)$  respectively, appear in the  $j$ -th block.

This structure also shows if all the SAKKT equations are written down. This entire set of linear SAKKT equations to be solved in each interior-point iteration, is given by (3.40), where in the left hand side  $W^i$  is equal to

$$\begin{bmatrix} R(0) & C_u^\top(0) & 0 & B_0^\top & 0 \\ C_u(0) & 0 & I_{n_c} & 0 & 0 \\ 0 & T_0^{i-1} & \Lambda_0^{i-1} & 0 & 0 \\ B_0 & 0 & 0 & 0 & -I_{n_x} \\ & & & -I_{n_x} & Q(j) & 0 & C_x^\top(j) & 0 & A_j^\top & 0 \\ & & & 0 & 0 & R(j) & C_u^\top(j) & 0 & B_j^\top & 0 \\ & & & 0 & C_x(j) & C_u(j) & 0 & I_{n_c} & 0 & 0 \\ & & & 0 & 0 & 0 & T_j^{i-1} & \Lambda_j^{i-1} & 0 & 0 \\ & & & 0 & A_j & B_j & 0 & 0 & 0 & -I_{n_x} \\ & & & & & & & \ddots & & \\ & & & & & & & -I_{n_x} & \bar{Q} & C_x^\top(N) & 0 \\ & & & & & & & 0 & C_x(N) & 0 & I_{n_c} \\ & & & & & & & 0 & 0 & T_N^{i-1} & \Lambda_N^{i-1} \end{bmatrix}$$

It can be seen that this matrix is a banded matrix, consisting of overlapping blocks.

Before trying to solve these equations, let us perform some row operations in the different blocks to eliminate some variables. These eliminations are performed for the blocks defined in (3.44), but where appropriate they will also be applied on the blocks defined in (3.41) and (3.47).

First, and analogous to what is done in classical interior point methods, the step directions belonging to the slack variables of the inequality constraints  $t_j$  are eliminated. From the equation of the inequality slacks

$$T_j^{i-1} \Delta \lambda_j^i + \Lambda_j^{i-1} \Delta t_j^i = r_{t_j}^i$$

we have that, similar to (2.29), for all  $j = 0, \dots, N$

$$\Delta t_j^i = (\Lambda_j^{i-1})^{-1} r_{t_j}^i - (\Lambda_j^{i-1})^{-1} T_j^{i-1} \Delta \lambda_j^i.$$

Elimination of  $\Delta t_j^i$  gives

$$\begin{bmatrix} -I_{n_x} & Q(j) & 0 & C_x^\top(j) & A_j^\top & 0 \\ 0 & 0 & R(j) & C_u^\top(j) & B_j^\top & 0 \\ 0 & C_x(j) & C_u(j) & -(\Lambda_j^{i-1})^{-1} T_j^{i-1} & 0 & 0 \\ 0 & A_j & B_j & 0 & 0 & -I_{n_x} \end{bmatrix} \begin{bmatrix} \Delta p_{j-1}^i \\ \Delta x^i(j) \\ \Delta u^i(j) \\ \Delta \lambda_j^i \\ \Delta p_j^i \\ \Delta x^i(j+1) \end{bmatrix} =$$



$$= \begin{bmatrix} r_{x(j)}^i \\ r_{u(j)}^i \\ r_{\lambda_j}^i - (\Lambda_j^{i-1})^{-1} r_{t_j}^i \\ r_{p_j}^i \end{bmatrix}. \quad (3.50)$$

and the SAKKT equation for  $\lambda_j, j = 0, \dots, N$  allows, from the third row of (3.50)

$$C_x(j)\Delta x^i(j) + C_u(j)\Delta u^i(j) - (\Lambda_j^{i-1})^{-1}T_j^{i-1}\Delta \lambda_j^i = r_{\lambda_j}^i - (\Lambda_j^{i-1})^{-1}r_{t_j}^i$$

to obtain

$$\Delta \lambda_j^i = S_j^{i-1}(-r_{\lambda_j}^i + (\Lambda_j^{i-1})^{-1}r_{t_j}^i + C_x(j)\Delta x^i(j) + C_u(j)\Delta u^i(j)),$$

where

$$S_j^{i-1} = (T_j^{i-1})^{-1}\Lambda_j^{i-1} = \begin{bmatrix} \frac{\lambda_j^{i-1}(1)}{t_j^{i-1}(1)} & & & \\ & \frac{\lambda_j^{i-1}(2)}{t_j^{i-1}(2)} & & \\ & & \ddots & \\ & & & \frac{\lambda_j^{i-1}(n_c)}{t_j^{i-1}(n_c)} \end{bmatrix}. \quad (3.51)$$

Note that  $n_c$  was representing the number of inequality constraints, as defined in Section 3.4.1. We are about to introduce some extra variables before we continue with the elimination.

Introducing

$$\begin{aligned} \tilde{R}(j) &= R(j) + C_u^\top(j)S_j^{i-1}C_u(j) & j = 0, \dots, N-1 \\ \tilde{Q}(j) &= Q(j) + C_x^\top(j)S_j^{i-1}C_x(j) & j = 1, \dots, N-1 \\ \tilde{Q}(N) &= \bar{Q} + C_x^\top(N)S_j^{i-1}C_x(N) \\ \tilde{M}(j) &= C_u^\top(j)S_j^{i-1}C_x(j) & j = 1, \dots, N-1 \\ \tilde{r}_{u(j)}^i &= r_{u(j)}^i + C_u^\top(j)S_j^{i-1}(r_{\lambda_j}^i - (\Lambda_j^{i-1})^{-1}r_{t_j}^i) \\ &= r_{u(j)}^i + C_u^\top(j)S_j^{i-1}r_{\lambda_j}^i - C_u^\top(j)(T_j^{i-1})^{-1}r_{t_j}^i & j = 0, \dots, N-1 \\ \tilde{r}_{x(j)}^i &= r_{x(j)}^i + C_x^\top(j)S_j^{i-1}(r_{\lambda_j}^i - (\Lambda_j^{i-1})^{-1}r_{t_j}^i) \\ &= r_{x(j)}^i + C_x^\top(j)S_j^{i-1}r_{\lambda_j}^i - C_x^\top(j)(T_j^{i-1})^{-1}r_{t_j}^i & j = 1, \dots, N \end{aligned} \quad (3.52)$$

one can eliminate  $\Delta \lambda_j^i$  from the expressions for  $\Delta u^i(j)$  and  $\Delta x^i(j)$  obtaining

$$\begin{aligned} j = 0 & : \tilde{R}(0)\Delta u^i(0) + B_0^\top \Delta p_0^i = \tilde{r}_{u(0)}^i \\ j = 1, \dots, N-1 & : \tilde{R}(j)\Delta u^i(j) + \tilde{M}(j)\Delta x^i(j) + B_j^\top \Delta p_j^i = \tilde{r}_{u(j)}^i \\ & \quad \tilde{Q}(j)\Delta x^i(j) + \tilde{M}^\top(j)\Delta u^i(j) + A_j^\top \Delta p_j^i - \Delta p_j^i = \tilde{r}_{x(j)}^i \\ j = N & : \tilde{Q}(N)\Delta x^i(N) - \Delta p_N^i = \tilde{r}_{x(N)}^i \end{aligned} \quad (3.53)$$

The set of equations that remains to be solved is now given by

$$\tilde{W}^i \Delta \tilde{w}^i = \tilde{r}^i,$$

where

$$\tilde{W}^i = \begin{bmatrix} \tilde{R}(0) & B_0^\top & 0 \\ B_0 & 0 & -I_{n_x} \\ & -I_{n_x} & \tilde{Q}(1) & \tilde{M}^\top(1) & A_j^\top & 0 \\ & 0 & \tilde{M}(1) & \tilde{R}(1) & B_j^\top & 0 \\ & 0 & A_j & B_j & 0 & -I_{n_x} \\ & & & & \ddots & \\ & & & & -I_{n_x} & \tilde{Q}(N) \end{bmatrix},$$

$$\Delta \tilde{w}^i = \begin{bmatrix} \Delta u^i(0) \\ \Delta p_0^i \\ \Delta x^i(1) \\ \Delta u^i(1) \\ \vdots \\ \Delta u^i(N-1) \\ \Delta p_{N-1}^i \\ \Delta x^i(N) \end{bmatrix}, \tilde{r}^i = \begin{bmatrix} \tilde{r}_{u(0)}^i \\ \tilde{r}_{p_0}^i \\ \tilde{r}_{x(1)}^i \\ \tilde{r}_{u(1)}^i \\ \vdots \\ \tilde{r}_{u(N-1)}^i \\ \tilde{r}_{p_{N-1}}^i \\ \tilde{r}_{x(N)}^i \end{bmatrix}.$$

The computational cost to solve this set of linear equations will vary linearly with the horizon length if matrix  $\tilde{W}^i$  is made block diagonal. Several approaches similar to Schur complement technique can be applied in this case. A Riccati recursion scheme can also be used to solve this structured set of equations by getting rid of the overlap. For this purpose, introduce the auxiliary variables  $\Pi_j$  and  $\pi_j$  giving the relation between  $\Delta p_{j-1}^i$  and  $\Delta x^i(j)$ ,

$$\Delta p_{j-1}^i = \Pi_j \Delta x^i(j) - \pi_j, \quad j = 1, \dots, N \quad (3.54)$$

How to deduce these auxiliary variables will become clear later on. Next, replace the overlap equation in  $x(j)$ , given by

$$-\Delta p_{j-1}^i + \tilde{Q}(j) \Delta x^i(j) + A_j^\top \Delta p_j^i + \tilde{M}^\top(j) \Delta u^i(j) = \tilde{r}_{x(j)}^i \quad (3.55)$$

in each block with this new equation. This way one obtains a set of equations that is easily solved as follows. First introduce the variables

$$\begin{aligned} \hat{R}(j) &= \tilde{R}(j) + B_j^\top \Pi_{j+1} B_j & j &= 0, \dots, N-1 \\ \hat{Q}(j) &= \tilde{Q}(j) + A_j^\top \Pi_{j+1} A_j & j &= 1, \dots, N-1 \\ \hat{M}(j) &= \tilde{M}(j) + B_j^\top \Pi_{j+1} A_j & j &= 1, \dots, N-1 \\ \hat{r}_{u(j)}^i &= \tilde{r}_{u(j)}^i + B_j^\top \Pi_{j+1} r_{p_j}^i + B^\top \pi_{j+1} & j &= 0, \dots, N-1 \\ \hat{r}_{x(j)}^i &= \tilde{r}_{x(j)}^i + A_j^\top \Pi_{j+1} r_{p_j}^i + A^\top \pi_{j+1} & j &= 1, \dots, N-1 \end{aligned} \quad (3.56)$$

The 0-th block has become

$$\tilde{W}_0^i \Delta \tilde{w}_0^i = \begin{bmatrix} \tilde{R}(0) & B_0^\top & 0 \\ B_0 & 0 & -I_{n_x} \\ 0 & -I_{n_x} & \Pi_1 \end{bmatrix} \begin{bmatrix} \Delta u^i(0) \\ \Delta p_0^i \\ \Delta x^i(1) \end{bmatrix} = \begin{bmatrix} \tilde{r}_{u(0)}^i \\ \tilde{r}_{p_0}^i \\ \pi_1 \end{bmatrix}, \quad (3.57)$$

which can be solved for  $\Delta u^i(0)$ ,  $\Delta p_0^i$  and  $\Delta x^i(1)$  easily as follows

$$\begin{aligned} \tilde{R}(0) \Delta u^i(0) &= \tilde{r}_{u(0)}^i - B_0^\top \Delta p_0^i \\ &= \tilde{r}_{u(0)}^i - B_0^\top (\Pi_1 \Delta x^i(1) - \pi_1) \\ &= \tilde{r}_{u(0)}^i - B_0^\top (\Pi_1 (B_0 \Delta u^i(0) - \tilde{r}_{p_0}^i) - \pi_1), \end{aligned}$$

which allows to write the solutions as

$$\begin{aligned} \Delta u^i(0) &= \hat{R}^{-1}(0) \hat{r}_{u(0)}^i \\ \Delta x^i(1) &= B_0 \Delta u^i(0) - \tilde{r}_{p_0}^i \\ \Delta p_0^i &= \Pi_1 \Delta x^i(1) - \pi_1. \end{aligned} \quad (3.58)$$

Having determined  $\Delta u^i(0)$ ,  $\Delta p_0^i$  and  $\Delta x^i(1)$ , the next step is to deduce a recursive scheme to find  $\Delta u^i(j)$ ,  $\Delta p_j^i$  and  $\Delta x^i(j+1)$  for  $j = 1, \dots, N$ , given the solution for  $\Delta u^i(j-1)$ ,  $\Delta p_{j-1}^i$  and  $\Delta x^i(j)$ .

If  $\Delta x^i(j)$  is given, then the  $j$ -block can be written as

$$\begin{bmatrix} \tilde{M}(j) & \tilde{R}(j) & B_j^\top & 0 \\ A_j & B_j & 0 & -I_{n_x} \\ 0 & 0 & -I_{n_x} & \Pi_{j+1} \end{bmatrix} \begin{bmatrix} \Delta x^i(j) \\ \Delta u^i(j) \\ \Delta p_j^i \\ \Delta x^i(j+1) \end{bmatrix} = \begin{bmatrix} \tilde{r}_{u(j)}^i \\ \tilde{r}_{p_j}^i \\ \pi_{j+1} \end{bmatrix}, \quad (3.59)$$

which can be easily transformed into

$$\begin{bmatrix} \tilde{R}(j) & B_j^\top & 0 \\ B_j & 0 & -I_{n_x} \\ 0 & -I_{n_x} & \Pi_{j+1} \end{bmatrix} \begin{bmatrix} \Delta u^i(j) \\ \Delta p_j^i \\ \Delta x^i(j+1) \end{bmatrix} = \begin{bmatrix} \tilde{r}_{u(j)}^i - \tilde{M}(j) \Delta x^i(j) \\ \tilde{r}_{p_j}^i - A_j \Delta x^i(j) \\ \pi_{j+1} \end{bmatrix} \quad (3.60)$$

and the solution is obtained analogous to that of (3.57) as

$$\begin{aligned} \Delta u^i(j) &= \hat{R}^{-1}(j) (\hat{r}_{u(j)}^i - \tilde{M}(j) \Delta x^i(j)) \\ \Delta x^i(j+1) &= B_j \Delta u^i(j) - \tilde{r}_{p_j}^i + A_j \Delta x^i(j) \\ \Delta p_j^i &= \Pi_{j+1} \Delta x^i(j+1) - \pi_{j+1}. \end{aligned} \quad (3.61)$$

The recursive scheme for determining the step directions is now complete.

The only thing that remains, is to find the auxiliary variables. To that purpose, the equations that were omitted are used as starting point. The omitted equation of the final block reads

$$-\Delta p_{N-1}^i + \tilde{Q}(N) \Delta x^i(N) = \tilde{r}_{x(N)}^i \quad (3.62)$$

Comparing with the equation (3.54), it is clear that

$$\Pi_N = \tilde{Q}(N), \quad \pi_N = \tilde{r}_{x(N)}^i. \quad (3.63)$$

We solve the system by starting at this last stage and working backwards. The omitted equation of the  $(N-1)$ -th block reads

$$-\Delta p_{N-2}^i + \tilde{Q}(N-1)\Delta x^i(N-1) + A_{N-1}^\top \Delta p_{N-1}^i + \tilde{M}(N-1)\Delta u^i(N-1) = \tilde{r}_{x(N-1)}^i. \quad (3.64)$$

To obtain  $\Pi_{N-1}$  and  $\pi_{N-1}$  from this equation, one should eliminate  $\Delta p_{N-1}^i$ . This can be done using (3.54) and the equations (3.61) for  $j = N-1$ ,

$$\begin{aligned} \Delta p_{N-1}^i &= \Pi_N \Delta x^i(N) - \pi_N \\ &= \Pi_N (A_{N-1} \Delta x^i(N-1) + B_{N-1} \Delta u^i(N-1) + \tilde{r}_{p_{N-1}}^i) - \pi_N \\ &= \Pi_N A_{N-1} \Delta x^i(N-1) \\ &\quad + \Pi_N B_{N-1} \hat{R}^{-1}(N-1) (\hat{r}_{u(N-1)}^i - \tilde{M}(N-1) \Delta x^i(N-1)) \\ &\quad + \Pi_N \tilde{r}_{p_{N-1}}^i - \pi_N. \end{aligned} \quad (3.65)$$

Equation (3.64) can now be transformed into

$$\begin{aligned} -\Delta p_{N-2}^i &+ (\hat{Q}(N-1) - A_{N-1}^\top \Pi_N B_{N-1} \hat{R}^{-1}(N-1) \hat{M}(N-1)) \Delta x^i(N-1) \\ &= \hat{r}_{x(N-1)}^i - A_{N-1}^\top \Pi_N B_{N-1} \hat{R}^{-1}(N-1) \hat{r}_{u(N-1)}^i \end{aligned} \quad (3.66)$$

and thus  $\Pi_{N-1}$  and  $\pi_{N-1}$  are obtained. The reasoning can be repeated for all  $j = 1, \dots, N-1$ , leading to the following recursive definition of the auxiliary variables

$$\begin{aligned} \Pi_j &= \hat{Q}(j) - A_j^\top \Pi_{j+1} B_j \hat{R}^{-1}(j) B_j^\top \Pi_{j+1} A \\ \pi_j &= \hat{r}_{x(j)}^i - A_j^\top \Pi_{j+1} B_j \hat{R}^{-1}(j) \hat{r}_{u(j)}^i. \end{aligned} \quad (3.67)$$

We are now able to solve the systems (3.57) and (3.60) by applying the recursion scheme and updating  $\Pi_j$  and  $\pi_j$  until we obtain their respective values at  $j = 1$ .

### 3.5 Implementation of the structured interior-point algorithm

We are about to formulate the complete structured interior-point algorithm to be used in MPC for nonlinear processes. The algorithm will be further tested on complex systems exhibiting strong nonlinearities and stiffness. A very common IPM technique is the primal-dual Mehrotra's prediction-corrector algorithm. As it was already defined before in (3.39), primal-dual interior-point methods solve the quadratic program by iterating from an initial guess to the optimal one. Different ways exist to calculate these iterates [60, 99, 69] inside the IPM.

### 3.5.1 IPM and calculation of the residuals

In the previous section a solution was found to the SAKKT equations to be solved in each iteration of an interior-point method for problem (3.37). This means a global IPM can now be deduced. Different choices of standard interior-point methods can be used, mainly differing in the way the steps in the primal-dual space are obtained, which is done by defining the residuals (see, for example [69, 67]). Here we adopt the method presented in Section 2.4.7.

This leads to solving the SAKKT equations where the residuals for the predictor step, denoted by  ${}^pr^i$ , can be written as

$$\begin{aligned}
 {}^pr_{u(j)}^i &= R(j)u_{\text{sig}}(j) - R(j)u^{i-1}(j) + B_j^\top p_j^{i-1} + C_u^\top(j)\lambda_j^{i-1}, j = 0, \dots, N-1 \\
 {}^pr_{p_0}^i &= A_0x(0) + B_0u^{i-1}(0) + x^{i-1}(1) \\
 {}^pr_{p_j}^i &= A_jx^{i-1}(j) + B_ju^{i-1}(j) + x^{i-1}(j+1), \quad j = 1, \dots, N-1 \\
 {}^pr_{x(j)}^i &= Q_{\text{ref}}(j)y_{\text{sig}}(j) - Q(j)x^{i-1}(j) - A_j^\top p_j^{i-1} + p_{j-1}^{i-1} - C_x^\top(j)\lambda_j^{i-1} \\
 {}^pr_{x(N)}^i &= \bar{Q}x^{i-1}(N) + p_{N-1}^{i-1} - C_x^\top(N)\lambda_N^{i-1} \\
 {}^pr_{\lambda_0}^i &= C_c(0) - C_x(0)x(0) - C_u(0)u^{i-1}(0) - t_0^{i-1} \\
 {}^pr_{\lambda_j}^i &= C_c(j) - C_x(j)x^{i-1}(j) - C_u(j)u^{i-1}(j) - t_j^{i-1} \\
 {}^pr_{\lambda_N}^i &= C_c(N) - C_x(N)x^{i-1}(N) - t_N^{i-1} \\
 {}^pr_{t_j}^i &= -T_j^{i-1}\lambda_j^{i-1}, \quad j = 0, \dots, N
 \end{aligned} \tag{3.68}$$

The centering-corrector step is defined by putting all residuals  ${}^{cc}r^i$  to zero, except those of the slack equations, which become

$${}^{cc}r_{t_j}^i = -(\Delta^p t_j^i)^\top \Delta^p \lambda_j^i - \sigma^i \mu^i \mathbf{1} \tag{3.69}$$

for  $j = 0, \dots, N$ , where  $\Delta^p w$  is the step direction solution obtained from the predictor step.

Once the step direction is obtained, the step length can be calculated analogous to standard IPMs and similar to (2.30) as

$$\alpha = \max(\max(-\frac{\Delta \lambda_j^i}{\lambda_j^i}, -\frac{\Delta t_j^i}{t_j^i})/(1-\delta), 1), \tag{3.70}$$

where  $\delta$  is a small number, typically 0.05, and the inner maximum is taken over all  $j = 1, \dots, n_{\text{in}}$ .

### 3.5.2 The overall structured interior-point algorithm

The overall IPM that will be used now consists of the following steps.

#### Structured Interior-Point Algorithm for MPC:

1. Calculate the residuals  ${}^pr^i$  using (3.68).
2. Calculate  $\Pi_N$  and  $\pi_N$  using (3.63).

3. For  $j = N - 1 : 1$ ,
  - 3.1 Calculate  $\Pi_j$  and  $\pi_j$  using (3.67).
  - 3.2 Calculate  $\hat{R}(j), \hat{Q}(j), \hat{M}(j), \hat{r}_{u(j)}^i$  and  $\hat{r}_{x(j)}^i$  using (3.56) and (3.52).
4. For  $j = 0 : N$ ,
  - 4.1 Calculate all step directions  $\Delta^p w^i$  using (3.58) and (3.61).
5. Calculate  $\mu^i$  and  $\sigma^i$ .
6. Calculate the residuals  $^{cc}r^i$  using (3.69).
7. Repeat steps 2-4 for the centering step to determine  $\Delta^{cc}w^i$ .
8. Calculate the step length  $\alpha$ .
9. Calculate a new iterate  $w^i$ .

It was shown that in many cases the structured method can solve MPC problem much faster for longer prediction horizons than standard QP solvers [87, 10]. In the next chapter the effectiveness of the proposed SIPM based MPC algorithm will be demonstrated on several industrial chemical processes, such as a continuous stirred tank reactor and a stiff nonlinear batch reactor.

## 3.6 MPC application results

### 3.6.1 Evaporation process

We come back to the design example introduced in the previous chapter. In the the evaporation process the level  $L_2$  is kept at the value of 1m. The setpoint for  $X_2$  is ramped down linearly from 25% to 15% over a period of 20 minutes, and the operating pressure  $P_2$  is simultaneously ramped up from 50.5 kPa to 70 kPa. These specifications determined the reference trajectory  $Y_{\text{ref}}$ . Every time step the MPC controller obtained a new linear model, which was used for the whole prediction horizon. The tuning parameters include prediction horizon  $N = 30$  and the following weighting matrices:

$$Q = \begin{bmatrix} 200 & 0 & 0 \\ 0 & 50 & 0 \\ 0 & 0 & 50 \end{bmatrix}, \quad R = \begin{bmatrix} 10 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0.1 \end{bmatrix}.$$

Different QP solvers gave good performance results, where the reference trajectories were followed quite closely. Figure 3.3 shows the controlled outputs after using the structured interior-point based MPC on the evaporation process.

Figure 3.4 gives the comparison of the maximal time required to solve one MPC problem using ASM, Mosek IPM [3] and structured IPM with different

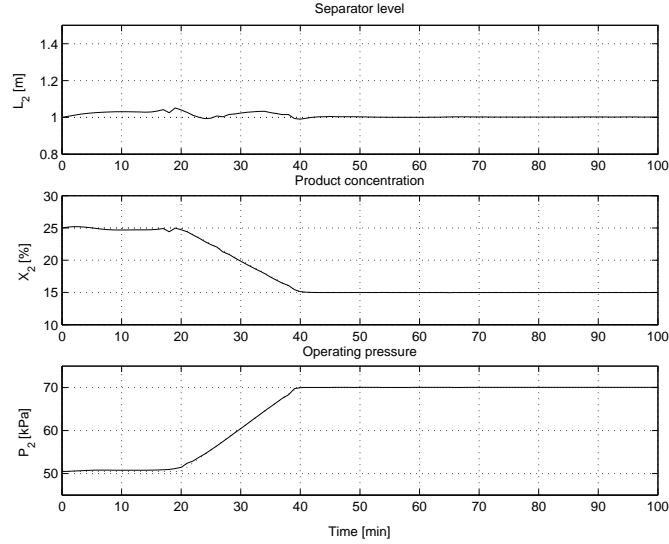


Figure 3.3: Outputs simulated with SIPM based MPC.

number of variables ( $Nn_u$ ). We see that although Mosek is much faster than ASM, these two standard QPs exhibit the behavior related to the third power of the number of variables.

At the same time we also noticed that it took the structured IPM in the MPC controller approximately 5 seconds to solve each optimization problem. The tests also showed that the time increased linearly with the horizon length. For horizons  $N > 25$  the structured IPM became faster than MATLAB's ASM, and so it could be implemented online with longer horizons, if required, considering the sampling time  $T = 1$  minute for this process. The structured IPM algorithm also became faster than Mosek for  $N > 160$ . Thus the evaporation process is an example which can prove the efficiency of the structured IPM for longer prediction horizons.

### 3.6.2 High-purity binary distillation column

The MPC objective is to bring the controlled outputs  $X_d$  and  $X_b$  to the set-points of 0.99 and 0.01, respectively (see Figure 3.5). The process was sampled with  $T = 2$  minutes. The prediction horizon was chosen to be  $N = 20$  and the weighting matrices are

$$Q = 10^{-4} \begin{bmatrix} 10 & 0 \\ 0 & 20 \end{bmatrix}, \quad R = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

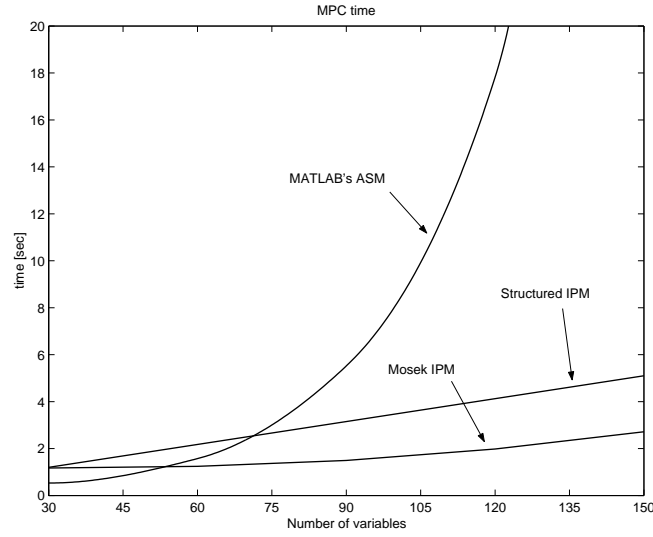


Figure 3.4: MPC computational time comparison for the evaporation process.

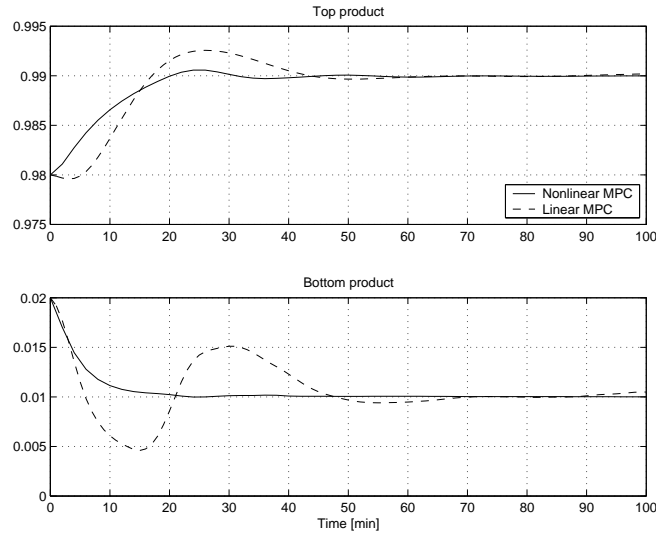


Figure 3.5: Top and bottom product compositions simulated with SIPM based MPC (solid) and linear MPC (dashed).

As it was shown in Section 2.6.2, it could be difficult to make both products  $X_d$  and  $X_b$  purer. Figure 3.6 illustrates such a rather successful attempt,



although it really becomes quite difficult for the MPC controller to reach purer product compositions.

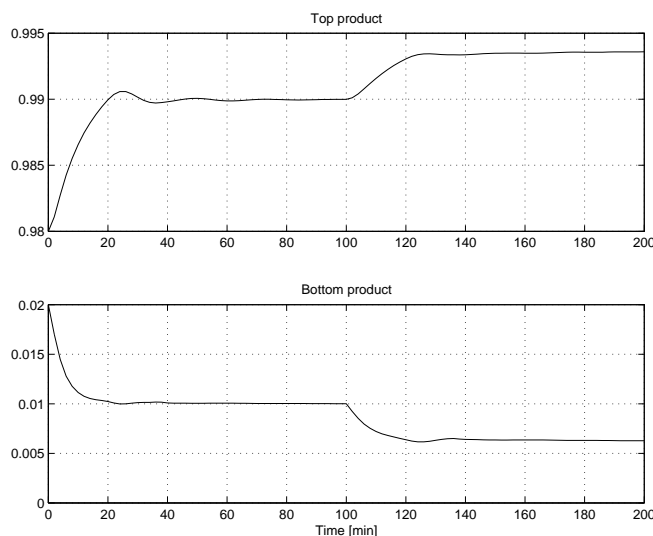


Figure 3.6: High-purity grade change simulated with SIPM based MPC.

Figure 3.7 gives the comparison of the maximal time required to solve one MPC problem using ASM and the structured IPM with different number of variables ( $Nn_u$ ), where we have similar behavior to the one in the previous example. We also found that it took the structured IPM in the MPC controller approximately 10.6 seconds to solve each optimization problem for the distillation column. We noticed that the structured IPM became faster than MATLAB's ASM only for  $N > 140$  and it failed to compete with Mosek even for much longer horizons. This could definitely be attributed to the fact that the algorithm retains states as optimization variables. In this process we have 2 inputs and 82 states which give this big increase in the number of optimization variables. So, the structured IPM is definitely not a good choice for the MPC optimizer unless we use reduced models of the distillation process.

### 3.6.3 MPC with reduced linear models

To reduce the computational complexity of the MPC problem for the distillation column we are going to use model reduction software from SLICOT library [95]. Three basic model reduction algorithms belonging to the class of methods based on or related to balancing techniques can be found in [62, 49, 33]. These methods are primarily intended for the reduction of linear, stable, continuous-

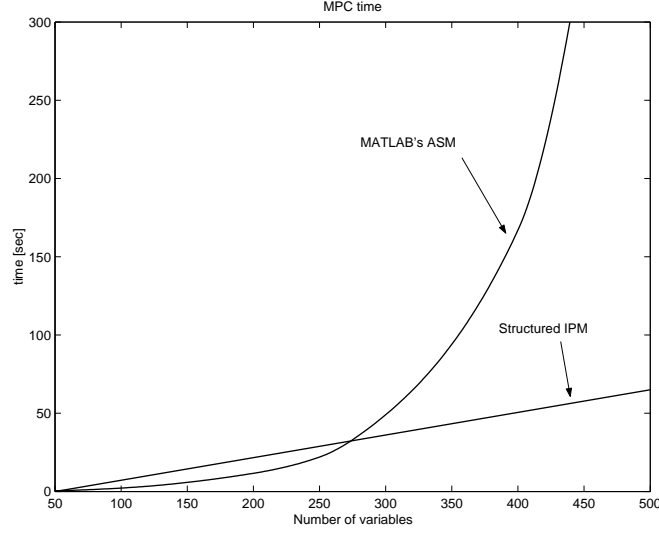


Figure 3.7: MPC computational time comparison for the distillation process.

or discrete-time systems. They rely on guaranteed error bounds and have particular features which recommend them for use in specific applications. Here we present the main features of balancing related model reduction.

#### Overview of balanced model reduction methods

Consider the  $n$ -th order original state-space model  $(A, B, C, D)$  with the transfer function matrix  $G(s) = C(sI - A)^{-1}B + D$ , and let  $(A_r, B_r, C_r, D_r)$  be an  $r$ -th order approximation of the original model ( $r < n$ ), with the transfer function  $G_r = C_r(sI - A_r)^{-1}B_r + D_r$ . A large class of model reduction methods can be interpreted as performing first a similarity transformation  $Z$  yielding

$$\left[ \begin{array}{c|c} Z^{-1}AZ & Z^{-1}B \\ \hline CZ & D \end{array} \right] := \left[ \begin{array}{cc|c} A_{11} & A_{12} & B_1 \\ A_{21} & A_{22} & B_2 \\ \hline C_1 & C_2 & D \end{array} \right]$$

and then defining the reduced model  $(A_r, B_r, C_r, D_r)$  as the diagonal system  $(A_{11}, B_1, C_1, D)$ . When writing  $Z := \begin{bmatrix} Z_1 & Z_2 \end{bmatrix}$  and  $Z^{-1} := \begin{bmatrix} L^\top & V^\top \end{bmatrix}^\top$ , then  $Z_1 L$  is a projector on  $Z_1$  along  $L$  since  $LZ_1 = I_r$ . Thus the reduced system is  $(A_r, B_r, C_r, D_r) = (LAZ_1, LB, CZ_1, D)$ .

Partitioned forms as above can be used to construct a so-called singular perturbation approximation (SPA). The matrices of the reduced model in this

case are given by

$$\begin{aligned} A_r &= A_{11} + A_{12}(\gamma I - A_{22})^{-1}A_{21} \\ B_r &= B_1 + A_{12}(\gamma I - A_{22})^{-1}B_2 \\ C_r &= C_1 + C_2(\gamma I - A_{22})^{-1}A_{21} \\ D_r &= D + C_2(\gamma I - A_{22})^{-1}B_2 \end{aligned} \quad (3.71)$$

where  $\gamma = 0$  for a continuous-time system and  $\gamma = 1$  for a discrete-time system. Note that SPA formulas preserve the DC-gains of stable original systems.

Specific requirements for model reduction algorithms are formulated and discussed in [96]. Such requirements are: (1) applicability of methods regardless the original system is minimal or not; (2) emphasis on enhancing the numerical accuracy of computations; (3) relying on numerically reliable procedures.

The first requirement can be fulfilled by computing  $L$  and  $Z_1$  directly, without determining  $Z$  or  $Z^{-1}$ . In particular, if the original system is not minimal, then  $L$  and  $Z_1$  can be chosen to compute an exact minimal realization of the original system [92].

The emphasis on improving the accuracy of computations led to so-called algorithms with enhanced accuracy. In many model reduction methods, the matrices  $L$  and  $Z_1$  are determined from two positive semi-definite matrices  $P$  and  $Q$ , called generically gramians. The gramians can be always determined in Cholesky factorized forms  $P = S^\top S$  and  $Q = R^\top R$ , where  $S$  and  $R$  are upper-triangular matrices. The computation of  $L$  and  $Z_1$  can be done by computing the singular value decomposition (SVD)

$$SR^\top = \begin{bmatrix} U_1 & U_2 \end{bmatrix} \text{diag}(\Sigma_1, \Sigma_2) \begin{bmatrix} V_1 & V_2 \end{bmatrix}^\top,$$

where

$$\Sigma_1 = \text{diad}(\sigma_1, \dots, \sigma_r), \quad \Sigma_2 = \text{diad}(\sigma_{r+1}, \dots, \sigma_n)$$

and  $\sigma_1 \geq \dots \geq \sigma_r > \sigma_{r+1} \geq \dots \geq \sigma_n \geq 0$  are the Hankel singular values of the system.

The so-called square-root (SR) methods determine  $L$  and  $Z_1$  as [90]

$$L = \Sigma_1^{-1/2} V_1^\top R, \quad Z_1 = S^\top U_1 \Sigma_1^{-1/2}.$$

If  $r$  is the order of a minimal realization of  $G$  then the gramians corresponding to the resulting realization are diagonal and equal. In this case the minimal realization is called balanced. The SR approach is usually very accurate for well-equilibrated systems. However if the original system is highly unbalanced, potential accuracy losses can be induced in the reduced model if either  $L$  or  $Z_1$  is ill-conditioned.

In order to avoid ill-conditioned projections, a balancing-free (BF) approach has been proposed in [81] in which always well-conditioned matrices  $L$  and  $Z_1$  can be determined. These matrices are computed from orthogonal matrices

whose columns span orthogonal bases for the right and left eigenspaces of the product  $PQ$  corresponding to the first  $r$  largest eigenvalues  $\sigma_1^2, \dots, \sigma_r^2$ . Because of the need to compute explicitly  $P$  and  $Q$  as well as their product, this approach is usually less accurate for moderately ill-balanced systems than the SR approach.

A balancing-free square-root (BFSR) algorithm which combines the advantages of the BF and SR approaches has been introduced in [92]. Matrices  $L$  and  $Z_1$  are determined as

$$L = (Y^\top X)^{-1} Y^\top, \quad Z_1 = X,$$

where  $X$  and  $Y$  are  $n \times r$  matrices with orthogonal columns computed from the QR decompositions  $S^\top U_1 = XW$  and  $R^\top V_1 = YZ$ , while  $W$  and  $Z$  are non-singular upper-triangular matrices. The accuracy of the BFSR algorithm is usually better than either of SR or BF approaches.

The SPA formulas can be used directly on a balanced minimal order realization of the original system computed with the SR method. A BFSR method to compute SPAs has been proposed in [91]. The matrices  $L$  and  $Z_1$  are computed such that the system  $(LAZ_1, LB, CZ_1, D)$  is minimal and the product of corresponding gramians has a block-diagonal structure which allows the application of the SPA formulas.

Provided the Cholesky factors  $R$  and  $S$  are known, the computation of matrices  $L$  and  $Z_1$  can be done by using exclusively numerically stable algorithms. Even the computation of the necessary SVD can be done without forming the product  $SR^\top$ . Thus the effectiveness of the SR or BFSR techniques depends entirely on the accuracy of the computed Cholesky factors of the gramians.

### **Algorithms for stable and unstable systems**

In the Balance & Truncate (B&T) method for stable systems [62]  $P$  and  $Q$  are the controllability and observability gramians satisfying a pair of continuous- or discrete-time Lyapunov equations

$$\begin{aligned} AP + PA^\top + BB^\top &= 0 & A^\top Q + QA + C^\top C &= 0 \\ APA^\top + BB^\top &= P & A^\top QA + C^\top C &= Q. \end{aligned}$$

These equations can be solved directly for the Cholesky factors of the gramians by using numerically reliable algorithms proposed in [34]. The BFSR version of the B&T method is described in [92]. Its SR version [90] can be used to compute balanced minimal representations. Such representations are also useful for computing reduced order models by using the SPA formulas [49] or the Hankel-norm approximation (HNA) method [33]. A BFSR version of the SPA method is described in [91]. Note that the B&T, SPA and HNA methods belong to the family of absolute error methods which try to minimize  $\|\Delta_a\|_\infty$ ,

where  $\Delta_a$  is the absolute error  $\Delta_a = G - G_r$ . For an  $r$ -th order approximation, we have generally

$$\|G - G_r\|_\infty \leq 2 \sum_{k=r+1}^n \sigma_k.$$

In case of optimal HNA method, the optimum  $G_r$  achieves

$$\inf \|G - G_r\|_H = \sigma_{r+1}$$

and even a feed-through matrix  $D_r$  can be chosen (see [33] for details) such that the error bound is one half of the bound for B&T and SPA.

The reduction of unstable systems can be performed by using the methods for stable systems in conjunction with two embedding techniques. The first approach consists in reducing only the stable projection of  $G$  and then including the unstable projection unmodified in the resulting reduced model. The following is a simple procedure for this computation:

1. Decompose additively  $G$  as  $G = G_1 + G_2$ , such that  $G_1$  has only stable poles and  $G_2$  has only unstable poles.
2. Determine  $G_{1r}$ , a reduced order approximation of the stable part  $G_1$ .
3. Assemble the reduced model  $G_r$  as  $G_r = G_{1r} + G_{2r}$ .

Note that for the model reduction at step 2 any of methods available for stable systems can be used.

The second approach is based on computing a stable rational coprime factorization (RCF) of  $G$ . The following procedure can be used to compute an  $r$ -th order approximation  $G_r$  of an  $n$ -th order (not necessarily stable) system  $G$ :

1. Compute a left coprime factorization of the transfer function matrix  $G$  in the form  $G = M^{-1}N$ , where  $M, N$  are stable and proper rational transfer function matrices.
2. Approximate the stable system of order  $n$   $[N \ M]$  with  $[N_r \ M_r]$  of order  $r$ .
3. Form the  $r$ -th order approximation  $G_r = M_r^{-1}N_r$ .

The coprime factorization approach used in conjunction with the B&T or BST methods fits in the general projection formulation described before. The gramians necessary to compute the projection are the gramians of the system  $[N \ M]$ . The computed matrices  $L$  and  $Z_1$  by using either the SR or BFSR methods can be directly applied to the matrices of the original system. The main computational problem is how to compute the RCF to allow a smooth and

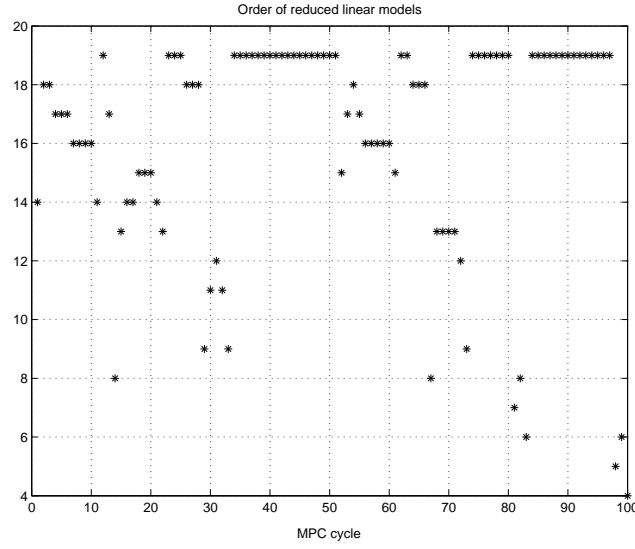


Figure 3.8: Order of reduced linear models of the distillation process.

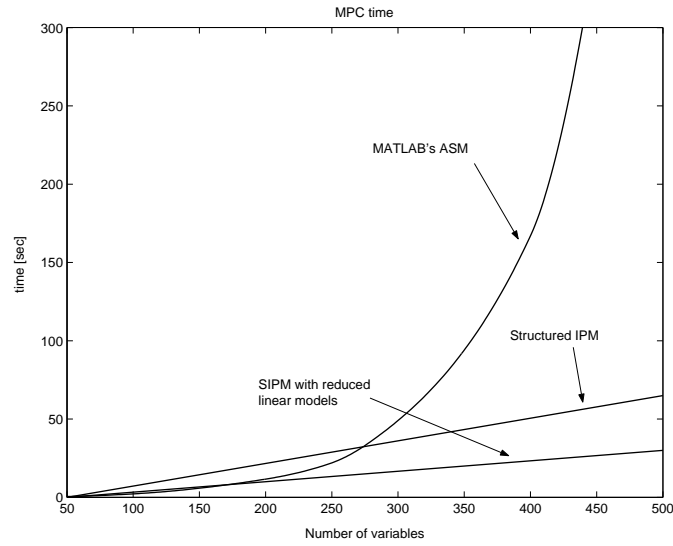


Figure 3.9: MPC computational time comparison for the distillation process with reduced linear models.

efficient embedding which prevents computational overheads. Two factoriza-

tion algorithms proposed recently compute particular RCFs which fulfill these aims: the RCF with prescribed stability degree [93] and the RCF with inner denominator [94]. Both are based on a numerically reliable Schur technique for pole assignment. The state matrix of the resulting factors is already in a real Schur form, thus the method has no overhead if the system is already stable since this reduction is always necessary even for stable systems. Note that the approximations computed for the factors of a coprime factorization with inner denominator by using the SPA method preserve these property also at the level of the reduced factors.

Using the described above balanced model reduction techniques [95], we managed to decrease the MPC computational time for the distillation column. The model order was constantly changing, which is reflected in Figure 3.8. The previous and the new computational results of MPC simulation based on reduced linear models are presented in Figure 3.9.

## Chemical Process Applications

---

4.1	MPC of a benchmark continuous stirred tank reactor	4.2	A stiff nonlinear DAE test problem
		4.3	Polymerization process

---

### 4.1 *MPC of a benchmark continuous stirred tank reactor*

We consider a benchmark problem for nonlinear control system design, which is based on a specific continuous stirred tank reactor (CSTR) that is described in [18]. The benchmark problem is characterized by a number of interesting features:

- The steady state gain changes its sign at the operating point. Thus, linear controllers with integral action will not be able to stabilize this reactor and accomplish satisfying performance [63].
- The zero dynamics changes its stability property at this operating point. Therefore, the qualitative behavior of the CSTR differs considerably for different setpoints and disturbances.
- The problem has a “real world” background.
- A complete set of performance objectives is given.

A discussion on the reasons and implications of the first two points can be found in [18].

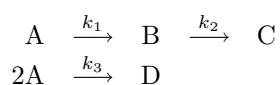
#### 4.1.1 *Description of a CSTR*

The reactor under consideration is a continuous stirred tank reactor with a cooling jacket in which cyclopentenol is produced from cyclopentadiene by acid-catalyzed electrophilic hydration in aqueous solution. The reaction scheme and



parameters are derived by theoretical modeling based on physical properties described in the literature for a real process.

Figure 4.1 shows a schematic diagram of the reactor. The main reaction is given by the transformation of cyclopentadiene (substance A) to the product cyclopentenol (substance B). The initial reactant cyclopentadiene also reacts in an unwanted parallel reaction to the by-product dicyclopentadiene (substance D). Furthermore, cyclopentanediol (substance C) is formed in an unwanted consecutive reaction from the product cyclopentenol. This, so-called van der Vusse reaction, is described by the following reaction scheme:



The input flow with its rate  $\dot{V}$  is fed to the reactor. It contains only cyclopenten-

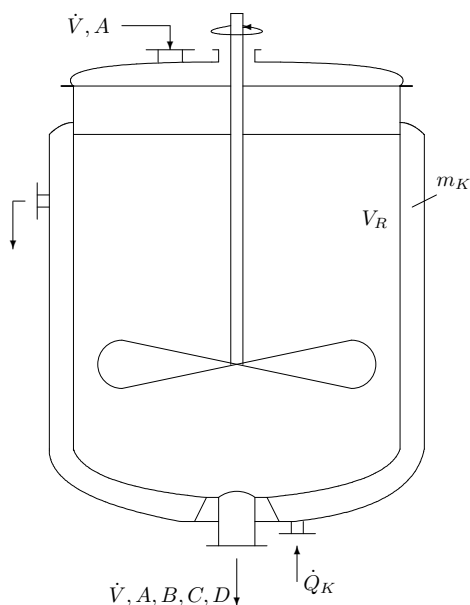


Figure 4.1: Schematic representation of the CSTR.

tadiene (substance A) with concentration  $c_{A0}$  at temperature  $T_0$ . Heat can be withdrawn from the coolant by an external heat exchanger with rate  $\dot{Q}_K$ .

In order to control concentrations and temperatures of the reactor, its dynamics are described by the following nonlinear differential equations that are derived from component balances for substances A and B and from energy balances for the reactor and cooling jacket.

**Component balances:**

$$\dot{c}_A = \frac{\dot{V}}{V_R}(c_{A0} - c_A) - k_1(T)c_A - k_3(T)c_A^2 \quad (4.1)$$

$$\dot{c}_B = -\frac{\dot{V}}{V_R}c_B + k_1(T)c_A - k_2(T)c_B \quad (4.2)$$

**Energy balances:**

$$\dot{T} = \frac{\dot{V}}{V_R}(T_0 - T) - \frac{1}{\rho C_p}(k_1(T)c_A\Delta H_{RAB} \quad (4.3)$$

$$+ k_2(T)c_B\Delta H_{RBC} + k_3(T)c_A^2\Delta H_{RAD}) \\ + \frac{k_w A_R}{\rho C_p V_R}(T_K - T)$$

$$\dot{T}_K = \frac{1}{m_K C_{PK}}(\dot{Q}_K + k_w A_R(T - T_K)), \quad (4.4)$$

$$c_A \geq 0, \quad c_B \geq 0.$$

The concentrations of substances A and B are  $c_A$  and  $c_B$  respectively. The temperature in the reactor is denoted by  $T$ , while the temperature in the cooling jacket is given by  $T_K$ . The reaction velocities  $k_i$  are assumed to depend on the temperature via the Arrhenius law

$$k_i(T) = k_{i0} \exp\left(-\frac{E_i}{R(T + 273.15)}\right), \quad i = 1, 2, 3. \quad (4.5)$$

Values for the physical and chemical parameters in equations (4.1)–(4.5) are given in Table 4.1. It should be noted that most parameters are only known within bounds.

**4.1.2 Control problem at the point of optimal yield**

The reactor is operated at a point where optimal yield (see [15, 13]) with respect to the product B is achieved (within a tolerance of 0.02 mol/l). The yield  $\Phi$  of product B is defined as the ratio between product concentration  $c_B$  and concentration of initial reactant  $c_{A0}$  of the feed in steady-state

$$\Phi = \frac{c_B|_S}{c_{A0}},$$

and is a measure for the effectiveness of the production. This optimal operating point was found in [13] by optimization of the steady state yield with respect to the design variables steady state feed flow  $\frac{\dot{V}}{V_R}|_S$ , steady state heat rate  $\dot{Q}_K|_S$

Name of parameter	Symbol	Value of parameter
pre-exponential factor	$k_{10}$	$(1.287 \pm 0.04) \cdot 10^{12} \text{ h}^{-1}$
pre-exponential factor	$k_{20}$	$(1.287 \pm 0.04) \cdot 10^{12} \text{ h}^{-1}$
pre-exponential factor	$k_{30}$	$(9.043 \pm 0.27) \cdot 10^9 \text{ h}^{-1}$
activation energy for reaction $k_1$	$E_1$	$81.1344 \text{ kJ mol}^{-1}$
activation energy for reaction $k_2$	$E_2$	$81.1344 \text{ kJ mol}^{-1}$
activation energy for reaction $k_3$	$E_3$	$71.1712 \text{ kJ mol}^{-1}$
enthalpies of reaction $k_1$	$\Delta H_{RAB}$	$(4.2 \pm 2.36) \frac{\text{kJ}}{\text{mol l}}$
enthalpies of reaction $k_2$	$\Delta H_{RBC}$	$-(11.0 \pm 1.92) \frac{\text{kJ}}{\text{mol l}}$
enthalpies of reaction $k_3$	$\Delta H_{RAD}$	$-(41.85 \pm 1.41) \frac{\text{kJ}}{\text{mol l}}$
density	$\rho$	$(0.9342 \pm 4.0 \cdot 10^{-4}) \frac{\text{kg}}{\text{l}}$
heat capacity	$C_p$	$(3.01 \pm 0.04) \frac{\text{kJ}}{\text{kg K}}$
heat transfer coeff. for cooling jacket	$k_w$	$(4032 \pm 120) \frac{\text{kJ}}{\text{h m}^2 \text{ K}}$
surface of cooling jacket	$A_R$	$0.215 \text{ m}^2$
reactor volume	$V_R$	$0.01 \text{ m}^3$
coolant mass	$m_K$	$5.0 \text{ kg}$
heat capacity of coolant	$C_{PK}$	$(2.0 \pm 0.05) \frac{\text{kJ}}{\text{kg K}}$
universal gas constant	$R$	$8.3144 \frac{\text{J}}{\text{mol K}}$

Table 4.1: CSTR model parameters.

and feed temperature  $T_0|_S$ . It is described by the following values [18]:

$$\begin{aligned}
 c_{A0}|_S &= 5.10 \frac{\text{mol}}{\text{l}} & c_A|_S &= 2.14 \frac{\text{mol}}{\text{l}} \\
 T_0|_S &= 104.9 \text{ }^\circ\text{C} & c_B|_S &= 1.09 \frac{\text{mol}}{\text{l}} \\
 \frac{\dot{V}}{V_R}|_S &= 14.19 \text{ h}^{-1} & T|_S &= 114.2 \text{ }^\circ\text{C} \\
 \dot{Q}_K|_S &= -1113.5 \frac{\text{kJ}}{\text{h}} & T_K|_S &= 112.9 \text{ }^\circ\text{C}
 \end{aligned} \tag{4.6}$$

The operating point of optimal yield is very desirable for economic reasons. The reactors behave strongly nonlinear with unstable zero dynamics leading to a difficult control problem at this operating point. A nonlinear MPC controller was proposed in [13] to drive the yield to the maximum level even in the presence of disturbances. It is clear that this control problem was especially challenging, as the controller always tries to drive the reactor to the point of optimal yield, and that point is characterized by difficulties like changing steady-state gain. The prediction horizon was chosen to be  $N = 200$ , the control horizon of only one sample, that means the controller had only one degree of freedom. The sampling interval was 20 seconds. In order to make real-time control possible, the manipulated variables stayed constant over two sampling periods as is done for the “blocking” technique. It is very important for the maximum yield controller that the high yield has to be maintained in steady-state and not only in a short time. This was achieved in the proposed controller by hard restrictions on the degrees of freedom of the controller. That

is, control horizon very short, and large enough prediction horizon.

We assume that the concentration  $c_A$  and  $c_B$ , the jacket temperature  $T_K$  and the reactor temperature  $T$  can be measured. So, all the states are measured outputs in this example. The feed of the reactor is assumed to come from an upstream unit. Therefore, the feed temperature  $T_0$  is assumed to be able to vary between

$$100\text{ }^\circ\text{C} \leq T_0 \leq 115\text{ }^\circ\text{C}$$

and is considered as an unmeasurable disturbance. Its nominal value from (4.6) will be used further in our tests. We use the flow rate normalized by the reactor volume  $\frac{\dot{V}}{V_R}$  and the heat rate  $\dot{Q}_K$  as manipulated variables, which are constrained by

$$\begin{aligned} 3\text{ h}^{-1} &\leq \frac{\dot{V}}{V_R} \leq 35\text{ h}^{-1} \\ -9000\text{ } \frac{\text{kJ}}{\text{h}} &\leq \dot{Q}_K \leq 0\text{ } \frac{\text{kJ}}{\text{h}}. \end{aligned}$$

Although we can calculate system Jacobians numerically, it is worth obtaining their analytical representation:

$$A = \begin{bmatrix} -\frac{\dot{V}}{V_R} - k_1 - 2k_3c_A & 0 & -\frac{\partial k_1}{\partial T}c_A - \frac{\partial k_3}{\partial T}c_A^2 & 0 \\ k_1 & -\frac{\dot{V}}{V_R} - k_2 & \frac{\partial k_1}{\partial T}c_A - \frac{\partial k_2}{\partial T}c_B & 0 \\ A(3,1) & -\frac{1}{\rho C_p}k_2\Delta H_{RBC} & A(3,3) & \frac{k_w A_R}{\rho C_p V_R} \\ 0 & 0 & \frac{k_w A_R}{m_K C_{PK}} & -\frac{k_w A_R}{m_K C_{PK}} \end{bmatrix},$$

where

$$\begin{aligned} A(3,1) &= -\frac{1}{\rho C_p}(k_1\Delta H_{RAB} + 2k_3c_A\Delta H_{RAD}) \\ A(3,3) &= -\frac{\dot{V}}{V_R} - \frac{1}{\rho C_p} \left( \frac{\partial k_1}{\partial T}c_A\Delta H_{RAB} + \frac{\partial k_2}{\partial T}c_B\Delta H_{RBC} + \frac{\partial k_3}{\partial T}c_A^2\Delta H_{RAD} \right) \\ &\quad - \frac{k_w A_R}{\rho C_p V_R} \end{aligned}$$

and

$$B = \begin{bmatrix} c_{A0} - c_A & 0 \\ -c_B & 0 \\ T_0 - T & 0 \\ 0 & \frac{1}{m_K C_{PK}} \end{bmatrix},$$

where

$$\frac{\partial k_i}{\partial T} = \frac{k_i(T)E_i}{R(T + 273.15)^2}, \quad i = 1, 2, 3.$$

A typical step response shows a slight difference in time constants of the product compositions and temperatures (see Figure 4.2).

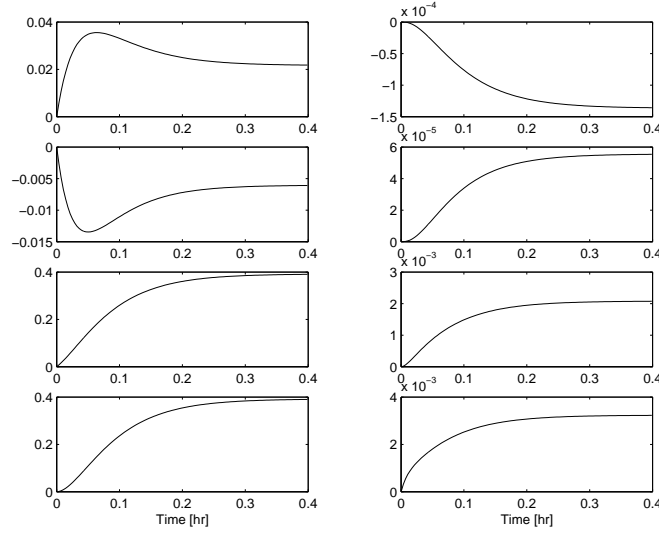


Figure 4.2: Step response of the CSTR.

### 4.1.3 Process start-up

First, we demonstrate a start-up for this process. We will start from some initial conditions different from the nominal point of optimal yield. This is also useful and important for optimal design of reactors and the control of reactors in the start-up phase. We assume that initial conditions are such that the reactor contains no species A or B. The jacket and reactor temperature are both equal to the feed temperature. The problem is characterized by a change of sign in the steady-state gain at the operating point, meaning linear controllers cannot stabilize this reactor without sacrificing performance. The MPC control objective is to steer the process to the optimal operating point defined in (4.6).

Figures 4.3 through 4.5 show the results of applying the structured IPM based MPC proposed in Section 3.4 to this problem. The sampling time is 20 seconds and the horizon length is chosen to be  $N = 30$ . All the states are assumed to be measured and there is no plant-model mismatch. The weighting matrices were chosen to be

$$Q = \begin{bmatrix} 10 & 0 & 0 & 0 \\ 0 & 5 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 2 \end{bmatrix}, \quad R = \begin{bmatrix} 10 & 0 \\ 0 & 0.1 \end{bmatrix}.$$

Note that the controller is able to stabilize the process after saturating one of the input constraints for the first half of the simulation. Despite the theoretic

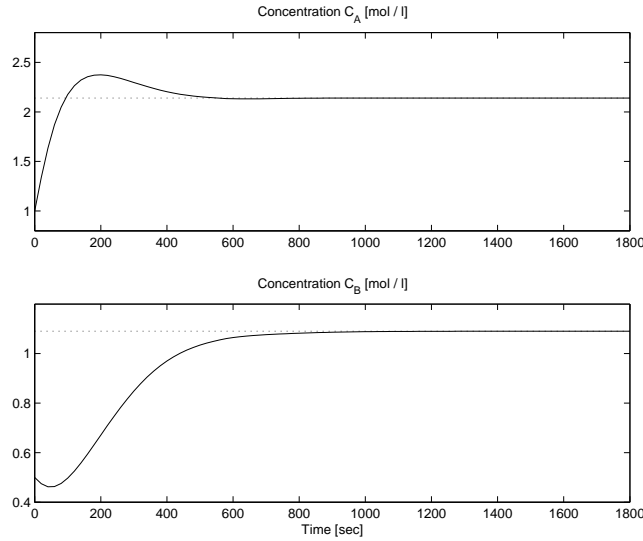


Figure 4.3: Concentrations  $c_A$  and  $c_B$  simulated with SIPM based MPC.

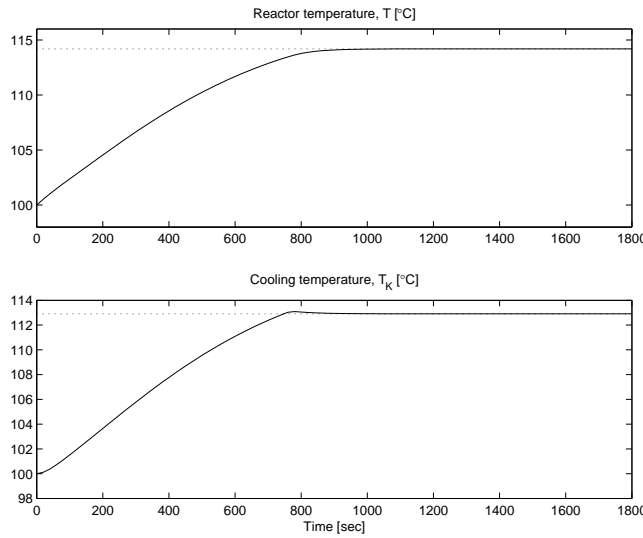


Figure 4.4: Reactor and jacket temperatures simulated with SIPM based MPC.

difficulties, like a change of sign of the steady-state gain, it was shown that with simulations that the closed-loop system is stable and has good nominal performance. Overall, the performance of the MPC controller on this example

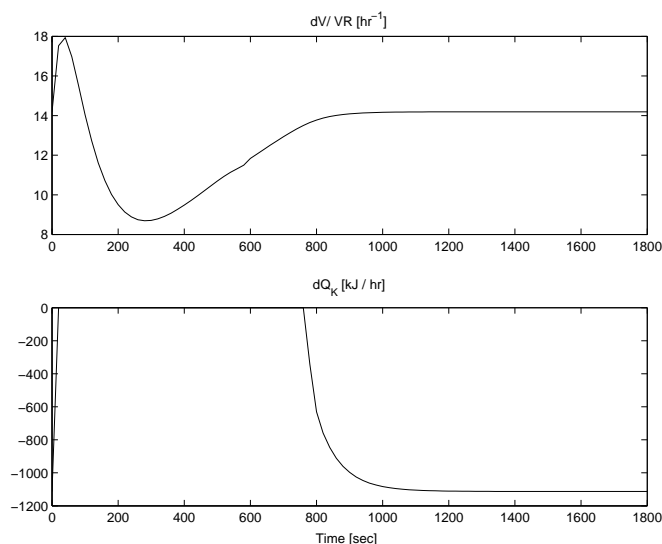


Figure 4.5: Manipulated variables simulated with SIPM based MPC.

is excellent.

The computational burden is a very important criterion for real-time implementation. We are going to demonstrate the effectiveness of the nonlinear MPC algorithm proposed in Section 3.4 comparing with the quasi-infinite horizon NMPC scheme described in [15]. That scheme was presented as computationally advantageous for real-time implementation, where the control horizon was chosen to be considerably shorter than the prediction horizon, namely  $N_c = 1$  and  $N = 70$ . Having much more degrees of freedom in the structured IPM based MPC controller, we still managed to solve the problem faster than in [15] with comparable performance results. This demonstrates that the MPC technology developed in this thesis has far-ranging advantages for real-time control implementation.

#### 4.1.4 Feed temperature disturbance attenuation

It becomes more difficult to steer the process to the optimal operating point in the presence of feed temperature disturbance. Figure 4.7 shows the MPC controlled response of the CSTR to a normally distributed feed disturbance (Figure 4.6).

The same tuning parameters as in the previous case were used for comparison. It is clear from Figure 4.8 that the manipulated variable is oscillating again after saturating one of the input constraints. Although we observe a slight oscillation in the reactor and cooling temperatures, the MPC controller

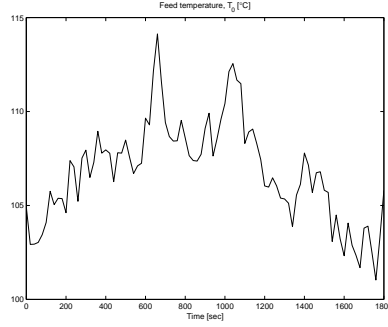


Figure 4.6: Feed temperature disturbance.

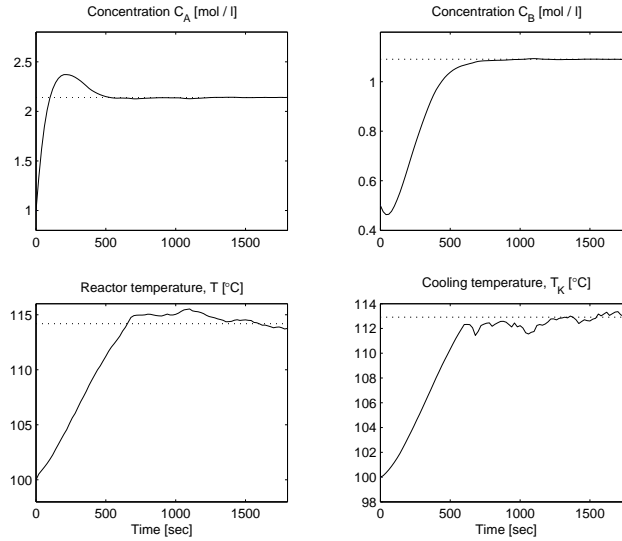


Figure 4.7: Output variables simulated with SIPM based MPC in the presence of feed temperature disturbance.

shows a good disturbance attenuation performance.

#### 4.1.5 Reference tracking problem

Next, we want to be able to produce substance B with concentration  $c_B$  in the following range

$$0.8 \frac{\text{mol}}{\text{l}} \leq c_B|_S \leq 1.09 \frac{\text{mol}}{\text{l}}.$$



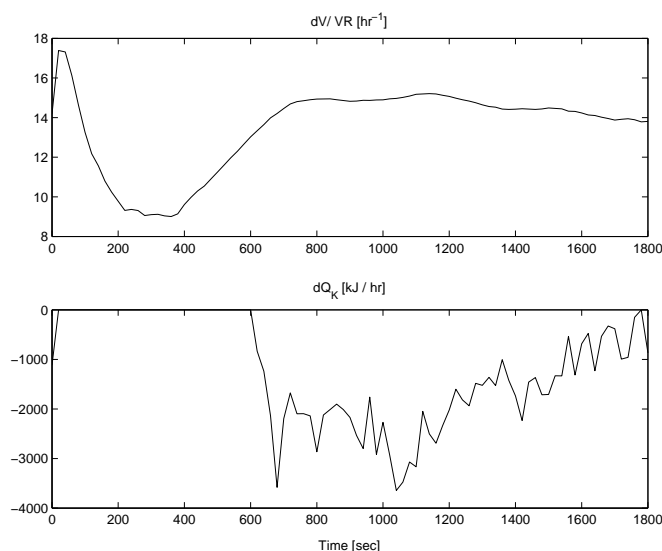


Figure 4.8: Manipulated variables simulated with SIPM based MPC in the presence of feed temperature disturbance.

The controller has to compensate the effects of changes in the setpoint value  $c_B|_S$ . In order to test the performance of the controller, we suggest step changes in the setpoint from their maximal value to the minimal value and back. The maximal steady state offset should not exceed  $0.02 \text{ mol/l}$  (control tolerance).

A “reference” solution to this benchmark problem was described in [18]. It was very time consuming to solve the on-line optimization problem posed. The complexity depends mainly on the number of independent variables  $N_c n_u$ . In order to reduce the number of independent variables, a technique called “blocking” was used. The idea is not to allow the manipulated variables to vary at every future sampling time but to require them to be constant over several sampling periods. In addition, the control horizon  $N_c$  was chosen to be smaller than the prediction horizon  $N$  and the manipulated variables were kept constant after the control horizon. In physical terms, a manipulated variable sequence of only  $N_c$  steps could not make the system follow the setpoint exactly over all  $N$  steps, when  $N_c < N$ . Therefore, only the setpoint change in the control horizon was considered in [18] for each optimization.

It is known that for linear non-minimum phase systems the closed-loop system can become unstable if the control action is too aggressive (i.e. the prediction horizon is too “short” or the control horizon is too “long”). This happens also in the nonlinear case, especially when the operating point is at the point of optimal yield as in the case of the CSTR considered. Similar to [13],

the prediction horizon of  $N = 200$  and the control horizon of  $N_c = 3$  were chosen in [18]. The manipulated variables were also kept constant over two sampling periods.

In our tests we are trying to exploit the long horizon capabilities of the MPC algorithm developed in the previous chapter. So, we do not use “blocking” technique and consider the setpoint change over the whole prediction horizon for each optimization. Here, we chose the prediction horizon to be  $N = 50$  and further observed better tracking performance with the increase of the horizon.

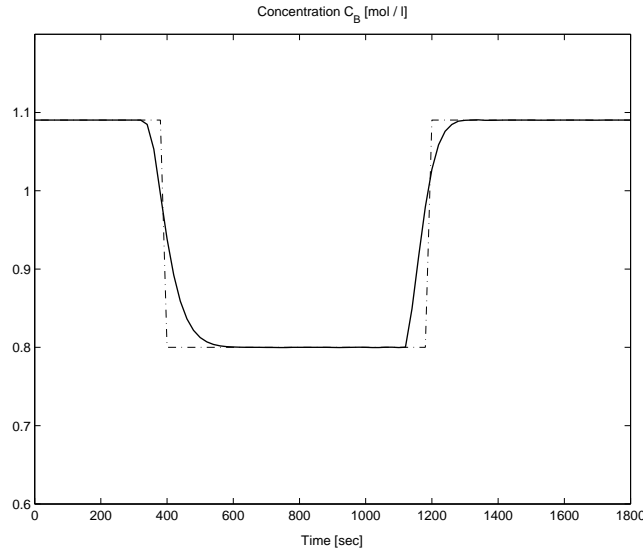


Figure 4.9: Concentration  $c_B$  simulated with SIPM based MPC.

Figures 4.9 and 4.10 show the MPC controlled response of the CSTR to step changes in the reference trajectory for  $c_B|_S$  from maximum to minimum and back to maximum value. We see that the concentration of product B is in the required control tolerance of the reference. Long horizon controller capabilities made it possible to observe almost no overshoot. Due to the operation at this highly nonlinear operating point, the control problem is very challenging. Despite the difficulty of the problem very satisfying control performance is achieved with the developed MPC technique. The bigger number of optimization variables ( $Nn_u$ ) gives us flexibility to obtain good reference tracking results. Thus, a large number of degrees of freedom allowed achieving excellent performance results with a significant reduction of computational complexity at the same time.

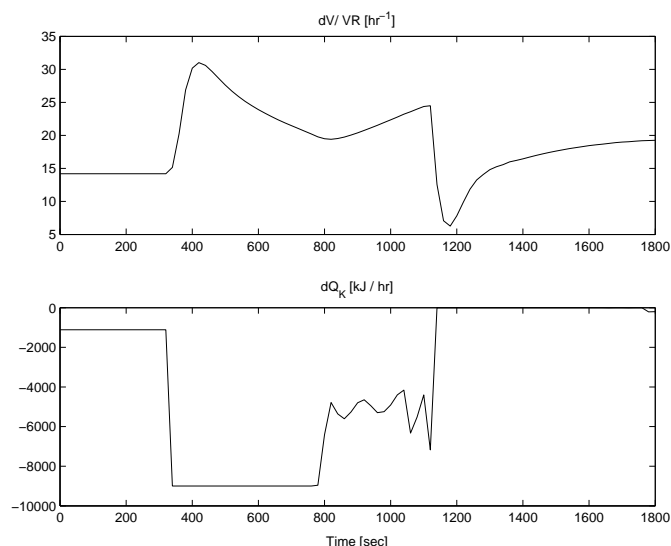
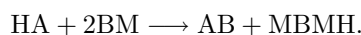


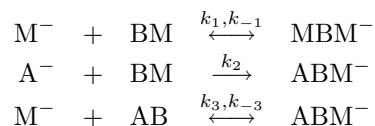
Figure 4.10: Manipulated variables simulated with SIPM based MPC.

## 4.2 A stiff nonlinear DAE test problem

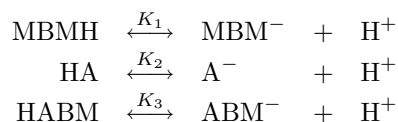
Using the kinetic model of a batch reactor system presented in [7, 48], we have formulated an MPC control problem for stiff nonlinear DAEs. The example in its original form was given by the Dow Chemical Company as a challenging test problem for parameter estimation software [12, 7]. The desired product AB is formed in the reaction



(For proprietary reasons, the true nature of the reacting species has been disguised.) The reaction is initiated by adding a catalyst QM to the mixture of reactants; QM is assumed to be completely dissociated to  $\text{Q}^+$  and  $\text{M}^-$  ions. The proposed reaction mechanism involves three slow reactions



and three rapid acid-base reactions



which are assumed to be at equilibrium. The equilibrium constants  $K_1, K_2$  and  $K_3$  are taken as constants over the range of temperatures employed. For the reaction rates  $k_1, k_{-1}$  and  $k_2$ , an Arrhenius temperature dependence is assumed

$$k_i = \hat{k}_i \exp\left(-\frac{E_i}{RT}\right), \quad i = 1, -1, 2,$$

where  $T$  is the reaction temperature in K, and  $\hat{k}_i, E_i$  and  $R$  denote the frequency factor, the activation energy, and the universal gas constant, respectively. Furthermore, based on similarities of the reacting species, it is supposed that

$$k_3 = k_1, \quad k_{-3} = \frac{1}{2}k_{-1}.$$

In the kinetic model describing the batch reactor system, the following state variables are used:

$$\begin{aligned} x_0 &= [\text{HA}] + [\text{A}^-] \\ x_1 &= [\text{BM}] \\ x_2 &= [\text{HABM}] + [\text{ABM}^-] \\ x_3 &= [\text{AB}] \\ x_4 &= [\text{MBMH}] + [\text{MBM}^-] \\ x_5 &= [\text{M}^-] \\ z_0 &= -\log([\text{H}^+]) \\ z_1 &= [\text{A}^-] \\ z_2 &= [\text{ABM}^-] \\ z_3 &= [\text{MBM}^-] \end{aligned}$$

Differential and algebraic states are denoted by  $x_j$  and  $z_j$ , respectively. All species concentrations  $[\cdot]$  are given in gmol per kg of the reaction mixture. Note that a logarithmic transformation is employed in case of  $z_0$ , since  $\text{H}^+$  is close to zero and changes rapidly over several orders of magnitude in the course of the reaction.

#### 4.2.1 Batch reactor model

Now the kinetic model can be stated in the form of six differential mass balance equations

$$\begin{aligned} \dot{x}_0 &= -k_2 x_1 z_1 \\ \dot{x}_1 &= -k_1 x_1 x_5 + k_{-1} z_3 - k_2 x_1 z_1 \\ \dot{x}_2 &= k_2 x_1 z_1 + k_3 x_3 x_5 - k_{-3} z_2 \\ \dot{x}_3 &= -k_3 x_3 x_5 + k_{-3} z_2 \\ \dot{x}_4 &= k_1 x_1 x_5 - k_{-1} z_3 \\ \dot{x}_5 &= -k_1 x_1 x_5 + k_{-1} z_3 - k_3 x_3 x_5 + k_{-3} z_2, \end{aligned} \tag{4.7}$$

then an electroneutrality condition and three equilibrium conditions

$$\begin{aligned}
 0 &= [\text{Q}^+] - x_5 + 10^{-z_0} - z_1 - z_2 - z_3 \\
 0 &= z_1 - K_2 x_0 / (K_2 + 10^{-z_0}) \\
 0 &= z_2 - K_3 x_2 / (K_3 + 10^{-z_0}) \\
 0 &= z_3 - K_1 x_4 / (K_1 + 10^{-z_0}).
 \end{aligned} \tag{4.8}$$

Reaction time is measured in hr. The fixed model parameters are given in Table 4.2 (see [7]). In order to formulate an MPC control problem based on

Parameter name	Symbol	Value
frequency factor for $k_1$	$\hat{k}_1$	$1.3708 \cdot 10^{12} \text{ kg gmol}^{-1} \text{ hr}^{-1}$
frequency factor for $k_{-1}$	$\hat{k}_{-1}$	$1.6215 \cdot 10^{20} \text{ kg gmol}^{-1} \text{ hr}^{-1}$
frequency factor for $k_2$	$\hat{k}_2$	$5.2282 \cdot 10^{12} \text{ kg gmol}^{-1} \text{ hr}^{-1}$
activation energy for $k_1$	$\beta_1 = E_1/R$	$9.2984 \cdot 10^3 \text{ K}$
activation energy for $k_{-1}$	$\beta_{-1} = E_{-1}/R$	$1.3108 \cdot 10^4 \text{ K}$
activation energy for $k_2$	$\beta_2 = E_2/R$	$9.5999 \cdot 10^3 \text{ K}$
equilibrium reaction constant	$K_1$	$2.575 \cdot 10^{-16} \text{ gmol kg}^{-1}$
equilibrium reaction constant	$K_2$	$4.876 \cdot 10^{-14} \text{ gmol kg}^{-1}$
equilibrium reaction constant	$K_3$	$1.7884 \cdot 10^{-16} \text{ gmol kg}^{-1}$

Table 4.2: Batch reactor parameters.

this model, we choose the reaction temperature  $T$  as control function

$$u = T \quad \text{with} \quad 293.15 \text{ K} \leq u(t) \leq 393.15 \text{ K}.$$

The model (4.7) and (4.8) is a set of non-linear differential and algebraic equations. These can be written in the form

$$f(\dot{x}, x, z, u) = 0. \tag{4.9}$$

Here  $x$  and  $z$  are the sets of differential and algebraic variables respectively, while  $\dot{x}$  are the derivatives of  $x$  with respect to time. On the other hand,  $u$  is the set of input variables that are given functions of time. Now consider a point  $(\dot{x}^*, x^*, z^*, u^*)$  on the simulation trajectory that satisfies equation (4.9). By linearizing the above equations at this point, we can obtain a linear model of the form

$$\frac{\partial f}{\partial \dot{x}} \delta \dot{x} + \frac{\partial f}{\partial x} \delta x + \frac{\partial f}{\partial z} \delta z + \frac{\partial f}{\partial u} \delta u = 0, \tag{4.10}$$

where, for example,  $\delta x$  denotes the deviation of the variable  $x$  from the linearization point  $x^*$  and all the partial derivatives are evaluated on the reference trajectory.

For most DAE systems of index 1, the matrix  $\begin{bmatrix} \frac{\partial f}{\partial \dot{x}} & \frac{\partial f}{\partial z} \end{bmatrix}$  is non-singular, and consequently the above system can be re-arranged to the form

$$\begin{bmatrix} \delta \dot{x} \\ \delta z \end{bmatrix} = - \begin{bmatrix} \frac{\partial f}{\partial \dot{x}} & \frac{\partial f}{\partial z} \end{bmatrix}^{-1} \begin{bmatrix} \frac{\partial f}{\partial x} & \frac{\partial f}{\partial u} \end{bmatrix} \begin{bmatrix} \delta x \\ \delta u \end{bmatrix}, \quad (4.11)$$

which is a linearized form of the original non-linear system (4.9). Although we can calculate the linearized model in (4.10) numerically, using technique similar to (3.20), we derive its analytical representation here as well:

$$\begin{aligned} \frac{\partial f}{\partial \dot{x}} &= \begin{bmatrix} 0 & k_2 z_1 & 0 & 0 & 0 & 0 \\ 0 & (k_1 x_5 + k_2 z_1) & 0 & 0 & 0 & k_1 x_1 \\ 0 & -k_2 z_1 & 0 & -k_3 x_5 & 0 & -k_3 x_3 \\ 0 & 0 & 0 & k_3 x_5 & 0 & k_3 x_3 \\ 0 & -k_1 x_5 & 0 & 0 & 0 & -k_1 x_1 \\ 0 & k_1 x_5 & 0 & k_3 x_5 & 0 & (k_1 x_1 + k_3 x_3) \\ 0 & 0 & 0 & 0 & 0 & -1 \\ \frac{-K_2}{(K_2 + 10^{-z_0})} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{-K_3}{(K_3 + 10^{-z_0})} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{-K_1}{(K_1 + 10^{-z_0})} & 0 \end{bmatrix} \\ \frac{\partial f}{\partial z} &= \begin{bmatrix} 0 & k_2 x_1 & 0 & 0 \\ 0 & k_2 x_1 & 0 & -k_{-1} \\ 0 & -k_2 x_1 & k_{-3} & 0 \\ 0 & 0 & -k_{-3} & 0 \\ 0 & 0 & 0 & k_{-1} \\ 0 & 0 & -k_{-3} & -k_{-1} \\ -10^{-z_0} \log(10) & -1 & -1 & -1 \\ \frac{-10^{-z_0} \log(10) K_2 x_0}{(K_2 + 10^{-z_0})^2} & 1 & 0 & 0 \\ \frac{-10^{-z_0} \log(10) K_3 x_2}{(K_3 + 10^{-z_0})^2} & 0 & 1 & 0 \\ \frac{-10^{-z_0} \log(10) K_1 x_4}{(K_1 + 10^{-z_0})^2} & 0 & 0 & 1 \end{bmatrix} \\ \frac{\partial f}{\partial u} &= \begin{bmatrix} \frac{\partial k_2}{\partial u} x_1 z_1 \\ \frac{\partial k_1}{\partial u} x_1 x_5 - \frac{\partial k_{-1}}{\partial u} z_3 + \frac{\partial k_2}{\partial u} x_1 z_1 \\ -\frac{\partial k_2}{\partial u} x_1 z_1 - \frac{\partial k_3}{\partial u} x_3 x_5 + \frac{\partial k_{-3}}{\partial u} z_2 \\ \frac{\partial k_3}{\partial u} x_3 x_5 - \frac{\partial k_{-3}}{\partial u} z_2 \\ -\frac{\partial k_1}{\partial u} x_1 x_5 + \frac{\partial k_{-1}}{\partial u} z_3 \\ \frac{\partial k_1}{\partial u} x_1 x_5 - \frac{\partial k_{-1}}{\partial u} z_3 + \frac{\partial k_3}{\partial u} x_3 x_5 - \frac{\partial k_{-3}}{\partial u} z_2 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \end{aligned}$$

where

$$\frac{\partial k_i}{\partial u} = \hat{k}_i \frac{E_i}{RT^2} \exp\left(-\frac{E_i}{RT}\right), \quad i = 1, -1, 2,$$

and

$$\frac{\partial k_3}{\partial u} = \frac{\partial k_1}{\partial u}, \quad \frac{\partial k_{-3}}{\partial u} = \frac{1}{2} \frac{\partial k_{-1}}{\partial u}.$$

#### 4.2.2 Setpoint tracking problem

The aim of this section is to demonstrate the applicability of the structured IPM based MPC for control of stiff DAE systems. The initial catalyst concentration (which is equal to  $[Q^+]$ ) is chosen as

$$p = [Q^+] = 0.014178 \text{ gmol kg}^{-1}, \quad 0 \text{ gmol kg}^{-1} \leq p \leq 0.0262 \text{ gmol kg}^{-1}$$

for our MPC control problem (see [48]). At initial time  $t = 0$  hr, we impose the conditions

$$\begin{aligned} x_0(0) &= 1.5778 \\ x_1(0) &= 8.32 \\ x_2(0) = x_3(0) = x_4(0) &= 0 \\ x_5(0) &= p \\ z_0(0) &= 7 \\ z_1(0) &= 1.0 \cdot 10^{-8} \\ z_2(0) = z_3(0) &= 0. \end{aligned}$$

A dynamic optimization problem was solved in [48] to determine optimal steady-state values for all the variables. The MPC objective in our case is to bring the differential and algebraic variables to those values as fast as possible. The sampling time for this process was 1 minute and the prediction horizon was chosen to be  $N = 30$ . Figures 4.11 and 4.12 present the results of the developed MPC application to the DAE test problem. The achieved performance is similar to the optimal solution found in [48]. The differential variables reached the optimal steady-state in approximately 1.75 hr. The example shows that the MPC algorithm developed in the previous section is fully capable to control such complex stiff systems without significant computational burden comparing to the results in [48].

### 4.3 Polymerization process

The purpose of this section is to present the experimental results obtained with the INCOOP software architecture [36, 88], where the MPC technology developed in Section 3.4 has also been tested. Here we describe Bayer continuous polymerization process (see Figure 4.13) with a subsequent separation unit and monomer recycle [35]. We will not be able to present specific details on the process and all figures will be scaled to conceal the real values of the variables.

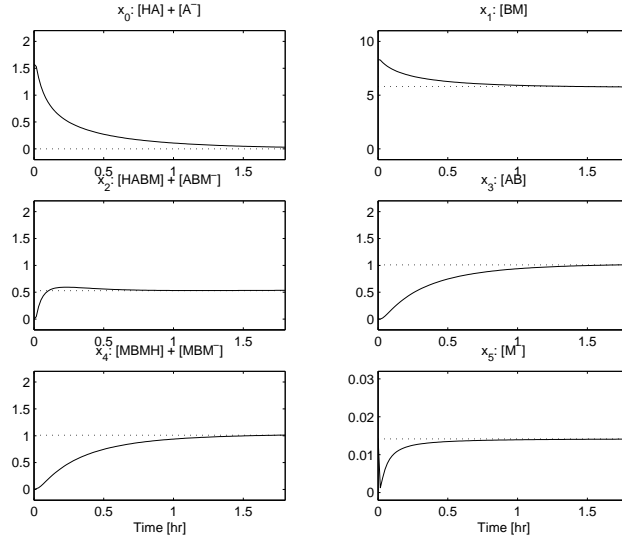


Figure 4.11: State variables simulated with SIPM based MPC.

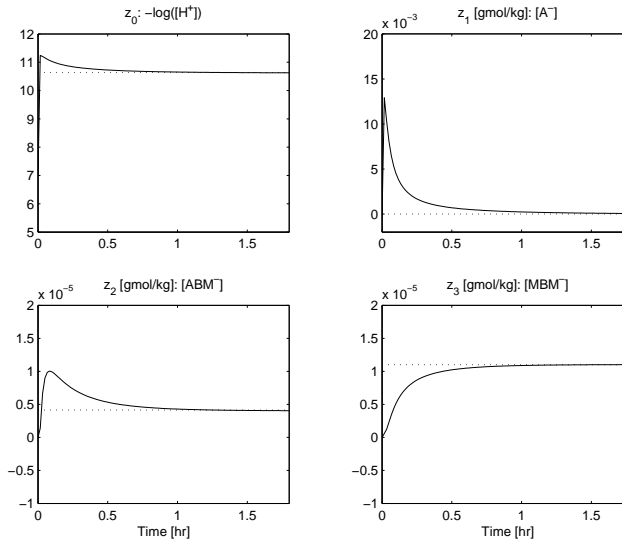


Figure 4.12: Algebraic variables simulated with SIPM based MPC.

The process uses monomers, catalyst and solvent to produce a specialty polymer. Complex reaction mechanism consists of more than 80 reactions.



The plant is operated in an open loop unstable point due to runaway reaction. Various polymer grades are produced, where frequent load changes are required due to fluctuating polymer demand. The process also becomes chal-

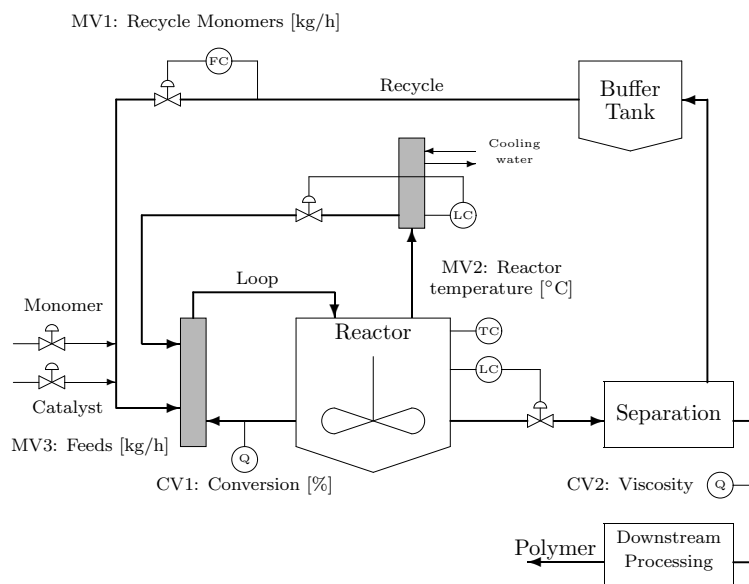


Figure 4.13: Polymerization process.

lenging for on-line optimization and control because of the presence of external disturbances due to sudden load changes and scheduled grade changes along with nominal process disturbances.

The process is described by an empirical DAE model with rigorous reactor system and an approximate model for the cooling unit. The whole model consists of 200 differential variables and approximately 2500 algebraic variables. Embedded base-level controllers are used along with a steady state filter for checking if a steady-state is achieved.

Process measurements include

- reactor temperature and holdup (online, with fast sampling time without delay),
- polymer quality in terms of viscosity (online and with lab samples, approximately 5 minutes sampling time with 30 minute delay),
- reactor conversion (online, fast sampling time without delay),
- inlet monomer flow rate (online), recycle and outlet flow rates (online, with uncertainty and bias).

Most of the flow measurements are affected by noise.

### 4.3.1 Optimal load change problem

Frequent changeovers are necessary and currently result in off-spec production during transition due to suboptimal trajectories. Open loop dynamics of reactor are unstable and stabilized by cascaded temperature control thus limiting the maximum speed of transitions. Polymer production rate needs to be adjusted spontaneously due to planned or unplanned throughput changes in downstream processing units. The trajectories have to be optimized to avoid production of off-spec material. Reactor residence time should reach new set-point as soon as possible. Product quality (molecular weight/viscosity) should not leave product specification.

The objective of the optimal load change problem is a quick transition without off-spec polymer production. Path constraints are imposed on monomer inlet and recycle flow-rates, reactor temperature and recycle buffer vessel holdup. The transition endpoint constraint is formalized in terms of a subsequent steady-state after the transition. The manipulated variables include monomer inlet and recycle flow rates, catalyst flow rate. Reactor temperature is controlled by a base-level controller.

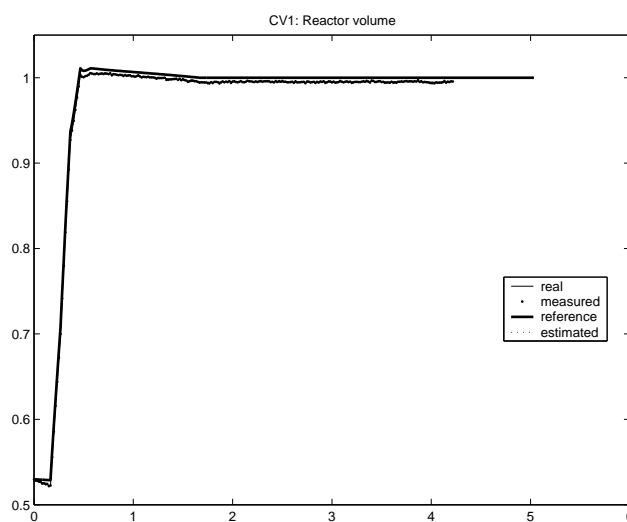


Figure 4.14: CV1: Reactor volume.

The disturbances during on-line control include 6 samples delay in viscosity measurement, where all measurements are obtained with high level of noise (5% bias in output flow-rate measurement). Besides that the problem is ill-

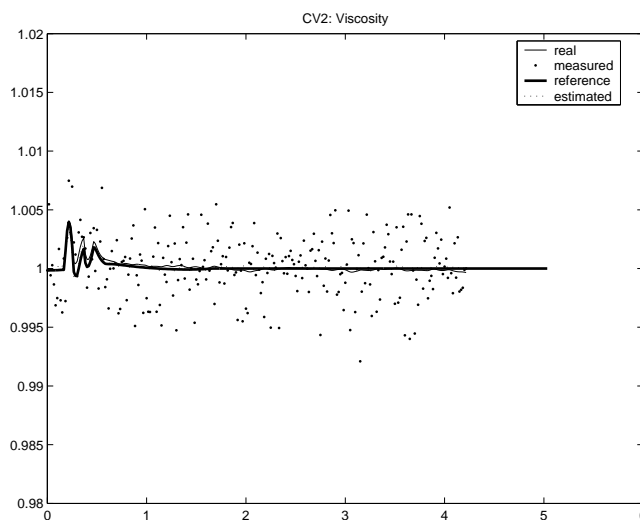


Figure 4.15: CV2: Viscosity.

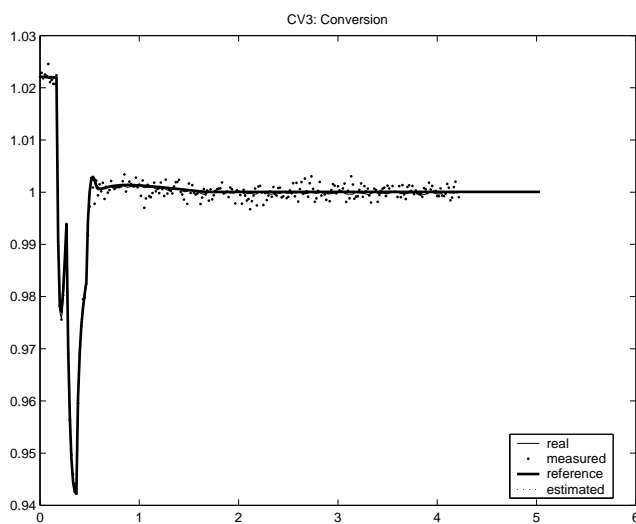


Figure 4.16: CV3: Conversion.

conditioned, the main difficulty is due to an unstable control system in case of tracking reference trajectories with level control being switched off. The tested scenario is represented by a problem with optimal load change from 50% to 90% and catalyst concentration of 95% of its nominal value.

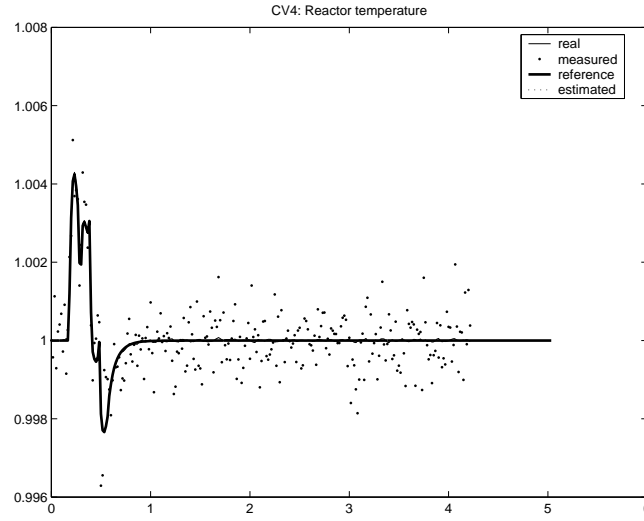


Figure 4.17: CV4: Reactor temperature.

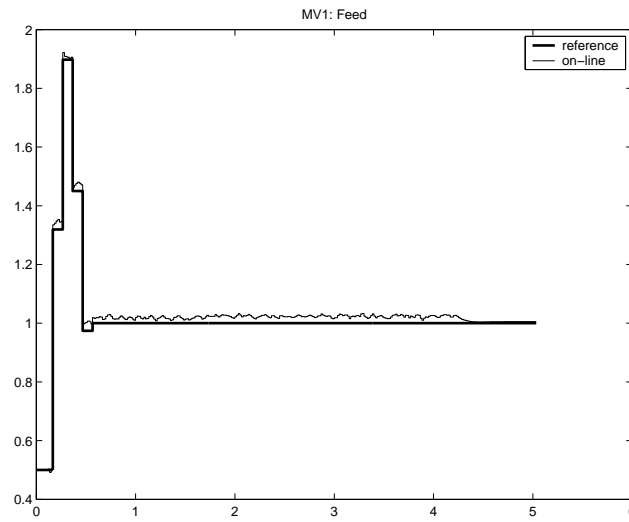


Figure 4.18: MV1: Feed.

The developed INCOOP solution was a two-level approach for model-based dynamic optimization and model predictive control [36]. The EKF-MPC algorithm applied to this process is described in [88] and is similar to the one

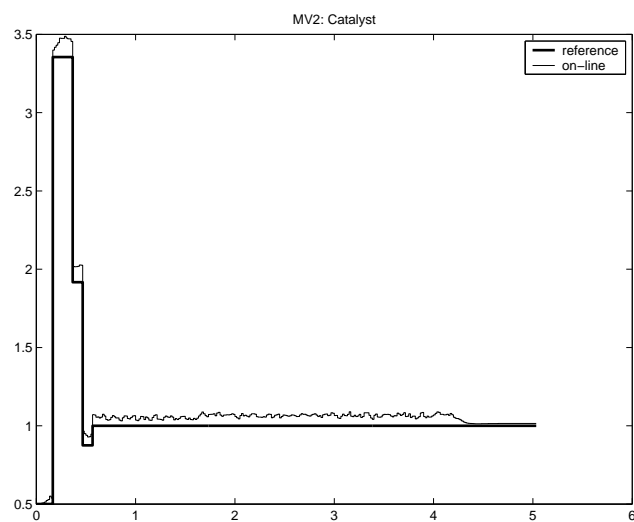


Figure 4.19: MV2: Catalyst.

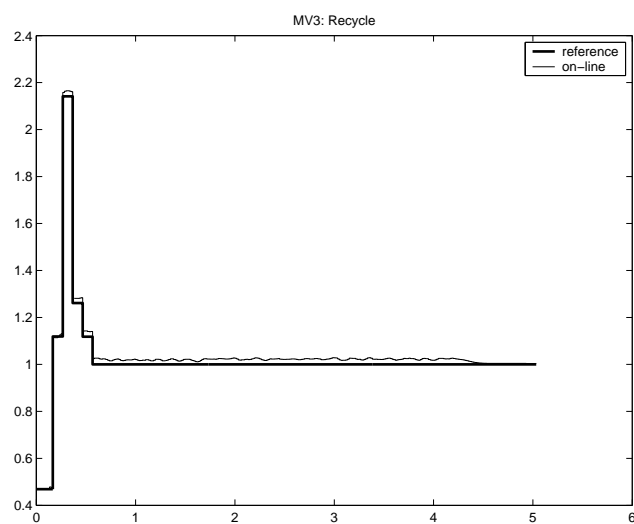


Figure 4.20: MV3: Recycle.

proposed in the previous chapter. The simulation results are presented in Figures 4.14–4.20 (see also [35]). The application of the developed technology shows that load changes can be realized fast and with no off-spec product.

Optimized profiles are significantly different from conventional operating procedure. Optimization results for several investigated scenarios predict that load changes can be realized with no off-spec material (compared to several hours of off-spec production in conventional manual operation).

The problem has practical relevance and is also related to an optimal grade change problem. Continuous polymerization plant produces different grades of the same polymer with frequent changeovers. The trajectories need to be optimized to minimize production of intermediate grade during transition. So the objective here is to achieve minimum grade transition time with a subsequent steady-state. Path constraints and manipulated variable are all the same. The problem is complicated by uncertain reaction rate parameters and major process disturbances such as solvent concentration, unreliable flow measurements.



## *Conclusions and Outlook*

---

5.1 Review of the results

5.2 Recommendations for  
further research

---

### **5.1 Review of the results**

This thesis presents a strategy for efficient model predictive control for nonlinear processes. Nonlinear model predictive control is a technology that is receiving increased attention from both academic researchers and industrial practitioners. Its strengths include its ability to handle constraints in input, state and output variables and the use of nonlinear process models that capture the behavior of the system in wider operating regions than linear models. While simpler methods for nonlinear control and estimation have been applied in industry for at least a decade, these methods are, in general, not reliable for process models.

One of the research objectives was to develop a computationally efficient MPC technology for a broad class of nonlinear systems. Real-time implementation of model predictive control is currently only possible for relatively slow systems with low dynamic performance specification, e.g. petrochemical plants. This is due to the high computational load of the MPC technology. In other industries, faster dynamics need to be controlled within the constraints of the system. In this research several possibilities are discussed to improve the computational efficiency of model predictive control technology. More specifically, in this thesis it is investigated what kind of techniques in model predictive control can be used to improve the efficiency of MPC applied to large-scale industrial systems exhibiting both fast and slow dynamics.

In this thesis we extended the MPC algorithm with capabilities for control of different system dynamics and for flexible constraint handling. Many formulations of nonlinear MPC have already been investigated by researchers, some incorporating terminal state constraints, others narrowing constraints so as to force the state to its setpoint. The formulations for the fully nonlinear MPC have been available for several years, but due to computational limitations, this technology has not yet been practiced. As an alternative to fully nonlinear



MPC, we introduced a different MPC algorithm that uses linear time-varying prediction models, obtained by applying a local linearization along a nominal input and state trajectory. In this way, the problematic setup of the standard NMPC approach is avoided, performance analysis is simplified, and computational efficiency is significantly improved. Local linear approximation of the state equation was used to develop an optimal prediction of the future states. The output prediction was made linear with respect to the undecided control input moves, which allowed to reduce the MPC optimization to a quadratic programming problem (QP). When compared with the standard local linearization based MPC which uses linear time-invariant prediction models, the proposed MPC provides more accurate approximations to the nonlinear system.

In Chapter 3 a technique has been proposed to improve computational efficiency in optimization to allow online MPC application. Solving QPs with standard methods typically requires computational time that increases with the third power of the number of optimization variables. The constrained optimization programs tend to become too large to be solved in real-time when these standard QP solvers are used. Many industrial examples show that large-scale, usually stiff, nonlinear systems may require long prediction horizons to fulfill certain performance specifications. These requirements increase the number of variables in the optimization. Naive implementations of standard QP solvers could be inefficient for such MPC problems. A structured interior-point method (IPM) has been developed in this thesis to solve the MPC problem for large-scale nonlinear systems to reduce the computational complexity. The developed optimization algorithm explicitly takes the structure of the given problem into account such that the computational cost varies linearly with the number of optimization variables, compared with the cubic growth for the standard optimizers. The quadratic programs that are solved by this method possess a special overlapping block-diagonal structure and are decomposed and solved efficiently using a tailored primal-dual interior-point methods. The algorithm also easily allows to introduce multiple linear models, thus making the control more flexible. The state elimination was not carried out and the structure given by the dynamics of the plant reflected in the Karush-Kuhn-Tucker (KKT) equations that were used to solve the QP using an interior-point method. The structured IPM was implemented using primal-dual Mehrotra's algorithm including prediction, correction and centering steps. The optimization variables consist of the inputs and the states over the horizon, but the optimization problem becomes sparse to allow computational time reduction, which is an important issue for on-line implementations of MPC for nonlinear stiff systems. Finally, this computational approach was used to explore the fundamental differences between linear and nonlinear MPC performance properties. In the thesis the effectiveness of the structured IPM based model predictive controller was demonstrated on several industrial chemical processes. We compared nonlinear and linear MPC and concluded that a nonlinear model

generally performed better than a linear model obtained from linearizing the original nonlinear model. We developed MPC for nonlinear systems and we showed in Chapters 3 and 4 that this performs better than a linear MPC for many examples relevant to the chemical process community.

## 5.2 Recommendations for further research

Nonlinear models have significant advantages over linear models for control and estimation. At the same time, they also present new problems that must be addressed to guarantee their usefulness. From an algorithmic point of view, pure nonlinear model predictive control requires the repeated solution of nonlinear optimal control problems. At certain times during the control period, the state of the system is estimated, and an optimal control problem is solved over a finite time horizon, using this state estimate as the initial state. The control component at the current time is used as the input to the system. Algorithms for nonlinear optimal control, which are often specialized nonlinear programming algorithms, can therefore be used in the context of nonlinear model predictive control, with the additional imperatives that the problem must be solved in “real time” and that good estimates of the solution may be available from the state and control profiles obtained at the previous sample.

An important aspect of future research in this area is the adaptation of the proposed optimization algorithm for solving true nonlinear MPC problems by directly exploiting the system structure. The standard SQP algorithms could be extended and approximated by a sequence of quadratic programming subproblems based on the proposed structured interior-point method.

In recent years, researchers have investigated computational strategies for model predictive control, leaving the dual problem of moving horizon state estimation (MHE) largely untouched. The nonlinear state estimation problem often is assumed solved by the extended Kalman filter (EKF). On the other hand, moving horizon estimation, handles constraints and nonlinear models naturally in its framework, and has been successful in estimating states of nonlinear models for which the EKF fails. Among other things, it would be interesting to address the practical issues of implementing MHE on a constrained nonlinear system and investigate how the state estimation problem is tied to the nonlinear model predictive control framework.

The practical implementation of model predictive control and moving horizon state estimation involves interaction between the individual elements. When these components are connected to form a closed-loop system, the nonlinear model can exhibit different behavior than is possible using a linear model. Although the behavior of the closed-loop linear MPC system is well studied, the behavior of the corresponding nonlinear system is not well characterized.

An other important question is the closed-loop behavior of nonlinear model predictive control, especially in combination with moving horizon state estima-

tion. The focus should be on the aspects of the resulting closed-loop system that are specific to nonlinear models. One such issue is the effect of choosing integrating disturbance models to account for unmeasured nonzero mean disturbances. The choice of disturbance models is an important issue for nonlinear plants. The choice of disturbance models can determine whether a system can be stabilized and will be crucial for a certain class of nonlinearities. More research is required on the proper choice of a disturbance model in case of unmeasured disturbances or plant-model mismatch. While input and output disturbance models often perform adequately, process knowledge could be used to model integrating disturbances more effectively. For instance, if disturbances are known to enter a process through a specific parameter, it seems reasonable to add the integrator to that parameter. Methods should also be developed to analyze aspects of closed-loop stability.

An other interesting aspect is the development of feedback strategies to efficiently handle disturbances in closed-loop. From receding horizon perspective the open-loop strategy produces an optimal control input, whereas the feedback strategy gives an optimal state law. Both depend on the initial state only, but the feedback strategies explicitly take the state on the prediction horizon into consideration to define the input. Feedback strategies have therefore the advantage that the effect of disturbances can be taken into account. However, the computation and synthesis of feedback strategies is much more difficult and involved than the synthesis of open-loop strategies.

While this study has answered some open questions about practical implementation of model predictive control for nonlinear processes, other questions remain open, and still others have been raised in this investigation. There are a lot of open issues, such as efficient constraint handling and robustness to plant-model mismatch. Current nonlinear MPC formulations do not permit to derive controllers that guarantee stability for a given uncertainty description with given bounds on the size of this uncertainty. Typically, for all these approaches no explicit quantitative nonlinear uncertainty models or only very simple ones are used. This is not surprising as the definition of meaningful uncertainty descriptions for nonlinear systems is an open problem not only in the NMPC context, but also in other areas of control.

One of the biggest questions about nonlinear model predictive control concerns the ability to develop a reliable nonlinear model. Until nonlinear models are more easily identified, the impact of nonlinear solution methods will not be fully realized. To better understand the advantages and disadvantages of different formulations of nonlinear MPC, the control community needs to focus on a useful set of benchmark problems. In the future, these benchmark examples will further define the new challenges of nonlinear MPC.

# A

## *Dynamic Programming*

---

A.1 Interior-Point Methods

A.2 Active Set Methods

---

### ***A.1 Interior-Point Methods***

In the 1980s it was discovered that many large linear programs could be solved efficiently by formulating them as nonlinear problems and solving them with various modifications of nonlinear algorithms such as Newton's method. One characteristic of these methods was that they required all iterates to satisfy the inequality constraints in the problem strictly, so they soon became known as interior-point methods. By the early 1990s, one class, namely primal-dual methods, had distinguished itself as the most efficient practical approach and proved to be a strong competitor to the simplex method on large problems. These methods are the focus of this chapter.

The motivation for interior-point methods arose from the desire to find algorithms with better theoretical properties than the simplex method. The simplex method can be quite inefficient on certain problems. Roughly speaking, the time required to solve a linear program may be exponential in the size of the problem, as measure by the number of unknowns and the amount of storage needed for the problem data. In practice, the simplex method is much more efficient than this bound would suggest, but its poor worst-case complexity motivated the development of new algorithms with better guaranteed performance. Among them is the ellipsoid method proposed, which finds a solution in time that is at worst polynomial in the problem size. Unfortunately this method approaches its worst-case bound on all problems and is not competitive with the simplex method.

Interior-point methods share common features that distinguish them from the simplex method. Each interior-point iteration is expensive to compute and can make significant progress towards the solution, while the simplex method usually requires a larger number of inexpensive iterations. The simplex method works its way around the boundary of the feasible polytope, testing a sequence of vertices in turn until it finds the optimal one. Interior-point methods approach the boundary of the feasible set only in the limit. They may approach

the solution either from the interior or the exterior of the feasible region, but they never actually lie on the boundary of this region.

In this chapter we outline some of the basic ideas behind primal-dual interior-point methods. We describe in detail a practical predictor-corrector algorithm proposed by Mehrotra, which is the basis of much of the current generation of software.

### A.1.1 Linear programming: primal-dual methods

Consider the linear programming problem in standard form

$$\min c^\top x, \quad \text{subject to } Ax = b, x \geq 0, \quad (\text{A.1})$$

where  $c$  and  $x$  are vectors in  $\mathbb{R}^n$ ,  $b$  is a vector in  $\mathbb{R}^m$ , and  $A$  is an  $m \times n$  matrix. The dual problem for (A.1) is

$$\max b^\top \lambda, \quad \text{subject to } A^\top \lambda + s = c, s \geq 0, \quad (\text{A.2})$$

where  $\lambda$  is a vector in  $\mathbb{R}^m$  and  $s$  is a vector in  $\mathbb{R}^n$ . Primal-dual solutions of (A.1), (A.2) are characterized by the Karush-Kuhn-Tucker conditions

$$A^\top \lambda + s = c, \quad (\text{A.3a})$$

$$Ax = b, \quad (\text{A.3b})$$

$$x_i s_i = 0, \quad i = 1, 2, \dots, n, \quad (\text{A.3c})$$

$$(x, s) \geq 0. \quad (\text{A.3d})$$

Primal-dual methods find solutions  $(x^*, \lambda^*, s^*)$  of this system by applying variants of Newton's method to the three equalities in (A.3) and modifying the search directions and step lengths so that the inequalities  $(x, s) \geq 0$  are satisfied strictly at every iteration. The equations (A.3a), (A.3b), (A.3c) are only mildly nonlinear and so are not difficult to solve by themselves. However, the problem becomes much more difficult when we add the nonnegativity requirement (A.3d). The nonnegativity condition is the source of all the complications in the design and analysis of interior-point methods.

To derive primal-dual interior-point methods, we restate the optimality conditions, (A.3) in a slightly different form by means of a mapping  $F$  from  $\mathbb{R}^{2n+m}$  to  $\mathbb{R}^{2n+m}$ :

$$F(x, \lambda, s) = \begin{bmatrix} A^\top \lambda + s - c \\ Ax - b \\ XSe \end{bmatrix} = 0 \quad (\text{A.4a})$$

$$(x, s) \geq 0, \quad (\text{A.4b})$$

where

$$X = \text{diag}(x_1, x_2, \dots, x_n), \quad S = \text{diag}(s_1, s_2, \dots, s_n), \quad (\text{A.5})$$

and  $e = (1, 1, \dots)^\top$ . Primal-dual methods generate iterates  $(x^k, \lambda^k, s^k)$  that satisfy the bounds (A.4b) strictly, that is,  $x^k > 0$  and  $s^k > 0$ . This property is the origin of the term *interior-point*. By respecting these bounds, the methods avoid spurious solutions, that is, points that satisfy  $F(x, \lambda, s) = 0$  but not  $(x, s) \geq 0$ . Spurious solutions abound, and do not provide useful information about solutions of (A.1) or (A.2), so it makes sense to exclude them altogether from the region of search.

Many interior-point methods actually require the iterates to be *strictly feasible*, that is, each  $(x^k, \lambda^k, s^k)$  must satisfy the linear equality constraints for the primal and dual problems. If we define the primal-dual feasible set  $\mathcal{F}$  and strictly feasible set  $\mathcal{F}^o$  by

$$\mathcal{F} = \{(x, \lambda, s) | Ax = b, A^\top \lambda + s = c, (x, s) \geq 0\}, \quad (\text{A.6a})$$

$$\mathcal{F}^o = \{(x, \lambda, s) | Ax = b, A^\top \lambda + s = c, (x, s) > 0\}, \quad (\text{A.6b})$$

the strict feasibility condition can be written concisely as

$$(x^k, \lambda^k, s^k) \in \mathcal{F}^o.$$

Like most iterative algorithms in optimization, primal-dual interior-point methods have two basic ingredients: a procedure for determining the step and a measure of the desirability of each point in the search space. As mentioned above, the search direction procedure has its origins in Newton's method for the nonlinear equations (A.4a). Newton's method forms a linear model for  $F$  around the current point and obtains the search direction  $(\Delta x, \Delta \lambda, \Delta s)$  by solving the following system of linear equations:

$$J(x, \lambda, s) \begin{bmatrix} \Delta x \\ \Delta \lambda \\ \Delta s \end{bmatrix} = -F(x, \lambda, s),$$

where  $J$  is the Jacobian of  $F$ . If the current point is strictly feasible (that is,  $(x, \lambda, s) \in \mathcal{F}^o$ ), the Newton step equations become

$$\begin{bmatrix} 0 & A^\top & I \\ A & 0 & 0 \\ S & 0 & X \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta \lambda \\ \Delta s \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -XSe \end{bmatrix}. \quad (\text{A.7})$$

A full step along this direction usually is not permissible, since it would violate the bound  $(x, s) \geq 0$ . To avoid this difficulty, we perform a line search along the Newton direction so that the new iterate is

$$(x, \lambda, s) + \alpha(\Delta x, \Delta \lambda, \Delta s),$$

for some line search parameter  $\alpha \in (0, 1]$ . Unfortunately, we often can take only a small step along the direction ( $\alpha \ll 1$ ) before violating the condition

$(x, s) > 0$ . So, the pure Newton direction (A.7), which is known as the *affine scaling direction*, often does not allow us to make much progress toward a solution.

Primal-dual methods modify the basic Newton procedure in two important ways:

1. They bias the search direction toward the interior of the nonnegative orthant  $(x, s) \geq 0$ , so that we can move further along the direction before one of the components of  $(x, s)$  becomes negative.
2. They keep the components of  $(x, s)$  from moving “too close” to the boundary of the nonnegative orthant.

### A.1.2 The central path

The central path  $\mathcal{C}$  is an arc of strictly feasible points that plays a vital role in primal-dual algorithms. It is parametrized by a scalar  $\tau > 0$ , and each point  $(x_\tau, \lambda_\tau, s_\tau) \in \mathcal{C}$  solves the following system:

$$A^\top \lambda + s = c, \quad (\text{A.8a})$$

$$Ax = b, \quad (\text{A.8b})$$

$$x_i s_i = \tau, \quad i = 1, 2, \dots, n, \quad (\text{A.8c})$$

$$(x, s) > 0. \quad (\text{A.8d})$$

These conditions differ from the KKT conditions only in the term  $\tau$  on the right-hand-side of (A.8c). Instead of the complementarity condition (A.3c), we require that the pairwise products  $x_i s_i$  have the same value for all indices  $i$ . From (A.8), we can define the central path as

$$\mathcal{C} = \{(x_\tau, \lambda_\tau, s_\tau) | \tau > 0\}.$$

It can be shown that  $(x_\tau, \lambda_\tau, s_\tau)$  is defined uniquely for each  $\tau > 0$  if and only if  $\mathcal{F}^\circ$  is nonempty.

Another way of defining  $\mathcal{C}$  is to use the mapping  $F$  defined in (A.4) and write

$$F(x_\tau, \lambda_\tau, s_\tau) = \begin{bmatrix} 0 \\ 0 \\ \tau e \end{bmatrix}, \quad (x_\tau, s_\tau) > 0. \quad (\text{A.9})$$

The equations (A.8) approximate (A.3) more and more closely as  $\tau$  goes to zero. If  $\mathcal{C}$  converges to anything as  $\tau \downarrow 0$ , it must converge to a primal-dual solution of the linear program. The central path thus guides us to a solution along a route that steers clear of spurious solutions by keeping all  $x$  and  $s$  components strictly positive and decreasing the pairwise products  $x_i s_i, i = 1, 2, \dots, n$ , to zero at roughly the same rate.

Primal-dual algorithms take Newton steps toward points on  $\mathcal{C}$  for which  $\tau > 0$ , rather than pure Newton steps for  $F$ . Since these steps are biased toward the interior of the nonnegative orthant defined by  $(x, s) \geq 0$ , it usually is possible to take longer steps along them than along the pure Newton steps for  $F$ , before violating the positivity condition.

To describe the biased search direction, we introduce a *centering parameter*  $\sigma \in [0, 1]$  and a *duality measure*  $\mu$  defined by

$$\mu = \frac{1}{n} \sum_{i=1}^n x_i s_i = \frac{x^\top s}{n}, \quad (\text{A.10})$$

which measures the average value of the pair wise products  $x_i s_i$ . By writing  $\tau = \sigma \mu$  and applying Newton's method to the system (A.9), we obtain

$$\begin{bmatrix} 0 & A^\top & I \\ A & 0 & 0 \\ S & 0 & X \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta \lambda \\ \Delta s \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -XSe + \sigma \mu e \end{bmatrix}. \quad (\text{A.11})$$

The step  $(\Delta x, \Delta \lambda, \Delta s)$  is a Newton step toward the point  $(x_{\sigma\mu}, \lambda_{\sigma\mu}, s_{\sigma\mu}) \in \mathcal{C}$ , at which the pairwise products  $x_i s_i$  are all equal to  $\sigma\mu$ . In contrast, the step (A.7) aims directly for the point at which the KKT conditions (A.3) are satisfied.

If  $\sigma = 1$ , the equations (A.11) define a *centering direction*, a Newton step toward the point  $(x_\mu, \lambda_\mu, s_\mu) \in \mathcal{C}$ , at which all the pairwise products  $x_i s_i$  are identical to  $\mu$ . Centering directions are usually biased strongly toward the interior of the nonnegative orthant and make little, if any, progress in reducing the duality measure  $\mu$ . However, by moving closer to  $\mathcal{C}$ , they set the scene for substantial progress on the next iteration. (Since the next iteration starts near  $\mathcal{C}$ , it will be able to take a relatively long step without leaving the nonnegative orthant.) At the other extreme, the value  $\sigma = 0$  gives the standard Newton step (A.7), sometimes known as the *affine-scaling direction*. Many algorithms use intermediate values of  $\sigma$  from the open interval  $(0, 1)$  to trade off between the twin goals of reducing  $\mu$  and improving centrality.

So far, we have assumed that the starting point  $(x^0, \lambda^0, s^0)$  is strictly feasible and, in particular, that it satisfies the linear equations  $Ax^0 = b$ ,  $A^\top \lambda^0 + s^0 = c$ . All subsequent iterates also respect these constraints, because of the zero right-hand-side terms in (A.11).

For most problems, however, a strictly feasible starting point is difficult to find. Infeasible-interior-point methods require only that the components of  $x^0$  and  $s^0$  be strictly positive. The search direction needs to be modified so that it improves feasibility as well as centrality at each iteration, but this requirement entails only a slight change to the step equation (A.11). If we define the residuals for the two linear equations as

$$r_b = Ax - b, \quad r_c = A^\top \lambda + s - c, \quad (\text{A.12})$$



the modified step equation is

$$\begin{bmatrix} 0 & A^\top & I \\ A & 0 & 0 \\ S & 0 & X \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta \lambda \\ \Delta s \end{bmatrix} = \begin{bmatrix} -r_c \\ -r_b \\ -XSe + \sigma \mu e \end{bmatrix}. \quad (\text{A.13})$$

The search direction is still a Newton step toward the point  $(x_{\sigma\mu}, \lambda_{\sigma\mu}, s_{\sigma\mu}) \in \mathcal{C}$ . It tries to correct all the infeasibility in the equality constraints in a single step. If a full step is taken at any iteration (that is,  $\alpha = 1$ ), the residuals  $r_b$ , and  $r_c$  become zero, and all subsequent iterates remain strictly feasible in the primal-dual algorithm.

### A.1.3 A practical primal-dual algorithm

Most existing interior-point codes for general-purpose linear programming problems are based on a predictor-corrector algorithm proposed by Mehrotra [60]. A *corrector step* is added to the search direction, so that the algorithm more closely follows a trajectory to the primal-dual solution set. An important feature of Mehrotra's algorithm is that it chooses the centering parameter  $\sigma$  adaptively. At each iteration, Mehrotra's algorithm first calculates the affine-scaling direction (the predictor step) and assesses its usefulness as a search direction. If this direction yields a large reduction in  $\mu$  without violating the positivity condition  $(x, s) > 0$ , the algorithm concludes that little centering is needed, so it chooses  $\sigma$  close to zero and calculates a centered search direction with this small value. If the affine-scaling direction is not so productive, the algorithm enforces a larger amount of centering by choosing a value of  $\sigma$  closer to 1.

The algorithm thus combines three steps to form the search direction: a predictor step, which allows us to determine the centering parameter  $\sigma_k$ , a corrector step using second-order information of the path leading toward the solution, and a centering step in which the chosen value of  $\sigma_k$  is substituted in (A.13). We will see that computation of the centered direction and the corrector step can be combined, so adaptive centering does not add further to the cost of each iteration.

In concrete terms, the computation of the search direction  $(\Delta x, \Delta \lambda, \Delta s)$  proceeds as follows. First, we calculate the predictor step  $(\Delta x^{\text{aff}}, \Delta \lambda^{\text{aff}}, \Delta s^{\text{aff}})$  by setting  $\sigma = 0$  in (A.13), that is,

$$\begin{bmatrix} 0 & A^\top & I \\ A & 0 & 0 \\ S & 0 & X \end{bmatrix} \begin{bmatrix} \Delta x^{\text{aff}} \\ \Delta \lambda^{\text{aff}} \\ \Delta s^{\text{aff}} \end{bmatrix} = \begin{bmatrix} -r_c \\ -r_b \\ -XSe + \sigma \mu e \end{bmatrix}. \quad (\text{A.14})$$

To measure the effectiveness of this direction, we find  $\alpha_{\text{aff}}^{\text{pri}}$  and  $\alpha_{\text{aff}}^{\text{dual}}$  to be the longest step lengths that can be taken along this direction before violating the nonnegativity conditions  $(x, s) \geq 0$ , with an upper bound of 1. Explicit

formulae for these values are as follows:

$$\alpha_{\text{aff}}^{\text{pri}} = \min \left( 1, \min_{i: \Delta x_i^{\text{aff}} < 0} -\frac{x_i}{\Delta x_i^{\text{aff}}} \right), \quad (\text{A.15a})$$

$$\alpha_{\text{aff}}^{\text{dual}} = \min \left( 1, \min_{i: \Delta s_i^{\text{aff}} < 0} -\frac{s_i}{\Delta s_i^{\text{aff}}} \right). \quad (\text{A.15b})$$

We define  $\mu_{\text{aff}}$  to be the value of  $\mu$  that would be obtained by a full step to the boundary, that is

$$\mu_{\text{aff}} = (x + \alpha_{\text{aff}}^{\text{pri}} \Delta x^{\text{aff}})^\top (s + \alpha_{\text{aff}}^{\text{dual}} \Delta s^{\text{aff}}) / n, \quad (\text{A.16})$$

and set the centering parameter  $\sigma$  to be

$$\sigma = \left( \frac{\mu_{\text{aff}}}{\mu} \right)^3.$$

It is easy to see that this choice has the effect mentioned above: when good progress is made along the predictor direction, we have  $\mu_{\text{aff}} \ll \mu$ , so the  $\sigma$  obtained from this formula is small and conversely.

The corrector step is obtained by replacing the right-hand-side in (A.14) by  $(0, 0, -\Delta X^{\text{aff}} \Delta S^{\text{aff}} e)$ , while the centering step requires a right-hand-side of  $(0, 0, \sigma \mu e)$ . We can obtain the complete Mehrotra step, which includes the predictor, corrector and centering step components, by adding the right-hand-sides for these three components and solving the following system

$$\begin{bmatrix} 0 & A^\top & I \\ A & 0 & 0 \\ S & 0 & X \end{bmatrix} \begin{bmatrix} \Delta x^{\text{aff}} \\ \Delta \lambda^{\text{aff}} \\ \Delta s^{\text{aff}} \end{bmatrix} = \begin{bmatrix} -r_c \\ -r_b \\ -X S e - \Delta X^{\text{aff}} \Delta S^{\text{aff}} e + \sigma \mu e \end{bmatrix}. \quad (\text{A.17})$$

We calculate the maximum steps that can be taken along these directions before violating the nonnegativity condition  $(x, s) > 0$  by formulae similar to (A.15), namely

$$\alpha_{\text{max}}^{\text{pri}} = \min \left( 1, \min_{i: \Delta x_i < 0} -\frac{x_i^k}{\Delta x_i} \right), \quad (\text{A.18a})$$

$$\alpha_{\text{max}}^{\text{dual}} = \min \left( 1, \min_{i: \Delta s_i < 0} -\frac{s_i^k}{\Delta s_i} \right). \quad (\text{A.18b})$$

and then choose the primal and dual step lengths as follows:

$$\alpha_k^{\text{pri}} = \min(1, \eta \alpha_{\text{max}}^{\text{pri}}), \quad \alpha_k^{\text{dual}} = \min(1, \eta \alpha_{\text{max}}^{\text{dual}}), \quad (\text{A.19})$$

where  $\eta \in [0.9, 1)$  is chosen so that  $\eta \rightarrow 1$  near the solution, to accelerate the asymptotic convergence. (For details of the choice of  $\eta$  and other elements of the algorithm such as the choice of starting point  $(x^0, \lambda^0, s^0)$  see Mehrotra [60].)

We summarize this discussion by specifying Mehrotra's algorithm:

Given  $(x^0, \lambda^0, s^0)$  with  $(x^0, s^0) > 0$ ;  
**for**  $k = 0, 1, 2, \dots$   
    Set  $(x, \lambda, s) = (x^k, \lambda^k, s^k)$  and solve (A.14) for  $(\Delta x^{\text{aff}}, \Delta \lambda^{\text{aff}}, \Delta s^{\text{aff}})$ ;  
    Calculate  $\alpha_{\text{aff}}^{\text{pri}}, \alpha_{\text{aff}}^{\text{dual}}$  and  $\mu_{\text{aff}}$  as in (A.15) and (A.16);  
    Set centering parameter to  $\sigma = (\mu_{\text{aff}}/\mu)^3$ ;  
    Solve (A.17) for  $(\Delta x, \Delta \lambda, \Delta s)$ ;  
    Calculate  $\alpha_k^{\text{pri}}$  and  $\alpha_k^{\text{dual}}$  from (A.19);  
    Set

$$\begin{aligned} x^{k+1} &= x^k + \alpha_k^{\text{pri}} \Delta x, \\ (\lambda^{k+1}, s^{k+1}) &= (\lambda^k, s^k) + \alpha_k^{\text{dual}} (\Delta \lambda, \Delta s). \end{aligned}$$

**end (for).**

It is important to note that no convergence theory is available for Mehrotra's algorithm at least in the form in which it is described above. In fact, there are examples for which the algorithm diverges. Simple safeguards could be incorporated into the method to force it into the convergence framework of existing methods. However, most programs do not implement these safeguards, because the good practical performance of Mehrotra's algorithm makes them unnecessary.

#### A.1.4 Quadratic programming: IPM

An optimization problem with a quadratic objective function and linear constraints is called a quadratic program. Problems of this type are important in their own right, and they also arise as subproblems in methods for general constrained optimization, such as sequential quadratic programming and augmented Lagrangian methods.

The primal-dual interior-point approach can be applied to convex quadratic programs through a simple extension of the linear-programming algorithms. The resulting algorithms are simple to describe, relatively easy to implement, and quite efficient on certain types of problems.

For simplicity, we restrict our attention to convex quadratic programs with inequality constraints, which we write as follows

$$\min_x \frac{1}{2} x^\top G x + x^\top d, \quad (\text{A.20a})$$

$$\text{subject to } A x \geq b, \quad (\text{A.20b})$$

where  $G$  is symmetric and positive semidefinite,  $A$  is a  $m \times n$  matrix. (Equality constraints can be accommodated with simple changes to the approaches described below.) We can specialize the KKT conditions to obtain the following

set of necessary conditions for (A.20): If  $x^*$  is a solution of (A.20), there is a Lagrange multiplier vector  $\lambda^*$  such that the following conditions are satisfied for  $(x, \lambda) = (x^*, \lambda^*)$ :

$$\begin{aligned} Gx - A^\top \lambda + d &= 0, \\ Ax - b &\geq 0, \\ (Ax - b)_i \lambda_i &= 0, \quad i = 1, 2, \dots, m, \\ \lambda &\geq 0. \end{aligned}$$

By introducing the slack vector  $y = Ax - b$ , we can rewrite these conditions as

$$Gx - A^\top \lambda + d = 0, \quad (\text{A.21a})$$

$$Ax - y - b \geq 0, \quad (\text{A.21b})$$

$$y_i \lambda_i = 0, \quad i = 1, 2, \dots, m, \quad (\text{A.21c})$$

$$(y, \lambda) \geq 0. \quad (\text{A.21d})$$

It is easy to see the close correspondence between (A.21) and the KKT conditions (A.3) for the linear programming problem (A.1). As in the case of linear programming, the KKT conditions are not only necessary but also sufficient, because the objective function is convex and the feasible region is convex. Hence, we can solve the convex quadratic program (A.20) by finding solutions of the system (A.21).

We can rewrite (A.21) as a constrained system of nonlinear equations and derive primal-dual interior-point algorithms by applying modifications of Newton's method to this system. Analogously to (A.4), we define

$$F(x, y, \lambda) = \begin{bmatrix} Gx - A^\top \lambda + d \\ Ax - y - b \\ Y\Lambda e \end{bmatrix} = 0, \quad (y, \lambda) \geq 0,$$

where

$$Y = \text{diag}(y_1, y_2, \dots, y_m), \quad \Lambda = \text{diag}(\lambda_1, \lambda_2, \dots, \lambda_m), \quad e = (1, 1, \dots, 1)^\top.$$

Given a current iterate  $(x, y, \lambda)$  that satisfies  $(y, \lambda) > 0$ , we can define a duality measure  $\mu$  by

$$\mu = \frac{1}{m} \sum_{i=1}^m y_i \lambda_i = \frac{y^\top \lambda}{m}, \quad (\text{A.22})$$

similarly to (A.10).

The central path  $\mathcal{C}$  is the set of points  $(x_\tau, y_\tau, \lambda_\tau)$  ( $\tau > 0$ ) such that

$$F(x_\tau, y_\tau, \lambda_\tau) = \begin{bmatrix} 0 \\ 0 \\ \tau e \end{bmatrix}, \quad (y_\tau, \lambda_\tau) > 0.$$

The generic step  $(\Delta x, \Delta y, \Delta \lambda)$  is a Newton-like step from the current point  $(x, y, \lambda)$  toward the point  $(x_{\sigma\mu}, y_{\sigma\mu}, \lambda_{\sigma\mu})$  on the central path, where  $\sigma \in [0, 1]$  is a parameter chosen by the algorithm. As in (A.13), we find that this step satisfies the following linear system:

$$\begin{bmatrix} G & 0 & -A^\top \\ A & -I & 0 \\ 0 & \Lambda & Y \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta \lambda \end{bmatrix} = \begin{bmatrix} -r_d \\ -r_b \\ -\Lambda S e + \sigma \mu e \end{bmatrix}, \quad (\text{A.23})$$

where

$$r_d = Gx - A^\top \lambda + d, \quad r_b = Ax - y - b.$$

We obtain the next iterate by setting

$$(x^+, y^+, \lambda^+) = (x, y, \lambda) + \alpha(\Delta x, \Delta y, \Delta \lambda),$$

where  $\alpha$  is chosen to retain the inequality  $(y^+, \lambda^+) > 0$  and possibly to satisfy various other conditions.

Mehrotra's predictor-corrector algorithm can also be extended to convex quadratic programming with the exception of one aspect: The step lengths in primal variables  $(x, y)$  and dual variables  $\lambda$  cannot be different, as they are in the linear programming case. The reason is that the primal and dual variables are coupled through the matrix  $G$ , so different step lengths can disturb feasibility of the equation (A.21a).

The major computational operation is the solution of the system (A.23) at each iteration of the interior-point method. This system may be restated in more compact forms. The augmented system form is

$$\begin{bmatrix} G & -A^\top \\ A & \Lambda^{-1}Y \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} = \begin{bmatrix} -r_d \\ -r_b + (-y + \sigma \mu \Lambda^{-1}e) \end{bmatrix}, \quad (\text{A.24})$$

and a symmetric indefinite factorization scheme can be applied to the coefficient matrix. The normal equations form is

$$(G + A^\top(Y^{-1}\Lambda)A)\Delta x = -r_d + A^\top(Y^{-1}\Lambda)[-r_b - y + \sigma \mu \Lambda^{-1}e],$$

which can be solved by means of a modified Cholesky algorithm. Note that the factorization must be recomputed at each iteration, because the change in  $y$  and  $\lambda$  leads to changes in the nonzero components of  $A^\top(Y^{-1}\Lambda)A$ .

## A.2 Active Set Methods

We now describe active set methods (ASM), which are generally the most effective methods for small- to medium-scale problems. The general quadratic

program (QP) can be stated as

$$\min_x q(x) = \frac{1}{2}x^\top Gx + x^\top d, \quad (\text{A.25a})$$

$$\text{subject to } a_i^\top x = b_i, \quad i \in \mathcal{E} \quad (\text{A.25b})$$

$$a_i^\top x \geq b_i, \quad i \in \mathcal{J} \quad (\text{A.25c})$$

where  $G$  is a symmetric  $n \times n$  matrix,  $\mathcal{E}$  and  $\mathcal{J}$  are finite sets of indices, and  $d$ ,  $x$ , and  $\{a_i\}, i \in \mathcal{E} \cup \mathcal{J}$ , are vectors with  $n$  elements. Quadratic programs can always be solved (or can be shown to be infeasible) in a finite number of iterations, but the effort required to find a solution depends strongly on the characteristics of the objective function and the number of inequality constraints. If the Hessian matrix  $G$  is positive semidefinite, we say that (A.25) is a convex QP, and in this case the problem is sometimes not much more difficult to solve than a linear program. Nonconvex QPs, in which  $G$  is an indefinite matrix, can be more challenging, since they can have several stationary points and local minima. In this chapter we describe algorithms that find the solution of a convex quadratic program or a stationary point of a general (nonconvex) quadratic program.

### A.2.1 Constrained QP

We begin our discussion of algorithms for quadratic programming by considering the case where only equality constraints are present. As we will see in this chapter, active set method for general quadratic programming solve an equality-constrained QP at each iteration.

Let us denote the number of constraints by  $m$ , assume that  $m \leq n$ , and write the quadratic program as

$$\min_x \frac{1}{2}x^\top Gx + x^\top d, \quad (\text{A.26a})$$

$$\text{subject to } Ax = b, \quad (\text{A.26b})$$

where  $A$  is the  $m \times n$  Jacobian of constraints defined by  $A = [a_i]_{i \in \mathcal{E}}^\top$ . For the present, we assume that  $A$  has full row rank (rank  $m$ ), and that the constraints (A.26b) are consistent.

The first-order necessary conditions for  $x^*$  to be a solution of (A.26) state that there is a vector  $\lambda^*$  such that the following system of equations is satisfied

$$\begin{bmatrix} G & -A^\top \\ A & 0 \end{bmatrix} \begin{bmatrix} x^* \\ \lambda^* \end{bmatrix} = \begin{bmatrix} -d \\ b \end{bmatrix}. \quad (\text{A.27})$$

It is easy to derive these conditions as a consequence of the general result for first-order optimality conditions. As before, we call  $\lambda^*$  the vector of Lagrange multipliers. The system (A.27) can be rewritten in a form that is useful for

computation by expressing  $x^*$  as  $x^* = x + p$ , where  $x$  is some estimate of the solution and  $p$  is the desired step. By introducing this notation and rearranging the equations, we obtain

$$\begin{bmatrix} G & -A^\top \\ A & 0 \end{bmatrix} \begin{bmatrix} -p \\ \lambda^* \end{bmatrix} = \begin{bmatrix} g \\ c \end{bmatrix}. \quad (\text{A.28})$$

where

$$c = Ax - b, \quad g = d + Gx, \quad p = x^* - x. \quad (\text{A.29})$$

The matrix in (A.28) is called the Karush-Kuhn-Tucker (KKT) matrix.

We also discuss an algorithm for solving QPs that contain inequality constraints and possibly equality constraints. Classical active-set methods can be applied both to convex and non-convex problems, and they have been the most widely used methods since the 1970s. We begin our discussion with a brief review of the optimality conditions for inequality-constrained quadratic programming. Note that the Lagrangian for the problem (A.25) is

$$\mathcal{L}(x, \lambda) = \frac{1}{2}x^\top Gx + x^\top d - \sum_{i \in \mathcal{J} \cup \mathcal{E}} \lambda_i (a_i^\top x - b_i). \quad (\text{A.30})$$

In addition, we define the active set  $\mathcal{A}(x^*)$  at an optimal point  $x^*$  as the indices of the constraints at which equality holds, that is,

$$\mathcal{A}(x^*) = \{i \in \mathcal{J} \cup \mathcal{E} : a_i^\top x^* = b_i\}. \quad (\text{A.31})$$

We conclude that any solution  $x^*$  of (A.25) satisfies the following first-order conditions

$$Gx^* + d - \sum_{i \in \mathcal{A}(x^*)} \lambda_i^* a_i = 0, \quad (\text{A.32a})$$

$$a_i^\top x^* = b_i, \quad \text{for all } i \in \mathcal{A}(x^*) \quad (\text{A.32b})$$

$$a_i^\top x^* \geq b_i, \quad \text{for all } i \in \mathcal{J} \setminus \mathcal{A}(x^*) \quad (\text{A.32c})$$

$$\lambda_i^* \geq 0, \quad \text{for all } i \in \mathcal{J} \cap \mathcal{A}(x^*). \quad (\text{A.32d})$$

In the optimality conditions for quadratic programming given above we need not assume that the active constraints are linearly dependent at the solution. Second-order sufficient conditions for  $x^*$  to be a local minimizer are satisfied if  $Z^\top GZ$  is positive definite, where  $Z$  is defined to be a null-space basis matrix for the active constraint Jacobian matrix

$$[a_i]_{i \in \mathcal{A}(x^*)}^\top.$$

The point  $x^*$  is actually a global solution for the equality-constrained case when this condition holds. When  $G$  is not positive definite, the general problem (A.25) may have more than one strict local minimizer at which the second-order necessary conditions are satisfied. Such problems are referred to as being “nonconvex” or “indefinite”, and they cause some complication for algorithms.

### A.2.2 Specification of ASM for convex QP

We start by discussing the convex case, in which the matrix  $G$  in (A.25a) is positive semidefinite. Since the feasible region defined by (A.25b), (A.25c) is a convex set, any local solution of the QP is a global minimizer. The case in which  $G$  is an indefinite matrix raises complications in the algorithms.

Recall the definition (A.31) above of the active set  $\mathcal{A}(x)$  at the optimal point  $x^*$ , it will be called an *optimal* active set. If  $\mathcal{A}(x^*)$  were known in advance, we could find the solution by applying one of the techniques for equality-constrained QP to the problem

$$\min_x q(x) = \frac{1}{2}x^\top Gx + x^\top d \quad \text{subject to} \quad a_i^\top x = b_i, \quad i \in \mathcal{A}(x^*).$$

Of course, we usually don't have prior knowledge of  $\mathcal{A}(x^*)$ , and as we will now see, determination of this set is the main challenge facing algorithms for inequality-constrained QP.

An active-set method starts by making a guess of the optimal active set, and if this guess turns out to be incorrect, it repeatedly uses gradient and Lagrange multiplier information to drop one index from the current estimate of  $\mathcal{A}(x^*)$  and add a new index. An active-set approach for linear programming is the simplex method. Active-set methods for QP differ from the simplex method in that the iterates may not move from one vertex of the feasible region to another. Some iterates (and, indeed, the solution of the problem) may lie at other points on the boundary or interior of the feasible region.

Active-set methods for QP come in three varieties, known as primal, dual, and primal-dual. We restrict our discussion to primal methods, which generate iterates that remain feasible with respect to the primal problem (A.25) while steadily decreasing the primal objective function.

Primal active-set methods usually start by computing a feasible initial iterate  $x_0$ , and then ensure that all subsequent iterates remain feasible. They find a step from one iterate to the next by solving a quadratic subproblem in which a subset of the constraints in (A.25b), (A.25c) is imposed as equalities. This subset is referred to as the *working set* and is denoted at the  $k$ th iterate  $x_k$  by  $\mathcal{W}_k$ . It consists of all the equality constraints  $i \in \mathcal{E}$  (see A.25b) together with some—but not necessarily all—of the active inequality constraints. An important requirement we impose on  $\mathcal{W}_k$  is that the gradients  $a_i$  of the constraints in the working set be linearly independent, even when the full set of active constraints at that point has linearly dependent gradients.

Given an iterate  $x_k$  and the working set  $\mathcal{W}_k$ , we first check whether  $x_k$  minimizes the quadratic  $q$  in the subspace defined by the working set. If not, we compute a step  $p$  by solving an equality-constrained QP subproblem in which the constraints corresponding to the working set  $\mathcal{W}_k$  are regarded as equalities and all other constraints are temporarily disregarded. To express



this subproblem in terms of the step  $p$ , we define

$$p = x - x_k, \quad g_k = Gx_k + d,$$

and by substituting for  $x$  into the objective function (A.25a), we find that

$$q(x) = q(x_k + p) = \frac{1}{2}p^\top Gp + g_k^\top p + c,$$

where  $c = \frac{1}{2}x_k^\top Gx_k + d^\top x_k$  is a constant term. Since we can drop  $c$  from the objective without changing the solution of the problem, we can write the QP subproblem to be solved at the  $k$ th iteration as follows

$$\min_p \frac{1}{2}p^\top Gp + g_k^\top p, \quad (\text{A.33a})$$

$$\text{subject to } a_i^\top p = 0 \text{ for all } i \in \mathcal{W}_k. \quad (\text{A.33b})$$

We denote the solution of this subproblem by  $p_k$ . Note that for each  $i \in \mathcal{W}_k$ , the term of  $a_i^\top x$  does not change as we move along  $p_k$ , since we have  $a_i^\top(x_k + p_k) = a_i^\top x_k = b_i$ . It follows that since the constraints in  $\mathcal{W}_k$  were satisfied at  $x_k$ , they are also satisfied at  $x_k + \alpha p_k$ , for any value of  $\alpha$ .

Suppose for the moment that the optimal  $p_k$  from (A.33) is nonzero. We need to decide how far to move along this direction. If  $x_k + p_k$  is feasible with respect to all the constraints, we set  $x_{k+i} = x_k + p_k$ . Otherwise, we set

$$x_{k+i} = x_k + \alpha_k p_k, \quad (\text{A.34})$$

where the step-length parameter  $\alpha_k$  is chosen to be the largest value in the range  $[0, 1]$  for which all constraints are satisfied. We can derive an explicit definition of  $\alpha_k$  by considering what happens to the constraints  $i \notin \mathcal{W}_k$ , since the constraints  $i \in \mathcal{W}_k$  will certainly be satisfied regardless of the choice of  $\alpha_k$ . If  $a_i^\top p_k \geq 0$  for some  $i \notin \mathcal{W}_k$ , then for all  $\alpha_k \geq 0$  we have  $a_i^\top(x_k + \alpha_k p_k) \geq a_i^\top x_k \geq b_i$ . Hence, this constraint will be satisfied for all nonnegative choices of the step-length parameter. Whenever  $a_i^\top p_k < 0$  for some  $i \notin \mathcal{W}_k$ , however, we have that  $a_i^\top(x_k + \alpha_k p_k) \geq b_i$  only if

$$\alpha_k \leq \frac{b_i - a_i^\top x_k}{a_i^\top p_k}.$$

Since we want  $\alpha_k$  to be as large as possible in  $[0, 1]$  subject to retaining feasibility, we have the following definition

$$\alpha_k = \min \left( 1, \min_{i \notin \mathcal{W}_k, a_i^\top p_k < 0} \frac{b_i - a_i^\top x_k}{a_i^\top p_k} \right). \quad (\text{A.35})$$

We call the constraints  $i$  for which the minimum in (A.35) is achieved the *blocking constraints*. (If  $\alpha_k = 1$  and no new constraints are active at  $x_k + \alpha_k p_k$ ,

then there are no blocking constraints on this iteration.) Note that it is quite possible for  $\alpha_k$  to be zero, since we could have  $a_i^\top p_k < 0$  for some constraint  $i$  that is active at  $x_k$  but not a member of the current working set  $\mathcal{W}_k$ .

If  $\alpha_k < 1$ , that is, the step along  $p_k$  was blocked by some constraint not in  $\mathcal{W}_k$ , a new working set  $\mathcal{W}_{k+1}$  is constructed by adding one of the blocking constraints to  $\mathcal{W}_k$ .

We continue to iterate in this manner, adding constraints to the working set until we reach a point  $\hat{x}$  that minimizes the quadratic objective function over its current working set  $\hat{\mathcal{W}}$ . It is easy to recognize such a point because the subproblem (A.33) has solution  $p = 0$ . Since  $p = 0$  satisfies the optimality conditions (A.28) for (A.33), we have that

$$\sum_{i \in \hat{\mathcal{W}}} a_i \hat{\lambda}_i = g = G\hat{x} + d, \quad (\text{A.36})$$

for some Lagrange multipliers  $\hat{\lambda}_i, i \in \hat{\mathcal{W}}$ . It follows that  $\hat{x}$  and  $\hat{\lambda}$  satisfy the first KKT condition (A.32a), if we define the multipliers corresponding to the inequality constraints that are not in the working set to be zero. Because of the control imposed on the step-length,  $\hat{x}$  is also feasible with respect to all the constraints, so the second and third KKT conditions (A.32b) and (A.32c) are satisfied by  $\hat{x}$ .

We now examine the signs of the multipliers corresponding to the inequality constraints in the working set, that is, the indices  $i \in \hat{\mathcal{W}} \cap \mathcal{I}$ . If these multipliers are all nonnegative, the fourth KKT condition (A.32d) is also satisfied, so we conclude that  $\hat{x}$  is a KKT point for the original problem (A.25). In fact, since  $G$  is positive semidefinite, we can show that  $\hat{x}$  is a local minimizer. When  $G$  is positive definite,  $\hat{x}$  is a strict local minimizer.

If, on the other hand, one of the multipliers  $\hat{\lambda}_j, j \in \hat{\mathcal{W}} \cap \mathcal{I}$ , is negative, the condition (A.32d) is not satisfied, and the objective function  $q$  may be decreased by dropping this constraint. We then remove an index  $j$  corresponding to one of the negative multipliers from the working set and solve a new subproblem (A.33) for the new step. This strategy produces a direction  $p$  at the next iteration that is feasible with respect to the dropped constraint (see [69]).

Having given a complete description of the active-set algorithm for convex QP, we can give the following formal specification:

```

Compute a feasible starting point  $x_0$ ;
Set  $\mathcal{W}_0$  to be a subset of the active constraints at  $x_0$ ;
for  $k = 0, 1, 2, \dots$ 
    Solve (A.33) to find  $p_k$ ;
    if  $p_k = 0$ 
        Compute Lagrange multipliers  $\hat{\lambda}_i$  that satisfy (A.36),
        set  $\hat{\mathcal{W}} = \mathcal{W}_k$ ;
    if  $\hat{\lambda}_i \geq 0$  for all  $i \in \mathcal{W}_k \cap \mathcal{I}$ ;

```

```

        STOP with solution  $x^* = x_k$ ;
    else
        Set  $j = \arg \min_{i \in \mathcal{W}_k \cap \mathcal{J}} \hat{\lambda}_j$ ;
         $x_{k+1} = x_k$ ;  $\mathcal{W}_{k+1} \leftarrow \mathcal{W}_k \setminus \{j\}$ ;
    else ( $*p_k \neq 0^*$ )
        Compute  $\alpha_k$  from (A.35);
         $x_{k+1} \leftarrow x_k + \alpha_k p_k$ ;
        if there are blocking constraints
            Obtain  $\mathcal{W}_{k+1}$  by adding one of the blocking constraints to  $\mathcal{W}_{k+1}$ ;
        else
             $\mathcal{W}_{k+1} \leftarrow \mathcal{W}_k$ ;
end (for)

```

Various techniques can be used to determine an initial feasible point. One such is to use the “Phase I” approach. Though no significant modifications are needed to generalize this method from linear programming to quadratic programming, we describe a variant here that allows the user to supply an initial estimate  $\tilde{x}$  of the vector  $x$ . This estimate need not be feasible, but prior knowledge of the QP may be used to select a value of  $\tilde{x}$  that is “not too infeasible”, which will reduce the work needed to perform the Phase I step.

Given  $\tilde{x}$ , we define the following feasibility linear program:

$$\begin{aligned}
 & \min_{(x,z)} && e^\top z \\
 & \text{subject to} && a_i^\top x + \gamma_i z_i = b_i, \quad i \in \mathcal{E}, \\
 & && a_i^\top x + \gamma_i z_i \geq b_i, \quad i \in \mathcal{J}, \\
 & && z \geq 0,
 \end{aligned}$$

where  $e = (1, \dots, 1)^\top$ ,  $\gamma_i = -\text{sign}(a_i^\top \tilde{x} - b_i)$  for  $i \in \mathcal{E}$ , while  $\gamma_i = 1$  for  $i \in \mathcal{J}$ . A feasible initial point for this problem is then

$$x = \tilde{x}, \quad z_i = |a_i^\top \tilde{x} - b_i| \quad (i \in \mathcal{E}), \quad z_i = \max(b_i - a_i^\top \tilde{x}, 0) \quad (i \in \mathcal{J}).$$

It is easy to verify that if  $\tilde{x}$  is feasible for the original problem (A.25), then  $(\tilde{x}, 0)$  is optimal for the feasibility subproblem. In general, if the original problem has feasible points, then the optimal objective value in the subproblem is zero, and any solution of the subproblem yields a feasible point for the original problem. The initial working set  $\mathcal{W}_0$  for the given above algorithm can be found by taking a linearly independent subset of the active constraints at the  $x$  component of the solution of the feasibility problem. (Further remarks on the active-set method can be found in [69]).

# *Notation*

## *Model and controller symbols*

$A, B, C, D$	Plant state space matrices
$A_w, B_w, C_w$	Disturbance model matrices
$J$	Performance objective function
$Q$	Output weighting matrix
$R$	Input weighting matrix
$\bar{Q}$	End-point state weighting matrix
$N$	Prediction horizon
$N_c$	Control horizon
$k$	Discrete time instant
$j$	Time instant on the horizon
$H$	Hessian matrix
$f$	Nonlinear model state equation
$g$	Nonlinear model output equation
$T$	Sampling time
$F_T$	One-step ahead nonlinear prediction
$\Phi, \Gamma$	Prediction matrices in MPC and EKF
$E$	Output error matrix
$G_y$	Output prediction matrix
$P$	ARE solution
$\Sigma$	Covariance matrix
$L$	Kalman filter gain
$t, \epsilon$	Slack variables
$\lambda, p$	Lagrange multipliers
$\mu$	Duality gap
$C_u, C_x, C_c$	Constraint matrices
$\Delta w$	IPM step direction
$\alpha$	Step length
$W$	Linearized KKT matrix
$r$	IPM residuals
$\Pi, \pi$	Structured IPM auxiliary variables

## *Signals*

$d$	Disturbance signal
$u$	Input signal

---

$\delta u$	Optimizing control input
$u_{\text{nom}}$	Nominal input trajectory
$v$	Measurement noise
$w$	Discrete-time white noise
$x$	Plant state variable
$x_w$	Disturbance model state variable
$y$	Measured output
$y_{\text{nom}}$	Nominal output trajectory
$y_{\text{ref}}$	Output reference trajectory
$z$	Performance output

### ***Superscripts***

$i$	IPM iteration
$d$	Discrete-time system

### ***Abbreviations***

APC	Advanced Process Control
ARE	Algebraic Riccati Equation
ASM	Active Set Method
DAE	Differential Algebraic Equations
DARE	Discrete Algebraic Riccati Equation
DCS	Distributed Control System
DMC	Dynamic Matrix Control
EKF	Extended Kalman Filter
FIR	Finite Impulse Response
IDCOM	Identification and Command
INCOOP	INtegration of process COntrol and plantwide OPTimization
IPM	Interior-Point Method
KKT	Karush-Kuhn-Tucker conditions
LP	Linear Programming
LPV	Linear Parameter-Varying
LQ	Linear Quadratic
LQG	Linear Quadratic Gaussian
LQR	Linear Quadratic Regulator
LTV	Linear Time-Varying
MIMO	Multi Input Multi Output
MP	Mathematical Programming
MPC	Model Predictive Control
NLP	Nonlinear Programming
NMPC	Nonlinear Model Predictive Control

---

QP	Quadratic Programming
RDE	Riccati Difference Equation
RTDO	Real-Time Dynamic Optimization
SIPM	Structured Interior-Point Method
SQP	Sequential Quadratic Programming



## *Bibliography*

- [1] F. Allgöwer, T.A. Badgwell, S.J. Qin, J.B. Rawlings, and S.J. Wright. Nonlinear predictive control and moving horizon estimation – an introductory overview. In P.M. Frank, editor, *Advances in Control: Highlights of ECC'99*, pages 391–449. Springer, London, 1999.
- [2] F. Allgöwer and A. Zheng, editors. *Nonlinear model predictive control*, volume 26 of *Progress in Systems and Control Theory*. Birkhauser, 2000.
- [3] E.D. Andersen and K.D. Andersen. The MOSEK interior-point optimizer for linear programming: an implementation of the homogeneous algorithm. In H. Frenk, K. Roos, T. Terlaky, and S. Zhang, editors, *High Performance Optimization*, pages 197–232. Kluwer Academic Publishers, 2000.
- [4] B.D.O. Anderson and J.B. Moore. *Optimal filtering*. Prentice Hall, 1979.
- [5] T.A. Badgwell. A robust model predictive control algorithm for stable linear plants. In *Proc. American Control Conference*, pages 1618–1622, Albuquerque, New Mexico, June 1997.
- [6] T.A. Badgwell. A robust model predictive control algorithm for stable nonlinear plants. In *Proc. IFAC Advanced Control of Chemical Processes*, pages 535–539, Banff, Canada, 1997.
- [7] L.T. Biegler, J.J. Damiano, and G.E. Blau. Nonlinear parameter estimation: a case study comparison. *AIChE Journal*, 32:29–45, 1986.
- [8] R.R. Bitmead, M. Gevers, and V. Wertz. *Adaptive optimal control, the thinking man's GPC*. Prentice Hall, Englewood Cliffs, New Jersey, 1990.
- [9] A.E. Bryson and Y-C. Ho. *Applied optimal control: optimization, estimation and control*. Hemisphere, Washington, 1975.
- [10] J. Buijs, J. Ludlage, W. van Brempt, and B. De Moor. Quadratic programming in model predictive control for large scale systems. In *Proc. IFAC World Congress*, Barselona, Spain, 2002.
- [11] E.F. Camacho and C. Bordons. *Model predictive control*. Springer, London, 1999.
- [12] M. Caracotsios and W.E. Stewart. Sensitivity analysis of initial value problems with mixed odes and algebraic equations. *Computers and Chemical Engineering*, 9:359–365, 1985.



- [13] H. Chen and F. Allgöwer. Maximal yield control of nonlinear chemical reactors. In *Proc. 1st IFAC YAC'95*, pages 764–769, Beijing, China, 1995.
- [14] H. Chen and F. Allgöwer. A quasi-infinite horizon predictive control scheme for constrained nonlinear systems. In *Proc. 16th Chinese Control Conference CCC'96*, pages 309–316, Qindao, China, 1996.
- [15] H. Chen and F. Allgöwer. A quasi-infinite horizon nonlinear predictive control scheme for stable systems: Application to a CSTR. In *Proc. IFAC Advanced Control of Chemical Processes*, pages 529–534, Banff, Canada, 1997.
- [16] H. Chen and F. Allgöwer. Nonlinear model predictive control schemes with guaranteed stability. In R. Berber and C. Kravaris, editors, *Nonlinear Model Based Process Control*, pages 465–494. Kluwer Academic Publishers, Dordrecht, 1998.
- [17] H. Chen and F. Allgöwer. A quasi-infinite horizon nonlinear model predictive control scheme with guaranteed stability. *Automatica*, 34(10):1205–1218, 1998.
- [18] H. Chen, A. Kremling, and F. Allgöwer. Nonlinear predictive control of a benchmark CSTR. In *Proc. 3rd European Control Conference*, pages 3247–3252, Rome, Italy, 1995.
- [19] H. Chen, C.W. Scherer, and F. Allgöwer. A game theoretic approach to nonlinear robust receding horizon control of constrained systems. In *Proc. American Control Conference*, pages 3073–3077, Albuquerque, New Mexico, June 1997.
- [20] H. Chen, C.W. Scherer, and F. Allgöwer. A robust model predictive control scheme for constrained linear systems. In *5th IFAC Symposium on Dynamics and Control of Process Systems, DYCOPS-5*, pages 60–65, Korfu, 1998.
- [21] L.L. Chisci, A. Lombardi, and E. Mosca. Dual receding horizon control of constrained discrete-time systems. *European Journal of Control*, 2:278–285, 1996.
- [22] R. de Keyser. A gentle introduction to model based predictive control. In *EC-PADI2 Int. Conference on Control Engineering and Signal Processing*, Peru, 1998.
- [23] G. De Nicolao, L. Magni, and R. Scattolini. On the robustness of receding-horizon control with terminal constraints. *IEEE Transactions on Automatic Control*, 41(3):451–453, 1996.

- [24] G. De Nicolao, L. Magni, and R. Scattolini. Stabilizing receding-horizon control of nonlinear time-varying systems. In *Proc. 4th European Control Conference ECC'97*, Brussels, Belgium, 1997.
- [25] G. De Nicolao, L. Magni, and R. Scattolini. Stability and robustness of nonlinear receding horizon control. In F. Allgöwer and A. Zheng, editors, *Nonlinear model predictive control*, volume 26 of *Progress in Systems and Control Theory*. Birkhauser, 2000.
- [26] S.L. De Oliveira. *Model predictive control for constrained nonlinear systems*. PhD thesis, Swiss Federal Institute of Technology (ETH), Zürich, Switzerland, 1996.
- [27] E. Demoro, C. Alexrud, D. Johnston, and G. Martin. Neural network modeling and control of polypropylene process. In *Society of Plastic Engineers Int'l Conference*, Houston, Texas, February 1997.
- [28] R. Findeisen and F. Allgöwer. Suboptimal infinite horizon nonlinear model predictive control for discrete-time systems. Technical Report 97.13, Automatic Control Laboratory, Swiss Federal Institute of Technology (ETH), Zürich, Switzerland, 1997. Presented at the NATO Advanced Study Institute on Nonlinear Model Based Process Control.
- [29] W. Fred Ramirez. *Process control and identification*. Academic Press, New York, NY, 1994.
- [30] C.E. García and M. Morari. Internal model control: A unifying review and some new results. *Ind. Eng. Chem. Proc. Des. Dev.*, 21:308–323, 1982.
- [31] H. Genceli and M. Nikolaou. Robust stability analysis of constrained  $L_1$ -norm model predictive control. *AIChE J.*, 39(12):1954–1965, 1993.
- [32] H. Genceli and M. Nikolaou. Design of robust constrained model-predictive controllers with volterra series. *AIChE J.*, 41(9):2098–2107, 1995.
- [33] K. Glover. All optimal Hankel-norm approximations of linear multivariable systems and their  $l_\infty$ -error bounds. *Int. J. Control*, 39:1115–1193, 1974.
- [34] S.J. Hammarling. Numerical solution of the stable, non-negative definite Lyapunov equation. *IMA J. Numerical Analysis*, 2:303–323, 1982.
- [35] J. Kadam, M. Schlegel, and W. Marquardt. Integrated dynamic optimization and control applied to the Bayer process. In *INCOOP Workshop materials*, Düsseldorf, Germany, 2003.

- [36] J.V. Kadam, W. Marquardt, M. Schlegel, T. Backx, O.H. Bosgra, P.-J. Brouwer, G. Dünnebier, D. van Hessem, A. Tiagounov, and S. de Wolf. Towards integrated dynamic real-time optimization and control of industrial processes. In I.E. Grossmann and C.M. McDonald, editors, *Proc. FOCAPO*, pages 593–596, 2003.
- [37] R.E. Kalman. Contributions to the theory of optimal control. *Bull. Soc. Math. Mex.*, 5:102–119, 1960.
- [38] R.E. Kalman and R.S. Bucy. New results in linear filtering and prediction theory. *Trans. ASME, J. Basic Engineering*, pages 95–108, March 1961.
- [39] S.S. Keerthi and E.G. Gilbert. Optimal infinite-horizon feedback laws for a general class of constrained discrete-time systems: Stability and moving horizon approximations. *J. Opt. Theory and Appl.*, 57(2):265–293, May 1988.
- [40] D.L. Kleinman. An easy way to stabilize a linear constant system. *IEEE Trans. Automat. Contr.*, 15(692), 1970.
- [41] M.V. Kothare, V. Balakrishnan, and M. Morari. Robust constrained model predictive control using linear matrix inequalities. *Automatica*, 32(10):1361–1379, 1996.
- [42] W.H. Kwon. Advances in predictive control: Theory and application. In *Proceedings of first Asian Control Conference*, Tokyo, 1994.
- [43] W.H. Kwon and A.E. Pearson. A modified quadratic cost problem and feedback stabilization of a linear system. *IEEE Transactions on Automatic Control*, 22(5):838–842, 1977.
- [44] L.S. Lasdon and A.D. Waren. *GRG2 User’s Guide*. Department of Computer and Information Science, Cleveland State University, Cleveland, Ohio, 1986.
- [45] J.H. Lee and B. Cooley. Recent advances in model predictive control and other related areas. In J.C. Kantor, C.E. Garcia, and B. Carnahan, editors, *Fifth International Conference on Chemical Process Control*, pages 201–216. AIChE and CACHE, 1997.
- [46] J.H. Lee and N.L. Ricker. Extended kalman filter based nonlinear model predictive control. *Ind. Eng. Chem. Res.*, 33:1530–1541, 1994.
- [47] J.H. Lee and Z.H. Yu. Worst-case formulations of model predictive control for systems with bounded parameters. *Automatica*, 33(5):763–781, 1997.

- [48] D.B. Leineweber. *Efficient reduced SQP methods for the optimization of chemical processes described by large sparse DAE models*. PhD thesis, Universität Heidelberg, Germany, 1998.
- [49] Y. Liu and B.D.O. Anderson. Singular perturbation approximation of balanced systems. *Int. J. Control*, 50:1379–1405, 1989.
- [50] J.M. Maciejowski. *Predictive control with constraints*. Prentice Hall, 2002.
- [51] L. Magni. *Nonlinear receding horizon control: Theory and Application*. PhD thesis, Università degli Studi di Pavia, 1998.
- [52] G. Martin, G. Boe, J. Keeler, D. Timmer, and J. Havener. Method and apparatus for modelling dynamic and steady-state processes for prediction, control and optimization. US Patent, 1998.
- [53] G. Martin and D. Johnston. Continuous model-based optimization. In *Process Optimization Conference*, Houston, Texas, March 1998. Gulf Publishing Co. and Hydrocarbon Processing.
- [54] D.Q. Mayne. Nonlinear model predictive control: An assessment. In J.C. Kantor, C.E. Garcia, and B. Carnahan, editors, *Fifth International Conference on Chemical Process Control*, pages 217–231. AIChE and CACHE, 1997.
- [55] D.Q. Mayne. Nonlinear model predictive control: Challenges and opportunities. In F. Allgöwer and A. Zheng, editors, *Nonlinear model predictive control*, volume 26 of *Progress in Systems and Control Theory*. Birkhauser, 2000.
- [56] D.Q. Mayne and H. Michalska. Receding horizon control of nonlinear systems. *IEEE Transactions on Automatic Control*, 35(7):814–824, July 1990.
- [57] D.Q. Mayne, J.B. Rawlings, C.V. Rao, and P.O.M. Scokaert. Constrained model predictive control: Stability and optimality. *Automatica*, 36:789–814, 2000.
- [58] E.S. Meadows, M.A. Henson, J.W. Eaton, and J.B. Rawlings. Receding horizon control and discontinuous state feedback stabilization. *Int. J. Contr.*, 62(5):1217–1229, 1995.
- [59] E.S. Meadows and J.B. Rawlings. Model predictive control. In M.A. Henson and D.E. Seborg, editors, *Nonlinear process control*, chapter 5, pages 233–310. Prentice Hall, 1997.

- [60] S. Mehrotra. On the implementation of a primal-dual interior point method. *SIAM J. Optim.*, 2(1):575–601, 1992.
- [61] H. Michalska and D.Q. Mayne. Robust receding horizon control of constrained nonlinear systems. *IEEE Transactions on Automatic Control*, 38(11):1623–1633, November 1993.
- [62] B.C. Moore. Principal component analysis in linear system: controllability, observability and model reduction. *IEEE Trans. Autom. Control*, 26:17–32, 1981.
- [63] M. Morari. Robust stability of systems with integral control. In *Proc. 22nd IEEE Conf. on Decision and Control*, pages 865–869, Los Angeles, CA, 1983.
- [64] M. Morari and J.H. Lee. Model predictive control: Past, present and future. In *Proceedings of PSE/Escap '97*, Trondheim, Norway, 1997.
- [65] E. Mosca. *Optimal, predictive and adaptive control*. Information and System Science Series. Prentice Hall, Englewood Cliffs, New Jersey, 1994.
- [66] K.R. Muske and J.B. Rawlings. Model predictive control with linear models. *AIChE Journal*, 39(2):262–287, 1993.
- [67] S.G. Nash and A. Sofer. *Linear and nonlinear programming*. The McGraw-Hill Companies, Inc., 1996.
- [68] R.B. Newell and P.L. Lee. *Applied process control. A case study*. Prentice Hall, 1989.
- [69] J. Nocedal and S.J. Wright. *Numerical optimization*. Springer Series in Operations Research, 1999.
- [70] N.M.C. Oliveira and L.T. Biegler. Constraint handling and stability properties of model predictive control. *AIChE J.*, 40:1138–1155, 1994.
- [71] N.M.C. Oliveira and L.T. Biegler. An extension of newton-type algorithms for nonlinear process control. *Automatica*, 31:281–286, 1995.
- [72] D.M. Prett and R.D. Gillette. Optimization and constrained multivariable control of a catalytic cracking unit. In *Proceedings of the Joint Automatic Control Conference*, 1980.
- [73] J.S. Qin and T.A. Badgwell. An overview of nonlinear model predictive control applications. In F. Allgöwer and A. Zheng, editors, *Nonlinear model predictive control*, volume 26 of *Progress in Systems and Control Theory*. Birkhauser, 2000.

- [74] S.J. Qin and T.A. Badgwell. An overview of industrial model predictive control technology. In J.C. Kantor, C.E. Garcia, and B. Carnahan, editors, *AIChE Symposium Series: Fifth Int. Conf. on Chemical Process Control*, volume 316, pages 232–256, 1997.
- [75] C.V. Rao, S.J. Wright, and J.B. Rawlings. Application of interior-point methods to model predictive control. *Journal of Optimization Theory and Applications*, 99:723–757, 1998.
- [76] J.B. Rawlings. Tutorial: Model predictive control technology. In *Proc. American Control Conference*, pages 662–676, San Diego, California, June 1999.
- [77] J.B. Rawlings, E.S. Meadows, and K.R. Muske. Nonlinear model predictive control: A tutorial and survey. In *Proc. IFAC Advanced Control of Chemical Processes*, pages 185–197, Kyoto, Japan, 1994.
- [78] J.B. Rawlings and K.R. Muske. The stability of constrained receding horizon control. *IEEE Transactions on Automatic Control*, 38(10):1512–1516, 1993.
- [79] J. Richalet, A. Rault, J.L. Testud, and J. Papon. Model predictive heuristic control: Applications to industrial processes. *Automatica*, 14:413–428, 1978.
- [80] N.L. Ricker and J.H. Lee. Nonlinear model predictive control of the tennessee eastman challenge process. *Computers and Chemical Engineering*, 19(9):961–981, 1995.
- [81] M.G. Safonov and R.Y. Chiang. A schur method for balanced-truncation model reduction. *IEEE Trans. Autom. Control*, 34:729–733, 1989.
- [82] L.O. Santos, N.M.C. de Oliveira, and L.T. Biegler. Reliable and efficient optimization strategies for nonlinear model predictive control. In *Proc. of the IFAC Symposium DYCORN+95*, pages 33–38, Helsingor, Denmark, 1995.
- [83] P.O.M. Scokaert, D.Q. Mayne, and J.B. Rawlings. Suboptimal model predictive control (feasibility implies stability). *IEEE Transactions on Automatic Control*, 44(3):648–654, March 1999.
- [84] S. Skogestad and I. Postlethwaite. *Multivariable feedback control*. Wiley, 1996.
- [85] O. Slupphaug and B.A. Foss. Bilinear matrix inequalities and robust stability of nonlinear multi-model MPC. In *Proc. American Control Conference*, pages 1689–1694, Philadelphia, June 1998.

- [86] R. Soeterboek. *Predictive control: A unified approach*. Prentice-Hall Inc., New York, 1992.
- [87] A. Tiagounov, J. Buijs, S. Weiland, and B. De Moor. Long horizon model predictive control for nonlinear industrial processes. In *Proc. European Control Conference*, Cambridge, UK, 2003.
- [88] A. Tiagounov, D. van Hessem, M. Balenovic, and S. Weiland. From state estimation to long horizon MPC for nonlinear industrial applications. In *INCOOP Workshop materials*, Düsseldorf, Germany, 2003.
- [89] A. Tiagounov and S. Weiland. Model predictive control algorithm for nonlinear chemical processes. In *Proc. Physics and Control Conference*, St.Petersburg, Russia, 2003.
- [90] M.S. Tombs and I. Postlethwaite. Truncated balanced realization of a stable non-minimal state-space system. *Int. J. Control*, 46:1319–1330, 1987.
- [91] A. Varga. Balancing-free square-root algorithm for computing singular perturbation approximations. In *Proc. of 30th IEEE CDC*, pages 1062–1065, Brighton, UK, 1991.
- [92] A. Varga. Efficient minimal realization procedure based on balancing. In *Prepr. of IMACS Symp. on Modelling and Control of Technological Systems*, volume 2, pages 42–47, 1991.
- [93] A. Varga. Coprime factors model reduction based on accuracy enhancing techniques. In *Systems Analysis, Modelling and Simulation*, volume 11, pages 303–311, 1993.
- [94] A. Varga. A schur method for computing coprime factorizations with inner denominators and applications in model reduction. In *Proc. 1993 American Control Conference*, pages 2130–2131, San Francisco, CA, 1993.
- [95] A. Varga. Model reduction routines for slicot. NICONET Report 1999-8, Katholieke Universiteit Leuven, Belgium, December 1999.
- [96] A. Varga and K.H. Fasol. A new square-root balancing-free stochastic truncation model reduction algorithm. In *Prepr. of 12th IFAC World Congress*, volume 7, pages 153–156, Sydney, Australia, 1993.
- [97] S. Weiland, A.A. Stoorvogel, and A. Tiagounov. End-point parametrization and guaranteed stability for a model predictive control scheme. In *Proc. IEEE Conference on Decision and Control*, pages 4832–4837, Orlando, Florida, 2001.

- [98] S.J. Wright. Applying new optimization algorithms to model predictive control. In J.C. Kantor, C.E. Garcia, and B. Carnahan, editors, *Chemical Process Control: Assessment and New Directions for Research*, AICHE Symposium Series 316, pages 147–155. AICHE and CACHE, 1997.
- [99] S.J. Wright. *Primal-dual interior-point methods*. SIAM, Philadelphia, 1997.
- [100] T.H. Yang and E. Polak. Moving horizon control of nonlinear systems with input saturation, disturbances and plant uncertainty. *Int. J. Contr.*, 58(4):875–903, 1993.
- [101] E. Zafiriou. Robust model predictive control of processes with hard constraints. *Comp. & Chem. Eng.*, 14(4/5):359–371, 1990.
- [102] H. Zhao, J.P. Guiver, and G.B. Sentoni. An identification approach to nonlinear state space model for industrial multivariable model predictive control. In *Proc. American Control Conference*, Philadelphia, June 1998.
- [103] A. Zheng and M. Morari. Robust stability of constrained model predictive control. In *Proc. American Control Conference*, pages 379–383, June 1993.
- [104] A. Zheng and W.-H. Zhang. Synthesis of robust nonlinear model predictive controllers for large-scale systems. In *Proc. 14th IFAC World Congress*, pages 25–30, Beijing, China, 1999.





# *Samenvatting*

De proces industrie heeft dringend behoefte aan technieken voor het nauwkeurig, efficient en flexibel bedrijven van haar plants. Dit vereist een doorlopende ontwikkeling van innovatieve technieken en de bijbehorende geïntegreerde software gereedschappen voor het modelleren van de processen, het optimaliseren van de transitiepaden en het modelgebaseerd regelen van de processen. Met de toenemende eisen neigen de modellen, die de procesdynamica beschrijven, te complex te worden voor de huidige generatie modelgebaseerde regel- en optimalisatie technieken. Het doel van het in dit proefschrift beschreven research project is de ontwikkeling van een nieuwe generatie regeltechniek, die aan de strakkere eisen ten aanzien van transitie gedrag en storingsonderdrukking voldoet en de realisatie van een prototype geïntegreerde real-time software omgeving. Het research project is uitgevoerd als onderdeel van een 5e kader programma Europees R&D project getiteld: “Integration of process Control and plantwide dynamic Optimization (INCOOP)”.

De enige geavanceerde regeltechniek, die breed door de procesindustrie is geaccepteerd is de Model Predictive Control (MPC). Het onderzoek beschreven in dit proefschrift spitst zich toe op MPC voor niet-lineaire processen. De optimalisatie binnen niet-lineaire MPC is in het algemeen een zeer rekenintensieve taak. Dit proefschrift beschrijft verschillende MPC algorithmen voor niet-lineaire processen, die gebruik maken van opeenvolgende linearisaties. De predicties worden berekend met behulp van simulaties met het niet-lineaire proces model. Voor het berekenen van een optimale voorspelling van de toekomstige toestandsvector wordt gebruik gemaakt van locale linearisaties van de toestandsvergelijkingen. De voorspellingen van de uitgangen worden berekend op basis van lineaire benaderingen voor de toekomstige ingangs manipulaties. Dit maakt het mogelijk het optimalisatie probleem binnen de MPC op te lossen als een Quadratisch Programmeer (QP) probleem.

In het proefschrift wordt beschreven hoe het QP probleem op verschillende manieren opgelost kan worden. In de eerste plaats kunnen de modelvergelijkingen worden gebruikt om toestanden te elimineren en daarmee het aantal variabelen in de optimalisatie te reduceren. Dit resulteert echter in een nagenoeg volledig gevulde matrices. Het oplossen van een QP met dit soort methoden resulteert in het algemeen in reketijden die kubisch toenemen met het aantal te optimaliseren variabelen. De reketijden worden daarmee in het algemeen veel te lang voor het real-time oplossen van de beoogde niet-lineaire regelproblemen met deze standaard QP algorithmen. Veel industriële processen hebben een dynamisch gedrag dat resulteert in grote, stijve optimalisatieproblemen, die een lange predictie horizon vereisen om aan de gestelde specificaties te kunnen voldoen. Dit soort eisen resulteert dan ook in optimalisatie problemen met veel

variabelen. Standaard implementaties van de QP algorithmen zijn dan ook niet toepasbaar voor het oplossen van de hier beoogde MPC optimalisatie problemen. In dit proefschrift is een gestructureerde interior-point methode (IPM) ontwikkeld, die het MPC probleem voor grote niet-lineaire systemen reduceert tot een complexiteit die real-time oplosbaar is. De in samenwerking met de KU-Leuven (Ir. Jeroen Buijs en Prof. Bart De Moor) ontwikkelde methode gebruikt expliciet de structuur van het op te lossen probleem, waardoor de rekentijd slechts lineair toeneemt met het aantal te optimaliseren variabelen. Het algoritme laat ook het gebruik van meerdere lineaire modellen toe, wat een bredere regeltechnische toepassing ondersteunt. De eliminatie van toestanden wordt hierbij niet gedaan en de structuur, die voortvloeit uit de dynamica van het proces wordt gereflecteerd in de Karush-Kuhn-Tucker (KKT) vergelijkingen, die worden gebruikt om het QP probleem op te lossen. De gestructureerde IPM is gecomplementeerd met behulp van het “primal-dual Mehrotra” algoritme inclusief de predictie, correctie en centreer stappen. De te optimaliseren variabelen zijn de ingangen en de toestanden van het proces over de volledige horizon. Het optimalisatie probleem wordt hierbij echter een ijl probleem. Dit resulteert in een significante reductie van de rekentijd, hetgeen zeer belangrijk is voor het toepassen van MPC op niet-lineaire stijve systemen.

In dit proefschrift wordt de effectiviteit van de gestructureerde IPM gebaseerde MPC aangetoond op diverse industriële processen. Als voorbeelden worden onder andere een CSTR (Continuous Stirred Tank Reactor) en een stijve, niet-lineaire batch reactor gebruikt. Dit soort systemen heeft in het algemeen dynamica, waarvan de tijdconstanten meerdere decades uiteen kunnen liggen. Een aantal regelproblemen, zoals het volgen van een referentie trajectorie, het opstarten van het proces en het onderdrukken van verstoringen, worden efficiënt opgelost met de in dit proefschrift beschreven hoog presterende MPC. De regelaar is verder getest op de processen, die als testcases in INCOOP zijn gebruikt. Ook deze tests toonden dezelfde successen.

With this letter I would like to confirm that I was notified by my supervisors that some members of the promotion committee had remarks on my thesis:

“High-Performance Model Predictive Control for Process Industry”

after my PhD defence on 18 June 2004. The remarks are known to be about improper citation of bibliography references with some parts of text in this thesis copied from other sources.

Below is a list of pages in this thesis together with the bibliography references where the text on those pages was copied from. The references describe well-documented and concise examples of process models which were used only for illustration purposes in this thesis.

Pages 59–63 in Section 2.6.1 from:

- [68] R.B. Newell and P.L. Lee. *Applied process control. A case study*. Prentice Hall, 1989.

Pages 65–67 from:

- [84] S. Skogestad and I. Postlethwaite. *Multivariable feedback control*. Wiley, 1996.

Pages 69–73 from:

- [50] J.M. Maciejowski. *Predictive control with constraints*. Prentice Hall, 2002.

Pages 115–118 from:

- [18] H. Chen, A. Kremling, and F. Allgöwer. Nonlinear predictive control of a benchmark CSTR. In *Proc. 3rd European Control Conference*, pages 3247–3252, Rome, Italy, 1995.
- [13] H. Chen and F. Allgöwer. Maximal yield control of nonlinear chemical reactors. In *Proc. 1st IFAC YAC'95*, pages 764–769, Beijing, China, 1995.
- [15] H. Chen and F. Allgöwer. A quasi-infinite horizon nonlinear predictive control scheme for stable systems: Application to a CSTR. In *Proc. IFAC Advanced Control of Chemical Processes*, pages 529–534, 1997.


Pages 126–128 from:

- [48] D.B. Leineweber. *Efficient reduced SQP methods for the optimization of chemical processes described by large sparse DAE models*. PhD thesis, Universität Heidelberg, Germany, 1998.

Material in Appendix A is directly taken from some parts of the following book containing well-described mathematical background:

- [69] J. Nocedal and S.J. Wright. *Numerical optimization*. Springer Series in Operations Research, 1999.

This survey is complete.



Andrey A. Tyaguinov