INAUGURAL-DISSERTATION
zur
Erlangung der Doktorwürde
der
Naturwissenschaftlich-Mathematischen Gesamtfakultät
der
Ruprecht-Karls-Universität
Heidelberg

vorgelegt von
Dipl. Phys. Moritz Mathias Diehl
aus Hamburg

Tag der mündlichen Prüfung: 25. Juni 2001

Real-Time Optimization for Large Scale Nonlinear Processes

Gutachter: Prof. Dr. Dr. h.c. Hans Georg Bock
Prof. Dr.-Ing. Frank Allgöwer

Abstract

Efficient numerical methods for the real-time solution of optimal control problems arising in *nonlinear model predictive control* (NMPC) are presented. The practical applicability of the methods is demonstrated in an experimental application to a pilot plant distillation column, involving the real-time optimization of a large scale differential algebraic process model, with sampling times of only a few seconds.

The solution approach is based on the direct multiple shooting method, which allows to combine the use of advanced, fully adaptive DAE solvers with the advantages of a simultaneous strategy. The real-time approach is characterized by an initial value embedding strategy, that efficiently exploits solution information in subsequent optimization problems. Dovetailing of the solution iterations with the process development in a real-time iteration scheme allows to reduce sampling times to a minimum, but maintains all advantages of a fully nonlinear treatment of the optimization problems. It is shown how the computations in each real-time iteration can be divided into a preparation phase and a considerably shorter feedback phase, which avoids the delay of one sampling time that is present in all previous NMPC schemes. A Gauss-Newton approach for least squares integrals is realized which allows to compute an excellent Hessian approximation at negligible computational costs.

The contraction properties of the algorithm are investigated theoretically, and contractivity of the real-time iterates is shown under mild conditions. Bounds on the loss of optimality with respect to the optimal solution are established.

In an experimental proof-of-concept study the developed numerical methods are applied to the NMPC of a pilot plant distillation column situated at the *Institut für Systemdynamik und Regelungstechnik* at the University of Stuttgart. A suitable system model is developed, which is stiff and comprises more than 200 state variables, and the system parameters are fitted to experimental data. A variant of the Extended Kalman Filter (EKF) is developed for state estimation. Using the real-time optimization algorithm, sampling times of less than 20 seconds and feedback delays below 400 milliseconds could be realized under practical conditions. The scheme shows good closed-loop performance, especially for large disturbances.

In a numerical experiment, the periodic control of an unstable system, an airborne kite that is flying loopings, is investigated. The algorithm shows excellent robustness and real-time performance for this challenging on-line optimization example.

Keywords

Boundary Value Problems, Constrained Gauss-Newton, Differential Algebraic Equations, Direct Multiple Shooting, Distillation Columns, Dynamic Chemical Processes, Embedding Techniques, Index One DAEs, Large-Scale Systems, Newton Type Methods, Nonlinear Model Predictive Control, Optimal Feedback Control, Periodic Control, Real-Time Optimization

Acknowledgements

I thank my advisors Prof. Dr. Dr. h.c. Hans Georg Bock and Dr. Johannes Schlöder, *Interdisziplinäres Zentrum für Wissenschaftliches Rechnen* (IWR), University of Heidelberg, for their excellent support over the last three years, and for their contribution to major ideas of the presented real-time optimization scheme. It was a pleasure to build on their extraordinary knowledge in the area of dynamic optimization, and their open minded attitude towards applied research deeply influenced my view on mathematics and its applications. I also thank Prof. Dr. Frank Allgöwer, *Institute for Systems Theory in Engineering* (IST), University of Stuttgart, for the direct and indirect support that he provided for my work. His enthusiasm initially convinced me to enter the area of Nonlinear Model Predictive Control, and his talent to ask the right questions at the right time did decisively influence the course of my work.

I owe an enormous debt of gratitude to Rolf Findeisen from the IST, for his willingness and extraordinary ability to explain engineering issues to a numerical analyst, which helped considerably to make a fruitful and lasting collaboration possible. I want to thank him and Dr. Ilknur Uslu and Stefan Schwarzkopf from the *Institut für Systemdynamik und Regelungstechnik* (ISR), University of Stuttgart, for many inspiring discussions and for their commitment to our project, the experimental application of nonlinear model predictive control to a pilot plant distillation column at ISR. All experimental results with the distillation column were obtained together with them. Furthermore, I want to thank Prof. Dr.-Ing. Dr. h.c. mult. Ernst Dieter Gilles, Dr.-Ing. Achim Kienle and Erik Stein, *Max-Planck-Institut für Dynamik komplexer technischer Systeme,* Magdeburg, for support of the project.

Many people have directly or indirectly contributed to the success of this work: first and foremost, I thank Dr. Daniel Leineweber, who not only introduced me into the world of numerical process optimization, but also continued to be a constant source of advice. I had the chance to build on the software in the simulation and optimization group at IWR, especially on his well structured software package MUSCOD-II, which significantly helped towards the successful realization of the numerical methods developed in this thesis. I also thank Andreas Schäfer, the coauthor of MUSCOD-II, for many invaluable discussions about implementation details, and for his generous willingness to help with diverse software problems. It was a pleasure to share a room with him.

I am grateful to Tobias Bürner, who provided valuable help in the area of parameter and state estimation. Dr. Zoltan Nagy from Babes-Bolyai University, Cluj, Romania, developed large parts of the employed distillation model during a research stay in Heidelberg, and I want to thank him and Rolf Findeisen for the productive time we had together in Heidelberg. Moritz Wendt from the *Institut für Prozess- und Anlagentechnik*, Technical University Berlin, gave helpful advice about the modelling of hydrodynamics.

In the very final phase of this work, several colleagues helped with proofreading and critical remarks. I especially thank Wolfgang Bangerth, Ulrich Brandt-Pollmann, Dr. Chrys Correa, Julia Hartmann, Dr. Ekaterina Kostina, Jan Schrage and Dr. Michael Winckler.

I heartily thank our group's secretary Margret Rothfuß for her contribution to the friendly environment in the workgroup, and our system administrator Thomas Kloepfer for many hours he spent helping me in various respects. I like to extend my thanks to the other people of the group for creating the open and constructive atmosphere that made my time there very pleasant.

Here I like to express my gratitude to Prof. Dr. Dr. h.c. mult. Willi Jäger for having incited my interest in applied mathematics through his inspiring introductory lectures and for his support and advice at several stages of my career. I thank him and the managing board of directors of the IWR for creating a great environment for doing interdisciplinary research. I have been particularly lucky to be a member of the IWR's postgraduate program "Modeling and Scientific Computing in Mathematics and Natural Sciences" (Chairman: Prof. Dr. Dr. h.c. Hans Georg Bock), which provided many inspiring interdisciplinary contacts.

Financial support by the *Deutsche Forschungsgemeinschaft* (DFG) within this postgraduate program as well as within the *DFG-Schwerpunktprogramm* "Real-Time Optimization of Large Systems" in the project *"Echtzeit-Optimierung bei großen nichtlinearen DAE Modellen der Verfahrenstechnik am Beispiel gekoppelter Destillationskolonnen"* is gratefully acknowledged.

Finally, I happily thank my friends, my flat mates, my family and my girlfriend Anne for the wonderful support and encouragement at home.

Contents

Acronyms

BDF	Backward Differentiation Formulae
CSTR	Continuous Stirred Tank Reactor
DAE	Differential Algebraic Equation
EKF	Extended Kalman Filter
END	External Numerical Differentiation
HJB	Hamilton-Jacobi-Bellman
IND	Internal Numerical Differentiation
IPM	Interior-Point Method
IVP	Initial Value Problem
LMPC	Linear Model Predictive Control
LQR	Linear Quadratic Regulator
KKT	Karush-Kuhn-Tucker
NLP	Nonlinear Programming
NMPC	Nonlinear Model Predictive Control
ODE	Ordinary Differential Equation
PRSQP	Partially Reduced Sequential Quadratic Programming
QIH	Quasi-Infinite Horizon
QP	Quadratic Programming
RHC	Receding Horizon Control
SQP	Sequential Quadratic Programming
ZTC	Zero Terminal Constraint

List of Selected Symbols

Symbol	Meaning	Defined in
G	equality constraint function	Sec. 3.1
$\tilde{G}$	equalities and active inequalities	Sec. 3.1
$\tilde{G}^{\mathrm{s}}$	equalities and stricly active inequalities	Sec. 3.1
H	inequality constraint function	Sec. 3.1
H^{act}	active components of H,	Sec. 3.1
$H^{\mathrm{s.act}}$	strongly active components of H	Sec. 3.1
$H^{\mathrm{w.act}}$	weakly active components of H	Sec. 3.1
λ	Lagrange multiplier vector for equality constraints	Sec. 3.1
μ	Lagrange multiplier vector for inequality constraints	Sec. 3.1
N	number of multiple shooting intervals	Sec. 2.1
p	constant system parameter vector	Sec. 1.1
$P_{\mathrm{oc}}(x_0)$	optimal control problem for initial value x_0	Sec. 1.1.1
$P(x_0)$	multiple shooting NLP for initial value x_0	Sec. 2.2
$P(t)$	NLP for fixed homotopy parameter t	Chap. 3
$\breve{P}(\breve{t})$	augmented version of $P(t)$	Sec. 3.4
$P_k(x_k)$	k-th shrinking horizon problem, initial value x_k	Sec. 4.3
P^k	fixed control formulation of $P_k(x_k)$	Sec. 5.2.1
q	vector of "free" NLP variables (controls $q_0, \ldots, q_{N-1}$)	Sec. 2.2
q_i	control parameter value on i-th multiple shooting interval	Sec. 2.1
s	vector of "dependent" NLP variables (states $s_0, \ldots, s_N$)	Sec. 2.2
s_i	state node value $s_i = (s_i^x, s_i^z)$	Sec. 2.1
s_i^x	differential state node value	Sec. 2.1
s_i^z	algebraic state node value	Sec. 2.1

Symbol	Meaning	Defined in
T	length of the time horizon	Sec. 1.1.1
$T(x_0)$	optimal horizon length for problem $P_{\mathrm{oc}}(x_0)$	Sec. 1.1.1
t	1) physical time	Sec. 1.1
t	2) scalar parameter for problems $P(t)$	Chap. 3
τ	rescaled time $\tau = t/T$	Sec. 2.1.1
u	control vector	Sec. 1.1
u_S	steady state controls	Sec. 1.2
$u^{\mathrm{f}}(x)$	optimal feedback control (shrinking horizon)	Eq. (1.5)
$u^{\mathrm{f}}_{\infty}(x)$	optimal feedback control (infinite horizon)	Sec. 1.3.2
$u^{\mathrm{f}}_{T}(x)$	NMPC feedback control (horizon length T)	Sec. 1.4
$u^*(t; x_0)$	optimal control trajectory for problem $P_{\mathrm{oc}}(x_0)$	Sec. 1.1.1
w	vector of all NLP variables $w = (q, s)$	Sec. 2.2
x	differential state vector	Sec. 1.1
x_{cl}	differential state of closed-loop system	Sec. 1.3
x_0	initial value	Sec. 1.1
x_S	differential steady state	Sec. 1.2
$x^*(t; x_0)$	optimal differential state trajectory for problem $P_{\mathrm{oc}}(x_0)$	Sec. 1.1.1
y	KKT triple $y = (w, \lambda, \mu)$	Sec. 3.3
y^*	1) solution of off-line problem $(= y^*_0)$	Sec. 5.1
y^*	2) real-time iteration limit point	Sec. 5.2
y^*_k	solution of shrunk problem P^k	Sec. 5.3
z	algebraic state vector	Sec. 1.1
z_{cl}	algebraic state of closed-loop system	Sec. 1.3
$z^*(t; x_0)$	optimal algebraic state trajectory for problem $P_{\mathrm{oc}}(x_0)$	Sec. 1.1.1

Introduction

Optimization techniques have a fundamental impact on current industrial practice. Optimization plays a crucial role not only in operations research and in product design, but also in the design of *dynamic industrial processes*. In many cases, an optimal control problem is solved off-line, i.e., before the actual process operation begins, and a variety of highly developed algorithms have been developed to attack this task.

In practical applications, however, control trajectories that are the result of an off-line optimization are of limited applicability, as the real process does not typically coincide completely with the mathematical model and is most probably subject to disturbances. Therefore, the generation of optimization-based feedback controls is of major practical interest. As optimal feedback controls cannot usually be precalculated in advance for all possible disturbances, the need for *real-time optimization* of the controlled process arises.

Model Predictive Control

The idea of *model predictive control* (MPC) is to determine the control at time t_0 by solving an optimal control problem on a prediction horizon $[t_0, t_0 + T]$ (see Fig. 1). The resulting optimal controls are given to the real process for a short time δ only, and at time $t_0 + \delta$ a new problem is solved on a horizon $[t_0 + \delta, t_0 + T + \delta]$ that is moved forward. A sequence of optimization problems is formulated and solved in real-time, which provides the possibility of reacting to disturbances. *Linear* model predictive control (LMPC), that is based on constrained linear system models, has achieved a state of considerable maturity (cf. García et al [GPM89], Lee et al. [LMG94]). It has had a strong impact on industrial control practice and LMPC techniques are nowadays widely applied, especially in the process industries (cf. Qin and Badgwell [QB96]).

Nonlinear Model Predictive Control

For processes operating during load changes, or for batch and periodic processes, however, nonlinear models that are based on first principles are expected to capture the system behaviour more accurately than linear ones. *Nonlinear* model predictive control (NMPC) promises to increase productivity and control performance and has long been investigated theoretically (for overview articles see e.g. Rawlings et al. [RMM94], Allgöwer et al. [ABQ+99], De Nicolao et al. [DMS00], or Mayne [May00]).

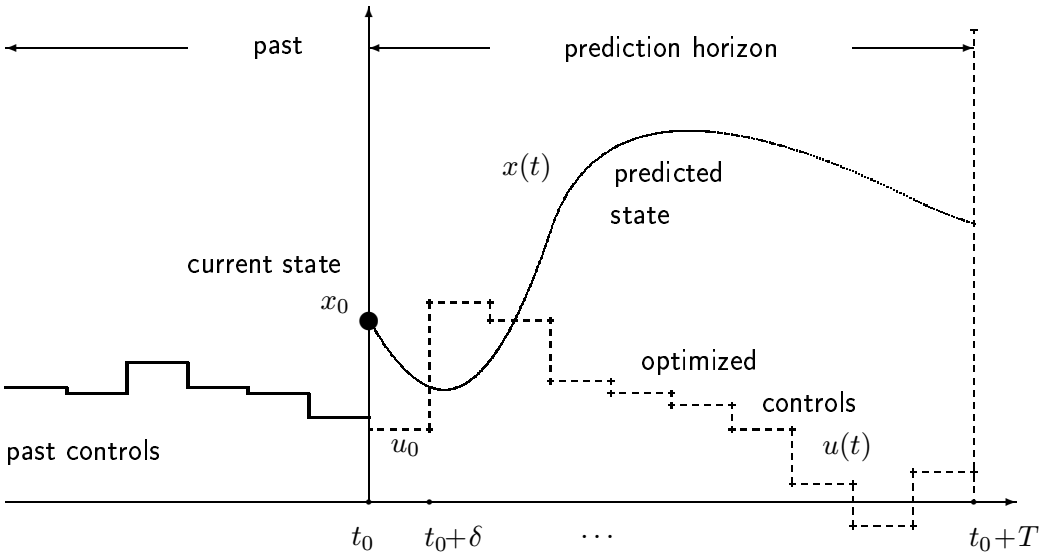


Figure 1: The principle of model predictive control: The optimization problem at time t_0, for the current system state x_0.

In the industry, however, NMPC is still being perceived as an academic concept rather than a practicable control strategy, and in a recent survey, Qin and Badgwell [QB00] report only 88 NMPC applications worldwide, only 5 of which are based on first principle models. As detailed nonlinear process models are increasingly being used for the design of industrial processes (see, e.g. Pantelides and Barton [PB93], Ross et al. [RBP+99], or Sorensen [Sor99]) they may, as a byproduct, also become easily available for NMPC applications.

The difficulty of solving the arising optimal control problems in real-time, however, is widely regarded as the principal impediment to a practical application of NMPC. In their survey, Qin and Badgwell [QB00] point out that "speed and the assurance of a reliable solution in real-time are major limiting factors in existing applications."

In this thesis, we present a new approach to respond to the challenge of real-time optimization in NMPC.

Existing Real-Time Optimization Approaches

In the last decade, the area of numerical techniques for the on-line solution of dynamic optimization problems in NMPC has undergone rapid development. Most real-time approaches are applications of optimal control methods which were originally developed for off-line use, and therefore they can be easily classified within the established framework of dynamic optimization methods.

Dynamic optimization algorithms based on the *direct* solution approach have proven to be particularly successful for the practical solution of constrained optimal control problems. In the direct approach, the originally infinite optimal control problem is parameterized to yield a finite dimensional *Nonlinear Programming* (NLP) problem, that can be solved efficiently by highly developed variants of *sequential quadratic programming* (SQP) (Han [Han76] and Powell [Pow78]). A variety of strategies for formulating the finite dimensional NLP exists. The approaches can roughly be classified into *sequential* and *simultaneous* solution strategies.

Sequential Approach

The sequential approach parameterizes the control trajectory and eliminates the corresponding state trajectory from the optimization problem by a numerical solution of the dynamic model equations (cf. Hicks and Ray [HR71], Sargent and Sullivan [SS78], Kraft [Kra85]). Only the control parameters remain as degrees of freedom in the NLP. Simulation and optimization calculations are performed *sequentially*, one after the other. The approach can easily be coupled with advanced simulation tools and is applied in many practical off-line applications (cf. e.g. Pantelides et al. [PSV94], Vassiliadis [VSP94a, VSP94b], Engl et al. [EKKvS99]; an overview of existing software packages can be found in Binder et al. [BBB$^+$01]).

Many real-time optimization schemes for NMPC are based on the sequential approach. We particularly mention the so called *multistep, Newton-type control algorithm* that was proposed by Li and Biegler [LB89] and de Oliveira and Biegler [OB95b] and which corresponds to a constrained Gauss-Newton method. This approach was often applied for numerical tests of NMPC, see e.g. Abel et al. [ADM95] and M'hamdi et al. [MHAM96]. A sequential approach was also used, e.g., by Weber [Web95] and Chen [Che97]. For a large-scale application of the sequential approach in real-time optimization see also Kronseder et al. [KvSB01].

Sequential optimization schemes for NMPC suffer from the drawback that poor initial guesses for the control trajectory may lead the predicted state trajectories far away from desired reference trajectories. This often causes an unnecessarily strong nonlinearity of the resulting NLPs and poor convergence behaviour, especially for unstable systems. In some cases, an open-loop simulation on a longer horizon is even impossible (see Fig 8.4 in Chap. 8 for an example).

Simultaneous Approach

The simultaneous approach avoids this difficulty by parameterizing both, the control *and* the state trajectory, and by solving the dynamic model equations and the control optimization problem *simultaneously* in a large constrained NLP. The parameterized state trajectory becomes a part of the optimization variables, and instability and nonlinearity of the dynamic model can be better controlled.

Many researchers have applied *collocation* in order to parameterize the dynamic model (see e.g. Tsang et al. [THE75], Bock [Boc83], Biegler [Bie84], Cuthrell and Biegler [CB89], Schulz [Sch96]), resulting in very large, but also very sparse NLPs. The collocation approach has been proposed for the solution of NMPC optimization problems by Biegler in [Bie00].

A second simultaneous approach to optimal control, the *direct multiple shooting* method, was presented by Plitt in 1981 [Pli81]. This method forms the basis for our real-time algorithm. The optimization horizon of interest is divided into a number of subintervals with local control parameters, and the dynamic model equations are solved independently on each of these subintervals, cf. Chap. 2. Continuity of the state trajectory from one interval to the next is enforced on the NLP level only, thus offering the possibility to deal with unstable and strongly nonlinear system models, as collocation. The method has long been known as a fast off-line optimization method in ODE and DAE (see e.g. Bock et al. [BP84, BES88, Boc87, BBLS99], Tanartkit and Biegler [TB95, TB96], Leineweber [Lei96, Lei99], Petzold et al. [PRG$^+$97], Hinsberger et al. [HMP96, Hin98]).

Recently, the direct multiple shooting method was proposed for real-time optimization problems and NMPC. Santos et al. [SOB95] emphasize the strength of the method in dealing with unstable modes and apply it to an NMPC simulation example, the Tennesse Eastman problem (Downs and Vogel [DV93]), but do not address the question of real-time feasibility. Leineweber et al. [LBS97] proposes a scheme for the fast reoptimization of batch processes after large disturbances and presents an application example from biochemical engineering.

In the last year, an *experimental* feasibility study of NMPC based on the conventional direct multiple shooting method has been presented by Santos et al. [SAC$^+$00, San00], for an experimentally simulated unstable continuous stirred tank reactor. The nonlinear first principle model consists of four differential states and two controls. Choosing short prediction horizons, optimization times of a few seconds are realized, which is sufficiently fast for the considered example.

Despite these successful applications of direct multiple shooting to real-time optimization, the use of an algorithm that was essentially designed for off-line use certainly has its limits, and this fact is reflected in the moderate system sizes of the above examples. Large scale problems with strict real-time constraints have therefore not been treated in experimental application examples so far.

The Conventional Scheme

Most numerical real-time optimization schemes are based on the idea that one moving horizon optimization problem can be formulated after the other, and that each of these problems can be solved independently, with higher or lower accuracy. The solution method itself is, in direct approaches, typically an iterative SQP type method. The following algorithmic scheme summarizes the conventional scheme:

1. Formulate an optimization problem according to the k-th data

   2. Initialize the solution procedure.
   3. Perform iterations.
   4. Stop when a termination criterion is satisfied (or when the time limit is reached)
5. Give the first control value to the plant.
6. Increase k by one and go to 1.

The focus is on chosing an efficient off-line method and to formulate the optimization problems in such a way that the real-time requirements can be met. Note that a delay of one sampling time is present in this scheme.

The Real-Time Iteration Scheme

In contrast to a conventional scheme, our *real-time iteration* approach shifts the focus from the sequence of optimization problems towards the solution algorithm itself. The algorithm is regarded to be iterating continuously – and while the algorithm is iterating, the problem data are modified from one iteration to the next. The scheme can be sketched as follows:

1. Prepare the k-th real-time iteration as far as possible without knowledge of the k-th data.
2. When the k-th data are available, modify the problem, and perform quickly those calculations that are necessary to obtain the first control value.
3. Give this control value immediately to the plant.
4. Perform the remaining calculations of the k-th iterate.
5. Increase k by one and go to 1.

This approach does only perform one iteration per sampling time and thus allows to reduce the sampling times considerably. Furthermore, the feedback step 2 is itself much shorter than a full iteration, so that the response delay can practically be avoided. Note that the scheme still offers the advantages of a fully nonlinear treatment of the optimization problems.

The approach can only perform well if the iteration scheme has good contraction properties – this is typically the case for simultaneous approaches like direct multiple shooting – and if the problem modifications are implemented in such a way that they have minimal interference on the iterates.

The Initial Value Embedding Strategy

The crucial observation is that essentially *one* parameter suffices to distinguish between different optimization problems, the initial value x_0 of the state trajectory (cf. Fig. 1). If derivative information with respect to x_0 is available, which is the case for simultaneous solution approaches, neighboring problems can be initialized very efficiently by a so called *initial value embedding* strategy. After each problem modification, the strategy obtains an excellent first order correction in the state and control trajectory that is based on the *previous* system linearization. Roughly spoken, the approach allows the inclusion of linear MPC feedback into the predicted trajectory, *before* a new system linearization is performed. The approach exploits the similarity between subsequent problems as much as possible. In conjunction with the excellent contraction properties of a simultaneous solution approach like direct multiple shooting, the real-time iterates stay very close to the exact solutions of the optimization problem.

The idea to dovetail the solution iterations by employing the initial value embedding idea was first proposed by Bock et al. [BDLS00], with a focus on shrinking horizon processes. The initial value embedding strategy *without* a dovetailing of iterations and process was implemented in a first version of the on-line direct multiple shooting method (Diehl [Die98]), and several numerical feasibility studies have been carried out with this algorithm: in Diehl et al. [DBLS99] real-time feasibility of the NMPC of a continuous stirred tank reactor is shown for rather long control horizons (cf. Sec 1.2); in Nagy et al. [NFD$^+$00], Allgöwer et al. [AFN$^+$00], and Findeisen et al. [FAD$^+$00] the NMPC of a large scale process control example, namely a binary distillation column, is considered, and real-time feasibility is demonstrated in numerical simulations.

In this thesis we present the newly developed real-time iteration scheme and investigate the contraction properties of the approach, and present experimental results that have been obtained by an application of the developed algorithm to the NMPC of a pilot plant distillation column at the *Institut für Systemdynamik und Regelungstechnik* (ISR), University of Stuttgart, employing a stiff DAE optimization model with over 200 states.

We mention here that several singular features of algorithm have been presented by other researchers in the area of practical real-time optimization.

In particular, a one-iteration scheme has been proposed by Li and Biegler [LB89], for the sequential approach. Their scheme, however, did not include the initial value embedding strategy for the initialization from one problem to the next, and it seems that the scheme was not further pursued in application examples. In a subsequent paper, de Oliveira and Biegler [OB95a] focus on the converged form of the algorithm, which essentially equals a conventional Gauss-Newton method for the sequential approach.[1]

In the application of conventional optimization schemes to on-line control, the question of how to initialize subsequent problems has found some attention in the literature. Liebman [LEL92] observes that warm starts of the optimization algorithm can save up to 80%

[1] Note that it would be possible to combine the initial value embedding idea with a sequential approach, though it is not as straightforward as for simultaneous approaches.

computation time, cf. also Biegler and Rawlings [BR91]. A shift strategy that accounts for the movement of the optimization horizon forward in time is proposed, e.g., by de Oliveira and Biegler [OB95a] for the sequential approach.

Highlights of the Thesis and Overview

The aim of this thesis is threefold. First, we want to describe in full detail how the real-time iteration scheme can be realized for the direct multiple shooting method. This is done in Chapters 2, 4 and 6. Secondly, the theoretical properties of the scheme are investigated in Chapter 5, which contains a contractivity result and bounds on the loss of optimality. Finally, we demonstrate the practical applicability of the approach in an *experimental* study that involves the NMPC of a pilot plant distillation column, which is modelled by a large scale process model (Chapter 7), and show in a simulation study that the approach can successfully be applied to an unstable periodic control example with strict real-time requirements (Chapter 8).

1. In Chapter 1, we introduce the class of real-time optimal control problems that can be treated with our approach. We also introduce a guiding example problem from chemical engineering that will be used several times in the thesis for illustrative purposes. Some theory regarding optimal feedback control and nonlinear model predictive control is briefly reviewed.

2. The direct multiple shooting parameterization is reviewed in Chapter 2 and the parameterized nonlinear programming (NLP) problem that will be regarded in the remainder of this thesis is formulated and discussed.

3. In Chapter 3, we recall optimality conditions for constrained NLPs and review a result from parametric optimization, which investigates the solution of neighboring optimization problems. The Sequential Quadratic Programming (SQP) technique is described, and its astonishing power in the solution of perturbed optimization problems is shown for a one dimensional analog of the initial value embedding strategy.

4. The new real-time iteration algorithm is presented in Chapter 4. We present the initial value embedding strategy and show how the approach can be realized on shrinking and on moving horizons.

5. Chapter 5 contains the major theoretical results of this thesis. After a review of the convergence properties for general off-line Newton type methods in Sec. 5.1, we show contractivity of the real-time iterates for the on-line problem on shrinking horizons (Sec. 5.2). The contractivity result is exploited to investigate the properties of the on-line solution, compared to the optimal off-line solution.

6. The specific algorithmic realization of one real-time iteration is described in Chapter 6. The chapter mostly presents well known techniques from the off-line direct

multiple shooting method in a new setting, leading to the division into preparation and feedback phase. However, a newly developed Gauss-Newton approach for least squares integrals is presented in Sec. 6.4, which can be employed for both, the off- and the on-line direct multiple shooting method.

7. Experimental results are presented in Chapter 7. The study involves the NMPC of a pilot plant distillation column using a stiff differential algebraic optimization model with over 200 states. We develop the system model and describe how the system parameters were determined using experimental data from the real column. The experimental results demonstrate that NMPC with a large scale process model is feasible.

8. To demonstrate the power and versatility of the proposed real-time iteration scheme, we present in Chapter 8 numerical simulations for an unstable periodic control example, namely an airborne kite. Control aim is to let the kite fly loopings. A new kite model is developed and a periodic orbit determined. Numerical tests show the real-time feasibility and an astonishing robustness of the real-time optimization approach even for large disturbances.

9. We finally conclude this thesis with a summary and an outlook of interesting future developments.

The developed real-time algorithm, that has also been presented in some publications (Bock et al. [BDS+00], Diehl et al. [DBS+01, DUF+01]), is currently considered for use in an industrial application.

Chapter 1

Real-Time Optimal Control

In this chapter we will first introduce a general class of optimal control problems for which our algorithms are designed, and review some theory regarding optimal feedback control and nonlinear model predictive control.

1.1 Optimal Control Problems in DAE

Differential Algebraic System Models

Let us assume that a system that we want to control can be described by a differential-algebraic equation (DAE) model of the following form:

$$\begin{aligned} B(x(t), z(t), u(t), p, t) \cdot \dot{x}(t) &= f(x(t), z(t), u(t), p, t) \\ 0 &= g(x(t), z(t), u(t), p, t). \end{aligned}$$

Here, $x \in \mathbb{R}^{n_x}$ and $z \in \mathbb{R}^{n_z}$ denote the differential and the algebraic state vectors, respectively, $u \in \mathbb{R}^{n_u}$ is the vector valued control function, whereas $p \in \mathbb{R}^{n_p}$ is a vector of constant system parameters such as reaction constants or material parameters.

We also assume that the Jacobian $\frac{\partial g}{\partial z}(\cdot)$ and the matrix $B(\cdot)$ are invertible, so that the DAE is of index-one and of semi-explicit type.

Objective Functional

Let us introduce a general Bolza type objective functional on a time horizon $[t_0, t_f]$ with start time t_0 and final time t_f

$$\int_{t_0}^{t_f} L(x(t), z(t), u(t), p, t)\, dt \quad + \quad E(x(t_f), z(t_f), p, t_f),$$

where L is often called the Lagrange term, and E the Mayer term of the objective. This objective functional defines the overall "costs" that shall be minimized.

Least Squares Objectives for Tracking Problems: An important subclass of optimal control problems are tracking problems that have as their aim to determine controls that lead the system state or more general an output function $l(x(t), z(t), u(t), p, t) \in \mathbb{R}^{n_l}$ "close" to some specified reference output trajectory $l^r(p,t) \in \mathbb{R}^{n_l}$ on the interval $t \in [t_0, t_f]$. Typically, the distance from the reference trajectory is measured by the integral of a squared difference, that may be weighted by a positive definite matrix Q, so that the integral

$$\int_{t_0}^{t_f} \|Q^{\frac{1}{2}} \cdot (l(x(t), z(t), u(t), p, t) - l^r(t, p))\|_2^2 \, dt$$

shall be minimized.[1]

By redefining $l(x(t), z(t), u(t), p, t)$, we can assume that $Q = \mathbb{I}$ and $l^r(p,t) = 0$, $\forall\, t \in [t_0, t_f]$. By also introducing a least squares Mayer term with a vector valued residual function $e(x(t_f), z(t_f), p, t_f) \in \mathbb{R}^{n_e}$, the general form of an objective functional in least squares form is given as

$$\int_{t_0}^{t_f} \|l(x(t), z(t), u(t), p, t)\|_2^2 \, dt \quad + \quad \|e(x(t_f), z(t_f), p, t_f)\|_2^2.$$

This form can be exploited for the efficient solution of the optimization problems by a *Gauss-Newton* approach that is presented in Section 6.4.

Path Constraints and Boundary Conditions

The state and control trajectories are required to satisfy so called *path constraints* on the horizon of interest

$$h(x(t), z(t), u(t), p, t) \geq 0, \quad t \in [t_0, t_f].$$

The most common form of this constraint type are minimum and maximum values for the controls, but also e.g. safety restrictions on the system state may enter here. In addition, terminal equality or inequality constraints

$$r^{\mathrm{e}}(x(t_f), z(t_f), p, t_f) = 0$$
$$r^{\mathrm{i}}(x(t_f), z(t_f), p, t_f) \geq 0$$

may be imposed, e.g. to specify that a semi-batch process should stop when the tank is full. In some nonlinear model predictive control formulations, the terminal constraints help to guarantee nominal stability (cf. Sec. 1.4.1).

One constraint that plays an important role in the presented algorithms is the initial value constraint

$$x(t_0) = x_0.$$

[1] We use the definition $\|l\|_2^2 := \sum_{i=1}^{n_l} l_i^2$.

We will also treat the fixed parameters p and the initial time t_0 as if they were constrained variables:

$$\begin{aligned} p &= \bar{p} \\ t_0 &= \bar{t}_0 \end{aligned}$$

The introduction of t_0 and p as trivially constrained optimization variables seems to be an unnecessary blow-up of the problem. However, this formulation will turn out to be crucial for the proposed real-time algorithms.

Elimination of Parameter and Time Dependence

For notational simplicity we will in the remainder of this thesis drop the dependence of the problem functions on the system parameters p and the time t. This is no loss of generality: by introducing an augmented state vector

$$\tilde{x} := \begin{pmatrix} x \\ p \\ \tilde{t} \end{pmatrix} \quad \text{and initial condition} \quad \tilde{x}_0 := \begin{pmatrix} x_0 \\ \bar{p} \\ \bar{t}_0 \end{pmatrix},$$

and introducing the augmented differential equation $\tilde{B}(\cdot)\,\dot{\tilde{x}} = \tilde{f}(\cdot)$ with

$$\tilde{B}(\cdot) := \begin{pmatrix} B(\cdot) & 0 & 0 \\ 0 & \mathbb{I}_{n_p} & 0 \\ 0 & 0 & 1 \end{pmatrix}, \text{ and } \quad \tilde{f}(\cdot) := \begin{pmatrix} f(\cdot) \\ 0 \\ 1 \end{pmatrix},$$

the original formulation of the initial value problem formulation can be recaptured if the "clock" variable $\tilde{t}(t)$ is inserted wherever a direct dependence of the time t was present.

Note, however, that the trivial additional differential equations are treated independently from the others in the numerical solution procedures, for reasons of efficiency. Furthermore, only those parameters p that may have different values at practically relevant process conditions should be kept in this way, whereas all definitely known parameters can be taken as constants that are "hidden" in the problem functions.

As the optimization problem has become time independent, the time horizon of interest may start at $t_0 = 0$. Let us define T to be the horizon length.

If the final time t_f should be fixed, this can now be achieved by formulating a terminal equality constraint

$$r^{\mathrm{e}}(\tilde{x}(T)) := \tilde{t}(T) - t_f = 0.$$

Note that in this case the duration T depends implicitly on the initial value $\tilde{x}_0$, because at a feasible solution $t_f = \tilde{t}(T) = \tilde{t}(0) + T = \bar{t}_0 + T$, so that $T = t_f - \bar{t}_0$.

1.1.1 Problem Formulation

We can now formulate an optimal control problem

$$P_{\mathrm{oc}}(x_0): \quad \min_{\substack{u(\cdot),x(\cdot),\\ z(\cdot),(T)}} \int_0^T L(x(t),z(t),u(t))\,dt \;+\; E(x(T),z(T)) \tag{1.1a}$$

subject to

$$\begin{aligned} B(x(t),z(t),u(t))\cdot\dot{x}(t) - f(x(t),z(t),u(t)) &= 0, \quad t\in[0,T], && \text{(1.1b)}\\ g(x(t),z(t),u(t)) &= 0, \quad t\in[0,T], && \text{(1.1c)}\\ x(0)-x_0 &= 0, && \text{(1.1d)}\\ r^{\mathrm{e}}(x(T),z(T)) &= 0, && \text{(1.1e)}\\ r^{\mathrm{i}}(x(T),z(T)) &\geq 0, && \text{(1.1f)}\\ h(x(t),z(t),u(t)) &\geq 0, \quad t\in[0,T]. && \text{(1.1g)} \end{aligned}$$

The length T may either be fixed, or appear as a degree of freedom in the optimization problem.

Solving the optimal control problem (1.1) for an initial value x_0 we obtain optimal trajectories $x^*(t;x_0)$ and $z^*(t;x_0)$ and an open-loop optimal control $u^*(t;x_0)$, for $t\in[0,T(x_0)]$. In order to keep the dependency of the optimal trajectories on the initial value x_0 in mind, we have taken them as additional arguments to the solution functions.

We shall now introduce as a guiding example an optimal control problem from chemical engineering, which will be cited several times in this thesis for illustrative purposes.

1.2 A Guiding Example: Continuous Stirred Tank Reactor

Let us consider a continuous stirred tank reactor (CSTR) model that was introduced by Chen et al. [CKA95] as a benchmark example for Nonlinear Model Predictive Control. The reactor is designed to produce cyclopentenol from cyclopentadiene by an acid-catalyzed electrophilic hydration in aqueous solution, an exothermal reaction that makes a cooling jacket necessary. The considered ODE model was originally introduced by Klatt and Engell [KE93].

1.2.1 Dynamic Model of the CSTR

A schematic diagram of the reactor (taken from [CKA95]) is shown in Fig. 1.1. The reaction and heat transfer scheme developed by Klatt and Engell [KE93] is based on physical modelling; it leads to an ODE model with four states and two controls.

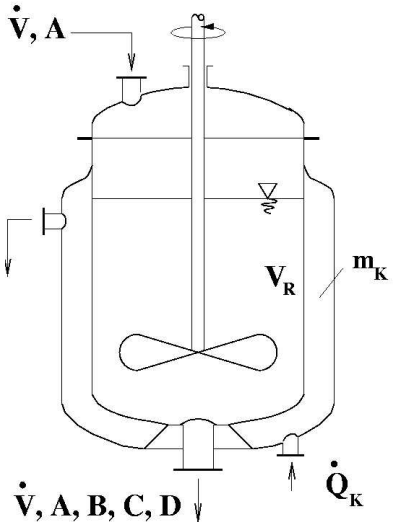


Figure 1.1: Schematic diagram of the CSTR (as shown in [CKA95])

The feed inflow has temperature ϑ_0 and contains only cyclopentadiene (substance A) with concentration c_{A0}. Its flow rate $\dot{V}$ can be controlled. In order to keep the liquid tank volume constant, the outflow is kept at the same rate as the inflow. The outflow contains a remainder of cyclopentadiene, the product cyclopentenol (substance B), and two unwanted by-products, cyclopentanediol (substance C) and dicyclopentadiene (substance D), with concentrations c_A, c_B, c_C, and c_D. The scheme for this so-called *van der Vusse reaction* is given as

$$\begin{aligned} A &\xrightarrow{k_1} B \xrightarrow{k_2} C \\ 2A &\xrightarrow{k_3} D. \end{aligned}$$

The reaction rates k_i depend on the reactor temperature ϑ via an Arrhenius law

$$k_i(\vartheta) = k_{i0} \cdot \exp\left(\frac{E_i}{\vartheta/^o\mathrm{C} + 273.15}\right), \quad i = 1, 2, 3.$$

The temperature ϑ_K in the cooling jacket is held down by an external heat exchanger whose heat removal rate $\dot{Q}_K$ can be controlled. As the substances C and D are unwanted and do not react further, it is not necessary to keep track of their concentrations.

The nonlinear ODE model can be derived from component balances for the substances A and B in the aqueous solution, and from enthalpy balances for the reactor and cooling jacket:

$$\begin{aligned}
\dot{c_A} &= \frac{\dot{V}}{V_R}(c_{A0} - c_A) \quad -k_1(\vartheta)c_A \qquad\qquad -k_3(\vartheta)c_A^2 \\
\dot{c_B} &= -\frac{\dot{V}}{V_R}c_B \qquad\quad +k_1(\vartheta)c_A \quad -k_2(\vartheta)c_B \\
\dot{\vartheta} &= \frac{\dot{V}}{V_R}(\vartheta_0 - \vartheta) + \frac{k_w A_R}{\rho C_p V_R}(\vartheta_K - \vartheta) \\
&\quad -\frac{1}{\rho C_p}\left(k_1(\vartheta)c_A H_1 + k_2(\vartheta)c_B H_2 + k_3(\vartheta)c_A^2 H_3\right) \\
\dot{\vartheta_K} &= \frac{1}{m_K C_{PK}}\left(\dot{Q}_K + k_w A_R(\vartheta - \vartheta_K)\right).
\end{aligned} \tag{1.2}$$

Here, C_{PK} and C_p denote the heat capacities of coolant and aqueous solution, ρ the solution's density, H_1,H_2, and H_3 the reaction enthalpies. Values of these parameters as well as for the Arrhenius coefficients k_{i0} and E_i for $i = 1, 2, 3$ and the employed reactor specific quantities (volume V_R, surface A_R and heat transfer coefficient k_W for cooling jacket and coolant mass m_K) are listed in Table 1.1. By introducing the system state x and the control vector u as

$$x = \begin{pmatrix} c_A \\ c_B \\ \vartheta \\ \vartheta_K \end{pmatrix} \quad \text{and} \quad u = \begin{pmatrix} \frac{\dot{V}}{V_R} \\ \dot{Q}_K \end{pmatrix}$$

we can summarize Eqs. (1.2) as

$$\dot{x} = f(x, u).$$

The result of a steady state optimization of the yield $= \frac{c_B|_S}{c_{A0}}$ with respect to the design parameter ϑ_0 (feed temperature) and the two controls yields the steady state and controls

$$x_S = \begin{pmatrix} 2.1402\ \frac{\text{mol}}{\text{l}} \\ 1.0903\ \frac{\text{mol}}{\text{l}} \\ 114.19\,^\circ\text{C} \\ 112.91\,^\circ\text{C} \end{pmatrix} \quad \text{and} \quad u_S = \begin{pmatrix} 14.19\,\text{h}^{-1} \\ -1113.5\ \frac{\text{kJ}}{\text{h}} \end{pmatrix}.$$

We will take this steady state as a desired reference value in the optimal control problem that follows – please note that it is of no importance in the following that x_S, u_S was itself the result of an optimization; the only property that is important for the optimal control problem is that $f(x_S, u_S) = 0$.

Note that we do not introduce the constant system parameters as additional variables, because we assume that they will never be subject to changes.

Symbol	Value	Symbol	Value
k_{10}	$1.287 \cdot 10^{12}\mathrm{h}^{-1}$	ρ	$0.9342\ \frac{\mathrm{kg}}{\mathrm{l}}$
k_{20}	$1.287 \cdot 10^{12}\mathrm{h}^{-1}$	C_p	$3.01\ \frac{\mathrm{kJ}}{\mathrm{kg \cdot K}}$
k_{30}	$9.043 \cdot 10^{9}\mathrm{h}^{-1}$	k_w	$4032\ \frac{\mathrm{kJ}}{\mathrm{h \cdot m^2 \cdot K}}$
E_1	-9758.3	A_R	$0.215\ \mathrm{m}^2$
E_2	-9758.3	V_R	$10\ l$
E_3	-8560	m_K	$5\ \mathrm{kg}$
H_1	$4.2\ \frac{\mathrm{kJ}}{\mathrm{mol}}$	C_{PK}	$2.0\ \frac{\mathrm{kJ}}{\mathrm{kg \cdot K}}$
H_2	$-11.0\ \frac{\mathrm{kJ}}{\mathrm{mol}}$	c_{A0}	$5.1\ \frac{\mathrm{mol}}{\mathrm{l}}$
H_3	$-41.85\ \frac{\mathrm{kJ}}{\mathrm{mol}}$	θ_0	$104.9\ ^\circ\mathrm{C}$

Table 1.1: Constant system parameters.

1.2.2 The Optimal Control Problem

Given an initial state x_0, the optimal control problem $P_{\mathrm{oc}}(x_0)$ is to steer the system safely and quickly into the steady state x_S. We take the formulation chosen by Chen in [Che97], that aims at minimizing the integrated weighted quadratic deviaton of the trajectory from the optimal steady state values. We define a Lagrange term

$$L(x,u) := (x - x_S)^T Q (x - x_S) + (u - u_S)^T R (u - u_S)$$

with diagonal matrices

$$Q := \begin{pmatrix} 0.2\,\mathrm{mol}^{-2}\,\mathrm{l}^2 & 0 & 0 & 0 \\ 0 & 1.0\,\mathrm{mol}^{-2}\,\mathrm{l}^2 & 0 & 0 \\ 0 & 0 & 0.5\,^\circ\mathrm{C}^{-2} & 0 \\ 0 & 0 & 0 & 0.2\,^\circ\mathrm{C}^{-2} \end{pmatrix}$$

and

$$R := \begin{pmatrix} 0.5\mathrm{h}^2 & 0 \\ 0 & 5.0 \cdot 10^{-7}\,\mathrm{kJ}^{-2}\,\mathrm{h}^2 \end{pmatrix}.$$

Control bounds $u_{\mathrm{LB}} \leq u(t) \leq u_{\mathrm{UB}}$ are given by

$$u_{\mathrm{LB}} := \begin{pmatrix} 3.0\,\mathrm{h}^{-1} \\ -9000\ \frac{\mathrm{kJ}}{\mathrm{h}} \end{pmatrix} \quad \text{and} \quad u_{\mathrm{UB}} := \begin{pmatrix} 35.0\,\mathrm{h}^{-1} \\ 0\ \frac{\mathrm{kJ}}{\mathrm{h}} \end{pmatrix},$$

so that we define the path inequality constraint function to be:

$$h(x(t), u(t)) := \begin{pmatrix} u(t) - u_{\mathrm{LB}} \\ u_{\mathrm{UB}} - u(t) \end{pmatrix} \geq 0.$$

We formulate the following optimal control problem $P_{\mathrm{oc}}(x_0)$:

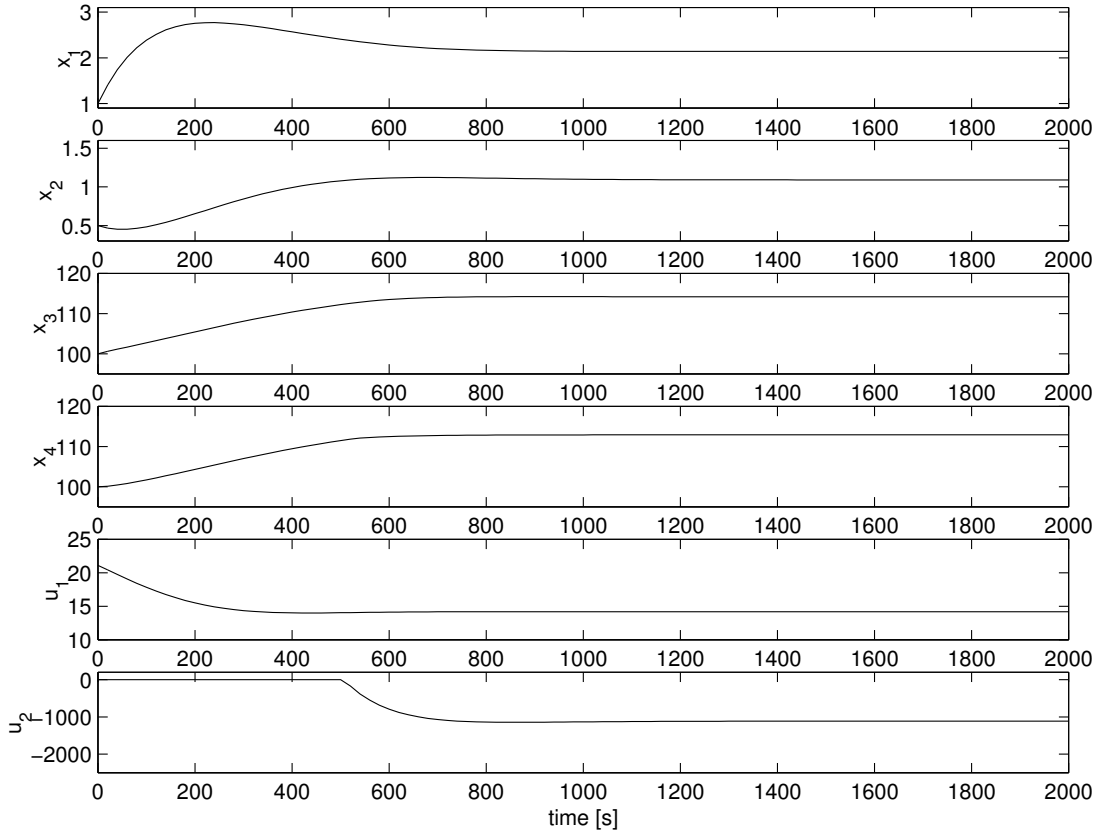


Figure 1.2: Example solution of the optimization problem (1.3). The first four graphs show the optimal state trajectories $x^*(t; x_0)$ and the last two the optimal control trajectories $u^*(t; x_0)$.

$$\begin{aligned} \min_{u(\cdot),x(\cdot)} & \int_0^T L\left(x(t),u(t)\right) dt \\ \text{subject to} & \\ \dot{x}(t) &= f\left(x(t),u(t)\right), \quad \forall\, t \in [0,T], \\ x(t_0) &= x_0, \\ h(x(t),u(t)) &\geq 0, \qquad\qquad \forall\, t \in [0,T]. \end{aligned} \tag{1.3}$$

In contrast to the formulation chosen in [Che97] we choose a considerably longer horizon length of $T = 2000$ seconds, which is sufficiently large to allow the assumption that our problem formulation is a good approximation for $T = \infty$.

The optimal trajectories $x^*(t;x_0)$ and $u^*(t;x_0)$ of an example solution of this optimization problem for the initial value

$$x_0 := \begin{pmatrix} 1.0\ \frac{\text{mol}}{\text{l}} \\ 0.5\ \frac{\text{mol}}{\text{l}} \\ 100\ ^\circ\text{C} \\ 100\ ^\circ\text{C} \end{pmatrix} \tag{1.4}$$

are shown in Figure 1.2.

1.3 Optimal Feedback Control

Let us for a moment assume that we can precompute, for all $x_0 \in \mathbb{R}^{n_x}$ for which the optimization problem $P_{\text{oc}}(x_0)$ has a solution, the optimal control trajectories $u^*(t;x_0)$ as well as the corresponding optimal state trajectories $x^*(t;x_0)$ and $z^*(t;x_0)$ on the time horizon $t \in [0,T(x_0)]$. We will assume that T is not fixed, but open to optimization. The length T may, however, be determined by the final state constraint (1.1e), e.g. in the case of a fixed end time t_f. Note that in this case the "clock" variable $\tilde{t}$ is part of the system state x.

Let us pick a fixed value of x_0 and consider the optimal solution trajectories $x^*(\cdot;x_0)$, $z^*(\cdot;x_0)$, and $u^*(\cdot;x_0)$ of $P_{\text{oc}}(x_0)$. Let us also pick a time $t_1 \in [0,T(x_0)]$ and the corresponding state $x_1 := x^*(t_1;x_0)$ on the optimal differential state trajectory $x^*(\cdot;x_0)$. Consider now the related optimization problem $P_{\text{oc}}(x_1)$. How are its optimal solution trajectories $x^*(\cdot;x_1)$, $z^*(\cdot;x_1)$, and $u^*(\cdot;x_1)$ on $[0,T(x_1)]$ related to those of $P_{\text{oc}}(x_0)$? From the *principle of optimality*, also known as the *optimality of subarcs*, it follows that $T(x_1) = T(x_0) - t_1$ and that the solution trajectories of $P_{\text{oc}}(x_1)$ coincide with the remaining part of the solution trajectories of $P_{\text{oc}}(x_0)$ after t_1, i.e.,

$$\left.\begin{aligned} x^*(t;x_1) &= x^*(t_1+t;x_0) \\ z^*(t;x_1) &= z^*(t_1+t;x_0) \\ u^*(t;x_1) &= u^*(t_1+t;x_0) \end{aligned}\right\} \forall\, t \in [0,T(x_1)].$$

By chosing $t = 0$ and formulating the last identity for all $t_1 \in [0, T(x_0)]$, we can conversely yield the optimal control trajectory $u(\cdot; x_0)$ by

$$u(t_1; x_0) = u^*(0; x^*(t_1; x_0)), \quad \forall\, t_1 \in [0, T(x_0)].$$

Hence, the result of the precomputation can be captured in an *optimal feedback control* (cf. [BH69]) function u^{f} that is defined as

$$u^{\mathrm{f}}(x_0) := u^*(0; x_0). \tag{1.5}$$

This function may be used as a feedback control that leads to the *closed-loop* DAE system

$$\begin{aligned} B(\cdot) \cdot \dot{x}_{\mathrm{cl}}(t) &= f(x_{\mathrm{cl}}(t), z_{\mathrm{cl}}(t), u^{\mathrm{f}}(x_{\mathrm{cl}}(t))) \\ 0 &= g(x_{\mathrm{cl}}(t), z_{\mathrm{cl}}(t), u^{\mathrm{f}}(x_{\mathrm{cl}}(t))). \end{aligned}$$

One computationally expensive and storage consuming possibility would be to precalculate such a feedback control law off-line on a sufficiently fine grid. The technique of choice to compute this feedback control would be dynamic programming [Bel57], or an approach using the Hamilton-Jacobi-Bellman (HJB) equation [LM68, Son90]. However, even for moderate state dimensions n_x this would require a prohibitively large computational effort.

In contrast to this our work is concerned with efficient ways to calculate the optimal feedback control $u^{\mathrm{f}}(x_0)$ *in real-time* while the real process runs.

1.3.1 Linearized Neighboring Feedback Control

One possibility to approximate the optimal feedback control law $u^{\mathrm{f}}(x_{\mathrm{cl}}(t))$ in the vicinity of a reference trajectory is provided by *linearized neighboring feedback control* (also called *perturbation feedback control* [BH69]). It requires a *nominal* or *reference solution* $x^*(\cdot; x_0)$, $z^*(\cdot; x_0)$, and $u^*(\cdot; x_0)$ of a nominal problem $P_{\mathrm{oc}}(x_0)$, and is a good approximation if the distance $\|x(t) - x^*(t; x_0)\|$ of the real trajectory $x(t)$ to the reference trajectory remains small. The idea is to approximate

$$u^{\mathrm{f}}(x_{\mathrm{cl}}(t)) = u^*(0; x_{\mathrm{cl}}(t)), \quad \forall t \in [0, T(x_0)],$$

by the linearization

$$u^{\mathrm{lnfc}}(x_{\mathrm{cl}}(t)) := u^*(t; x_0) + K(t)(x_{\mathrm{cl}}(t) - x^*(t; x_0)),$$

where

$$K(t) := \frac{\partial u^{\mathrm{f}}}{\partial x}(x^*(t; x_0))$$

that is defined for all $t \in [0, T(x_0)]$. Note that the constant term $u^*(t; x_0)$ is equal to $u^{\mathrm{f}}(x^*(t; x_0)) = u^*(0; x^*(t; x_0))$ due to the principle of optimality. The derivative or *gain matrix* K, if it exists, can efficiently be calculated by using first and second derivative

information along the reference trajectory $x^*(\cdot; x_0)$, $z^*(\cdot; x_0)$, and $u^*(\cdot; x_0)$, see e.g. Bryson and Ho [BH69].

The method can be extended to the case that the derivative K does not exist in a strict sense, e.g. in the case of bang-bang controls, but where it is still possible to use derivative information along a reference trajectory. Numerical techniques to compute linearized neighboring feedback controls have been developed, e.g., by Pesch [Pes78], Krämer-Eis et al. [KE85, KEB87], and Kugelmann and Pesch [KP90a, KP90b]. Linearized neighboring techniques have been applied for on-line control of batch reactors, e.g., by Terwiesch and Agarwal [TA94].

Note that the approximations provided by these techniques are only valid in the neighborhood of a reference trajectory. If the real-system has moved far away from the reference trajectory during process development, the approximation of the optimal feedback control may become very poor and may drive the system even into directions opposite to what is desired. Cf. Sec. 4.3.2 and Example 4.4.

Example 1.1 (Optimal and Linearized Neighboring Feedback)
As an example for a tabulation of the optimal feedback controls $u^{\mathrm{f}}(x)$, and for the linearized neighboring feedback approximation $u^{\mathrm{lnfc}}(x)$ we show in Figure 1.3 a one dimensional cut through the four dimensional state space of the CSTR of Section 1.2, for initial values

$$x_\epsilon := x_S + \epsilon(x_0 - x_S),$$

that interpolate between the steady state x_S and the disturbed initial value x_0 from (1.4).

The graphs for $u^{\mathrm{f}}(x_\epsilon) := u^*(0; x_\epsilon)$ have been obtained by a numerical solution of the optimal control problem (1.3) *(which we take as an approximation for $T = \infty$) for 141 initial values x_ϵ, $\epsilon \in \{-0.20, -0.19, \ldots, 1.19, 1.20\}$, whereas $u^{\mathrm{lnfc}}(x_\epsilon) := u_S + K(0)(x_\epsilon - x_S)$ is based on a linearization of $u^{\mathrm{f}}(\cdot)$ at the steady state x_S. For a closed-loop trajectory due to a linearized neighboring feedback technique, cf. Example 4.4.*

In our considerations about optimal feedback control we have assumed that the horizon length T of the optimization problem (1.1) is a variable in the optimization problem that is either determined by some constraints or a real degree of freedom. In this case we speak of "shrinking horizon problems", because the time horizon T is shrinking during optimal process development, as we have seen by the principle of optimality for subarcs. In chemical engineering this problem type arises typically for batch or semi-batch processes, in robotics e.g. for time optimal maneuvers.

1.3.2 Infinite Horizon Problems

On the other hand, a major application for feedback control systems are systems that run infinitely long, so that the choice $T = \infty$ would be appropriate. It is straightforward that the principle of optimality holds also for the solution trajectories $x^*_\infty(\cdot; x_0)$, $z^*_\infty(\cdot; x_0)$, and $u^*_\infty(\cdot; x_0)$, and we may accordingly define an optimal feedback control law for infinite horizon problems

$$u^{\mathrm{f}}_\infty(x_0) := u^*_\infty(0; x_0).$$

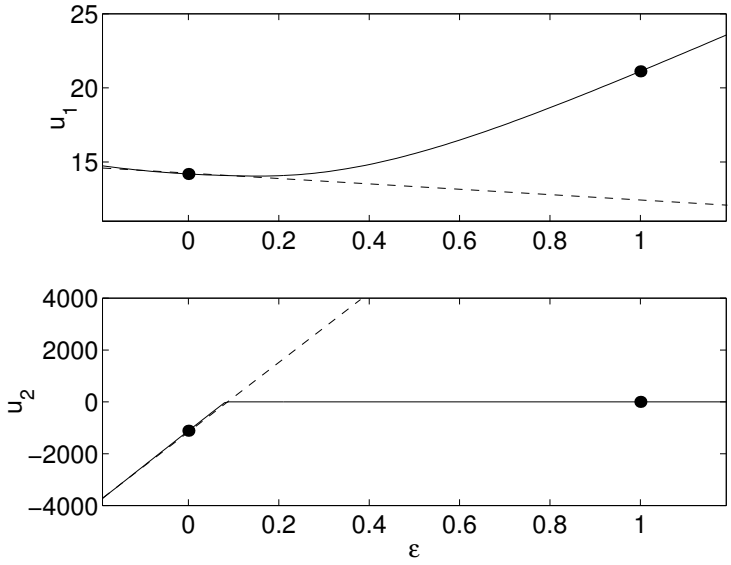


Figure 1.3: Optimal (solid) and linearized neighboring feedback controls (dashed), for the initial values $x_\epsilon := x_S + \epsilon(x_0 - x_S)$ of Example 1.1. The dots indicate the optimal feedback controls for the steady state x_S ($\epsilon = 0$) and the disturbed value x_0 ($\epsilon = 1$) (cf. Fig. 1.2, $t = 0$)

Nominal Stability

For steady state tracking problems with an objective $\int_0^\infty L(x(t), z(t), u(t))\, dt$ that satisfies $L(x, z, u) > 0$ for all $(x, z, u) \neq (x_S, z_S, u_S)$ and $L(x_S, z_S, u_S) = 0$ at the steady state, the principle of optimality ensures nominal stability of the corresponding closed-loop system, as the *optimal cost function*

$$V_\infty(x_0) := \int_0^\infty L(x^*_\infty(t; x_0), z^*_\infty(t; x_0), u^*_\infty(t; x_0))\, dt,$$

if it remains finite, serves as a *Lyapunov function* (as defined e.g. in [Son90]) for the closed-loop system.

To sketch the idea of the nominal stability proof, let us assume that $V_\infty(\cdot) \in C^1$ and that $u^*_\infty(\cdot; x_0) \in C^0 \forall\, x_0$, and furthermore that the level sets of $V_\infty(\cdot)$ are compact in $\mathbb{R}^{n_x}$.

First note that $V_\infty(x_S) = 0$ and $V_\infty(x_0) > 0, x_0 \neq x_S$. It will now be shown that

$$\frac{d}{dt} V_\infty(x_{\mathrm{cl}}(t)) < 0, \quad \forall\, x_{\mathrm{cl}}(t) \neq x_S, \tag{1.6}$$

so that the only accumulation point of the closed-loop trajectory $x_{\mathrm{cl}}(t)$, $t \in [0, \infty)$, can be x_S. As the level sets are compact, an accumulation point must exist, so that asymptotic

stability follows. To show the descent property (1.6), first note that the closed-loop trajectory $x_{\mathrm{cl}}(\cdot)$ for the initial value x_0 coincides with the optimal trajectory $x^*_\infty(\cdot; x_0)$. Therefore it needs only to be shown that

$$\frac{d}{dt} V_\infty(x^*_\infty(t; x_0)) < 0, \quad \forall\, x^*_\infty(t; x_0) \neq x_S. \tag{1.7}$$

Differentiation of the identity

$$V_\infty(x^*_\infty(t; x_0)) = V_\infty(x_0) - \int_0^t L(x^*_\infty(\tau; x_0), z^*_\infty(\tau; x_0), u^*_\infty(\tau; x_0))\, d\tau,$$

(which is a direct consequence of the principle of optimality) with respect to t yields

$$\frac{d}{dt} V_\infty(x^*_\infty(t; x_0)) = -L(x^*_\infty(t; x_0), z^*_\infty(t; x_0), u^*_\infty(t; x_0)) < 0, \quad \forall\, x^*_\infty(t; x_0) \neq x_S.$$

1.4 Nonlinear Model Predictive Control

Unfortunately, infinite horizon problems are in general very difficult to handle for nonlinear and constrained systems. Therefore, a so called "moving horizon" approach is often used instead, where a constant *control horizon* of length T is chosen in all optimization problems. If the constant T is sufficiently large, the computed optimal trajectories $x^*_T(\cdot; x_0)$, , $z^*_T(\cdot; x_0)$, and $u^*_T(\cdot; x_0)$ are expected to be similar to the corresponding infinite horizon values $x^*_\infty(\cdot; x_0)$, $z^*_\infty(\cdot; x_0)$, and $u^*_\infty(\cdot; x_0)$ on $[0, T]$ so that the definition for moving horizon problems,

$$u^{\mathrm{f}}_T(x_0) := u^*_T(0; x_0)$$

is a good approximation for the infinite horizon optimal feedback control $u^{\mathrm{f}}_\infty(x)$. We call the resulting feedback law $u^f_T(x)$ "optimal moving horizon feedback control" [BBB+01] or "Nonlinear Model Predictive Control" (NMPC). Often also the term "Receding Horizon Control" (RHC) [MM90] is used to denote this moving horizon scheme. The computation of the optimal moving horizon control law $u^{\mathrm{f}}_T(x_0)$ in real-time is the main application of our algorithms.

1.4.1 Schemes to Ensure Nominal Stability

Note that the principle of optimality does no longer hold for moving horizon problems; however, a variety of schemes to ensure nominal stability for steady state tracking problems has been devised. These schemes make strong use of artificially introduced end point constraints as formulated in Eqs. (1.1e) and (1.1f), and of the Mayer term $E(x(T), z(T))$ in the objective functional (1.1a). The principal idea is to formulate the optimization problems in such a way that the optimal cost function

$$V_T(x_0) := \int_0^T L(x^*_T(t; x_0), z^*_T(t; x_0), u^*_T(t; x_0))\, dt + E(x^*_T(T; x_0))$$

can serve as a Lyapunov function of the closed-loop system, as for infinite horizon problems. For an overview of such schemes, we refer to the articles by Mayne [May96, May00] or De Nicolao, Magni, and Scattolini [DMS00]. Here, we will briefly introduce three of these schemes. All three have in common that the so called *monotonicity property* [DMS00] holds in a neighborhood Ω_T of the steady state x_S:

$$V_T(x) \leq V_{T-\delta}(x), \quad \forall\, \delta \geq 0, \delta \leq T, x \in \Omega_T. \tag{1.8}$$

Nominal stability follows together with the principle of optimality for subarcs, which states that

$$V_T(x_0) = \int_0^\delta L(x_T^*(t;x_0), z_T^*(t;x_0), u_T^*(t;x_0))\, dt + V_{T-\delta}(x_T^*(\delta;x_0))$$

so that (using $x_0 = x_T^*(0;x_0)$)

$$\begin{aligned} V_T(x_T^*(\delta;x_0)) - V_T(x_T^*(0;x_0)) &\leq V_{T-\delta}(x_T^*(\delta;x_0)) - V_T(x_T^*(0;x_0)) \\ &= -\textstyle\int_0^\delta L(x_T^*(t;x_0), z_T^*(t;x_0), u_T^*(t;x_0))\, dt. \end{aligned}$$

Differentiating this inequality by δ we can deduce that

$$\begin{aligned} \tfrac{d}{dt} V_T(x_T^*(0;x_0)) &= \tfrac{\partial V_T}{\partial x}(x_T^*(0;x_0))\ \dot{x}_T^*(0;x_0) \\ &\leq -L(x_T^*(0;x_0), z_T^*(0;x_0), u_T^*(0;x_0)) < 0, \forall\, x_0 \neq x_S. \end{aligned}$$

Let us now choose $x_0 := x_{\mathrm{cl}}(t)$ to be one state of the closed-loop trajectory, at time t. The time development of the nominal closed-loop system obeys the same DAE as the model; because at time t the differential system state is $x_{\mathrm{cl}}(t) = x_0 = x_T^*(0;x_0)$ and the closed-loop control is chosen to be $u_T^{\mathrm{f}}(x_{\mathrm{cl}}(t)) := u_T^*(0;x_0)$, the algebraic state also coincides with the start of the optimal trajectory: $z_{\mathrm{cl}}(t) = z_T^*(0;x_0)$ (due to the algebraic consistency condition); therefore, the time derivatives coincide:

$$\dot{x}_{\mathrm{cl}}(t) = \dot{x}_T^*(0;x_0).$$

This allows to conclude that

$$\frac{d}{dt} V_T(x_{\mathrm{cl}}(t)) = \frac{\partial V_T}{\partial x}(x_{\mathrm{cl}}(t))\ \dot{x}_{\mathrm{cl}}(t) < 0, \forall x_{\mathrm{cl}}(t) \neq x_S.$$

Zero Terminal Constraint

The idea of the zero terminal constraint (ZTC) scheme is to formulate the terminal point constraint

$$r^{\mathrm{e}}(x(T)) := x(T) - x_S,$$

where x_S is the differential part of the desired steady state, and to employ no final penalty $E(x(T))$. Nominal stability for nonlinear continuous time systems was proven by Mayne and Michalska [MM90]. The monotonicity property (1.8) follows from the fact that the optimal solution $x_{T-\delta}^*(t;x_0)$, $z_{T-\delta}^*(t;x_0)$, and $u_{T-\delta}^*(t;x_0)$ of a problem $P_{\mathrm{oc},T-\delta}(x_0)$ on a short horizon $[0, T-\delta]$ can be prolonged to a feasible trajectory of the problem $P_{\mathrm{oc},T}(x_0)$ on a long horizon $[0,T]$, by adding for $t \in [T-\delta, T]$ the final parts $x_T^*(t;x_0) := x_S$, $z_T^*(t;x_0) := z_S$, and $u_T^*(t;x_0) := u_S$, that have no additional costs, cf. Fig.1.4.

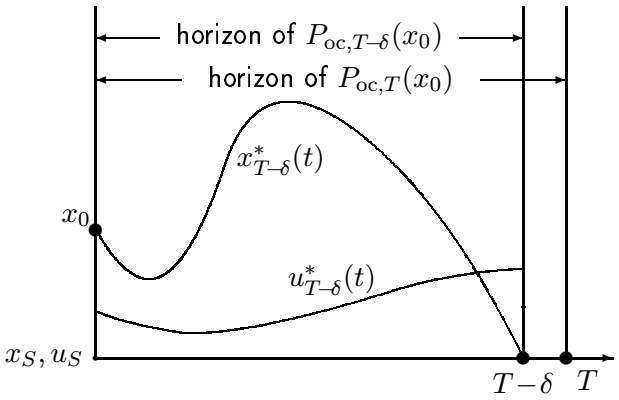


Figure 1.4: Monotonicity property for the zero terminal constraint (ZTC) scheme: the solution trajectories of problem $P_{\mathrm{oc},T-\delta}(x_0)$ can be prolonged to feasible trajectories for problem $P_{\mathrm{oc},T}(x_0)$ without increasing the objective.

Quasi-Infinite Horizon NMPC

The quasi-infinite horizon (QIH) scheme employs a positive definite penalty matrix $P \in \mathbb{R}^{n_x \times n_x}$ that allows to formulate a terminal penalty term

$$E(x(T)) := \|x(T) - x_S\|_P^2 := \|P^{\frac{1}{2}}(x(T) - x_S)\|_2^2$$

and a terminal constraint

$$r^{\mathrm{i}}(x(T)) := \alpha - \|x(T) - x_S\|_P^2 \geq 0,$$

(with $\alpha > 0$) that constrains $x(T)$ to be in an elliptic region $\Omega := \{x \in \mathbb{R}^{n_x} | \|x - x_S\|_P^2 \leq \alpha\}$. Chen and Allgöwer [CA98, Che97] have shown how the matrix P and the constant α can be computed so that the monotonicity property (1.8) is satisfied. Their approach, that was originally formulated for ODE systems, was generalized to DAE systems of index-one by Findeisen and Allgöwer [FA00]. Using the system linearization around the steady state and a linear closed-loop law $u(x) = u_S + K \cdot (x - x_S)$, the matrix P is computed as the solution of a Lyapunov equation, and the constant α is determined so that the elliptic region Ω is positively invariant for the linearly controlled closed-loop system, and so that the path constraints $h(x, z, u_S + K \cdot (x - x_S)) \leq 0$ are not violated in the set $\{(x, z) \in \mathbb{R}^{n_x} \times \mathbb{R}^{n_z} | x \in \Omega, g(x, z, u_S + K \cdot (x - x_S)) = 0\}$.

Infinite-Horizon Closed-Loop Costing

The idea of the infinite-horizon closed-loop costing approach, that was proposed by De Nicolao, Magni, and Scattolini [DMS96], is to introduce a terminal penalty that is itself the infinite integral of the Lagrange objective

$$E(x(T)) := \int_T^\infty L(\hat{x}(t; x(T)), \hat{z}(t; x(T)), \hat{u}(t; x(T)))\, dt$$

where $\hat{x}(t;x(T))$ and $\hat{z}(t;x(T))$ are the trajectories corresponding to the following closed-loop initial value problem on the horizon $[T,\infty)$:

$$\begin{aligned} B(\cdot)\cdot\dot{\hat{x}}(t) &= f(\hat{x}(t),\hat{z}(t),K(\hat{x}(t))),\\ 0 &= g(\hat{x}(t),\hat{z}(t),K(\hat{x}(t))),\\ \hat{x}(T) &= x(T). \end{aligned}$$

The function $K:\mathbb{R}^{n_x}\to\mathbb{R}^{n_u}$ is chosen so as to stabilize the system in the vicinity of the steady state (typically by a LQR control law for the system linearization). The final state constraint $r^{\mathrm{i}}(x(T))\geq 0$ must constrain $x(T)$ so that all predicted closed-loop trajectories $\hat{x}(t;x(T))$, $\hat{z}(t;x(T))$, and $\hat{u}(t;x(T)):=K(\hat{x}(t;x(T)))$ remain feasible and have finite costs. The monotonicity property (1.8) follows from the fact that a prolongation of the horizon from $[0,T-\delta]$ to $[0,T]$ only increases the degrees of freedom; the new degrees of freedom, the controls $u(t)$ for $t\in[T-\delta,T]$, can still be chosen to be $u(t)=K(x(t))$, which would yield equal costs as for the short horizon. A practical implementation of this approach must overcome the nontrivial problems of determining the final state constraint $r^{\mathrm{i}}(x(T))\geq 0$, and the on-line computation of the infinite integral to determine $E(x(T))$. Note, however, that the computation of a finite horizon approximation of $E(x(T))$ can be very cheap even on relatively long horizons, if adaptive implicit DAE solvers are used, as the stepsizes in the vicinity of the steady state can be made very large. We have employed such a scheme for the control experiments with a distillation column that are presented in Chap. 7, where the trivial closed-loop law $K(\cdot):=u_S$ could be chosen because the system is stable.

1.4.2 Alternative Feedback Strategies

Optimal feedback control and nonlinear model predictive control as defined above are not the only ways to derive feedback laws, and among these they are not necessarily the best. They suffer from an inherent contradiction: on the one hand the employed system model is deterministic, but on the other hand the necessity for feedback control is created by the non-deterministic behaviour of the system, or by the presence of model-plant mismatch. There are several strategies that include some sort of knowledge that the real system does *not* obey the deterministic model equations. We will briefly mention two of these here.

Stochastic Optimal Control

Stochastic optimal control techniques employ a stochastic system model instead of a deterministic one, and aim at optimizing the expectation value of an objective functional. The stochastic point of view makes it possible to design feedback controllers that take future disturbances into account – provided that realistic assumptions on the governing stochastics are available. The method of choice for the solution of stochastic optimal control problems is dynamic programming, which is originally due to Bellman [Bel57]. (We recommend the two excellent books on Optimal Control and Dynamic Programming by Bertsekas [Ber95a, Ber95b] and refer also to Bertsekas et al. [BT96, BS96]). For linear systems with quadratic costs the solution of stochastic optimal control problems is equivalent

to the solution of a corresponding deterministic optimal control problem (see e.g. Bryson and Ho, [BH69]), a fact that leads to the *separation theorem* or *certainty-equivalence principle* [Sim56, JT61] for linear systems. *Nonlinear* stochastic optimal control problems, however, are difficult to solve even for moderate system sizes.

Robust Control

The area of so called *robust control* techniques is vast and has undergone rapid development in the last two decades. Roughly spoken, robust control techniques aim at designing feedback control laws $u^{rc}(x)$ that are not only able to stabilize a nominal system model, but that show a good control performance for a whole set of perturbed/disturbed systems. For an introduction into linear robust control techniques we refer to Zhou et al. [ZDG96] or to Morari [Mor87]. Though robust control theory is highly developed for linear systems, only a few extensions exist that take explicitly nonlinear system models into account. The question of robustness of NMPC is mostly unsolved. Some preliminary steps have been outlined for example in [OM94, MM93, YP93], and even some approaches exist that try to synthesize robust NMPC controllers, e.g. Chen et al. [CSA97] (cf. also [May00]).

Chapter 2

Direct Multiple Shooting

In this chapter we will present, as a first step towards the numerical solution of the optimal control problem (1.1), the so called *direct multiple shooting* parameterization which is the basis for all algorithms presented later in this thesis. The direct multiple shooting parameterization transforms the original infinite optimal control problem $P_{\text{oc}}(x_0)$ (1.1) into a finite dimensional *Nonlinear Programming* (NLP) problem that we will denote by $P(x_0)$. The direct multiple shooting method is originally due to Bock and Plitt [Pli81, BP84], and its most recent form has been developed by Leineweber [Lei99] and implemented in his optimal control package MUSCOD-II, which also forms the basis for the actual implementation of the real-time algorithms presented in this work.

2.1 Problem Parameterization

In order to reformulate the infinite optimal control problem (1.1) as a finite dimensional *nonlinear programming* (NLP) problem, both its controls and its states are parameterized in the direct multiple shooting method. Let us first introduce a time transformation that prepares the control and state parameterization.

2.1.1 Time Transformation

In order to be able to treat problems with a variable horizon length T conveniently, we introduce a time transformation

$$t : [0,1] \to [0,T], \quad \tau \mapsto t(\tau, T) := T\tau$$

which allows us to regard an optimization problem on the fixed horizon $[0,1]$ only. By interpreting the trajectories of x, z, and u as functions of $\tau \in [0,1]$ we can formulate a problem on the horizon $[0,1]$ that is equivalent to problem (1.1). If the horizon length T is variable, we will treat it as a free global parameter, that can conceptually be localized by introduction of an additional trivial differential equation $\dot{T}(\tau) = 0$ with a free initial value. To keep notation simple we will in the following subsections assume that T is fixed, but keep in mind that the case of a variable horizon is captured by this approach, too.

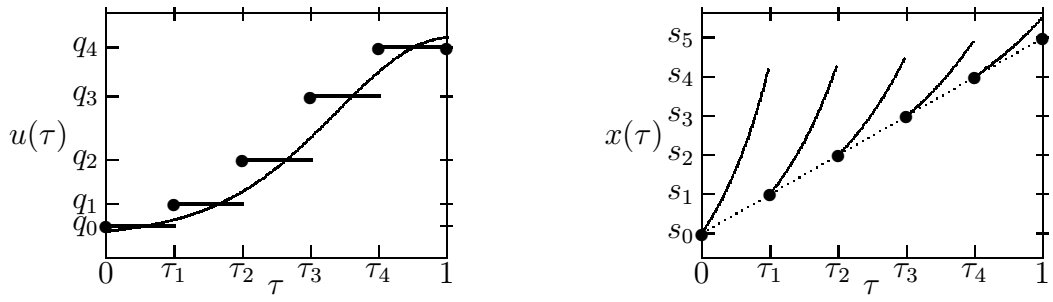


Figure 2.1: Control and state parameterization ($N = 5$).

2.1.2 Control Discretization

In the direct multiple shooting method (as in all direct solution approaches) the infinitely many degrees of freedom $u(\tau)$ for $\tau \in [0, 1]$ are approximated by a finite control representation. For this aim we choose a multiple shooting grid

$$0 = \tau_0 < \tau_1 < \ldots < \tau_N = 1, \tag{2.1}$$

and approximate the control $u(\tau)$ by a *piecewise* constant control representation, i.e., we set

$$u(\tau) := q_i \text{ for } \tau \in [\tau_i, \tau_{i+1}), \quad i = 0, 1, \ldots N-1, \tag{2.2}$$

with N vectors $q_i \in \mathbb{R}^{n_u}$, as sketched on the left hand side of Fig. 2.1. For completeness, we set as control at the final time

$$u(1) := q_N := q_{N-1},$$

where the vector q_N is introduced for notational convenience only and will not be regarded as a new parameter, but just as a second name for q_{N-1}. Note that the point value $u(1)$ of the control may directly influence the final algebraic state $z(1)$ (that is determined by $g(x(1), z(1), u(1)) = 0$) and can therefore not be neglected in the case of DAE models.

It is possible to use other, possibly higher order control parameterizations on the intervals (e.g. linear or cubic polynomials), but it is of crucial importance for the direct multiple shooting method that the control parameterization has local support on the multiple shooting intervals $[\tau_i, \tau_{i+1}]$, so that the control parameters have a local influence only (cf. Sec. 6.1).

Where continuity of the controls is desired, the control can e.g. be treated as an additional differential system state whose time derivative can be controlled.

2.1.3 State Parameterization

In a crucial second step, $2(N+1)$ additional vectors $s_0^x, s_1^x, \ldots, s_N^x$ and $s_0^z, s_1^z, \ldots, s_N^z$ of the same dimensions n_x and n_z as differential and algebraic system states are introduced, which

we will denote differential and algebraic *node values*. For brevity we will often combine them in the vectors $s_i := (s_i^x, s_i^z)$.

All but the last node value serve as initial values for N independent *relaxed* initial value problems on the intervals $[\tau_i, \tau_{i+1}]$:

$$\begin{aligned} B(\cdot)\cdot\dot{x}_i(\tau) &= T\, f(x_i(\tau), z_i(\tau), q_i) && (2.3)\\ 0 &= g(x_i(\tau), z_i(\tau), q_i) - \exp\left(-\beta\frac{\tau-\tau_i}{\tau_{i+1}-\tau_i}\right) g(s_i^x, s_i^z, q_i) && (2.4)\\ x_i(\tau_i) &= s_i^x. && (2.5)\end{aligned}$$

The decaying subtrahend in (2.4) with $\beta > 0$ is deliberately introduced to facilitate an efficient DAE solution for initial values and controls s_i^x, s_i^z, q_i that may temporarily violate the consistency conditions (1.1c) (note that (s_i^x, s_i^z, q_i) is per definition a consistent initial value for the relaxed initial value problem). This modification (Bock et al. [BES88]), is commonly referred to as a *relaxed* DAE formulation, cf. Schulz et al. [SBS98], Leineweber [Lei99].

The solutions of these initial value problems are N independent trajectories $x_i(\tau)$, $z_i(\tau)$ on $[\tau_i, \tau_{i+1}]$, which are a function of s_i and q_i only. In order to keep this dependency in mind, we will denote them often by $x_i(\tau; s_i, q_i)$ and $z_i(\tau; s_i, q_i)$. See the right hand side of Fig. 2.1 for an illustration.

By substituting the independent trajectories $x_i(\tau)$, $z_i(\tau)$ into the Lagrange term L in Eq. (1.1a) we can simultaneously calculate the integral objective contributions $L_i(s_i, q_i)$ that are given by

$$L_i(s_i, q_i) := \int_{\tau_i}^{\tau_{i+1}} T\, L(x_i(\tau), z_i(\tau), q_i))\, d\tau. \tag{2.6}$$

The introduction of the values s_i^x and s_i^z has introduced non-physical degrees of freedom that have to be removed by corresponding equality constraints in the NLP. First, we have to require that the relaxation terms in the relaxed DAE formulation (2.4) vanish, i.e., formulate the *algebraic consistency* conditions

$$g(s_i^x, s_i^z, q_i) = 0 \quad i = 0, 1, \ldots, N. \tag{2.7}$$

Secondly, we have to enforce continuity of the differential state trajectory by formulating the following *matching conditions* which require that each differential node value s_{i+1}^x should equal the final value of the preceding trajectory x_i:

$$s_{i+1}^x = x_i(\tau_{i+1}; s_i, q_i), \quad i = 0, \ldots, N-1. \tag{2.8}$$

The first differential node value s_0^x is required to be equal to the initial value x_0 of the optimization problem:

$$s_0 = x_0. \tag{2.9}$$

Together, the constraints (2.7), (2.8), and (2.9) remove the additional degrees of freedom which were introduced with the parameters s_i, $i = 0, \ldots, N$. It is by no means necessary that these constraints are satisfied *during* the optimization iterations – on the contrary, it is a crucial feature of the direct multiple shooting method that it can deal with *infeasible* initial guesses of the variables s_i.

2.1.4 Discretization of Path Constraints

Using the multiple shooting grid $\tau_0, \ldots, \tau_N$, the infinite dimensional path inequality constraints (1.1g) are transformed into $N+1$ vector inequality constraints

$$h(s_i^x, s_i^z, q_i) \geq 0, \quad i = 0, 1, \ldots, N.$$

Note that it would be equally possible to use a finer grid for the discretization of the path constraints.

2.2 The Nonlinear Programming Problem

The finite dimensional NLP in the direct multiple shooting parameterization is given as

$$P(x_0): \quad \min_{\substack{q_0,\ldots,q_{N-1},\\ s_0,\ldots,s_N}} \sum_{i=0}^{N-1} L_i(s_i^x, s_i^z, q_i) + E(s_N^x, s_N^z) \tag{2.10a}$$

subject to

$$\begin{aligned}
s_{i+1}^x - x_i(\tau_{i+1}; s_i^x, s_i^z, q_i) &= 0, \quad i = 0, \ldots, N-1, && \text{(2.10b)}\\
g(s_i^x, s_i^z, q_i) &= 0, \quad i = 0, \ldots, N, && \text{(2.10c)}\\
s_0^x - x_0 &= 0, && \text{(2.10d)}\\
r^{\mathrm{e}}(s_N^x, s_N^z) &= 0, && \text{(2.10e)}\\
r^{\mathrm{i}}(s_N^x, s_N^z) &\geq 0, && \text{(2.10f)}\\
h(s_i^x, s_i^z, q_i) &\geq 0, \quad i = 0, \ldots, N. && \text{(2.10g)}
\end{aligned}$$

This is the NLP problem formulation that we will use as a reference in the following chapters. For a visualization of the NLP variables, see Fig. 2.2. It turns out that the NLP has a very favourable sparse structure, due to the fact that all constraint functions and the additive terms of the objective function each depend only on a small number of variables, and conversely, each variable appears only in a few problem functions.

To conveniently write the NLP (2.10) in a shorter form let us introduce the vectors

$$q := \begin{pmatrix} q_0 \\ \vdots \\ q_{N-1} \end{pmatrix} \in \mathbb{R}^{n_q}, \quad s := \begin{pmatrix} s_0 \\ \vdots \\ s_N \end{pmatrix} \in \mathbb{R}^{n_s}, \quad \text{and} \quad w := \begin{pmatrix} q \\ s \end{pmatrix} \in \mathbb{R}^{n_w}$$

with $n_q := Nn_u$, $n_s := (N+1)(n_x + n_z)$, and $n_w = n_q + n_s$, and define $F(w) := \sum_{i=0}^{N-1} L_i(s_i, q_i) + E(s_N)$ and summarize all equality constraints in a function $G : \mathbb{R}^{n_w} \to$

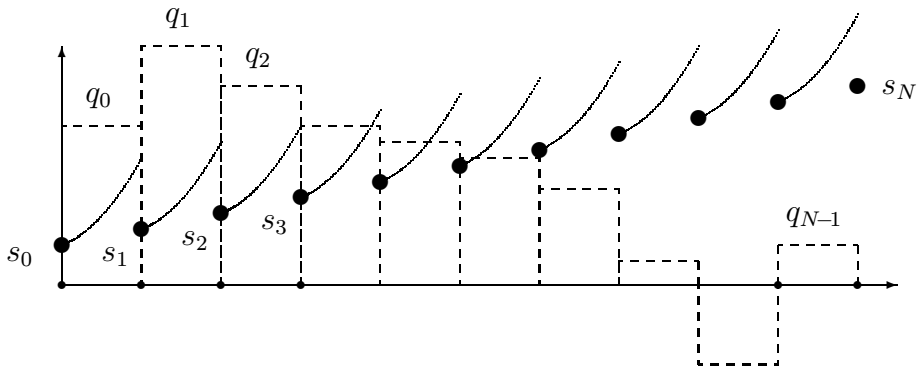


Figure 2.2: The NLP variables in the multiple shooting parameterization

$\mathbb{R}^{n_G}$ and all inequality constraints in a function $H : \mathbb{R}^{n_w} \to \mathbb{R}^{n_H}$. The NLP can then be summarized as

$$\min_{w \in \mathbb{R}^{n_w}} F(w) \quad \text{subject to} \quad \begin{cases} G(w) = 0, \\ H(w) \geq 0. \end{cases}$$

We will in the following Chapter 3 discuss how to decide if a point $w \in \mathbb{R}^{n_w}$ is a local optimum of the NLP 2.10. But let us beforehand briefly mention some structural features of the NLP, and also give an example for the multiple shooting parameterization.

Remark on the Initial Value Constraint

In a real-time application of our optimization algorithms, the problem $P(x_0)$ has to be solved several times, each time for a different initial value x_0. In our real-time strategies we will exploit the fact that the actual value of x_0 enters the problem $P(x_0)$ only via the initial value constraint (2.10d). We may therefore isolate this constraint, and summarize the optimization problem $P(x_0)$ as

$$P(x_0) : \min_{s_0^x \in \mathbb{R}^{n_x}, \tilde{w} \in \mathbb{R}^{(n_w - n_x)}} F(s_0^x, \tilde{w}) \quad \text{subject to} \quad \begin{cases} s_0^x - x_0 = 0, \\ \tilde{G}(s_0^x, \tilde{w}) = 0, \\ H(s_0^x, \tilde{w}) \geq 0. \end{cases}$$

This formulation will become crucial in our description of the real-time iteration scheme in Chapter 4.

Remark on Free Horizon Lengths

Note that in the case that the horizon length T is variable, we simply augment the differential state vectors by an additional component $\dot{T}(\tau) = 0$, $\forall \tau \in [0, 1]$; in the multiple

shooting formulation, the continuity conditions (2.10b) enforce that T is constant over the whole horizon; its initial value, however, is *free*, in contrast to the other initial values x_0. To capture problems with variable T in the above NLP formulation, we therefore only have to modify the initial value constraint $s_0^x - x_0 = 0$ to $(\mathbb{I}_{n_x} | \, 0) s_0^x - x_0 = 0$ (note that $x_0 \in \mathbb{R}^{n_x}$ and $s_i^x \in \mathbb{R}^{n_x+1}$).

2.2.1 Free and Dependent Variables

Note that the variables $q = (q_0, q_1, \ldots, q_{N-1})$ may be denoted the "free" components, and $s = (s_0, s_1, \ldots, s_N)$ the "dependent" components, since the constraints (2.10b)-(2.10d) allow to determine all variables s uniquely if q is given (in the case of a free horizon length, as discussed above, the last initial value $s_{0\,n_x+1}^x$ is also free and actually becomes a part of q). If we assume for a moment that no final equality constraint (2.10e) and no inequality constraints (2.10f),(2.10g) are present, we can write the optimization problem in the form

$$\min_{q \in \mathbb{R}^{n_q},\, s \in \mathbb{R}^{n_s}} F(q,s) \quad \text{subject to} \quad G(q,s) \;=\; 0,$$

where the function G has the useful property that its Jacobian $\frac{\partial G}{\partial s}$ with respect to the dependent variables, s, is invertible. To see this, note that $\frac{\partial G}{\partial s}$ is lower block triangular

$$\begin{pmatrix}
\mathbb{I}_{n_x} & & & & & \\
\frac{\partial g}{\partial s_0^x} & \frac{\partial g}{\partial s_0^z} & & & & \\
-\frac{\partial x_0(\tau_1)}{\partial s_0^x} & -\frac{\partial x_0(\tau_1)}{\partial s_0^z} & \mathbb{I}_{n_x} & & & \\
 & & \ddots & & & \\
 & & -\frac{\partial x_{N-1}(\tau_N)}{\partial s_{N-1}^x} & -\frac{\partial x_{N-1}(\tau_N)}{\partial s_{N-1}^z} & \mathbb{I}_{n_x} & \\
 & & & & \frac{\partial g}{\partial s_N^x} & \frac{\partial g}{\partial s_N^z}
\end{pmatrix}$$

with invertible blocks $\mathbb{I}_{n_x}$ and $\frac{\partial g}{\partial s_i^z}$ on the diagonal (the invertibility of $\frac{\partial g}{\partial s_i^z}$ follows from the index one assumption of the DAE system).

In the presence of final equality constraints (2.10e) some previously free variables of the vector q may be declared dependent and it may again be possible to find a separation into free and dependent variables q and s with the invertibility of $\frac{\partial G}{\partial s}$. The same may be done in the presence of active inequality constraints (2.10f) or (2.10g).

The separability into free and dependent components will be used for some convergence results in Chapter 5; it is also exploited by the numerical solution algorithms described in Sections 6.5 and 6.6.

Example 2.1 (Direct Multiple Shooting for the CSTR)
Let us again consider the guiding example of Section 1.2. Choosing N =100 multiple shooting intervals each of 20 seconds length, we arrive at an NLP formulation that comprises $n_w = n_q + n_s = Nn_u + (N+1)n_x = 100 \times 2 + 101 \times 4 = 604$ NLP variables.

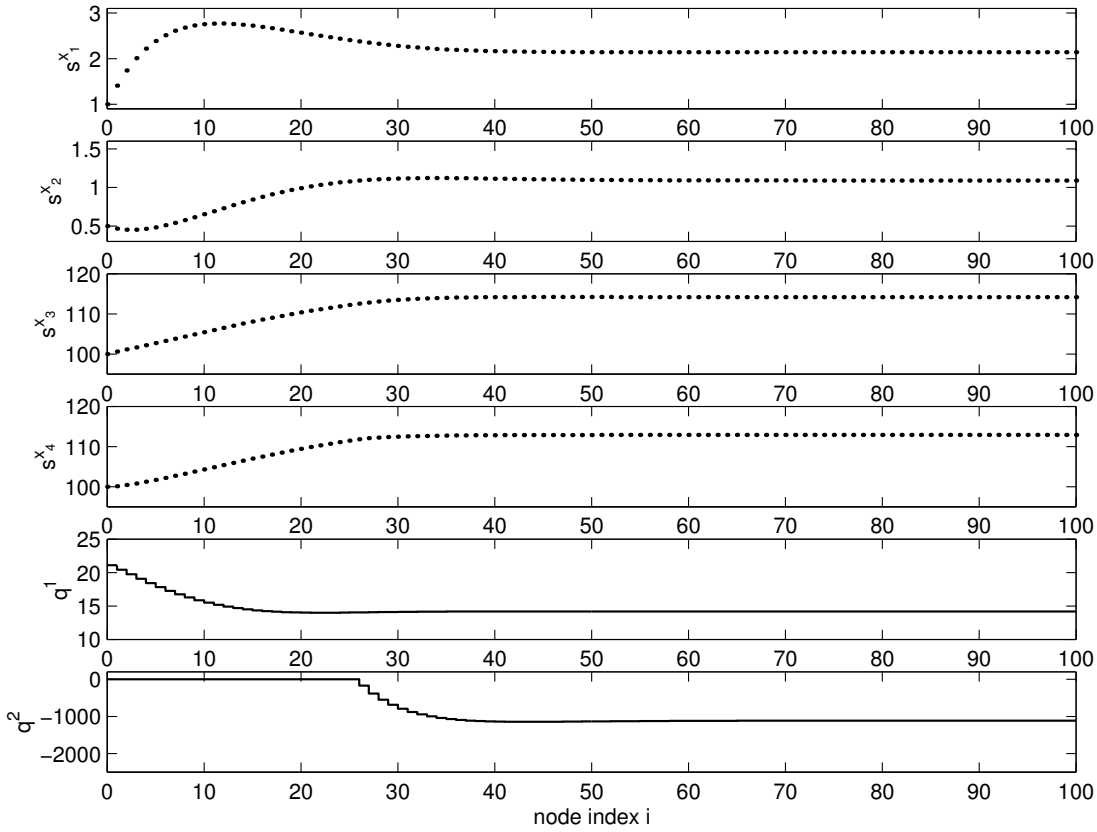


Figure 2.3: Content of NLP variables in the direct multiple shooting method, corresponding to the solution of Figure 1.2. The dots in the first four graphs indicate the multiple shooting node values s_i, the last two graphs show the piecewise constant controls q_i.

The overall number of continuity constraints (2.10b) *is* $Nn_x = 400$*, the initial value constraint* (2.10d) *has dimension* $n_x = 4$*. Together, they form* n_s *constraints, so that* $n_w - n_s = 604 - 404 = 200 = n_q$ *effective degrees of freedom remain. The values for all 604 multiple shooting variables at the solution of problem* $P(x_0)$*, with* x_0 *according to* (1.4)*, are visualized in Figure 2.3.*

Chapter 3

Local Optimality and SQP Methods

This chapter is aimed at the preparation of our numerical methods for the solution of neighboring optimization problems in real-time. We will therefore consider not only one single NLP problem, but a parameterized family of optimization problems

$$P(t): \quad \min_{w \in \mathbb{R}^{n_w}} F(t,w) \quad \text{subject to} \quad \begin{cases} G(t,w) = 0 \\ H(t,w) \geq 0 \end{cases}$$

with C^2 functions $F : \mathbb{R} \times \mathbb{R}^{n_w} \to \mathbb{R}$, $G : \mathbb{R} \times \mathbb{R}^{n_w} \to \mathbb{R}^{n_G}$, and $H : \mathbb{R} \times \mathbb{R}^{n_w} \to \mathbb{R}^{n_H}$. Please note that the scalar homotopy parameter t has *no* relation to the physical time t. It can be thought of as a one dimensional analog for the initial value x_0 that distinguishes between different optimization problems $P(x_0)$ as they arise in the multiple shooting parameterization. The vector w can be regarded as the multiple shooting variables $q_0, \ldots, q_{N-1}, s_0, \ldots, s_N$.

We will first review in Sec. 3.1 some conditions which allow to decide if a point $w^*(t)$ is a (local) solution of an optimization problem $P(t)$ for fixed t. In Section 3.2 we will review a result from parametric optimization that allows to conclude that the solution manifold $w^*(t)$ is continuous and piecewise differentiable with respect to t, in all "benign" points $w^*(t)$ that satisfy rather mild conditions. The nondifferentiable points are those points where the set of binding inequalities changes.

The so called "Sequential Quadratic Programming" (SQP) method is an approach to find a (local) minimum $w^*(t)$ of a problem $P(t)$ for fixed homotopy parameter t. It will be introduced in Sec. 3.3. We also show in Sec. 3.4 that its prototype algorithm, the so called *exact Hessian SQP method*, when applied to an optimization problem $P(t+\epsilon)$ and initialized with the solution of problem $P(t)$, is able to provide a prediction for $w^*(t+\epsilon)$ that is of $O(\|\epsilon\|^2)$, even if the set of binding inequalities changes at the point t. This astonishing property, however, requires a slight reformulation of the optimization problems, which leads directly to the idea of the initial value embedding strategy, a crucial feature of the real-time iteration approach presented in Chap. 4.

3.1 Local Optimality Conditions

For notational convenience, let us first drop the parameter t and treat a single NLP problem

$$\min_{w \in \mathbb{R}^{n_w}} F(w) \quad \text{subject to} \quad \begin{cases} G(w) = 0 \\ H(w) \geq 0 \end{cases} \tag{3.1}$$

where the functions $F : \mathbb{R}^{n_w} \to \mathbb{R}$, $G : \mathbb{R}^{n_w} \to \mathbb{R}^{n_G}$, and $H : \mathbb{R}^{n_w} \to \mathbb{R}^{n_H}$ are twice continuously differentiable.

Let us generalize the definition of the gradient ∇F of a scalar function F to the gradient $\nabla_w G$ of vector functions G as the transpose of the Jacobian matrix

$$\nabla_w G(w) := \left(\tfrac{\partial G}{\partial w}(w)\right)^T.$$

A *feasible point* is a point $w \in \mathbb{R}^{n_w}$ that satisfies $G(w) = 0$ and $H(w) \geq 0$. A *local minimum* of the NLP (3.1) is a feasible point w^* which has the property that $F(w^*) \leq F(w)$ for all feasible points w in a neighborhood of w^*. A *strict local minimum* satisfies $F(w^*) < F(w)$ for all neighboring feasible points $w \neq w^*$.

Active inequality constraints at a feasible point w are those components $H_j(w)$ of $H(w)$ with $H_j(w) = 0$. We will subsume the equality constraints and the active inequalities at a point w (the so called the *active set*) in a combined vector function of *active constraints*:

$$\tilde{G}(w) := \begin{pmatrix} G(w) \\ H^{\text{act}}(w) \end{pmatrix}.$$

Note that the active set may be different at different feasible points w.

Regular points are feasible points w that satisfy the condition that the Jacobian of the active constraints, $\nabla \tilde{G}(w)^T$, has full rank, i.e., that all rows of $\nabla \tilde{G}(w)^T$ are linearly independent.

To investigate local optimality in the presence of constraints, it is very useful to introduce the *Lagrangian multiplier* vectors $\lambda \in \mathbb{R}^{n_G}$ and $\mu \in \mathbb{R}^{n_H}$, that are also called *adjoint variables*, as they correspond one-to-one to the constraint functions G and H, and to define the so called *Lagrangian function* $\mathcal{L}$ by

$$\mathcal{L}(w, \lambda, \mu) := F(w) - \lambda^T G(w) - \mu^T H(w). \tag{3.2}$$

We will now state a variant of the *Karush-Kuhn-Tucker* necessary conditions for local optimality of a point w^*. These conditions have been first derived by Karush in 1939 [Kar39] – and independently by Kuhn and Tucker in 1951 [KT51]. (A proof of the following two theorems can be found in virtually any textbook on nonlinear programming, e.g. Bazaara and Shetty [BS79] or Nocedal and Wright [NW99].) For brevity, we will restrict our attention to regular points only.

Theorem 3.1 (Karush-Kuhn-Tucker Conditions)
If a regular point $w^* \in \mathbb{R}^{n_w}$ is a local optimum of the NLP (3.1), then there exist unique Lagrange multiplier vectors $\lambda^* \in \mathbb{R}^{n_G}$ and $\mu^* \in \mathbb{R}^{n_H}$ so that the triple (w^*, λ^*, μ^*) satisfies the following necessary conditions:

$$\begin{aligned} \nabla_w \mathcal{L}(w^*, \lambda^*, \mu^*) &= 0 && \text{(3.3a)} \\ G(w^*) &= 0, && \text{(3.3b)} \\ H(w^*) &\geq 0, && \text{(3.3c)} \\ \mu^* &\geq 0, && \text{(3.3d)} \\ \mu_j^* \, H_j(w^*) &= 0, \quad j = 1, 2, \ldots, n_H. && \text{(3.3e)} \end{aligned}$$

A triple (w^*, λ^*, μ^*) that satisfies the Karush-Kuhn-Tucker conditions (3.3) is called a KKT point. Note that the so called *complementarity condition* (3.3e) implies that $\mu_j^* = 0$ at inactive constraints $H_j(w^*) > 0$. At active constraints $H_j(w^*) = 0$ the corresponding multipliers μ^* may also become zero. Active constraints with zero multipliers are called *weakly* active, and those with positive multipliers *strongly* active. Let us subdivide the active set vector function $H^{\text{act}}(w^*)$ at a KKT point (w^*, λ^*, μ^*) into its strongly and weakly active parts, i.e., let us write

$$H^{\text{act}}(w^*) =: \begin{pmatrix} H^{\text{s.act}} \\ H^{\text{w.act}} \end{pmatrix} (w^*).$$

A KKT point for which all active constraints are strongly active is said to satisfy the *strict complementarity* condition.

Quadratic Programs

One special class of optimization problem plays a preeminent role in the SQP algorithms that are presented later in this chapter and deserves some remarks: *quadratic programs* (QP) are those optimization problems (3.1) that have a quadratic objective function and linear constraint functions, i.e., problems of the type

$$\min_{w \in \mathbb{R}^{n_w}} \frac{1}{2} w^T A w + a^T w \quad \text{subject to} \quad \begin{cases} b + Bw &= 0 \\ c + Cw &\geq 0 \end{cases} \qquad (3.4)$$

with vectors $a \in \mathbb{R}^{n_w}$, $b \in \mathbb{R}^{n_G}$, $c \in \mathbb{R}^{n_H}$, and matrices $A \in \mathbb{R}^{n_w} \times \mathbb{R}^{n_w}$, $B \in \mathbb{R}^{n_G} \times \mathbb{R}^{n_w}$, and $C \in \mathbb{R}^{n_H} \times \mathbb{R}^{n_w}$.

A variety of highly developed algorithms to solve QPs exists, and the success of SQP type methods is to a large part due to the fact that QPs are very efficiently solvable.

The conditions (3.3) for a point (w^*, λ^*, μ^*) to be a KKT point of the above QP are:

$$\begin{aligned} Aw^* + a - B^T\lambda^* - C^T\mu^* &= 0 \\ b + Bw^* &= 0, \\ c + Cw^* &\geq 0, \\ \mu^* &\geq 0, \\ \mu_j^* \left(c_j + C_{j,\cdot} w^*\right) &= 0, \quad j = 1, 2, \ldots, n_H. \end{aligned}$$

For QPs without inequalites, the KKT conditions can be written in the compact form

$$\begin{pmatrix} A & B^T \\ B & 0 \end{pmatrix} \begin{pmatrix} w^* \\ -\lambda^* \end{pmatrix} = \begin{pmatrix} -a \\ -b \end{pmatrix}.$$

The matrix on the left hand side is called the KKT matrix. It is invertible, if B has full rank n_G and A is positive definite on the null space of B, as stated in the following lemma. The invertibility of the KKT matrix implies that the equality constrained QP has a unique KKT point.

Lemma 3.2 (Invertibility of the KKT Matrix)
Let us assume that $A \in \mathbb{R}^n \times \mathbb{R}^n$ is a symmetric matrix and $B \in \mathbb{R}^m \times \mathbb{R}^n$ has full rank $m \leq n$. Let us furthermore assume that A is positive definite on the null space of B. Then the matrix

$$\begin{pmatrix} A & B^T \\ B & 0 \end{pmatrix}$$

is invertible.

A short proof of this lemma can be found in Nocedal and Wright [NW99, Lemma 16.1].

The existence and uniqueness of a KKT point can also be shown for inequality constrained QPs, e.g. under the two additional assumptions that the feasible set is non-empty, and that the combined constraint matrix (B^T, C^T) has full rank $n_G + n_H$. We will encounter such a uniquely solvable quadratic programming problem in Theorem 3.4.

First, however, let us review *sufficient* conditions for a KKT point to be a strict local optimizer.

Theorem 3.3 (Strong Second Order Sufficient Conditions)
Sufficient conditions for a point $w^* \in \mathbb{R}^{n_w}$ to be a strict local minimizer of (3.1) *are:*

- *w^* is a regular point,*
- *there exist multiplier vectors $\lambda^* \in \mathbb{R}^{n_G}$ and $\mu^* \in \mathbb{R}^{n_H}$, such that (w^*, λ^*, μ^*) is a KKT point, and*

- *the Hessian matrix* $\nabla_w^2 \mathcal{L}(w^*, \lambda^*, \mu^*)$[1] *is positive definite on the null space* $\mathcal{N}^{\mathrm{s}}$ *of the linearized strongly active constraints*

$$\tilde{G}^{\mathrm{s}}(w^*) := \begin{pmatrix} G \\ H^{\mathrm{s.act}} \end{pmatrix}(w^*),$$

i.e., for every non-zero vector $\Delta w \in \mathcal{N}^{\mathrm{s}}$,

$$\mathcal{N}^{\mathrm{s}} := \{\Delta w \in \mathbb{R}^{n_w} | \nabla_w \tilde{G}^{\mathrm{s}}(w^*)^T \Delta w = 0\},$$

it holds that

$$\Delta w^T \, \nabla_w^2 \mathcal{L}(w^*, \lambda^*, \mu^*) \, \Delta w \; > \; 0.$$

Remark: The sufficient conditions of the theorem are called "strong" second order sufficient conditions, because weaker sufficient conditions exists, which require only the positive definiteness of $\nabla_w^2 \mathcal{L}(w^*, \lambda^*, \mu^*)$ on a cone

$$\mathcal{N}^{\mathrm{w}} := \{\Delta w \in \mathcal{N}^{\mathrm{s}} | \nabla_w \tilde{H}^{\mathrm{w.act}}(w^*)^T \Delta w \geq 0\}.$$

We have chosen the strong formulation, as it turns out that the strong second order sufficient conditions for optimality, as stated in Theorem 3.3, have the desirable property that a KKT point (w^*, λ^*, μ^*) that satisfies them is stable against perturbations in the problem functions F, G and H, as we will investigate in the following section.

3.2 Piecewise Differentiable Dependence on Perturbations

Let us now consider a parameterized family of optimization problems $P(t)$

$$\min_{w \in \mathbb{R}^{n_w}} F(t, w) \quad \text{subject to} \quad \begin{cases} G(t, w) \; = \; 0 \\ H(t, w) \; \geq \; 0 \end{cases} \tag{3.6}$$

where the functions $F : \mathbb{R} \times \mathbb{R}^{n_w} \to \mathbb{R}$, $G : \mathbb{R} \times \mathbb{R}^{n_w} \to \mathbb{R}^{n_G}$, and $H : \mathbb{R} \times \mathbb{R}^{n_w} \to \mathbb{R}^{n_H}$ are C^2. We want to investigate how the solution points $(w^*(t), \lambda^*(t), \mu^*(t))$ depend on the variable t, or, in the language of parametric optimization, we want to investigate the set

$$\Sigma_{\mathrm{loc}} := \{(t, w) \in \mathbb{R} \times \mathbb{R}^{n_w} | w \text{ is a local minimizer for } P(t))\}.$$

We restrict our attention to the subset of points $(t, w^*(t))$ from Σ_{loc} that satisfy the strong second order sufficient conditions of Theorem 3.3. The main result of this section is that

[1] Here we use the definition $\nabla_w^2 \mathcal{L} := \frac{\partial^2 \mathcal{L}}{\partial w^2}$.

the points $(w^*(t), \lambda^*(t), \mu^*(t))$ form a continuous and piecewise differentiable curve on this subset, if an additional technical assumption is satisfied. For a much more detailed discussion of the properties of the set Σ_{loc} we refer to the book on parametric optimization by Guddat, Guerra Vasquez and Jongen [GVJ90].

Before we formulate this theorem, we will give a simple example for illustration.

Example 3.1 (Piecewise Differentiability)
Consider the family $P(t)$ of simple optimization problems

$$\min_{w \in \mathbb{R}} \ \frac{1}{2} w^2 \quad \text{subject to} \quad -t + \sinh(w) \ \geq \ 0$$

The solution curves $w^*(t), \mu^*(t)$ can easily be found to be

$$\begin{aligned} w^*(t) &= \max(0, \operatorname{arcsinh}(t)), \\ \mu^*(t) &= \frac{w^*(t)}{\cosh(w^*(t))}. \end{aligned}$$

These curves are continuous and piecewise differentiable with piecewise derivatives

$$\begin{aligned} \frac{\partial w^*}{\partial t}(t) &= \begin{cases} 0, & \text{if} \quad t < 0, \\ \cosh(\operatorname{arcsinh}(t))^{-1}, & \text{if} \quad t > 0, \end{cases} \\ \frac{\partial \mu^*}{\partial t}(t) &= \cosh(w^*(t))^{-1}(1 - \tanh(w^*(t))) \frac{\partial w^*}{\partial t}(t). \end{aligned}$$

The graph of $w^*(t)$ is depicted in Figure 3.1. How can the manifold be characterized in the vicinity of the continuous but non-differentiable point $w^*(0)$?

We will now formulate the basic theorem of this section, which is proved in Appendix C. A very similar formulation of the theorem and a proof can be found in [GVJ90] (Theorem 3.3.4 and Corollary 3.3.1 (2)).

Theorem 3.4 (One Sided Differentiability)
Consider a parameterized family of optimization problems $P(t)$ as defined in (3.6). Let us assume that we have found, for problem $P(0)$, a KKT point $(w^*(0), \lambda^*(0), \mu^*(0))$ that satisfies the sufficient optimality conditions of Theorem 3.3, with strongly and weakly active set vectors $H^{\text{s.act}}$ and $H^{\text{w.act}}$.

Let us furthermore assume that the solution $(\delta w_*, \delta \lambda_*, \delta \mu_*^{\text{s.act}}, \delta \mu_*^{\text{w.act}})$ of the following quadratic program (with all derivatives evaluated at the solution point $(w^*(0), \lambda^*(0), \mu^*(0))$ for $t = 0$)

$$\begin{aligned} \min_{\delta w \in \mathbb{R}^{n_w}} \ & \frac{1}{2} \ \delta w^T \ \nabla_w^2 \mathcal{L} \ \delta w \ + \ \left(\frac{\partial}{\partial t} \nabla_w \mathcal{L} \right)^T \delta w \\ \text{subject to} \ & \begin{cases} \frac{\partial G}{\partial t} + \nabla_w G^T \delta w &= 0 \\ \frac{\partial H^{\text{s.act}}}{\partial t} + (\nabla_w H^{\text{s.act}})^T \delta w &= 0. \\ \frac{\partial H^{\text{w.act}}}{\partial t} + (\nabla_w H^{\text{w.act}})^T \delta w &\geq 0. \end{cases} \end{aligned} \tag{3.7}$$

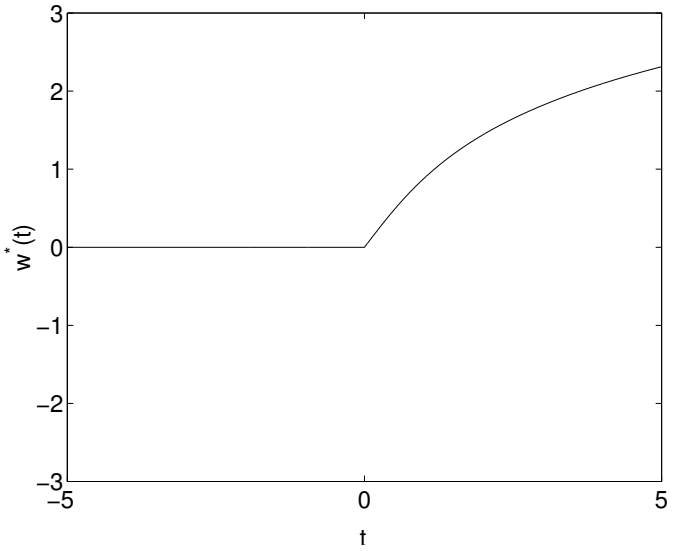


Figure 3.1: Graph of the solution manifold $w^*(t)$ of Example 3.1.

satisfies the strict complementarity condition for the multiplier vector $\delta\mu_*^{\text{w.act}}$ *of the inequality constraints.*

Then there exists an $\epsilon > 0$ *and a differentiable curve* $v : [0, \epsilon) \to \mathbb{R}^{n_w} \times \mathbb{R}^{n_G} \times \mathbb{R}^{n_H}$,

$$t \mapsto \begin{pmatrix} w^*(t) \\ \lambda^*(t) \\ \mu^*(t) \end{pmatrix}$$

of KKT points that satisfy the sufficient optimality conditions of Theorem 3.3 for the corresponding problems $P(t)$, $t \in [0, \epsilon)$. *At* $t = 0$ *the one sided derivative of this curve is given by*

$$\lim_{t \to 0, t>0} \frac{1}{t} \begin{pmatrix} w^*(t) - w^*(0) \\ \lambda^*(t) - \lambda^*(0) \\ \mu^*(t) - \mu^*(0) \end{pmatrix} = \begin{pmatrix} \delta w_* \\ \delta\lambda_* \\ \begin{pmatrix} \\ \delta\mu_* \\ \end{pmatrix} \end{pmatrix} := \begin{pmatrix} \delta w_* \\ \delta\lambda_* \\ \delta\mu_*^{\text{s.act}} \\ \delta\mu_*^{\text{w.act}} \\ 0 \end{pmatrix}.$$

Remark 1: Note that the QP (3.7) always has a unique solution $(\delta w_*, \delta\lambda_*, \delta\mu_*^{\text{s.act}}, \delta\mu_*^{\text{w.act}})$. This is due to the positive definiteness of $\nabla_w^2 \mathcal{L}$ on the null space of the equality constraint matrix $\nabla_w \tilde{G}^{\text{s}T}$, the feasibility of the QP ($\delta w = 0$ is feasible), and the fact that the constraint matrix $(\nabla_w G, \nabla_w H^{\text{s.act}}, \nabla_w H^{\text{w.act}})$ has full rank due to the assumption that $w^*(0)$ is a regular point.

Remark 2: The only further requirement in addition to the sufficient conditions of Theorem 3.3 is the – technical – assumption of strict complementarity in the solution of the QP (3.7). It is needed to guarantee that there exists an $\epsilon > 0$ so that the active set of the local solutions of $P(t)$ does not change for $t \in (0, \epsilon)$.

Remark 3: The theorem treats only the existence of the "right" hand side of the solution curve $(w^*(t), \lambda^*(t), \mu^*(t))$ on the interval $t \in [0, \epsilon)$. If the strict complementarity condition is also satisfied for the solution $(\delta w'_*, \delta\lambda'_*, \delta\mu_*^{\text{s.act}\prime}, \delta\mu_*^{\text{w.act}\prime})$ of an inverted version of the QP (3.7), namely of

$$\begin{aligned} \min_{\delta w \in \mathbb{R}^{n_w}} \quad & \frac{1}{2}\, \delta w^T\, \nabla_w^2 \mathcal{L}\, \delta w \;+\; \left(-\frac{\partial}{\partial t}\nabla_w \mathcal{L}\right)^T \delta w \\ \text{subject to} \quad & \begin{cases} \left(-\frac{\partial G}{\partial t}\right) + \nabla_w G^T \delta w &= 0 \\ \left(-\frac{\partial H^{\text{s.act}}}{\partial t}\right) + \left(\nabla_w H^{\text{s.act}}\right)^T \delta w &= 0. \\ \left(-\frac{\partial H^{\text{w.act}}}{\partial t}\right) + \left(\nabla_w H^{\text{w.act}}\right)^T \delta w &\geq 0, \end{cases} \end{aligned} \tag{3.8}$$

then also the "left" hand side solution curve $t \in (-\epsilon', 0] \mapsto (w^*(t), \lambda^*(t), \mu^*(t))$, $\epsilon' > 0$, exists, with the one sided derivative

$$\lim_{t \to 0, t<0} \frac{1}{t} \begin{pmatrix} w^*(t) - w^*(0) \\ \lambda^*(t) - \lambda^*(0) \\ \mu^*(t) - \mu^*(0) \end{pmatrix} = \begin{pmatrix} -\delta w'_* \\ -\delta\lambda'_* \\ -\delta\mu_*^{\text{s.act}\prime} \\ -\delta\mu_*^{\text{w.act}\prime} \\ 0 \end{pmatrix}.$$

This is an immediate consequence of the theorem, applied to the reversed problem family $P'(t) := P(-t)$.

Remark 4: If the reference point $(w^*(0), \lambda^*(0), \mu^*(0))$ itself satisfies the strict complementarity condition, then no weakly active constraints $H^{\text{w.act}}$ exist, and the original and inverted QP, (3.7) and (3.8), do *not* contain any inequality constraints. Therefore, the assumption of strict complementarity in the QP solution is trivially satisfied for both problems, and the solution curve exists on both sides, for $t \in (-\epsilon', \epsilon)$. Furthermore, from symmetry follows that the QP solutions coincide (up to a sign change),

$$\begin{pmatrix} \delta w_* \\ \delta\lambda_* \\ \delta\mu_*^{\text{s.act}} \end{pmatrix} = \begin{pmatrix} -\delta w'_* \\ -\delta\lambda'_* \\ -\delta\mu_*^{\text{s.act}\prime} \end{pmatrix},$$

so that the derivative of the curve $t \in (-\epsilon', \epsilon) \mapsto (w^*(t), \lambda^*(t), \mu^*(t))$ exists and is continuous everywhere, also at the point $t = 0$.

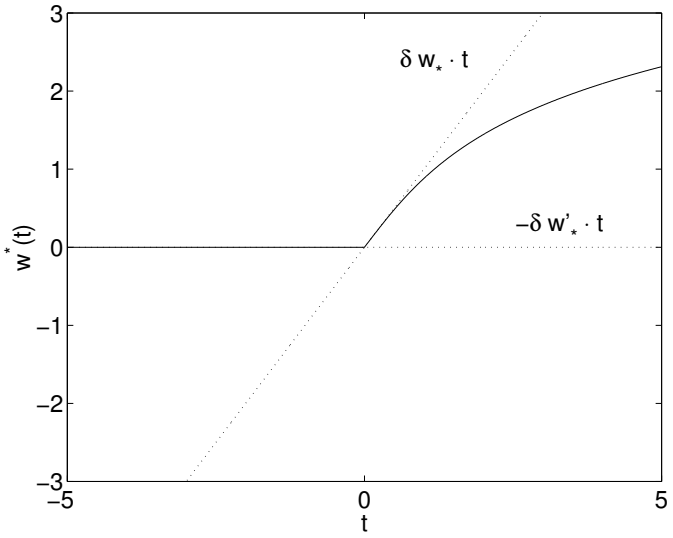


Figure 3.2: Two sided derivative of the solution manifold $w^*(t)$ of Example 3.1.

Example 3.2 (One Sided Differentiability)
Let us again consider the family $P(t)$ of simple optimization problems of Example 3.1. For $t = 0$ the reference solution is $w^*(0) = \mu^*(0) = 0$, and the quadratic programming subproblem (3.7) *as in Theorem 3.4 is*

$$\min_{\delta w \in \mathbb{R}} \ \frac{1}{2}\, \delta w^2 \quad \text{subject to} \quad -1 + \delta w \ \geq \ 0,$$

with the solution

$$\delta w_* = 1 \quad \text{and} \quad \delta \mu_*^{\text{w.act}} = 1,$$

which corresponds to the "right" hand side derivatives of $w^*(t), \mu^*(t)$ for $t \to 0, t > 0$.

Conversely, the inverted quadratic programming subproblem (3.8) *is*

$$\min_{\delta w \in \mathbb{R}} \ \frac{1}{2}\, \delta w^2 \quad \text{subject to} \quad 1 + \delta w \ \geq \ 0,$$

which has the solution

$$\delta w'_* = 0 \quad \text{and} \quad \delta \mu_*^{\text{w.act}'} = 0.$$

This solution corresponds to the (inverted) derivatives of $w^*(t), \mu^*(t)$ for $t \to 0, t < 0$. The two sides of the derivative are illustrated in Figure 3.2.

3.3 Sequential Quadratic Programming

Sequential Quadratic Programming (SQP) is an iterative technique to find a KKT point (w^*, λ^*, μ^*) of an NLP

$$\min_{w \in \mathbb{R}^{n_w}} F(w) \quad \text{subject to} \quad \begin{cases} G(w) &= 0 \\ H(w) &\geq 0 \end{cases}$$

Starting with an initial guess $y_0 = (w_0, \lambda_0, \mu_0)$, an SQP method iterates

$$y_{k+1} = y_k + \alpha_k \Delta y_k \tag{3.9}$$

where $\alpha_k \in (0, 1]$ and

$$\Delta y_k = \begin{pmatrix} \Delta w_k \\ \Delta \lambda_k \\ \Delta \mu_k \end{pmatrix} := \begin{pmatrix} \Delta w_k \\ \tilde{\lambda}_k - \lambda_k \\ \tilde{\mu}_k - \mu_k \end{pmatrix}$$

is obtained from the solution point $(\Delta w_k, \tilde{\lambda}_k, \tilde{\mu}_k)$ of the following quadratic program

$$\begin{array}{ll} \min\limits_{\Delta w \in \Omega_k} & \frac{1}{2}\ \Delta w^T\ A_k\ \Delta w \quad + \quad \nabla_w F(w_k)^T \Delta w \\ \text{subject to} & \begin{cases} G(w_k) + \nabla_w G(w_k)^T \Delta w &= 0 \\ H(w_k) + \nabla_w H(w_k)^T \Delta w &\geq 0 \end{cases} \end{array} \tag{3.10}$$

Existing SQP methods differ mainly by the choice of the steplength α_k, the choice of the so called Hessian matrix A_k and the choice of the set $\Omega_k \subset \mathbb{R}^{n_w}$. The iterates y_k according to Eq. (3.9) form a sequence that is expected to converge towards a KKT point $y^* = (w^*, \lambda^*, \mu^*)$ of the NLP. In practice, the iterations are stopped when a prespecified convergence criterion is fulfilled.

We will in this section introduce only one SQP method that is theoretically very appealing: the full step exact Hessian SQP method, that was first introduced by Wilson [Wil63].

3.3.1 The Full Step Exact Hessian SQP Method

The full step exact Hessian SQP method is distinguished by the choices $\alpha_k := 1$, $\Omega_k := \mathbb{R}^{n_w}$, and, most important,

$$A_k := \nabla_w^2 \mathcal{L}(w_k, \lambda_k, \mu_k).$$

To see why this choice is advantageous, let us first regard an equality constrained problem. In this case, the necessary optimality conditions for the QP solution $(\Delta w_k, \tilde{\lambda}_k)$ are

$$\begin{aligned} \nabla_w^2 \mathcal{L}(w_k, \lambda_k) \Delta w_k + \nabla_w F(w_k) - \nabla_w G(w_k) \tilde{\lambda}_k &= 0, \\ G(w_k) + \nabla_w G(w_k)^T \Delta w_k &= 0. \end{aligned}$$

By substituting $\tilde{\lambda}_k = \lambda_k + \Delta\lambda_k$ we can write this equivalently as

$$\begin{array}{llll} \nabla_w \mathcal{L}(w_k, \lambda_k) & +\nabla_w^2 \mathcal{L}(w_k, \lambda_k) \quad \Delta w_k & -\nabla_w G(w_k) \ \Delta\lambda_k & = 0, \\ G(w_k) & +\nabla_w G(w_k)^T \quad \Delta w_k & & = 0 \end{array}$$

which corresponds to the Newton-Raphson iteration rule

$$\begin{pmatrix} \nabla_w \mathcal{L}(w_k, \lambda_k) \\ G(w_k) \end{pmatrix} + \frac{\partial}{\partial(w, \lambda)} \begin{pmatrix} \nabla_w \mathcal{L}(w_k, \lambda_k) \\ G(w_k) \end{pmatrix} \begin{pmatrix} \Delta w_k \\ \Delta\lambda_k \end{pmatrix} = 0,$$

for the solution of the KKT system

$$\begin{pmatrix} \nabla_w \mathcal{L}(w, \lambda) \\ G(w) \end{pmatrix} = \begin{pmatrix} \nabla_w F(w) - \nabla_w G(w)\lambda \\ G(w) \end{pmatrix} = 0.$$

This equivalence proves that the full step exact Hessian SQP method shows the same excellent local convergence behaviour as the Newton-Raphson method, in the vicinity of a solution (w^*, λ^*) of the KKT system. Note, however, that it is necessary to start with a good initial guess not only for the primal variables w, but also for the multipliers λ. Fortunately, it turns out that the initial guess λ_0 of the multipliers is not as crucial as the initial guess w_0 for the primal variables, due to the special structure of the KKT system. This is expressed in the following theorem (for a proof we refer to Fletcher [Fle87]).

Theorem 3.5 (Convergence of the Exact Hessian SQP Method)
If a point (w^*, λ^*) satisfies the sufficient optimality conditions of Theorem 3.3 of an equality-constrained *NLP problem, and if w_0 is sufficiently close to w^*, and if λ_0 is chosen such that the matrix*

$$\begin{pmatrix} \nabla_w^2 \mathcal{L}(w_0, \lambda_0) & -\nabla_w G(w_0) \\ \nabla_w G(w_0)^T & 0 \end{pmatrix}$$

is invertible, then the sequence of iterates generated by the full step exact Hessian SQP method, i.e., the sequence (w_k, λ_k) of iterates that satisfies

$$\begin{pmatrix} \nabla_w^2 \mathcal{L}(w_k, \lambda_k) & -\nabla_w G(w_k) \\ \nabla_w G(w_k)^T & 0 \end{pmatrix} \begin{pmatrix} w_{k+1} - w_k \\ \lambda_{k+1} \end{pmatrix} = -\begin{pmatrix} \nabla_w F(w_k) \\ G(w_k) \end{pmatrix}$$

converges q-quadratically to (w^*, λ^*), i.e.,

$$\left\| \begin{pmatrix} w_{k+1} - w^* \\ \lambda_{k+1} - \lambda^* \end{pmatrix} \right\| \leq C \left\| \begin{pmatrix} w_k - w^* \\ \lambda_k - \lambda^* \end{pmatrix} \right\|^2$$

with some constant $C \geq 0$.

3.3.2 Active Set Determination

In Theorem 3.5, local convergence is only proven for equality constrained problems. In the presence of inequality constraints, however, we may assume that close to the solution the active set does not change, so that the reasoning for equality constrained problems is still applicable. This assumption is valid in the vicinity of a KKT point (w^*, λ^*, μ^*) that satisfies the second order sufficient conditions of Theorem 3.3 *and* the strict complementarity condition.

The strength of the QP formulation (3.10) during the SQP iterations is that it allows to determine the multipliers and the active set without prior knowledge of them. To show this, let us assume that we have found a KKT point $y^* = (w^*, \lambda^*, \mu^*)$ that satisfies the first order necessary conditions of Theorem 3.1:

$$\begin{aligned} \nabla F(w^*) - \nabla_w G(w^*)\lambda^* - \nabla H(w^*)\mu^* &= 0 \\ G(w^*) &= 0 \\ H(w^*) &\geq 0 \\ \mu^* &\geq 0 \\ \mu_j^* \, H(w^*)_j &= 0, \quad j = 1, 2, \ldots, n_H. \end{aligned}$$

Let us now assume that we formulate the first QP 3.10 for the determination of Δy_0, initialized at $y_0 = y^*$, with some Hessian matrix A_0. The necessary conditions of optimality for the QP solution $(\Delta w_0, \tilde{\lambda}_0, \tilde{\mu}_0)$ are

$$\begin{aligned} A_0 \, \Delta w_0 + \nabla F(w^*) - \nabla_w G(w^*)\tilde{\lambda}_0 - \nabla H(w^*)\tilde{\mu}_0 &= 0 \\ G(w^*) + \nabla_w G(w^*)^T \Delta w_0 &= 0 \\ H(w^*) + \nabla_w H(w^*)^T \Delta w_0 &\geq 0 \\ \tilde{\mu}_0 &\geq 0 \\ \tilde{\mu}_{0,j} \, (H(w^*) + \nabla_w H(w^*)^T \Delta w_0)_j &= 0, \quad j = 1, 2, \ldots, n_H. \end{aligned}$$

It can be seen that $(\Delta w_0, \tilde{\lambda}_0, \tilde{\mu}_0) = (0, \lambda^*, \mu^*)$ satisfies these conditions, and assuming positive definiteness of A_0 on the null space of the equality constraints $\nabla_w G(w^*)^T$, this solution is also the unique optimum of the QP: multipliers and active set are detected from knowledge of w^* only. We may therefore infer that even in a neighborhood of a local optimum w^*, the multipliers and active set can be determined by the SQP algorithm, under a weak positive definiteness assumption on the matrix A_0. This can indeed be proven, under the condition that (w^*, λ^*, μ^*) satisfies the second order sufficient conditions of Theorem 3.3 and the strict complementarity condition. For a detailed discussion and a proof we refer to Robinson [Rob74].

3.4 SQP for a Parameterized Problem Family

Let us in the sequel consider the parameterized family of augmented optimization problems

$$\breve{P}(\breve{t}): \quad \min_{t \in \mathbb{R},\, w \in \mathbb{R}^{n_w}} F(t,w) \quad \text{subject to} \quad \begin{cases} t - \breve{t} &= 0 \\ G(t,w) &= 0 \\ H(t,w) &\geq 0 \end{cases} \tag{3.14}$$

where the functions $F : \mathbb{R} \times \mathbb{R}^{n_w} \to \mathbb{R}$, $G : \mathbb{R} \times \mathbb{R}^{n_w} \to \mathbb{R}^{n_G}$, and $H : \mathbb{R} \times \mathbb{R}^{n_w} \to \mathbb{R}^{n_H}$ are in C^2. This family is equivalent to the family $P(t)$ of problems (3.6) in Section 3.2, with the only difference that t is now introduced as an additional variable which is fixed by the additional constraint $t - \breve{t} = 0$. This addition of t to the SQP variables has the consequence that derivatives with respect to t are evaluated in the SQP algorithm, which allows to perform the transition between different optimization problems in such a way that a first order approximation of the solution manifold as in Theorem 3.4 is provided by the first iterate.

Let us for brevity define

$$\breve{w} := \begin{pmatrix} t \\ w \end{pmatrix}, \quad \breve{\lambda} := \begin{pmatrix} \lambda_t \\ \lambda \end{pmatrix}, \quad \text{and} \quad \breve{G}(\breve{w}) := \begin{pmatrix} t - \breve{t} \\ G(t,w) \end{pmatrix},$$

so that the Lagrangian function $\breve{L}$ of problem $\breve{P}(\breve{t})$ can be written as

$$\begin{aligned} \breve{\mathcal{L}}(\breve{w}, \breve{\lambda}, \mu) &:= F(\breve{w}) - \breve{\lambda}^T \breve{G}(\breve{w}) - \mu^T H(\breve{w}) \\ &= F(t,w) - \lambda_t(t - \breve{t}) - \lambda^T G(t,w) - \mu^T H(t,w) \\ &= \mathcal{L}(t,w,\lambda,\mu) - \lambda_t(t - \breve{t}), \end{aligned}$$

where

$$\mathcal{L}(t,w,\lambda,\mu) = F(t,w) - \lambda^T G(t,w) - \mu^T H(t,w)$$

is the Lagrangian function for the parameterized family of optimization problems $P(t)$ of Section 3.2.

We will now show that the first full step exact Hessian SQP iterate for the enlarged problem $\breve{P}(\epsilon)$, when started at the solution $(\breve{w}^*(0), \breve{\lambda}^*(0), \mu^*(0))$ of $\breve{P}(0)$, is closely related to the one sided derivative of the solution manifold $(w^*(\cdot), \lambda^*(\cdot), \mu^*(\cdot))$ of the problems $P(t)$, as in Theorem 3.4.

Theorem 3.6 (First Order Prediction by Exact Hessian SQP)
Let us assume that we have found a KKT point $(\breve{w}^*(0), \breve{\lambda}^*(0), \mu^*(0))$ of problem $\breve{P}(0)$ that satisfies the sufficient optimality conditions of Theorem 3.3. If a full step SQP algorithm with exact Hessian for the solution of the problem $\breve{P}(\epsilon)$, with $\epsilon > 0$ sufficiently small, is started with this solution as an initial guess, then the nontrivial part of the first SQP step, $(\Delta w, \Delta\lambda, \Delta\mu)$, is identical to ϵ times the one sided derivative of the solution manifold $(w^*(\cdot), \lambda^*(\cdot), \mu^*(\cdot))$ of problems $P(t)$ as given in Theorem 3.4, i.e.,

$$\frac{1}{\epsilon} \begin{pmatrix} \Delta w \\ \Delta\lambda \\ \Delta\mu \end{pmatrix} = \begin{pmatrix} \delta w_* \\ \delta\lambda_* \\ \delta\mu_* \end{pmatrix} = \lim_{t \to 0, t>0} \frac{1}{t} \begin{pmatrix} w^*(t) - w^*(0) \\ \lambda^*(t) - \lambda^*(0) \\ \mu^*(t) - \mu^*(0) \end{pmatrix}$$

Remark: The first order prediction provided by the exact Hessian SQP is equivalent to one step of the *Euler predictor* pathfollowing method in parametric optimization [GVJ90, Sec. 3.3, p. 73].

Proof: For a proof first note that $\nabla^2_{\breve{w}}\breve{\mathcal{L}} = \nabla^2_{\breve{w}}\mathcal{L}$ due to the linearity of the constraint $t - \epsilon = 0$, so that the value of the additional multiplier λ_t plays no role in the QP (3.10). Furthermore, it can easily be seen that $(\breve{w}^*(0), \breve{\lambda}^*(0), \mu^*(0))$ satisfies the sufficient optimality conditions of Theorem 3.3 for problem $\breve{P}(0)$ if and only if $(w^*(0), \lambda^*(0), \mu^*(0))$ satisfies them for problem $P(0)$, and $\lambda_t^*(0) = \frac{\partial}{\partial t}\mathcal{L}(0, w^*(0), \lambda^*(0), \mu^*(0))$.

The QP (3.10) for the first SQP iterate can be written in the form

$$\min_{\Delta t, \Delta w} \; \frac{1}{2}\Delta w^T \nabla_w^2 \mathcal{L} \Delta w + \Delta t \frac{\partial}{\partial t}\nabla_w \mathcal{L}^T \Delta w + \nabla_w F^T \Delta w + \frac{\partial F}{\partial t}\Delta t + \frac{1}{2}\frac{\partial^2 \mathcal{L}}{\partial t^2}\Delta t^2$$
$$\text{subject to} \quad \begin{cases} \Delta t - \epsilon &= 0 \\ G + \frac{\partial G}{\partial t}\Delta t + \nabla_w G^T \Delta w &= 0 \\ H + \frac{\partial H}{\partial t}\Delta t + \nabla_w H^T \Delta w &\geq 0, \end{cases}$$

where all functions and derivatives are evaluated at the point $t = 0$, $w^*(0)$, $\lambda^*(0)$ and $\mu^*(0)$.

The variable $\Delta t = \epsilon$ can directly be eliminated, and using the fact that $G(0, w^*(0)) = 0$ and $H^{\text{act}}(0, w^*(0)) = 0$ as well as the fact that

$$\nabla_w F(0, w^*(0)) = \nabla_w G(0, w^*(0))\lambda^*(0) + \nabla_w H(0, w^*(0))\mu^*(0)$$

we can formulate the equivalent QP (dropping the constant $\frac{\partial F}{\partial t}\epsilon + \frac{1}{2}\frac{\partial^2 \mathcal{L}}{\partial t^2}\epsilon^2$ in the objective)

$$\min_{\Delta w} \; \frac{1}{2}\Delta w^T \nabla_w^2 \mathcal{L} \Delta w + \epsilon \frac{\partial}{\partial t}\nabla_w \mathcal{L}^T \Delta w + \left(\nabla_w G \lambda^*(0) + \nabla_w H \mu^*(0)\right)^T \Delta w$$
$$\text{subject to} \quad \begin{cases} \frac{\partial G}{\partial t}\epsilon + \nabla_w G^T \Delta w &= 0 \\ \frac{\partial H^{\text{act}}}{\partial t}\epsilon + \nabla_w {H^{\text{act}}}^T \Delta w &\geq 0 \\ H^{\text{inact}} + \frac{\partial H^{\text{inact}}}{\partial t}\epsilon + \nabla_w {H^{\text{inact}}}^T \Delta w &\geq 0, \end{cases}$$

The conditions (3.3) according to Theorem 3.1 for a triple $(\Delta w, \lambda, \mu)$ to be a KKT point of this QP problem are, using $\lambda = \lambda^*(0) + \Delta\lambda$ and $\mu = \mu^*(0) + \Delta\mu$:

$$\begin{aligned}
\nabla_w^2 \mathcal{L} \Delta w + \epsilon \frac{\partial}{\partial t} \nabla_w \mathcal{L} - \nabla_w G \Delta\lambda - \nabla_w H \Delta\mu &= 0, && \text{(3.15a)}\\
\frac{\partial G}{\partial t}\epsilon + \nabla_w G^T \Delta w &= 0, && \text{(3.15b)}\\
\frac{\partial H^{\text{s.act}}}{\partial t}\epsilon + \nabla_w H^{\text{s.act}^T} \Delta w &\geq 0, && \text{(3.15c)}\\
\frac{\partial H^{\text{w.act}}}{\partial t}\epsilon + \nabla_w H^{\text{w.act}^T} \Delta w &\geq 0, && \text{(3.15d)}\\
H^{\text{inact}} + \frac{\partial H^{\text{inact}}}{\partial t}\epsilon + \nabla_w H^{\text{inact}^T} \Delta w &\geq 0, && \text{(3.15e)}\\
\mu^{*,\text{s.act}}(0) + \Delta\mu^{\text{s.act}} &\geq 0, && \text{(3.15f)}\\
\Delta\mu^{\text{w.act}} &\geq 0, && \text{(3.15g)}\\
\Delta\mu^{\text{inact}} &\geq 0, && \text{(3.15h)}\\
(\mu^*(0) + \Delta\mu)_j^{\text{s.act}} \left(\frac{\partial H^{\text{s.act}}}{\partial t}\epsilon + \nabla_w H^{\text{s.act}^T} \Delta w \right)_j &= 0, && \text{(3.15i)}\\
\Delta\mu_j^{\text{w.act}} \left(\frac{\partial H^{\text{w.act}}}{\partial t}\epsilon + \nabla_w H^{\text{w.act}^T} \Delta w \right)_j &= 0, && \text{(3.15j)}\\
\Delta\mu_j^{\text{inact}} \left(H^{\text{inact}} + \frac{\partial H^{\text{inact}}}{\partial t}\epsilon + \nabla_w H^{\text{inact}^T} \Delta w \right)_j &= 0. && \text{(3.15k)}
\end{aligned}$$

By assuming that Δw, $\Delta\mu$ can be made arbitrarily small by choosing ϵ small, we can assume that $H^{\text{inact}} + \frac{\partial H^{\text{inact}}}{\partial t}\epsilon + \nabla_w H^{\text{inact}^T} \Delta w > 0$ and therefore drop (3.15e), and replace (3.15k) by $\Delta\mu_j^{\text{inact}} = 0$. Additionally, we conclude that $(\mu^*(0) + \Delta\mu)_j^{\text{s.act}} > 0$, so that (3.15i) and (3.15c) can be replaced by

$$\frac{\partial H^{\text{s.act}}}{\partial t}\epsilon + \nabla_w H^{\text{s.act}^T} \Delta w = 0.$$

By a division by ϵ and a redefinition

$$\delta w_* := \frac{\Delta w}{\epsilon}, \quad \delta\lambda_* := \frac{\Delta\lambda}{\epsilon}, \quad \text{and} \quad \delta\mu_* := \frac{\Delta\mu}{\epsilon},$$

we can write the necessary conditions as

$$\begin{aligned}
\nabla_w^2 \mathcal{L} \delta w_* + \frac{\partial}{\partial t} \nabla_w \mathcal{L} - \nabla_w G \delta \lambda_* - \nabla_w H \delta \mu_* &= 0, \\
\frac{\partial G}{\partial t} + \nabla_w G^T \delta w_* &= 0, \\
\frac{\partial H^{\text{s.act}}}{\partial t} + \nabla_w H^{\text{s.act}^T} \delta w_* &= 0, \\
\frac{\partial H^{\text{w.act}}}{\partial t} + \nabla_w H^{\text{w.act}^T} \delta w_* &\geq 0, \\
\delta \mu_*^{\text{w.act}} &\geq 0, \\
\delta \mu_{*j}^{\text{w.act}} \left(\frac{\partial H^{\text{w.act}}}{\partial t} + \nabla_w H^{\text{w.act}^T} \delta w_* \right)_j &= 0, \\
\delta \mu_*^{\text{inact}} &= 0,
\end{aligned}$$

which are exactly the KKT conditions for the QP (3.7) that is formulated in Theorem 3.4. By the unique existence of this solution we confirm our assumption that Δw and $\Delta \mu$ can be made arbitrarily small by chosing ϵ sufficiently small.

□

3.4.1 Large Disturbances and Active Set Changes

In the proof of Theorem 3.6 we have made ϵ sufficiently small to ensure that the active set of the first QP corresponds to the active set in the immediate vicinity of the solution point $w^*(0)$ – in this way it was possible to show that the first iterate of the exact Hessian SQP method, when started at a solution $y^*(t)$ delivers a prediction $y^*(t) + \Delta y(\epsilon)$ of the solution $y^*(t+\epsilon)$ that is $\|y^*(t+\epsilon) - (y^*(t) + \Delta y(\epsilon))\| = O(\|\epsilon\|^2)$ under rather mild conditions, even at the points where the active set changes, as treated in Theorem 3.4.

In practical applications, however, when we want to solve a problem $P(t_2)$ starting with the solution of a problem $P(t_1)$, we will typically encounter the case that the non-differentiable point of the solution curve $w^*(t)$ lies somewhere in the interval between t_1 and t_2. It is very important to note that the SQP method is in practice also able to treat this case, as it can even reproduce *distant* active set changes, which will be illustrated by the following example.

Example 3.3 (First Order Prediction of Exact Hessian SQP)
Let us again consider the family of simple optimization problems of Examples 3.1 and 3.2, but in an augmented formulation $\breve{P}(\breve{t})$:

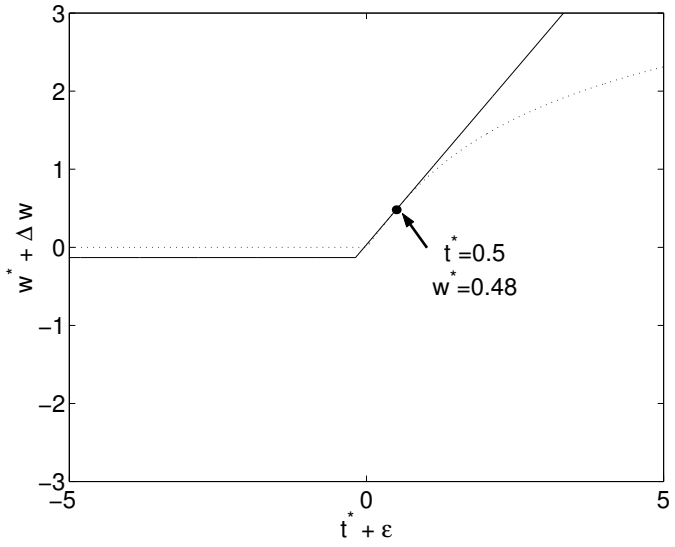


Figure 3.3: First step Δw of exact Hessian SQP method for Example 3.1, as a function of ϵ, for the initialization $t^* = 0.5, w^* = 0.48 = \arcsin(0.5)$.

$$\min_{t \in \mathbb{R}, w \in \mathbb{R}} \ \frac{1}{2} w^2 \quad \text{subject to} \quad \begin{cases} t - \breve{t} &= 0 \\ -t + \sinh(w) &\geq 0 \end{cases}$$

Initialized at a solution $t^* = \breve{t}$, $w^* = \max(0, \arcsin(t^*)$, $\mu^* = w^*/\cosh(w^*)$ *of* $\breve{P}(\breve{t})$, *the QP of the first exact Hessian SQP iteration for the solution of* $\breve{P}(\breve{t} + \epsilon)$ *is*

$$\min_{\Delta t \in \mathbb{R}, \Delta w \in \mathbb{R}} \ \frac{1}{2} \ \Delta w^T \left(1 - \tanh(w^*) w^* \ \Delta w \ + \ w^* \ \Delta w\right)$$

$$\text{subject to} \quad \begin{cases} \Delta t \ - \ \epsilon &= 0 \\ -t^* + \sinh(w^*) - \Delta t \ + \ \cosh(w^*) \, \Delta w &\geq 0, \end{cases}$$

which has the solution

$$\begin{aligned} \Delta t &= \epsilon, \\ \Delta w &= \max(-\tfrac{w^*}{1 - \tanh(w^*) w^*}, \tfrac{t^* - \sinh(w^*) + \epsilon}{\cosh(w^*)}), \\ \tilde{\mu} &= \tfrac{w^* + (1 - \tanh(w^*) w^*) \Delta w}{\cosh(w^*)} \end{aligned}$$

as depicted in Figure 3.3 for an initialization t^*, w^* *that is in the neighborhood of the "corner"* $t = 0, w = 0$.

Chapter 4

Real-Time Iterations

In this chapter we will develop the main algorithmic ideas of our real-time iteration approach. We will first present in Sec. 4.1 the challenges that every real-time optimal control scheme has to face, and motivate the idea of the *real-time iteration* scheme. In Sec. 4.2 we present the initial value embedding approach for perturbed problems, that arises quite naturally in the framework of the direct multiple shooting method. The algorithm is described in Sec. 4.3 for shrinking horizon problems, and in Sec. 4.4 for moving horizon problems, that are typical for practical NMPC applications.

After the presentation of the real-time iteration idea in this chapter, we will in Chap. 5 prove that the proposed approach leads to a contractive algorithm under suitable conditions, and in Chap. 6 we will have a close look at one real-time iteration.

4.1 Practical Real-Time Optimal Control

In a real-time scenario we aim at not only solving one optimization problem, but a whole sequence of problems. Let us denote the differential state of the plant at time t by $x_0(t)$. Then, ideally, at *every time* t, the optimal control problem of Sec. 1.1.1 with an initial value $x_0(t)$ would be solved instantaneously, and the optimal control $u^*(0; x_0(t))$ be given as a control to the real plant at time t. This strategy would yield an optimal feedback control, or, for moving horizons, a Receding Horizon Control (RHC) law as e.g. defined in [MM90]. In all real implementations of NMPC, however, two approximations to this ideal approach are made:

- First, it is not the infinite optimal control problem from Sec. 1.1.1 that is solved, but a parameterized, finite dimensional formulation of it. In our approach it is the NLP from Sec. 2.2 that arises after the direct multiple shooting parameterization, which was denoted $P(x_0(t))$.

- Secondly, the optimization problems cannot be solved instantaneously, so that the problems are solved only at discrete *sampling times* $\ldots, t_i, t_{i+1}, \ldots$, with interval

durations $\delta_i = t_{i+1} - t_i$ that are long enough to perform the necessary computations for the solution of problem $P(x_0(t_i))$.

Note that in this framework the optimal control corresponding to the system state $x_0(t_i)$ at time t_i is usually only available at time t_{i+1}, after the computations have been performed. This leads to a delay that may result in poor real-time performance, if the sampling intervals δ_i are not short enough.

In principle, it is possible to predict the state $x_0(t_{i+1})$ already at time t_i and to solve the corresponding problem $P(x_0(t_{i+1}))$ during the time interval $[t_i, t_{i+1}]$, so that at time t_{i+1} the optimal solution for the problem $P(x_0(t_{i+1}))$ is already available. However, unpredicted disturbances that have occured in the interval $[t_i, t_{i+1}]$ are not taken into account, so that the feedback delay of one sampling time is still present.

4.1.1 A Conventional Approach

A straightforward approach to real-time optimal control would be to just employ a fast off-line algorithm to solve the arising optimization problems, and use the completely converged solution of the optimization problem to provide the feedback. We call this approach the *conventional approach to NMPC*, and it is for example described by Binder et al. in [BBB+01]. Note, however, that the duration δ_i may not be known in advance, if it is insisted that each solution should satisfy a prespecified convergence criterion: in fact, the number of SQP iterations cannot be bounded at all! In all practical implementations some safeguards must exist, that stop the solution algorithm in time, e.g. after a fixed number of SQP iterations, even if the convergence criterion is not met.

Example 4.1 (Conventional NMPC)
Let us consider again the scenario that was presented in Sec. 1.2 and introduce it as a real-time example. We assume that the system state is disturbed at time $t_0 = 0$, so that it suddenly jumps to the disturbed initial value x_0, that is known immediately, but could not be known in advance. We choose a multiple shooting parameterization with $N = 100$ intervals each of 20 seconds length. Let us assume that one SQP iteration takes 20 seconds computation time, and that after 5 iterations all occuring optimization problems are solved with satisfying accuracy: therefore, we can choose a sampling time of $\delta = 100$ seconds. During this time we have to apply the best available controls to the real plant, which are the steady state controls in the first 100 seconds, and in the following sampling times the outcome of the previous optimization. The optimizations are carried out for the predicted initial values after *100 seconds, to alleviate the effect of the delay. We will assume that the model and the real plant coincide, so that the open-loop solution that is available after the first 100 seconds corresponds already to the closed-loop trajectory and is not modified in the following sampling intervals. The resulting closed-loop trajectory is shown in Fig. 4.1, and compared to the optimal feedback control. The integrated least squares objective that we can regard as a performance measure of the closed-loop trajectories, is for the conventional NMPC scheme increased by 17 % compared to the optimal feedback control.*

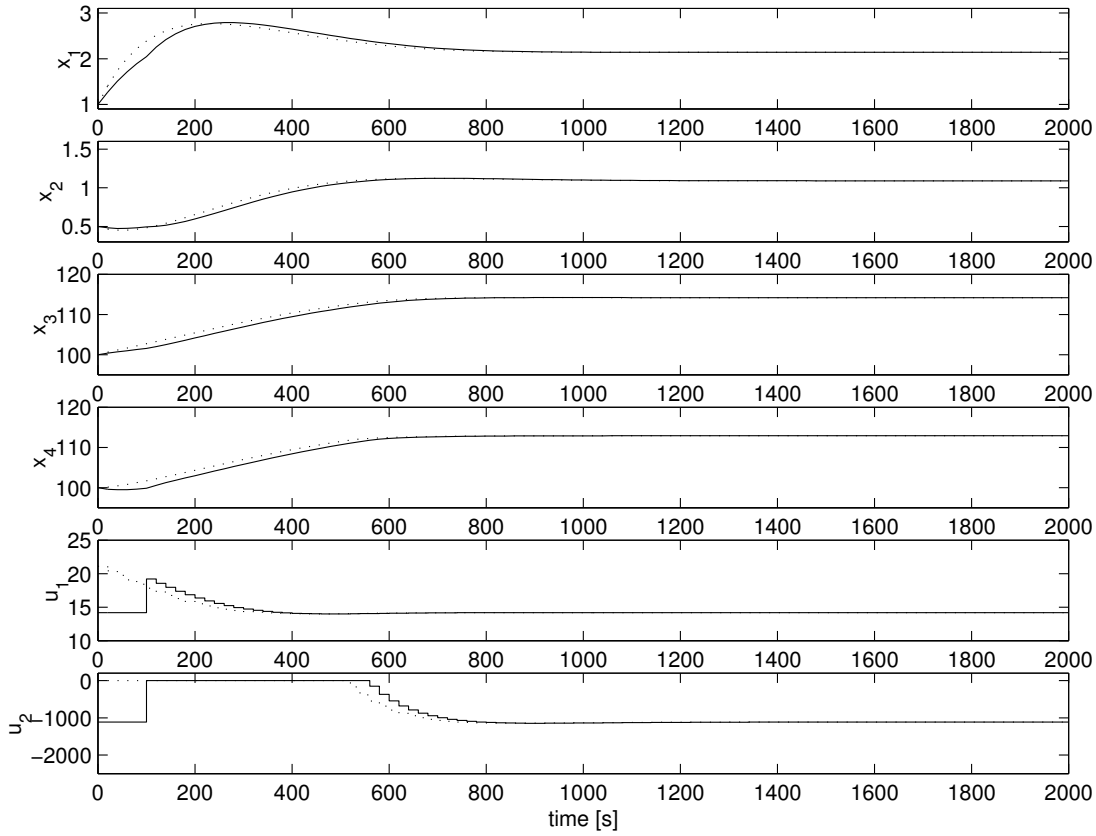


Figure 4.1: State and control trajectories of a conventional NMPC scheme, compared with optimal feedback control (which is dotted).

4.1.2 The Real-Time Iteration Idea

The reason for the poor performance of the conventional NMPC scheme is that we have to wait a long time until a feedback to disturbances is delivered, and in the meantime we have to apply a rather arbitrary, uncorrected control. Would it not be possible to use some other feedback control that is not necessarily optimal, but better than the uncorrected values? In Example 3.3 at the end of Chap. 3 we have seen that the first QP solution of a full step exact Hessian SQP algorithm provides already a rather good approximation of the exact solution, if the algorithm is initialized in a neighborhood of this solution. Motivated by this observation, we conclude that – in a real-time scenario – it would probably be better to use the result of this first correction instead of waiting until the SQP algorithm has converged (without reacting to disturbances). After the first SQP iteration, there would already be the chance to react to new disturbances – and if no further disturbance occurs, the algorithm could continue to improve the outcome of the previous iterates. Compared

with the conventional approach, our real-time algorithm differs therefore in two important respects:

- We restrict the number of solution iterations that are performed for each problem to *one single SQP iteration*, allowing to reduce the sampling intervals δ_i to a minimum. This approach is only possible if we ensure that the subsequent optimization problems are carefully initialized in order to maintain the excellent convergence properties of the direct multiple shooting method in the absence of disturbances.

- Secondly, we divide the necessary computations during each real-time iteration into a (long) *preparation phase* that can be performed without knowledge of x_0, and a considerably shorter *feedback phase* that allows to make the delay even shorter than the sampling time δ_i. As this remaining delay is typically orders of magnitude smaller than δ_i, we will in the following neglect it and assume that the result of each real-time iteration is *immediately* available, and that the sampling time δ_i is only needed to prepare the *following* real-time iteration.

Both algorithmic features are based on an initialization strategy that can be understood as an *initial value embedding*, which will be described in the following section.

4.2 The Initial Value Embedding

In Theorem 3.6 of Sec. 3.4 we have shown that the first iterate of a full step exact Hessian SQP algorithm that is initialized at a neighboring solution delivers a first order approximation of the exact solution, if an augmented problem formulation (3.14) is used. The crucial feature of this augmented formulation is that the actual value of the parameter that distinguishes between different problems is introduced as an additional NLP variable, that is fixed by a trivial equality constraint, so that derivatives with respect to the parameter are present in the SQP framework. Fortunately, in the direct multiple shooting NLP formulation of Sec. 2.2, the distinguishing parameter of the NLPs $P(x_0)$ is the initial value x_0, that is itself constraining s_0^x by a trivial equality constraint $s_0^x - x_0 = 0$. Therefore, we may regard the NLP formulation (2.10) as an embedded problem formulation of the form

$$\min_{s_0^x \in \mathbb{R}^{n_x}, \tilde{w} \in \mathbb{R}^{(n_w - n_x)}} F(s_0^x, \tilde{w}) \quad \text{subject to} \quad \left\{ \begin{array}{rcl} s_0^x - x_0 & = & 0, \\ \tilde{G}(s_0^x, \tilde{w}) & = & 0, \\ H(s_0^x, \tilde{w}) & \geq & 0, \end{array} \right.$$

with $w = (s_0^x, \tilde{w})$ (cf. Sec. 2.2). Comparing with the notation of (3.14), s_0^x has taken the place of t, and x_0 the place of $\breve{t}$.

Let us assume that we have found a solution $y^*(x_0) = (w^*(x_0), \lambda^*(x_0), \mu^*(x_0))$ of problem $P(x_0)$. If the SQP algorithm for the solution of a neighboring problem $P(x_0 + \epsilon)$ is initialized with this solution, the first full step exact Hessian SQP iterate provides already an excellent (first order) approximation of the solution $y^*(x_0 + \epsilon)$. This first iterate is even able to approximate distant active set changes, as was shown in Example 3.3, and will also be illustrated in the following example for a parameterized optimal control problem.

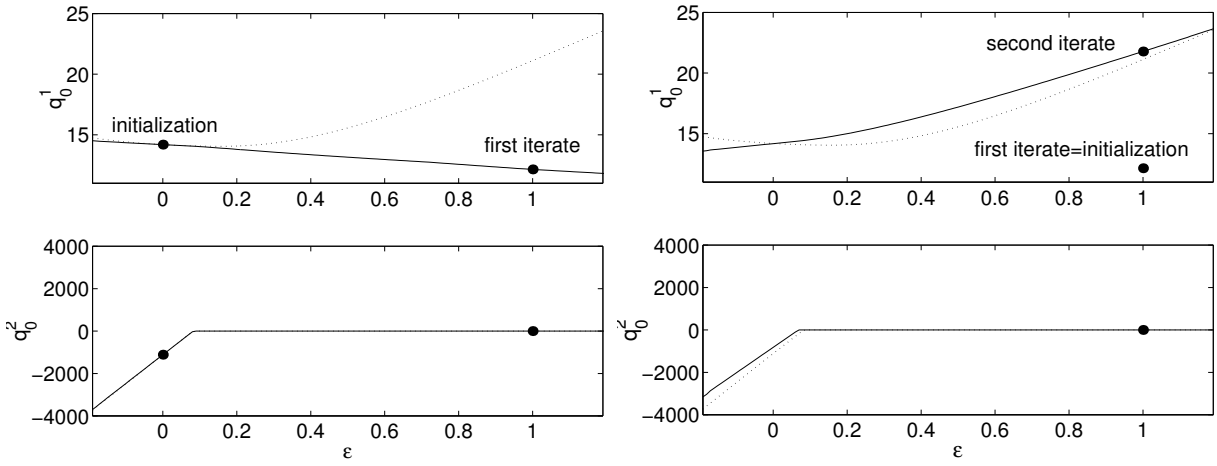


Figure 4.2: First order correction in q_0^1 and q_0^2 after initial value embedding as a function of ϵ, for two different initializations: on the left hand side the algorithm was initialized with the steady state trajectory, on the right hand side with the outcome of the first iterate for $\epsilon = 1$ from the left hand side. The exact solution manifold (cf. Fig. 1.3) is dotted.

Example 4.2 (Initial Value Embedding)
Let us again consider the continuous stirred tank reactor with a disturbed initial value x_0 as in Example 4.1. We will regard a whole family of optimization problems $P(x_\epsilon)$ with perturbed initial values

$$x_\epsilon := x_S + \epsilon(x_0 - x_S),$$

that interpolate between the steady state x_S and the disturbed initial value x_0 (cf. Example 1.1). Let us initialize the SQP algorithm with the steady state trajectory, which is the solution of $P(x_S)$ ($\epsilon = 0$). The number n_w of NLP variables is 604, as discussed in Example 2.1. We restrict our attention only to the first control vector, q_0. A comparison of the first correction in q_0 after the initial value embedding with the exact solution manifold (cf. Example 1.1) is shown on the left hand side of Fig. 4.2 for different values of ϵ; this illustrates that the initial value embedding strategy provides a first order approximation of the exact solution manifold, that takes active set changes into account (cf. lower left graph of Fig. 4.2). On the right hand side of Fig. 4.2 we investigate what happens if we choose the first iterate itself as an initialization of the algorithm, so that we obtain a second iterate. Note that the first iterate itself is not *the solution of any problem, so that the manifold of second iterates does not touch the exact solution manifold at a reference point, as before. But it can be seen that in the vicinity of $\epsilon = 1$ it provides already a quite good approximation of the exact solution manifold.*

4.3 Real-Time Iterations on Shrinking Horizons

So far we have assumed that the initialization of the SQP algorithm is given. In a real-time scenario, there is essentially only one source which can provide an initial guess for the current problem: the outcome of the previous iterate. Depending on the problem class, different strategies come to mind to use the previous real-time iteration to initialize the current one.

In the case of shrinking horizon problems, there exists a very natural initialization that is based on the principle of optimality: if a solution $x^*(t)$, $z^*(t)$, $u^*(t)$ is optimal on the time horizon $t \in [t_i, t_f]$ for an initial value $x_0(t_i)$, its restriction to the shrunk horizon $[t_{i+1}, t_f] = [t_i + \delta_i, t_f]$ is still optimal for the initial value $x_0(t_{i+1}) = x^*(t_{i+1})$. This can be translated into the direct multiple shooting context, if the length of the multiple shooting intervals, $T(\tau_{i+1} - \tau_i)$, corresponds to the length of the sampling intervals, $\delta_i = t_{i+1} - t_i$. Let us for this scope regard a problem discretization with N multiple shooting intervals, and let us assume that for the optimization problem $P(x_0(t_0))$ on the full horizon $[t_0, t_0+T]$ a solution $w^* = (q_0, \ldots, q_{N-1}, s_0, \ldots, s_N)$ has been found. At time $t_k = t_0 + \sum_{i=1}^{k} \delta_i = t_0 + T\tau_k$, a reduced problem can be formulated, on a shrunk horizon with only $N-k$ multiple shooting intervals, for the initial value $x_k := x_0(t_k)$. We will denote this problem by $P_k(x_k)$. Let us adopt the convention that the multiple shooting variables w_k of the reduced NLP are numbered so that the indices start with k, i.e., $w_k = (q_k, \ldots, q_{N-1}, s_k, \ldots, s_N)$, so that the problem $P_k(x_k)$, $k = 0, \ldots, N-1$ can be written as:

$$P_k(x_k): \quad \min_{\substack{q_k, \ldots, q_{N-1}, \\ s_k, \ldots, s_N}} \sum_{i=k}^{N-1} L_i(s_i^x, s_i^z, q_i) + E(s_N^x, s_N^z) \tag{4.1a}$$

subject to

$$s_{i+1}^x - x_i(\tau_{i+1}; s_i^x, s_i^z, q_i) = 0, \quad i = k, \ldots, N-1, \tag{4.1b}$$
$$g(s_i^x, s_i^z, q_i) = 0, \quad i = k, \ldots, N, \tag{4.1c}$$
$$s_k^x - x_k = 0, \tag{4.1d}$$
$$r^{\mathrm{e}}(s_N^x, s_N^z) = 0, \tag{4.1e}$$
$$r^{\mathrm{i}}(s_N^x, s_N^z) \geq 0, \tag{4.1f}$$
$$h(s_i^x, s_i^z, q_i) \geq 0, \quad i = k, \ldots, N. \tag{4.1g}$$

Note that $P_0(x_0)$ corresponds to the original problem $P(x_0)$ formulated in (2.10).

Clearly, if we have found a solution $y_k^* = (w_k^*, \lambda_k^*, \mu_k^*)$ of problem $P_k(x_k)$, and if the state x_{k+1} corresponds to the predicted optimal value on this trajectory, i.e., $x_{k+1} = (s_{k+1}^x)_k^*$, the restriction of the solution to the remaining horizon provides the solution y_{k+1}^* for the shrunk problem $P_{k+1}(x_{k+1})$, which is a good initialization also for disturbed initial values $x_{k+1} + \epsilon$,

when the initial value embedding is employed. Let us introduce the "shrink" operator S_k, that just removes the first components q_k, s_k and the corresponding multipliers from a vector $y_k = (w_k, \lambda_k, \mu_k)$, i.e., the operator that projects the variables and multipliers of $P_k(\cdot)$ to the variable and multiplier space of $P_{k+1}(\cdot)$. Using S_k, the above statement can be expressed as

$$y^*_{k+1} = S_k y^*_k.$$

4.3.1 A Real-Time Algorithm

In the real-time iteration context, the algorithm would proceed as follows: Starting with an initial guess $y^0_0 = (w^0_0, \lambda^0_0, \mu^0_0)$ for the problems $P_0(\cdot)$ prepare the first real-time iteration as far as possible without knowledge of x_0. Then perform for for $k = 0, \ldots, N-1$ the following cycle:

1. At the moment t_k that x_k is known, perform the prepared real-time iteration (based on a linearization at y^k_k and the initial value embedding idea) towards the solution of $P_k(x_k)$. This yields the first order correction y^{k+1}_k.
2. Give the resulting value of the first control vector $(q_k)^{k+1}_k$ (which is contained in the vector y^{k+1}_k) immediately as a control to the plant.
3. Shrink the first order correction y^{k+1}_k to the variable space of the new problem $P_{k+1}(\cdot)$, i.e., define the new iterate
$$y^{k+1}_{k+1} := S_k y^{k+1}_k.$$
4. Prepare the first iterate of problem $P_{k+1}(\cdot)$ as far as possible without knowledge of x_{k+1}, using the shrunk vector y^{k+1}_{k+1} as an initialization.
5. Increase k by one and go to 1.

Note that in our algorithm the first two steps do only need a very short computation time compared to the fourth step (cf. Chap. 6).

Example 4.3 (Real-time iterations)
The closed-loop trajectory resulting from the real-time iteration approach for the scenario that was presented in Sec. 1.2 is shown in Fig. 4.3. The number of multiple shooting intervals is $N = 100$, with intervals of equal length $\delta_k = 20$ sec for $k = 0, \ldots, N-1$. The initialization y^0_0 was chosen to be the steady state trajectory, i.e., the solution of $P(x_S)$. We assume that the preparation time per real-time iteration is exactly 20 seconds (in reality, the computation time per iteration was always less than 1 second, cf. Fig. 4.5).

It can be seen that the real-time iteration approach delivers a trajectory that is nearly identical with an optimal feedback control (dotted line) – apart from the "outlier" of u_1 in the first interval which is due to linearization errors, and which corresponds to the first iterate in Fig. 4.2. In the second real-time iteration, when nonlinearities are taken into account, u_1 is already very close to its optimal value. The performance index is only increased by 3 % compared to optimal feedback control.

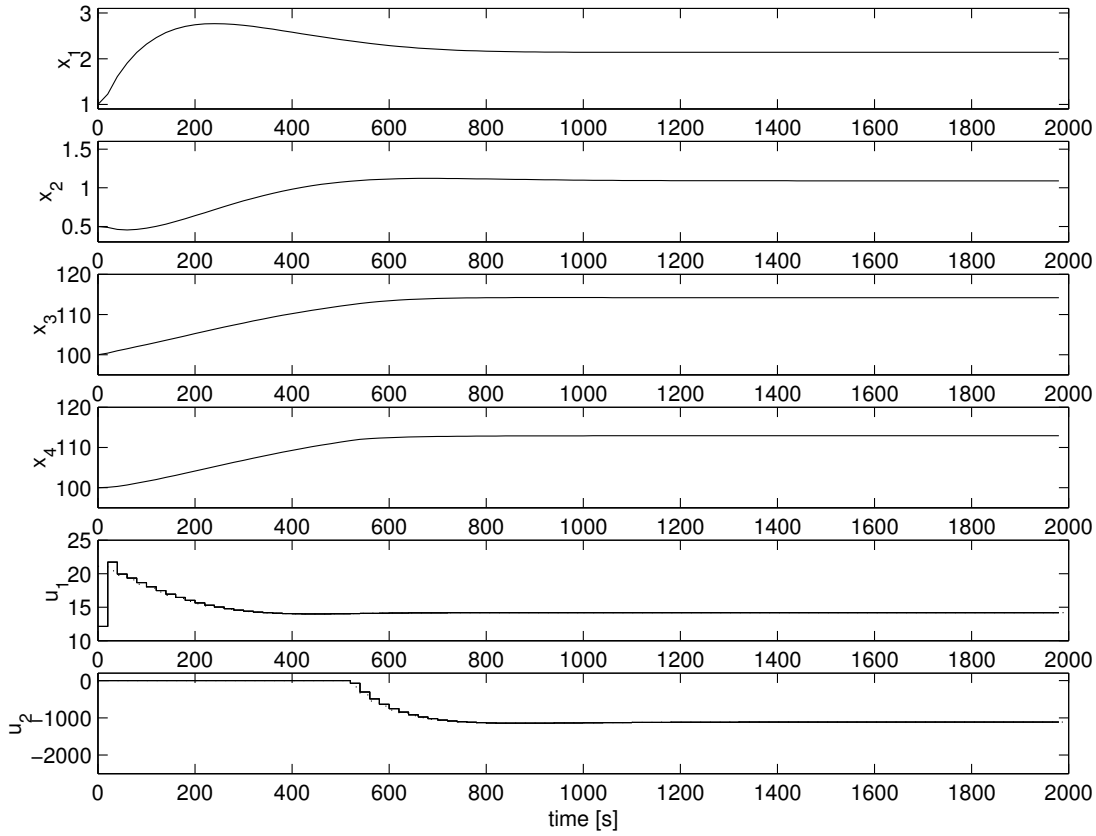


Figure 4.3: State and control trajectories of the real-time iteration approach. The dotted lines, that are nearly identical, show the trajectory due to optimal feedback control.

4.3.2 Comparison with Linearized Neighboring Feedback Control

It is interesting to compare the above real-time iteration scheme with a well-known approximation of optimal feedback control, namely with linearized neighboring feedback control as introduced in Sec. 1.3.1. For this aim let us assume that the initial guess y_0^0 is the solution of a *nominal* problem $P_0(\bar{x}_0)$, and that the predicted optimal trajectory goes through the points $\bar{x}_1, \ldots, \bar{x}_N$, that we call the *nominal* or *reference* trajectory. By the principle of optimality it is clear that for a given $k < N$ the restriction of y_0^0 to the variable space of $P_k(\cdot)$, that we denote by y_k^0, is the solution of the problem $P_k(\bar{x}_k)$. The idea of linearized neighboring feedback control is to use *only* the initial guess y_0^0 and its subvectors y_k^0 for the initialization of the real-time iterations. The preparation of the first iterate for *all* problems $P_k(\cdot)$ can be performed off-line, reducing the necessary on-line computations to a minimum.

In linearized neighboring feedback control, usually the assumption is made that the active set does not change during the on-line QP solutions, so that the QP can largely be

presolved, leaving only one matrix vector multiplication that has to be performed on-line: the control u_k on the interval $[t_k, t_{k+1}]$ is given by $u_k = \bar{u}_k - K_k(x_k - \bar{x}_k)$, where $\bar{u}_k$ is the nominal control, and the matrix K_k is the precomputed *gain matrix*. Virtually no on-line computations have to be performed in this case, and very short sampling times can be realized.

For larger deviations in $x_k - \bar{x}_k$, however, this may lead to control responses that exceed the control bounds – therefore we present here a modified linearized feedback control scheme that solves the prepared QPs on-line, so that all linearized constraints can be taken into account, when active set changes occur. Note that bounds are linear constraints and therefore exactly satisfied in each QP solution. The difference to the real-time iteration scheme is that all control responses are based on the same system linearization, at the reference solution y_0^0. This algorithm would proceed as follows:

Based on the reference solution $y_0^0 = (w_0^0, \lambda_0^0, \mu_0^0)$ and on its subvectors $y_k^0 = S_k S_{k-1} \dots S_1 y_0^0$ (that are the solutions of the nominal problems $P_k(\bar{x}_k)$), prepare the first QP solution of the problems $P_k(\cdot)$ as far as possible without knowledge of x_k. Then perform for for $k = 0, \dots, N-1$ the following cycle:

1. At the moment t_k that x_k is known, perform the prepared QP solution towards the solution of $P_k(x_k)$. This yields the first order correction y_k^1.

2. Give the resulting value of the first control vector $(q_k)_k^1$ (which is contained in the vector y_k^1) immediately as a control to the plant.

3. Increase k by one and go to 1.

Note that this linearized neighboring feedback control scheme is very closely related to *linear* model predictive control on shrinking horizons, as it is based on a linear system model, and only a QP has to be solved in each iteration. It is superior to what is commonly called linear model predictive control, however, in the respect that nonlinearities of the system equations along the nominal trajectory are taken into account, and that the Hessian matrix does not only represent a quadratic objective, but the full second order information of the Lagrangian function along the nominal trajectory. Note that the values y_k^1 are first order approximations of the optimal solutions y_k^* of the full nonlinear problems $P_k(x_k)$, i.e., $\|y_k^1 - y_k^*\| = O(\|x_k - \bar{x}_k\|^2)$, due to the initial value embedding.

The low computational on-line cost of linearized neighboring feedback control, however, comes along with the inability to adapt to large deviations from the nominal solution (i.e., for big $\|x_k - \bar{x}_k\|$), as the system nonlinearity is not taken into account in the on-line context.

Example 4.4 (Linearized neighboring feedback control)
In Fig. 4.4 a closed-loop trajectory corresponding to the described linearized neighboring feedback control for the same scenario as in Example 4.3 is shown, and compared with the real-time iteration scheme. It can be seen that the two trajectories differ significantly, and that the linearized neighboring feedback control shows considerably poorer performance, see

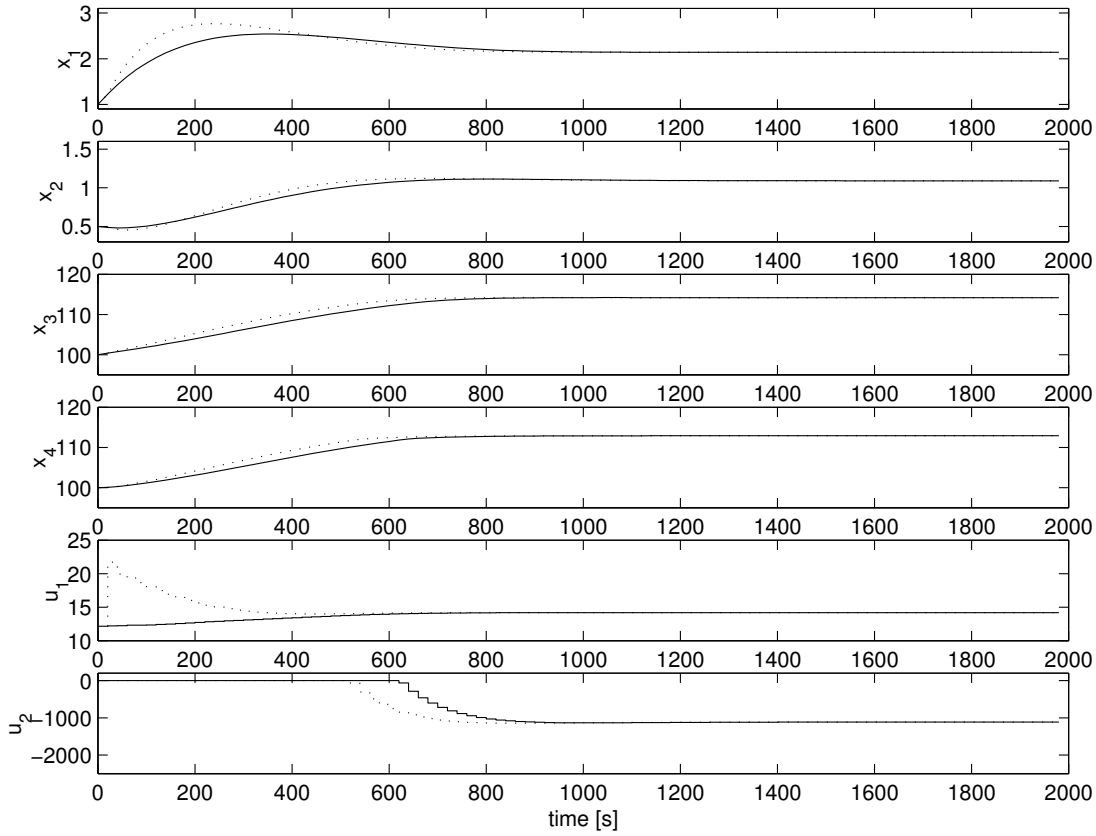


Figure 4.4: Linearized neighboring feedback control: state and control trajectories, compared with the real-time iteration trajectories (dotted). Both schemes coincide on the first interval, but the linearized neighboring scheme does not take nonlinearities into account.

Table 4.1. Note that the two schemes coincide on the first interval, where they both use the same initialization (cf. Fig. 4.2). In the linearized neighboring feedback control, this initialization is kept for the whole trajectory, whereas it is continuously updated during the real-time iteration cycles.

4.3.3 Problems with Free Final Time

So far we have focused on problems where the overall duration was prespecified. In practical shrinking horizon problems, however, often the final time is open to optimization, or is determined implicitly by terminal constraints. As discussed in Sec. 2.2, the formulation of a free final time can be achieved by an augmentation of the differential state vectors s_i^x by

Strategy	relative costs
Optimal Feedback Control	100 %
Real-Time Iterations	103 %
Conventional NMPC Approach	117 %
Linearized Neighboring Feedback	121 %

Table 4.1: Performance of different real-time strategies, measured in terms of the objective function, from Examples 4.1, 4.3, and 4.4.

one component, and the formulation (4.1) of the optimization problem $P_k(x_k)$ needs to be modified only at the initial value constraint (4.1d), which is changed to

$$(\mathbb{I}_{n_x}|0)s_k^x - x_k = 0.$$

As before, the principle of optimality holds, so that a solution of problem $P_k(x_k)$ provides a solution of the shrunk problem $P_{k+1}(x_{k+1})$, if x_{k+1} corresponds to the predicted value $(\mathbb{I}_{n_x}|0)s_{k+1}^x$. If disturbances occur, the initial value embedding and real-time iteration scheme can be applied without modifications; however, it should be kept in mind that the interval durations may now change during the real-time iterations.

Apart from the shrinking problem formulation as described above, there exists an interesting second possibility to formulate the series of optimization problem on a shrinking horizon with free final time: instead of the problems $P_0(x_0), P_1(x_1), \ldots, P_{N-1}(x_{N-1})$ with shrinking multiple shooting interval numbers, we can consider always the *same* multiple shooting discretization and regard only one type of parameterized problem, $P(\cdot) = P_0(\cdot)$, i.e., we treat successively $P(x_0), P(x_1), \ldots, P(x_{N-1})$. In this case it is not straightforward how to initialize the subsequent real-time iterations. One way would be to take the variable and multipliers from the previous iterate without any modification, i.e., to perform successive *warm starts* and to rely on the approximation capacities of the initial value embedding (cf. Sec. 4.4.2). Though such a scheme can be successful in practical applications, especially for short sampling times, it is difficult to prove convergence, as it will be done for the shrinking problem formulation in Chap. 5.

4.4 Real-Time Iterations on Moving Horizons

In applications of nonlinear model predictive control (NMPC) to continuous processes the optimization problems are typically formulated on moving horizons, which aim to approximate an infinite prediction horizon. This results in problems which all have the same horizon length, and which are only distinguished by the initial value x_k. We will therefore only treat one type of optimization problem $P(\cdot)$, and adopt the convention that the subvectors of the primal variables w are denoted by $q_0, \ldots, q_{N-1}$ and $s_0, \ldots, s_N$ in all problems, i.e., we disregard the absolute position of the moving horizon in time, in contrast to the shrinking horizon case.

We will present two basic strategies how to proceed from one optimization problem to the next. Both show their advantages in different circumstances: the first, the shift strategy, is especially advantageous for periodic or time dependent processes, as it considers the movement of the horizon in time explicitly. The second strategy, the warm start, is especially useful in applications where the multiple shooting intervals are chosen to be considerably longer than the sampling times.

4.4.1 Shift Strategy

The principle of optimality does not hold for finite moving horizons, but it is approximately valid if the horizon length is long enough to justify the assumption that the remaining costs on the infinite horizon can be neglected. This motivates an adaptation of the shrinking horizon initialization strategy to moving horizons that we call the *shift strategy*.

For the initialization of a problem $P(x_{k+1})$ it uses the iterate $y_k^{k+1} = (w_k^{k+1}, \lambda_k^{k+1}, \mu_k^{k+1})$, that is the outcome of the previous iteration towards the solution of problem $P(x_k)$, to initialize the new problem with y_{k+1}^{k+1} as follows.

Shift in the Primal Variables

If the primal variables w_k^{k+1} are denoted by

$$w_k^{k+1} = (q_0, q_1, \ldots, q_{N-2}, q_{N-1}; s_0, s_1, \ldots, s_{N-2}, s_{N-1}, s_N)$$

then the shift initialization sets

$$w_k^{k+1} := (q_1, q_2, \ldots, q_{N-1}, q_{N-1}^{\text{new}}; s_1, s_2, \ldots, s_{N-1}, s_{N-1}^{\text{new}}, s_N^{\text{new}}),$$

where the new values q_{N-1}^{new}, s_{N-1}^{new}, and s_N^{new} can be specified in different ways:

- One straightforward way is to keep the old values at their place, i.e., to initialize

  $$q_{N-1}^{\text{new}} := q_{N-1}, \quad s_{N-1}^{\text{new}} := s_{N-1}, \quad \text{and} \quad s_N^{\text{new}} := s_N,$$

  which has the advantage that the only infeasibility that is introduced into a feasible trajectory is the violation of the continuity condition at the start of the last interval, i.e., at τ_{N-1}.

- A second possibility would be to solve the DAE on the new last interval, starting with s_N, and employing the control q_{N-1}, which yields the final value $s_N^{\text{new}}(s_N, q_{N-1})$ for differential and algebraic states, i.e., we initialize:

  $$q_{N-1}^{\text{new}} := q_{N-1}, \quad s_{N-1}^{\text{new}} := s_N, \quad \text{and} \quad s_N^{\text{new}} := s_N^{\text{new}}(s_{N-1}^{\text{new}}, q_{N-1}^{\text{new}}).$$

  In this case, both the continuity condition and the algebraic consistency condition at τ_{N-1} are fullfilled, if the previous solution was feasible: for the continuity condition this is trivially true, and for the consistency condition note that previously, at τ_N, $0 = g(s_N, q_N) = g(s_N, q_{N-1})$, as q_{N-1} provides per definition the control q_N at the final multiple shooting node. However, path and terminal constraints may be violated by the new final value s_N^{new}.

- Yet another possibility is to solve the DAE on the last interval starting at s_N, but to employ a different control than q_{N-1}. Though this may sacrifice algebraic consistency at τ_{N-1}, this strategy may be advantageous, e.g. for periodic processes, where a time dependent nominal control may be taken. This strategy was employed for the periodic control example of Chap. 8.

Note that the initial violation of constraints is naturally treated in the direct multiple shooting framework and does not create any additional difficulty in the following SQP iterations. During the real-time iterations the outcome of the previous iterate will usually not be feasible anyway, and additional infeasibility is introduced by initial values x_k that are not in accordance with the predictions.

Shift in the Multipliers

The initialization of the multipliers is also performed by a shift; for the multiplier values on the final interval we usually keep the old values. It is clear that a shifted solution, even if it may be feasible, is in general not optimal. For sufficiently long horizons, however, we expect the principle of optimality to hold, so that the shifted primal variables and the shifted multipliers y_{k+1}^{k+1} are close to a solution of $P(x_{k+1})$ if y_k^{k+1} was a solution of $P(x_k)$ and the system has developed as predicted (i.e., $x_{k+1} = s_1^x$).

4.4.2 Warm Start Technique

On the other hand, if the horizon length is relatively short, so that the principle of optimality does not hold at all, subsequent optimization problems may have very similar solutions, that are mainly determined by terminal conditions, such as e.g. a Mayer term and terminal constraints that are introduced to bound the neglected future costs (cf. Sec. 1.4). In this case the best initialization of subsequent problems should be a *warm start strategy*, which takes the result of the previous iteration, y_k^{k+1}, without further changes to initialize the current iteration: $y_{k+1}^{k+1} := y_k^{k+1}$.

If y_k^{k+1} was the solution to problem $P(x_k)$, then the only infeasibility in problem $P(x_{k+1})$ is introduced by the initial value constraint, as in general $x_{k+1} \neq x_k$. In this case, however, the next iterate y_{k+1}^{k+2} is identical to the first order correction to the optimal solution, as we proved in Theorem 3.6, i.e., its distance to the optimal solution is $\|y_{k+1}^{k+2} - y_{k+1}^*\| = O(\|x_{k+1} - x_k\|^2)$, if an exact Hessian SQP method is used. To shed more light on this desirable property, we regard the (unsolved) problem $P(x_{k+1})$ as a member in a family of perturbed problems $P(x_k + \epsilon(x_{k+1} - x_k))$, where a solution for $\epsilon = 0$ exists, and the solution for $\epsilon = 1$ is desired (cf. Example 4.2). The warm start strategy therefore has a very natural connection to the initial value embedding strategy.

Interpretation as Modified SQP Iterations

Another interesting property of the warm start technique occurs if the initial values x_k, x_{k+1} coincide during some iterates $k, k+1, \ldots$. In this case, all real-time iterations treat the

same problem, so that the standard convergence properties of SQP methods can be expected. This observation motivates a new look on the real-time iteration idea: rather than interpreting the real-time iterations as many prematurely stopped solution attempts of subsequent optimization problems, we regard them as a continuous series of SQP iterates with the particularity that one parameter, the initial value x_0, is slightly modified during the iterations. This interpretation captures very well the philosophy of the real-time iterations; in every practical implementation of a real-time iteration algorithm it has meticulously to be taken care that the initialization from one problem to the next does preserve *all* informations that are necessary to guarantee the convergence properties of an off-line method. This is most easily realized for the warm start strategy.

Short Sampling Times

In some practical NMPC applications it may be desirable to choose the multiple shooting intervals longer than the sampling time; this allows e.g. long prediction horizons with a limited number of multiple shooting nodes, which may be a crucial real-time advantage, as the computation time generally grows with the number of multiple shooting nodes. Another practical reason for choosing relatively long control intervals may be to detune the NMPC controller, whose aggressive response may otherwise excite unmodelled system modes with short timescales.

In the warm start technique, short sampling times can be treated without difficulty. Even sampling times of variable size are allowed – the only requirement for good performance is that the problems (i.e., the initial value x_0) do not change too much from one iteration to the next. Therefore, the shorter the sampling time, the better the contraction behaviour.

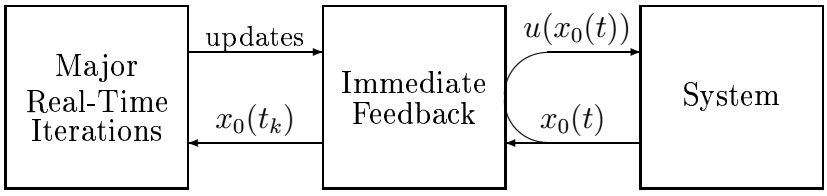
Self-Synchronization of the Real-Time Iterations

Where a continuous stream of state estimates is available, the warm start technique even offers the possibility to let the sampling times be determined on-line by the optimizer itself. Whenever a new real-time iterate is prepared, say at time t_k, the current state estimate $x_0(t_k)$ is used to perform the next real-time iteration towards the solution of $P(x_0(t_k))$, whose response is given immediately to the real-plant, and then the next iterate is prepared, until the algorithm is ready to perform, at time t_{k+1}, the next feedback phase. Note that in this scheme it is not necessary to know the computation time $t_{k+1} - t_k$ in advance. In our practical implementation of NMPC in Chap. 7 we have employed this scheme.

Successive Generation of Feedback Laws

Yet another possibility, that can be employed if a full real-time iteration takes too long to be able to respond to relevant disturbances, is to separate the preparation phase and the feedback phase of each real-time iteration completely. Then, the self-synchronized major SQP iterations are performed as one process, that gives all data that are necessary for the immediate feedback to another process. This feedback process delivers a feedback $u(x_0(t))$

with a frequency that can be considerably higher than that of the major nonlinear iterates. Only at the end of each major SQP iteration, say at the time points t_k, all updated data are transferred from the SQP process to the feedback process, and simultaneously the the current system state $x_0(t_k)$ is given to the SQP process, to modify the next major real-time iteration. The scheme can be visualized as follows:



Note that between the updates (that occur only at the major sampling times t_k) the feedback is based on a linear system model that is obtained by a linearization along the best available predicted trajectory, similar to the linearized neighboring feedback control scheme that was presented in Sec. 1.3.1.

Example 4.5 (Comparison of Moving Horizon Strategies)
The CSTR real-time scenario that was treated in the previous examples can in a straightforward way be formulated as a moving horizon problem. Instead of shrinking the time horizon of the problems, the time horizon is kept at constant length and moved forward. We can imagine that we continuously "append" multiple shooting intervals at the end of the horizon. With the chosen horizon length of 2000 *seconds the closed loop system was already at steady state in the middle of the horizon of the first optimization problem; therefore the appended parts do practically not matter at all, and for the shift strategy with exact Hessian SQP we obtain exactly the same closed loop trajectory as before in the real-time iteration Example 4.3 on a shrinking horizon. We have carried out closed-loop simulations for the same scenario also with the warm start technique, and the result is that the closed-loop trajectories are practically identical. We also carried out tests with an algorithm where the exact Hessian matrix was replaced by a Gauss-Newton approximation, which again yields no visible difference of the trajectories. The performance of the different strategies can be measured by the objective function on the considered interval of 2000 seconds compared to the optimal value, as in Table 4.1. For all four moving horizon strategies we have observed nearly identical values of 103 % of the optimal costs.*

For the chosen application, all strategies require more or less the same computational costs per real-time iteration, which is dominated by linear algebra, because the ODE solution and sensitivity computation do not require much time for a system of such a small size. The necessary CPU time per iteration, and the share of it which is needed to deliver the "immediate feedback" are depicted in Fig. 4.5 for the above scenario, where a Gauss-Newton method was employed. First, it can be seen that the overall cost of at most one second is much below the 20 seconds that we have chosen as sampling time, thus ensuring practical applicability of the algorithm for this type of problem. But it can also be seen that the "immediate" feedback requires a considerable proportion of the overall CPU time for this example problem and is therefore not as immediate as postulated. Note, however, that in

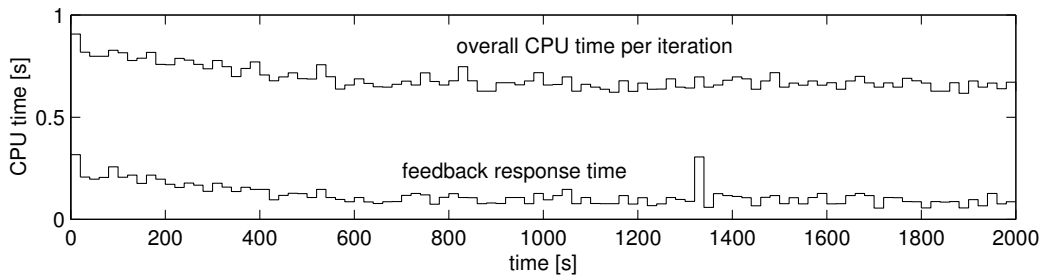


Figure 4.5: Preparation and feedback times for the real-time iterations for the moving horizon CSTR scenario of Example 4.5, on an AMD Athlon processor with 1009 MHz. The horizon length was chosen to be $N = 100$ multiple shooting intervals.

large scale applications the lion's share of the computational cost is incurred by the DAE solution, which does not contribute to the response time, so that the immediate feedback is indeed orders of magnitude smaller than the overall computation time (cf. Figs. 7.7 and 7.16 in Chap. 7). Details on the separation into preparation and feedback phase are given in Chap. 6.

Chapter 5

Contractivity of the Real-Time Iterations

In the last chapter we have presented a new scheme for the approximate solution of optimization problems in real-time, which shows very promising performance in numerical examples. From the previous discussion, however, it is far from clear how this scheme behaves theoretically.

To motivate why this is important, let us imagine that we apply the real-time iteration scheme in a NMPC framework to stabilize a system, and that the real system coincides with the employed system model. If the subsequent optimization problems could be solved exactly in real-time, proofs exist that ensure nominal stability of the closed-loop system for different NMPC schemes (cf. Sec. 1.4). Now the question arises if it is also possible to establish nominal stability results if the optimization problems are not solved exactly, but with our real-time iteration scheme. Otherwise, it may be possible that the real-time controller does *not* stabilize the system, but drives it in the worst case even away from the desired operating point: linearization errors may increase from iteration to iteration, and the approximations to the exact solutions may become worse and worse. We will show that this need not be feared, and we will prove that the real-time iterations deliver approximations of the exact solutions that become better and better, under reasonable conditions.

Unfortunately, standard convergence results for off-line SQP methods cannot be applied, because each real-time iteration belongs to a *different* optimization problem. Nevertheless, we will start the chapter in Sec. 5.1 by reviewing the local convergence properties for a class of off-line optimization algorithms that are commonly referred to as "Newton type methods", which comprise the exact Hessian SQP method and the Constrained Gauss-Newton method. We will then in Sec. 5.2 present the real-time iteration scheme in a new setting, which allows to compare the real-time iterates on shrinking horizons with those of the off-line method. This makes it possible to prove *contractivity* of the real-time iterations. We usually avoid the term "convergence" in the real-time framework on shrinking horizons, because the iterates stop after N cycles, when the time horizon of interest is over.

The principal result is that the real-time iteration scheme on shrinking horizons is contracting under the same sufficient conditions as the corresponding off-line scheme. This result can conceptually be generalized to the shift strategy on infinite moving horizons and allows to conclude that the real-time iteration scheme leads to a convergent closed-loop behaviour in this case. Finally, in Sec. 5.3 we investigate how far the result of the real-time iterations deviates from the theoretical optimal solutions.

Throughout the chapter we will assume that the iterates are started sufficiently close to a KKT point that satisfies the sufficient conditions of Theorem 3.3 and the strict complementarity condition, so that we can assume that the active set is known and we can restrict our attention to equality constrained problems. Furthermore, we will assume that the variables w can be split into free variables $q \in \mathbb{R}^{n_q}$ and dependent ones $s \in \mathbb{R}^{n_s}$, so that the off-line optimization problem on the full horizon of interest can be formulated as follows:

$$P(x_0): \qquad \min_{q,s} F(q,s) \quad \text{s.t.} \quad G(q,s) = 0 \tag{5.1}$$

with $F : \hat{D} \subset \mathbb{R}^{n_q} \times \mathbb{R}^{n_s} \to \mathbb{R}$ and $G : \hat{D} \subset \mathbb{R}^{n_q} \times \mathbb{R}^{n_s} \to \mathbb{R}^{n_s}$ twice continuously differentiable, where we assume that the constraint function G is such that $\frac{\partial G}{\partial s}$ is invertible for all $(q,s) \in \hat{D}$. This last property can naturally be achieved for the direct multiple shooting method, as discussed in Sec. 2.2.1, where q are the controls, and s the state variables. This separation helps to formulate the shrinking of the time horizon in the real-time setting; the shrinking will be expressed by decreasing step-by-step the degrees of freedom for the controls.

5.1 The Off-Line Problem

5.1.1 Newton Type Optimization Methods

Using the Lagrangian function $\mathcal{L} : \hat{D} \times \mathbb{R}^{n_s} \to \mathbb{R}$

$$\mathcal{L}(q,s,\lambda) := F(q,s) - \lambda^T G(q,s)$$

we can formulate necessary optimality conditions of first order, according to Theorem 3.1:

$$\nabla_{(q,s,-\lambda)} \mathcal{L}(q,s,\lambda) = \begin{pmatrix} \nabla_q F(q,s) - \nabla_q G(q,s)\lambda \\ \nabla_s F(q,s) - \nabla_s G(q,s)\lambda \\ G(q,s) \end{pmatrix} = 0. \tag{5.2}$$

Let us define for later convenience the vector $y \in \mathbb{R}^n$ with $n := n_q + n_s + n_s$ and the function $R : D \subset \hat{D} \times \mathbb{R}^{n_s} \to \mathbb{R}^n$ as

$$y := \begin{pmatrix} q \\ s \\ -\lambda \end{pmatrix} \quad \text{and} \quad R(y) := \begin{pmatrix} \nabla_q F(q,s) - \nabla_q G(q,s)\lambda \\ \nabla_s F(q,s) - \nabla_s G(q,s)\lambda \\ G(q,s) \end{pmatrix}, \tag{5.3}$$

so that the above system (5.2) is equivalent to $R(y) = 0$. To solve this system, the exact Newton-Raphson method would start at an initial guess y^0 and compute a sequence of iterates $y^1, y^2, \ldots$ according to

$$y^{k+1} = y^k + \Delta y^k, \tag{5.4}$$

where each Δy^k is the solution of the linearized system

$$R(y^k) + \frac{\partial R}{\partial y}(y^k)\Delta y^k = 0, \tag{5.5}$$

or, fully written,

$$\begin{pmatrix} \nabla_q \mathcal{L} \\ \nabla_s \mathcal{L} \\ G(q,s) \end{pmatrix} + \begin{pmatrix} \frac{\partial^2 \mathcal{L}}{\partial q^2} & \frac{\partial^2 \mathcal{L}}{\partial q \partial s}^T & \frac{\partial G}{\partial q}^T \\ \frac{\partial^2 \mathcal{L}}{\partial q \partial s} & \frac{\partial^2 \mathcal{L}}{\partial s^2} & \frac{\partial G}{\partial s}^T \\ \frac{\partial G}{\partial q} & \frac{\partial G}{\partial s} & 0 \end{pmatrix} \begin{pmatrix} \Delta q^k \\ \Delta s^k \\ -\Delta \lambda^k \end{pmatrix} = 0. \tag{5.6}$$

We have seen in Sec. 3.3.1 that these Newton-Raphson iterates are identical to the full step exact Hessian SQP method.

The Newton type methods considered in this chapter differ from the exact Newton-Raphson method in the way that the exact Hessian $\frac{\partial^2 \mathcal{L}}{\partial (q,s)^2}$ is replaced by a (symmetric) approximation

$$A(y) = \begin{pmatrix} A_{qq} & A_{qs}^T \\ A_{qs} & A_{ss} \end{pmatrix},$$

so that we can define an approximate derivative of R by:

$$J(y) := \begin{pmatrix} A_{qq} & A_{qs}^T & \frac{\partial G}{\partial q}^T \\ A_{qs} & A_{ss} & \frac{\partial G}{\partial s}^T \\ \frac{\partial G}{\partial q} & \frac{\partial G}{\partial s} & 0 \end{pmatrix}. \tag{5.7}$$

For our Newton type method, Eq. (5.5) is replaced by the approximation

$$R(y^k) + J(y^k)\Delta y^k = 0. \tag{5.8}$$

5.1.2 The Constrained Gauss-Newton Method

An important special case of the Newton type methods considered in this chapter is the constrained Gauss-Newton method, which is applicable for problems with a least squares form of the objective function F:

$$F(q,s) = \frac{1}{2}\|l(q,s)\|_2^2 \tag{5.9}$$

with $l : \hat{D} \to \mathbb{R}^{n_l}$, $n_l \geq 1$, a vector valued function. For this case, the Hessian approximation A is defined to be

$$A(q,s) := \left(\frac{\partial l}{\partial (q,s)}\right)^T \left(\frac{\partial l}{\partial (q,s)}\right). \tag{5.10}$$

The error $\frac{\partial^2 \mathcal{L}}{\partial (q,s)^2} - A$ can be quantified by calculation of $\frac{\partial^2 \mathcal{L}}{\partial (q,s)^2}$

$$\frac{\partial^2 \mathcal{L}}{\partial (q,s)^2} - \left(\frac{\partial l}{\partial (q,s)}\right)^T \left(\frac{\partial l}{\partial (q,s)}\right) = \sum_{i=1}^{n_l} l_i \frac{\partial^2 l_i}{\partial (q,s)^2} + \sum_{i=1}^{n_s} \lambda_i \frac{\partial^2 G_i}{\partial (q,s)^2}.$$

At a solution $y^* = (q^*, s^*, \lambda^*)$, the necessary optimality conditions (5.2) require that

$$\nabla_s F - \nabla_s G \lambda^* = \left(\frac{\partial l}{\partial s}\right)^T l(q^*, s^*) - \nabla_s G \lambda^* = 0,$$

so that

$$\lambda^* = -\nabla_s G^{-1} \frac{\partial l}{\partial s}^T l(q^*, s^*) = O\left(\|l(q^*, s^*)\|\right),$$

which allows to conclude that

$$\left\| \frac{\partial^2 \mathcal{L}}{\partial (q,s)^2} - A(q^*, s^*)) \right\| = O\left(\|l(q^*, s^*)\|\right).$$

Thus we expect the Gauss-Newton method to work well for small residual vectors $l(q,s)$. Note that $A(q,s)$ is independent of the multiplier vector λ.

Remark: The least squares function $l(q,s)$ needs not to be a mapping into a finite dimensional space $\mathbb{R}^{n_l}$, but may more generally be a mapping into any Hilbert space H. If $\langle \cdot, \cdot \rangle_H$ is the inner product in H, the least squares objective function of Eq. (5.9) is then written as

$$F(q,s) = \frac{1}{2} \langle l(q,s), l(q,s) \rangle_H,$$

and the Gauss-Newton approximation of the Hessian in Eq. (5.10) is given by the symmetric matrix

$$A(q,s)_{ij} := \mathrm{Re} \left\langle \frac{\partial l}{\partial (q,s)_i} \ , \ \frac{\partial l}{\partial (q,s)_j} \right\rangle_H,$$

where the indices i, j run through all components of (q,s). Note that this matrix is finite dimensional, which allows to treat this general case with the presented numerical methods. In Sec. 6.4 it is shown how to compute $A(q,s)$ efficiently in the presence of integral least squares terms as introduced in Sec. 1.1.

5.1.3 Sufficient Conditions for Local Convergence

Let us now state sufficient conditions for convergence of a series of general Newton type iterates (y^k), $k = 0, 1, \ldots$ in a space $\mathbb{R}^n$ defined by

$$y^{k+1} = y^k + \Delta y^k = y^k - J(y^k)^{-1} R(y^k), \tag{5.11}$$

towards a solution of the system

$$R(y) = 0. \tag{5.12}$$

Theorem 5.1 (Local Convergence of Newton Type Methods)
Let us assume that $R : D \subset \mathbb{R}^n \to \mathbb{R}^n$ is continuously differentiable and that the approximation of the derivative $J : D \subset \mathbb{R}^n \to \mathbb{R}^{(n\times n)}$ is continuous and has a continuous inverse on D. Furthermore, let us make the following assumptions:

$$\left\| J(y_1)^{-1} \left(J(y_2) - \frac{\partial R}{\partial y}(y_2) \right) \right\| \leq \kappa < 1, \quad \forall\, y_1, y_2 \in D \tag{5.13a}$$

and

$$\left\| J(y_1)^{-1} \left(J(y_2) - J(y_3) \right) \right\| \leq \omega \| y_2 - y_3 \|, \quad \forall\, y_1, y_2, y_3 \in D. \tag{5.13b}$$

Additionally, we suppose that the first step $\Delta y^0 := -J(y^0)^{-1} R(y^0)$ starting at an initial guess y^0 is sufficiently small, so that

$$\delta_0 := \kappa + \frac{\omega}{2} \| \Delta y^0 \| < 1 \tag{5.13c}$$

and that

$$D_0 := \left\{ y \in \mathbb{R}^n \,|\, \| y - y^0 \| \leq \frac{\| \Delta y^0 \|}{1 - \delta_0} \right\} \subset D. \tag{5.13d}$$

Under these conditions the sequence of Newton type iterates (y^k) defined by Eq. (5.11) remains inside D_0 and converges towards a $y^* \in D_0$ satisfying the sytem (5.12), $R(y^*) = 0$.

Proof: Slightly modifying a proof that can be found in Bock [Boc87], we first show that the norm of the steps Δy^k contracts, and show then that (y^k) is a Cauchy sequence. The contraction can be shown as follows:

$$\begin{aligned}
\| \Delta y^{k+1} \| &= \| J(y^{k+1})^{-1} \cdot R(y^{k+1}) \| \\
&= \| J(y^{k+1})^{-1} \cdot (R(y^{k+1}) - R(y^k) - J(y^k) \cdot \Delta y^k) \| \\
&= \| J(y^{k+1})^{-1} \cdot \textstyle\int_0^1 (\frac{\partial R}{\partial y}(y^k + t\Delta y^k) - J(y^k)) \cdot \Delta y^k \, dt \| \\
&= \| J(y^{k+1})^{-1} \cdot \textstyle\int_0^1 (\frac{\partial R}{\partial y}(y^k + t\Delta y^k) - J(y^k + t\Delta y^k)) \Delta y^k \, dt \\
&\quad + J(y^{k+1})^{-1} \cdot \textstyle\int_0^1 (J(y^k + t\Delta y^k) - J(y^k)) \Delta y^k \, dt \| \\
&\leq \textstyle\int_0^1 \| J(y^{k+1})^{-1} \, (\frac{\partial R}{\partial y}(y^k + t\Delta y^k) - J(y^k + t\Delta y^k)) \| \| \Delta y^k \| \, dt \\
&\quad + \textstyle\int_0^1 \| J(y^{k+1})^{-1} \, (J(y^k + t\Delta y^k) - J(y^k)) \| \| \Delta y^k \| \, dt \\
&\leq \kappa \| \Delta y^k \| + \textstyle\int_0^1 \omega t \| \Delta y^k \|^2 \, dt \\
&= \left(\kappa + \tfrac{\omega}{2} \| \Delta y^k \| \right) \| \Delta y^k \| =: \delta_k \| \Delta y^k \|.
\end{aligned} \tag{5.14}$$

If $\delta_k \leq 1$, then $\|\Delta y^{k+1}\| \leq \|\Delta y^k\|$, and $\delta_{k+1} \leq \delta_k \leq 1$. Therefore, we can inductively deduce that

$$\|\Delta y^{k+1}\| \leq \delta_0 \|\Delta y^k\|, \quad \forall\, k \geq 0$$

so that

$$\|y^{k+m} - y^k\| \leq \frac{1}{1-\delta_0}\|\Delta y^k\| \leq \frac{\delta_0^k}{1-\delta_0}\|\Delta y^0\|, \quad \forall\, k, m \geq 0. \tag{5.15}$$

In particular,

$$\|y^m - y^0\| \leq \frac{1}{1-\delta_0}\|\Delta y^0\|, \quad \forall m \geq 0.$$

Therefore, (y^k) is a Cauchy sequence that remains inside the compact set D_0 and hence converges towards a limit point y^*. By continuity of R and J^{-1},

$$0 = \lim_{k\to\infty} \Delta y^k = \lim_{k\to\infty} -J(y^k)^{-1}R(y^k) = -J(y^*)^{-1}R(y^*)$$

so that $R(y^*) = 0$. □

We will state a second, stricter form of the above theorem, which is applicable to optimization problems only, and implies local convergence of the Newton type iterates towards a strict local minimum. Before this second version of the theorem can be formulated, we have to give an explicit formula for the inverse of the approximate derivative $J(y)$.

Lemma 5.2 (Inverse of the KKT Matrix)
Let us assume that $J(y)$ is a matrix as defined in Eq. (5.7), i.e.,

$$J(y) = \begin{pmatrix} A_{qq} & A_{qs}^T & \frac{\partial G}{\partial q}^T \\ A_{qs} & A_{ss} & \frac{\partial G}{\partial s}^T \\ \frac{\partial G}{\partial q} & \frac{\partial G}{\partial s} & 0 \end{pmatrix}$$

with $\frac{\partial G}{\partial s}$ invertible, and let us also assume that the so called reduced Hessian matrix

$$A_r(y) := \left(\mathbb{I} \,\middle|\, -\tfrac{\partial G}{\partial q}^T \left(\tfrac{\partial G}{\partial s}\right)^{-T} \right) \begin{pmatrix} A_{qq} & A_{qs}^T \\ A_{qs} & A_{ss} \end{pmatrix} \begin{pmatrix} \mathbb{I} \\ -\left(\frac{\partial G}{\partial s}\right)^{-1} \frac{\partial G}{\partial q} \end{pmatrix} \tag{5.16}$$

is positive definite. Then the inverse of $J(y)$ exists and is given by the formula

$$J(y)^{-1} = C_1(y) A_r(y)^{-1} C_1(y)^T + C_2(y) \tag{5.17}$$

with

$$C_1(y) := \begin{pmatrix} \mathbb{I} \\ -\left(\frac{\partial G}{\partial s}\right)^{-1} \frac{\partial G}{\partial q} \\ -\left(\frac{\partial G}{\partial s}\right)^{-T} \left(A_{qs} - A_{ss} \left(\frac{\partial G}{\partial s}\right)^{-1} \frac{\partial G}{\partial q} \right) \end{pmatrix} \tag{5.18}$$

and

$$C_2(y) := \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & \left(\frac{\partial G}{\partial s}\right)^{-1} \\ 0 & \left(\frac{\partial G}{\partial s}\right)^{-T} & -\left(\frac{\partial G}{\partial s}\right)^{-T} A_{ss} \left(\frac{\partial G}{\partial s}\right)^{-1} \end{pmatrix}. \tag{5.19}$$

Remark: Note that the assumptions of this lemma coincide with those of Lemma 3.2, with the constraint matrix B set to $B = (\frac{\partial G}{\partial q} | \frac{\partial G}{\partial s})$. They are also closely related to the sufficient optimality conditions of Theorem 3.3. The positive definiteness of A_r will be used in the proof of Theorem 5.3 to show that the Newton type iterates converge towards a local minimum.

Proof: The invertibility follows from Lemma 3.2. The inversion formula (5.17) can be verified by checking that

$$J(y)(C_1(y)A_r(y)^{-1}C_1(y)^T + C_2(y)) = \mathbb{I}$$

and using the fact that

$$J(y)C_1(y) = \begin{pmatrix} A_r(y) \\ 0 \\ 0 \end{pmatrix} \quad \text{and} \quad J(y)C_2(y) = \mathbb{I} - \begin{pmatrix} C_1(y)^T \\ 0 \\ 0 \end{pmatrix}.$$

Theorem 5.3 (Off-Line Convergence)
Let us assume that $R : D \to \mathbb{R}^n$ is defined according to (5.3) *to be the residual of the necessary optimality conditions of the equality constrained optimization problem* (5.1).

We assume that the reduced Hessian approximation $A_r(y)$ from Eq. (5.16) *is positive definite on the whole domain D, with bounded inverse:*

$$\|A_r(y)^{-1}\| \leq \beta_A < \infty, \quad \forall\, y \in D. \tag{5.20a}$$

We also assume boundedness of $\|C_1\|$ and $\|C_2\|$ as defined in Eqs. (5.18) *and* (5.19) *on D:*

$$\|C_1(y)\| \leq \beta_{C_1} < \infty, \quad \forall\, y \in D, \tag{5.20b}$$

and

$$\|C_2(y)\| \leq \beta_{C_2} < \infty, \quad \forall\, y \in D. \tag{5.20c}$$

Let us define $\beta := \beta_{C_1}\beta_A\beta_{C_1} + \beta_{C_2}$. Let us suppose that a Lipschitz condition

$$\beta\, \|J(y_1) - J(y_2)\| \leq \omega \|y_1 - y_2\|, \quad \omega < \infty, \quad \forall\, y_1, y_2 \in D, \tag{5.20d}$$

holds for the derivative approximation $J(y)$ and that

$$\beta \left\| A(y) - \frac{\partial^2 \mathcal{L}}{\partial (q,s)^2}(y) \right\| \leq \kappa, \quad \kappa < 1, \quad \forall\, y \in D. \tag{5.20e}$$

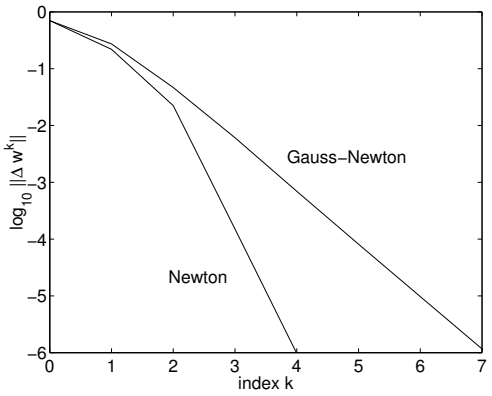


Figure 5.1: Convergence rates for off-line Newton type methods, for the solution of Example 5.1. Comparison of Newton and Gauss-Newton method, when started at the steady state trajectory.

Additionally, we assume as in Theorem 5.1 that the first step $\Delta y^0 := -J(y^0)^{-1}R(y^0)$ *starting at an initial guess* y^0 *is sufficiently small, so that*

$$\delta_0 := \kappa + \frac{\omega}{2}\|\Delta y^0\| < 1 \tag{5.20f}$$

and that

$$D_0 := \left\{ y \in \mathbb{R}^n \middle| \, \|y - y^0\| \leq \frac{\|\Delta y^0\|}{1-\delta_0} \right\} \subset D. \tag{5.20g}$$

Under these circumstances the Newton type iterates (y^k) *according to Eq.* (5.8) *converge towards a KKT point* $y^* = (q^*, s^*, \lambda^*) \in D$ *whose primal part* (q^*, s^*) *is a strict local minimum of Problem* (5.1).

A proof of the theorem is given in Appendix D.

Example 5.1 (Continuous Stirred Tank Reactor)
Let us again consider the optimal control problem that was introduced in Sec. 2.1, respectively its multiple shooting parameterization as described in Example 2.1. The solution w^* *of the NLP with initial value as in Sec. 2.1 was shown in Fig. 2.3; a comparison of the convergence rates for the Newton and Gauss-Newton method is given in Fig. 5.1, where the algorithm is started at the steady state trajectory. The plot for the Gauss-Newton method allows us to estimate that* $\kappa \approx 0.1$.

5.2 The On-Line Problem

We will now regard real-time iterations on shrinking horizons as introduced in Sec. 4.1.2, in the framework of the off-line optimization problem. It is shown in Sec. 5.2.1 that the essential difference to the off-line iterations is that after each Newton type iteration some components of the free variables q are fixed, i.e., that the optimization problem (5.1) is changed to a problem in the same space, but with some more (trivial) equality constraints. In the following Sec. 5.2.2 we will show that the nonlinearity and incompatibility constants ω and κ for the off-line problem are still valid for a problem with some fixed controls. This allows to conclude in Theorem 5.6 that the real-time iterations contract if the sufficient conditions for off-line convergence of Theorem 5.3 are satisfied, which is the main result of this chapter. In Sec. 5.3 we investigate how far the result of the real-time iterations deviates from the theoretical optimal solutions.

Let us first discuss why fixing of some free components is equivalent to the real-time iteration scheme as introduced in Sec. 4.1.2.

5.2.1 The Fixed Control Formulation

In the real-time iteration framework for shrinking horizons of Sec. 4.3, we have reduced the number of multiple shooting nodes from one problem to the next, in order to keep pace with the process development.

We regard a problem discretization with N multiple shooting intervals on a *fixed length* time horizon with duration T, and assume that the computation time for the k-th real-time iteration is δ_k, and that $\sum_{k=1}^{N} \delta_k = T$ (this is e.g. the case if all iterations take the same time δ and the time horizon of interest has the length $T = N\delta$). The multiple shooting points are chosen so that the times $t_k := T\tau_k$ satisfy $t_k - t_{k-1} = \delta_k$, i.e.,

$$t_0 = 0, \quad t_k = \sum_{i=1}^{k} \delta_i, \quad \text{for} \quad k = 1, \dots, N.$$

Let $x_0, x_1, \dots, x_N$ denote the differential system states of the real system at times $t_0, t_1, \dots, t_N$, that serve as initial values for the parameterized optimization problems $P_k(x_k)$ of shrinking length, as defined in Sec. 4.3, Eq. (4.1).

At time t_0 the state x_0 is known, and the initial value embedding strategy quickly yields the control value u_0 that will be implemented on the first time interval, up to time t_1. At time t_1 the next immediate feedback has been prepared, and is applied to the shrunk problem $P_1(x_1)$. If the model and the real system coincide, the new system state x_1 is identical to the final value $x(t_1)$ of the initial value problem

$$\begin{aligned} B(\cdot)\dot{x}(t) &= f(x(t), z(t), u_0), \quad t \in [t_0, t_1], \\ 0 &= g(x(t), z(t), u_0), \quad t \in [t_0, t_1], \\ x_0(t_0) &= x_0, \end{aligned}$$

and the initial value constraint for the problem $P_1(x_1)$ is

$$s_1^x = x_1.$$

Let us now regard the original problem $P_0(x_0)$ on the full horizon, but with an additional constraint that fixes the control q_0 on the first interval to be equal to the implemented value $u_0 = q_0^0 + \Delta q_0^0 = q_0^1$. This problem then contains the constraints

$$\begin{aligned} s_1^x - x_0(t_1; s_0^x, s_0^z, q_0) &= 0, \\ g(s_0^x, s_0^z, q_0) &= 0, \\ s_0^x - x_0 &= 0, \\ q_0 - u_0 &= 0, \end{aligned}$$

which constrain s_0^x, s_0^z, q_0 and s_1^x uniquely. In the solution, $s_1^x = x_1$, because the relaxed initial value problem

$$\begin{aligned} B(\cdot)\dot{x_0}(t) &= f(x_0(t), z_0(t), q_0), \quad t \in [t_0, t_1], \\ 0 &= g(x_0(t), z_0(t), q_0) - e^{-\beta \frac{t-t_0}{t_1-t_0}} g(s_0^x, s_0^z, q_0), \quad t \in [t_0, t_1], \\ x_0(t_0) &= s_0^x, \end{aligned}$$

is equivalent to the real system dynamics if $s_0^x = x_0$, $q_0 = u_0$, $g(s_0^x, s_0^z, q_0) = 0$, so that $x_0(t_1; s_0^x, s_0^z, q_0) = x(t_1) = x_1$. Once the correct values for s_0, q_0 are found during the iterative solution procedure, they are not changed anymore, and the above constraints are completely equivalent to $s_1^x = x_1$.

For ODE models, the correct solution for s_0^x, q_0 is already found after the first iterate, due to the linearity of the initial value constraint, and due to the fact that u_0 was set just to the outcome of this first iterate. Therefore, fixing of q_0 is completely equivalent to the shrinking of the horizon. One slight complication arises, however, for DAE models: after the first iterate, s_0^z may still not be at its correct value in the fixed control formulation (i.e., $g(s_0^x, s_0^z, q_0) \neq 0$) and this may result in a value $x_0(t_1; s_0^x, s_0^z, q_0)$ that is slightly different from the correct value $x(t_1)$, due to the DAE relaxation. We will disregard this slight difference that is only present in the relaxed DAE case (and that could even be interpreted as a slight superiority of the real implementation over the fixed control formulation, which we only introduce here for theoretical purposes).

Let us therefore assume in the remaining part of this chapter that the real-time iterations on shrinking horizons are identical to a subsequent fixing of the controls $q_0, \ldots, q_{N-1}$ in the original off-line optimization problem (5.1), which we denoted by $P(x_0)$ or $P_0(x_0)$. For notational convenience, we will in the following define $P^0 := P_0(x_0) = P(x_0)$ to be the original (off-line) problem, and denote by P^k the problems with more and more fixed controls that are generated during the real-time iterations and which are equivalent to the shrinking horizon problems $P_k(x_k)$. A visualization of the outcome of the first two real-time iterations is given in Figures 5.2 and 5.3.

The series of problems P^k, $k = 0, \ldots, N$ are given by

$$P^k : \min_{q,s} F(q,s) \quad \text{subject to} \quad \left\{ \begin{array}{rcl} G(q,s) & = & 0, \\ q_i - u_i & = & 0, \quad i = 0, \ldots, k-1. \end{array} \right.$$

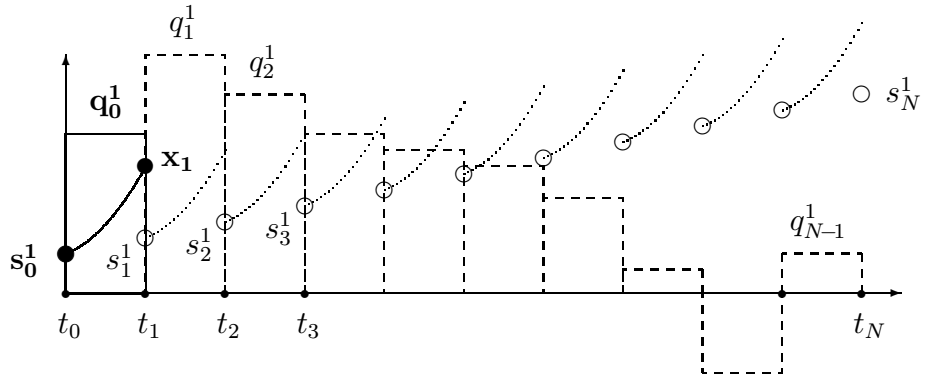


Figure 5.2: NLP variable content after the first real-time iteration. The next optimization problem, P^1, is formulated by fixing the first control value q_0, which implicitly fixes the value x_1.

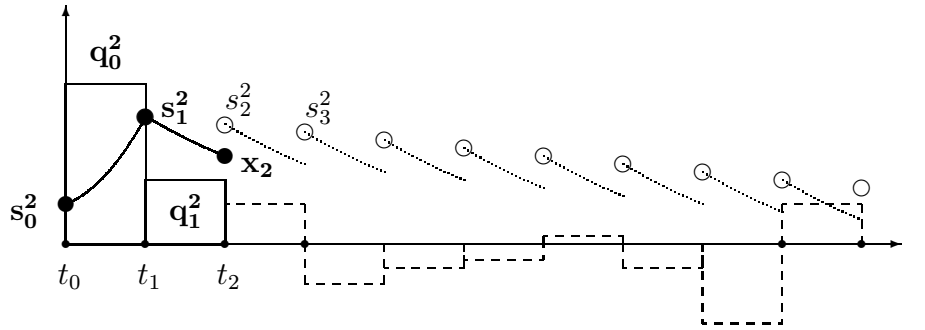


Figure 5.3: NLP variable content after the second real-time iteration. The problem P^2 can be formulated by fixing q_0 and q_1, which implicitly fixes x_2.

(Recall that $q = (q_0, q_1, \ldots, q_{N-1})$.) In a shorter formulation, we will also write these problems as

$$P^k : \min_{q,s} F(q,s) \quad \text{s.t.} \quad \left\{ \begin{array}{rcl} G(q,s) & = & 0, \\ Q_1^{k\,T}(q - u^k) & = & 0. \end{array} \right.$$

Here, the matrices $Q_1^k \in \mathbb{R}^{n_q \times m_k}$, $m_k = k n_u$, are of the form

$$Q_1^k := \begin{pmatrix} \mathbb{I}_{m_k} \\ 0 \end{pmatrix}, \tag{5.21}$$

and the vectors $u^k \in \mathbb{R}^{n_q}$ are defined as

$$u^k := \begin{pmatrix} u_0 \\ \vdots \\ u_{k-1} \\ * \end{pmatrix}, \quad \text{with} \quad Q_1^{k\,T}(q - u^k) = \begin{pmatrix} q_0 - u_0 \\ \vdots \\ q_{k-1} - u_{k-1} \end{pmatrix},$$

so that the last components of u^k, that are introduced only for later notational convenience, can carry arbitrary values. Note that in the last problem P^N no degrees of freedom remain, as all components of q are fixed.

5.2.2 Fixing Some Controls

We will now prove that the nonlinearity and incompatibility constants ω and κ from the off-line problem (5.1) are still valid for any modified problem P^k, when some controls are fixed. Let us in this subsection consider only one modified optimization problem P^k and drop the index k for notational simplicity:

$$\min_{q,s} F(q,s) \quad \text{s.t.} \quad \left\{ \begin{array}{rcl} G(q,s) & = & 0 \\ Q_1^T(q - u) & = & 0 \end{array} \right. \tag{5.22}$$

where the matrix $Q_1 \in \mathbb{R}^{n_q \times m}$, $m \leq n_q$ consists of m orthonormal columns, as in (5.21). We also introduce the orthonormal complement of Q_1 by

$$Q_2 := \begin{pmatrix} 0 \\ \mathbb{I}_{(n_q - m)} \end{pmatrix} \in \mathbb{R}^{n_q \times (n_q - m)}.$$

Let us formulate the necessary first-order conditions of optimality for the modified problem. We introduce the Lagrangian function $\tilde{\mathcal{L}}$ of the modified problem (5.22)

$$\tilde{\mathcal{L}}(q,s,\lambda,\mu) := \mathcal{L}(q,s,\lambda) - {\lambda_Q}^T Q_1^T(q - u),$$

where $\mathcal{L}$ is the Lagrangian function of the original problem (5.1). The necessary conditions of optimality for the modified problem are:

$$\nabla_{(q,s,-\lambda,-\lambda_Q)} \tilde{\mathcal{L}}(q,s,\lambda,\lambda_Q) = \begin{pmatrix} \nabla_q \mathcal{L} - Q_1 \lambda_Q \\ \nabla_s \mathcal{L} \\ G(q,s) \\ Q_1^T(q - u) \end{pmatrix} = 0. \tag{5.23}$$

Multiplying the first component of this vector by the orthogonal matrix $(Q_1|Q_2)^T \in \mathbb{R}^{n_q \times n_q}$ yields

$$\begin{pmatrix} \begin{pmatrix} -\lambda_Q + Q_1^T \nabla_q \mathcal{L} \\ Q_2^T \nabla_q \mathcal{L} \end{pmatrix} \\ \nabla_s \mathcal{L} \\ G(q,s) \\ Q_1^T (q-u) \end{pmatrix} = 0$$

and it can be seen that the upper part can always be made zero by choosing

$$\lambda_Q := Q_1^T \nabla_q \mathcal{L}.$$

Therefore, the trivial first condition can be omitted in the formulation of the necessary conditions for optimality of the modified problem and we do *not* have to regard the additional multipliers λ_Q. This allows us to treat the modified problem in the same primal-dual space of $y \in \mathbb{R}^n$ as the original problem, with $n = n_q + n_s + n_s$. Defining the essential part of the residual of the necessary optimality conditions to be

$$\tilde{R}(y) := \begin{pmatrix} Q_1^T(q-u) \\ Q_2^T \nabla_q \mathcal{L} \\ \nabla_s \mathcal{L} \\ G(q,s) \end{pmatrix} = \begin{pmatrix} Q_1^T(q-u) \\ 0 \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 0 & & & \\ Q_2^T & & & \\ & \mathbb{I} & & \\ & & \mathbb{I} & \end{pmatrix} R(y)$$

we can compute the derivative

$$\frac{\partial \tilde{R}}{\partial y} = \begin{pmatrix} Q_1^T \\ 0 \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 0 & & & \\ Q_2^T & & & \\ & \mathbb{I} & & \\ & & \mathbb{I} & \end{pmatrix} \frac{\partial R}{\partial y} \tag{5.24}$$

and provide an approximation of this derivative which uses the approximation $J(y)$ of the original problem

$$\begin{aligned} \tilde{J}(y) &:= \begin{pmatrix} Q_1^T \\ 0 \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 0 & & & \\ Q_2^T & & & \\ & \mathbb{I} & & \\ & & \mathbb{I} & \end{pmatrix} J(y) \\ &= \begin{pmatrix} Q_1^T & 0 & 0 \\ Q_2^T A_{qq} & Q_2^T A_{qs}^T & Q_2^T \frac{\partial G}{\partial q}^T \\ A_{qs} & A_{ss} & \frac{\partial G}{\partial s}^T \\ \frac{\partial G}{\partial q} & \frac{\partial G}{\partial s} & 0 \end{pmatrix}. \end{aligned} \tag{5.25}$$

Theorem 5.4 (Contraction Constants for the Modified Problem)
Let us assume that the sufficent conditions for local convergence of Theorem 5.3 are satisfied

for the original (off-line) problem (5.1). *Then the derivative approximation* $\tilde{J} : D \to \mathbb{R}^n$ *of the modified problem* (5.22) *satisfies the two conditions*

$$\left\| \tilde{J}(y_1)^{-1} \left(\tilde{J}(y_2) - \frac{\partial \tilde{R}}{\partial y}(y_2) \right) \right\| \leq \kappa < 1, \quad \forall\, y_1, y_2 \in D, \text{ and} \tag{5.26a}$$

$$\left\| \tilde{J}(y_1)^{-1} \left(\tilde{J}(y_2) - \tilde{J}(y_3) \right) \right\| \leq \omega \|y_2 - y_3\|, \quad \forall\, y_1, y_2, y_3 \in D, \tag{5.26b}$$

with the same values of κ *and* ω *as the off-line problem.*

Remark: These two bounds correspond to the properties (5.13a) and (5.13b) in Theorem 5.1.3, thus allowing to conclude that the contraction inequality

$$\|\Delta y^{k+1}\| \leq \left(\kappa + \frac{\omega}{2} \|\Delta y^k\| \right) \|\Delta y^k\|$$

also holds for the modified problem. This implies that the optimization problem does not become less tractable from the algorithmic viewpoint when we add additional constraints. However, we do not address the question a suitable problem initialization, yet.

To prove the theorem, let us first give an explicit formula of the inverse $\tilde{J}(y)^{-1}$.

Lemma 5.5 (KKT Inverse for the Modified Problem)
The inverse of the matrix $\tilde{J}(y)$ *as defined in Eq.* (5.25) *is given by the formula*

$$\tilde{J}(y)^{-1} = Q \left(\begin{array}{c|c} \mathbb{I} & \\ \hline & \tilde{C}_1 \tilde{A}_r^{-1} \tilde{C}_1^T + \tilde{C}_2 \end{array} \right) \left(\begin{array}{c|c} \mathbb{I} & \\ \hline -Q_2^T A_{qq} Q_1 & \mathbb{I} \\ -A_{qs} Q_1 & \quad \mathbb{I} \\ -\frac{\partial G}{\partial q} Q_1 & \quad\quad \mathbb{I} \end{array} \right), \tag{5.27}$$

with

$$Q := \begin{pmatrix} Q_1 & Q_2 & & \\ & & \mathbb{I} & \\ & & & \mathbb{I} \end{pmatrix},$$

$$\tilde{A}_r(y) := Q_2^T A_r(y) Q_2, \tag{5.28a}$$

$$\tilde{C}_1(y) := \begin{pmatrix} Q_2^T & & \\ & \mathbb{I} & \\ & & \mathbb{I} \end{pmatrix} C_1(y) Q_2, \tag{5.28b}$$

and

$$\tilde{C}_2(y) := \begin{pmatrix} Q_2^T & & \\ & \mathbb{I} & \\ & & \mathbb{I} \end{pmatrix} C_2(y) \begin{pmatrix} Q_2 & & \\ & \mathbb{I} & \\ & & \mathbb{I} \end{pmatrix}, \tag{5.28c}$$

where $A_r(y)$, $C_1(y)$, *and* $C_2(y)$ *are defined as in Lemma 5.2 for the original problem.*

Proof of Lemma 5.5: We will check that $\tilde{J}(y)\tilde{J}(y)^{-1} = \mathbb{I}$. First note that

$$\tilde{J}(y)\,Q = \begin{pmatrix} \mathbb{I} & & & \\ Q_2^T A_{qq} Q_1 & Q_2^T A_{qq} Q_2 & Q_2^T A_{qs}^T & Q_2^T \frac{\partial G}{\partial q}^T \\ A_{qs} Q_1 & A_{qs} Q_2 & A_{ss} & \frac{\partial G}{\partial s}^T \\ \frac{\partial G}{\partial q} Q_1 & \frac{\partial G}{\partial q} Q_2 & \frac{\partial G}{\partial s} & 0 \end{pmatrix}.$$

The inverse of the lower-right part of this matrix can be obtained by an application of Lemma 5.2 (using $Q_2^T Q_2 = \mathbb{I}$). Its inverse is given as

$$\begin{pmatrix} Q_2^T A_{qq} Q_2 & Q_2^T A_{qs}^T & Q_2^T \frac{\partial G}{\partial q}^T \\ A_{qs} Q_2 & A_{ss} & \frac{\partial G}{\partial s}^T \\ \frac{\partial G}{\partial q} Q_2 & \frac{\partial G}{\partial s} & 0 \end{pmatrix}^{-1} = \tilde{C}_1(y) \tilde{A}_r^{-1}(y) \tilde{C}_1^T(y) + \tilde{C}_2.$$

Therefore,

$$\tilde{J}(y)\,Q \left(\begin{array}{c|c} \mathbb{I} & \\ \hline & \tilde{C}_1 \tilde{A}_r^{-1} \tilde{C}_1^T + \tilde{C}_2 \end{array}\right) = \left(\begin{array}{c|c} \mathbb{I} & \\ \hline Q_2^T A_{qq} Q_1 & \mathbb{I} \\ A_{qs} Q_1 & \mathbb{I} \\ \frac{\partial G}{\partial q} Q_1 & \mathbb{I} \end{array}\right),$$

which is the inverse of the rightmost block Frobenius matrix in formula (5.27). □

Proof of Theorem 5.4: Note that $\tilde{C}_1(y)$ and $\tilde{C}_2(y)$ as defined in Eqs. (5.28b) and (5.28c) are projections of $C_1(y)$ and $C_2(y)$, so that their (spectral) matrix norm satisfies

$$\|\tilde{C}_1(y)\| \le \|C_1(y)\| \le \beta_{C_1}, \quad \forall y \in D,$$

and

$$\|\tilde{C}_2(y)\| \le \|C_2(y)\| \le \beta_{C_2}, \quad \forall y \in D.$$

To provide a bound on the inverse $\tilde{A}_r(y)^{-1}$ we have to use the fact that $A_r(y)$ is positive definite. First we show that the eigenvalues of the projection $\tilde{A}_r(y) = Q_2^T A_r(y) Q_2$ lie between the maximum and minimum eigenvalues of $A_r(y)$. To prove this we note that $\tilde{A}_r(y)$ is a submatrix of $A_r(y)$, as

$$A_r(y) = \begin{pmatrix} Q_1^T \\ Q_2^T \end{pmatrix} A_r(y) \; (Q_1 | Q_2) = \begin{pmatrix} * & * \\ * & \tilde{A}_r(y) \end{pmatrix}$$

By the interlacing property (see e.g. [Wil65], pp. 103–104), the eigenvalues $\tilde{\lambda}_1(y), \ldots, \tilde{\lambda}_{(n_q - m)}(y)$ of the submatrix $\tilde{A}_r(y)$ must lie in the spectrum of $A_r(y)$, i.e., $\tilde{\lambda}_k(y) \in [\lambda_1(y), \lambda_{n_q}(y)]$. In particular, $\lambda_1(y) \le \tilde{\lambda}_1(y)$ for the smallest eigenvalues. The

inverse of the smallest eigenvalue corresponds to the spectral norm of the inverse of a positive definite matrix, so that we deduce that

$$\|\tilde{A}_r(y)^{-1}\| = \frac{1}{\tilde{\lambda}_1(y)} \leq \frac{1}{\lambda_1(y)} = \|A_r^{-1}(y)\| \leq \beta_A, \quad \forall y \in D.$$

This allows to find a bound on the central part of the inverse $\tilde{J}(y)^{-1}$ in formula (5.27):

$$\left\|\tilde{C}_1(y)\tilde{A}_r(y)^{-1}\tilde{C}_1(y)^T + \tilde{C}_2(y)\right\| \leq \beta_{C_1}\beta_A\beta_{C_1} + \beta_{C_2} = \beta, \quad \forall y \in D.$$

From Eqs. (5.24) and (5.25) it follows that

$$\tilde{J}(y_2) - \frac{\partial \tilde{R}}{\partial y}(y_2) = \begin{pmatrix} 0 & & \\ Q_2^T & & \\ & \mathbb{I} & \\ & & \mathbb{I} \end{pmatrix} \left(J(y_2) - \frac{\partial R}{\partial y}(y_2)\right)$$

and

$$\tilde{J}(y_2) - \tilde{J}(y_3) = \begin{pmatrix} 0 & & \\ Q_2^T & & \\ & \mathbb{I} & \\ & & \mathbb{I} \end{pmatrix} (J(y_2) - J(y_3))$$

and from formula (5.27) that

$$\tilde{J}^{-1}(y_1) \begin{pmatrix} 0 & & \\ Q_2^T & & \\ & \mathbb{I} & \\ & & \mathbb{I} \end{pmatrix}$$

$$= \begin{pmatrix} Q_2 & & \\ & \mathbb{I} & \\ & & \mathbb{I} \end{pmatrix} \left(\tilde{C}_1(y_1)\tilde{A}_r(y_1)^{-1}\tilde{C}_1(y_1)^T + \tilde{C}_2(y_1) \right) \begin{pmatrix} Q_2^T & & \\ & \mathbb{I} & \\ & & \mathbb{I} \end{pmatrix},$$

which has a spectral norm less or equal to β. This allows to establish the desired bounds:

$$\begin{aligned} &\left\| \tilde{J}^{-1}(y_1) \left(\tilde{J}(y_2) - \frac{\partial \tilde{R}}{\partial y}(y_2) \right) \right\| \\ &= \left\| \tilde{J}^{-1}(y_1) \begin{pmatrix} 0 & & \\ Q_2^T & & \\ & \mathbb{I} & \\ & & \mathbb{I} \end{pmatrix} \left(J(y_2) - \frac{\partial R}{\partial y}(y_2) \right) \right\| \\ &= \beta \left\| J(y_2) - \frac{\partial \tilde{R}}{\partial y}(y_2) \right\| \leq \kappa < 1, \quad \forall y_1, y_2 \in D, \end{aligned}$$

and

$$\begin{aligned}&\left\| \tilde{J}^{-1}(y_1) \left(\tilde{J}(y_2) - \tilde{J}(y_3) \right) \right\| \\ &= \beta \, \|J(y_2) - J(y_3)\| \leq \omega \|y_2 - y_3\|, \quad \forall y_1, y_2, y_3 \in D.\end{aligned}$$

□

5.2.3 Convergence of the Real-Time Iterations

In this subsection we finally consider the scenario that we subsequently fix more and more free components during the Newton type iterations. To be able to speak conveniently about convergence, and to be able to define a limit point y^* of the real-time iterates y^k, we will regard an infinite sequence of optimization problems P^k, $k = 0, 1, \ldots$, where we define $P^{N+k} := P^N$ for $k > 0$. Following the N-th iterate, no degrees of freedom remain, but the iterates may still be converging towards feasibility.[1]

As discussed above, at the k-th iterate the problem P^k

$$P^k : \min_{q,s} F(q,s) \quad \text{s.t.} \quad \left\{ \begin{array}{rcl} G(q,s) & = & 0 \\ Q_1^{k\,T}(q - u^k) & = & 0 \end{array} \right. \tag{5.29}$$

is treated, where the matrices $Q_1^k \in \mathbb{R}^{n_q \times m_k}$ and their orthonormal complements $Q_2^k \in \mathbb{R}^{n_q \times (n_q - m_k)}$ are of the form

$$Q_1^k := \begin{pmatrix} \mathbb{I}_{m_k} \\ 0 \end{pmatrix}, \quad \text{and} \quad Q_2^k := \begin{pmatrix} 0 \\ \mathbb{I}_{(n_q - m_k)} \end{pmatrix}$$

with nondecreasing integers m_k that satisfy $0 = m_0 \leq m_k \leq n_q$. Note that

$$Q_2^{k+1} = Q_2^k \Pi_k \quad \text{with} \quad \Pi_k := \begin{pmatrix} 0 \\ \mathbb{I}_{(n_q - m_{k+1})} \end{pmatrix} \in \mathbb{R}^{(n_q - m_k) \times (n_q - m_{k+1})}.$$

The vectors u^k will be defined during the iterations with iterates $y^k = (q^k, s^k, \lambda^k)$, to be

$$u^k := q^k, \quad k = 0, \ldots$$

Note that the first problem P^0 has no additional constraint, because $m_0 = 0$, and corresponds to the original off-line problem (5.1) that was treated in Sec. 5.1. In contrast to P^0, the problems $P^1, P^2, \ldots$ are only generated during the iterations and therefore depend on the initialization y^0 and on the chosen Newton type method.

Each problem P^k is equivalent to finding the zero of a function R^k as follows:

$$R^k(y) := \begin{pmatrix} Q_1^{k\,T}(q - u^k) \\ Q_2^{k\,T} \nabla_q \mathcal{L} \\ \nabla_s \mathcal{L} \\ G(q,s) \end{pmatrix} = 0.$$

[1] Note, however, that for ODE models a feasible solution is already obtained after the N-th iterate.

Note that the necessary optimality conditions $R^0(y) = 0$ correspond to the off-line condition $R(y) = 0$ that was defined in Eq. (5.3) in Sec. 5.1.1. The derivative approximation $J^k(y)$ is defined according to Eq. (5.25) to be

$$J^k(y) \;=\; \begin{pmatrix} Q_1^{kT} \\ 0 \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 0 & & \\ Q_2^{kT} & & \\ & \mathbb{I} & \\ & & \mathbb{I} \end{pmatrix} J(y),$$

where $J(y)$ is the derivative approximation of $R(y)$.

During the real-time iterations, each step $\Delta y^k = y^{k+1} - y^k$ is generated by an attempt to attack problem P^k, starting at the current best guess y^k:

$$\Delta y^k := -J^k(y^k)^{-1}\, R^k(y^k). \tag{5.30}$$

Theorem 5.6 (Convergence of the Real-Time Iterations)
Let us assume that the sufficent conditions for local convergence of Theorem 5.3 are satisfied for the original (off-line) problem (5.1). *Then the sequence of real-time iterates* (y^k) *according to Eq.* (5.30) *converges towards a feasible point*

$$y^* \in D_0 = \left\{ y \in \mathbb{R}^n |\; \|y - y^0\| \le \frac{\|\Delta y^0\|}{1-\delta_0} \right\} \subset D, \quad \delta_0 = \kappa + \frac{\omega}{2}\|\Delta y^0\|.$$

Remark: Though this theorem uses the term "convergence" and regards the infinite sequence y^k, it is not the behaviour for $k \to \infty$ that causes the difficulty, as from $k = N$ on we treat always the same optimization problem P^N; the difficulty lies in showing that the *first* iterates $y^1, y^2, \ldots, y^N$ remain in the set D_0.

Proof: We will follow the lines of the proof of Theorem 5.1. Note that the first step Δy^0 of the real-time iterations coincides with the first step of the off-line iterations. Therefore, to prove convergence towards a limit point $y^* = (q^*, s^*, \lambda^*)$, we only have to show that the contraction propery

$$\|\Delta y^{k+1}\| \le \left(\kappa + \frac{\omega}{2}\|\Delta y^k\|\right) \|\Delta y^k\| \tag{5.31}$$

is satisfied for the real-time iterates. In a second step, we will show that the primal part (q^*, s^*) of the limit point is feasible.

In Theorem 5.4 in Sec. 5.2.2 we have already shown that fixing of components does not increase the constants $\kappa < 1$ and ω that are used to prove the contraction property for a single optimization problem. This means that all derivative approximations J^k satisfy the bounds (5.13a) and (5.13b).

But how to compare the steps Δy^k and Δy^{k+1} that correspond to different residual functions R^k and R^{k+1}?

The trick to prove the contraction property (5.31) is to treat two subsequent steps Δy^k and Δy^{k+1} *as if they were belonging to the same optimization problem* P^{k+1} *with residual function* R^{k+1}. If this is true, Eq. (5.14) can directly be used to prove the contraction property. This trick is paradoxical because it assumes that the constraint

$$Q_1^{k+1^T}(q - u^{k+1}) = Q_1^{k+1^T}(q - q^{k+1}) = 0$$

is already defined before the iterate $y^{k+1} = (q^{k+1}, s^{k+1}, \lambda^{k+1})$ is computed, i.e., before q^{k+1} is known!

Fortunately, it can be shown that the step Δy^k is not changed, if it would have been defined by

$$\Delta y^k := -J^{k+1}(y^k)^{-1}\, R^{k+1}(y^k)$$

instead of

$$\Delta y^k := -J^k(y^k)^{-1}\, R^k(y^k)$$

as in Eq. (5.30). To see this, note that Δy^k is the unique solution of $R^k(y^k)+J^k(y^k)\Delta y^k = 0$. We will show that it also satisfies $R^{k+1}(y^k) + J^{k+1}(y^k)\Delta y^k = 0$.

$$
\begin{aligned}
& R^{k+1}(y^k) \qquad\qquad + \qquad\qquad J^{k+1}(y^k) \quad \Delta y^k \\
&= \begin{pmatrix} Q_1^{k+1^T}(q^k - q^{k+1}) \\ Q_2^{k+1^T}\nabla_q \mathcal{L} \\ \nabla_s \mathcal{L} \\ G(q,s) \end{pmatrix} + \begin{pmatrix} Q_1^{k+1^T} & & \\ Q_2^{k+1^T}A_{qq} & Q_2^{k+1^T}A_{qs}^T & Q_2^{k+1^T}\frac{\partial G}{\partial q}^T \\ A_{qs} & A_{ss} & \frac{\partial G}{\partial s}^T \\ \frac{\partial G}{\partial q} & \frac{\partial G}{\partial s} & 0 \end{pmatrix} \begin{pmatrix} q^{k+1} - q^k \\ s^{k+1} - s^k \\ \lambda^k - \lambda^{k+1} \end{pmatrix} \\
&= \begin{pmatrix} 0 & & \\ Q_2^{k+1^T} & & \\ & \mathbb{I} & \\ & & \mathbb{I} \end{pmatrix} \left(\begin{pmatrix} \nabla_q \mathcal{L} \\ \nabla_s \mathcal{L} \\ G(q,s) \end{pmatrix} + \begin{pmatrix} A_{qq} & A_{qs}^T & \frac{\partial G}{\partial q}^T \\ A_{qs} & A_{ss} & \frac{\partial G}{\partial s}^T \\ \frac{\partial G}{\partial q} & \frac{\partial G}{\partial s} & 0 \end{pmatrix} \Delta y^k \right) \\
&= \begin{pmatrix} 0 & & \\ \Pi_k^T Q_2^{k^T} & & \\ & \mathbb{I} & \\ & & \mathbb{I} \end{pmatrix} \left(\begin{pmatrix} \nabla_q \mathcal{L} \\ \nabla_s \mathcal{L} \\ G(q,s) \end{pmatrix} + \begin{pmatrix} A_{qq} & A_{qs}^T & \frac{\partial G}{\partial q}^T \\ A_{qs} & A_{ss} & \frac{\partial G}{\partial s}^T \\ \frac{\partial G}{\partial q} & \frac{\partial G}{\partial s} & 0 \end{pmatrix} \Delta y^k \right) \\
&= \begin{pmatrix} 0 & & \\ \Pi_k^T & & \\ & \mathbb{I} & \\ & & \mathbb{I} \end{pmatrix} \left(\begin{pmatrix} Q_2^{k^T}\nabla_q \mathcal{L} \\ \nabla_s \mathcal{L} \\ G(q,s) \end{pmatrix} + \begin{pmatrix} Q_2^{k^T}A_{qq} & Q_2^{k^T}A_{qs}^T & Q_2^{k^T}\frac{\partial G}{\partial q}^T \\ A_{qs} & A_{ss} & \frac{\partial G}{\partial s}^T \\ \frac{\partial G}{\partial q} & \frac{\partial G}{\partial s} & 0 \end{pmatrix} \Delta y^k \right) \\
&= \begin{pmatrix} 0 & & \\ \Pi_k^T & & \\ & \mathbb{I} & \\ & & \mathbb{I} \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \quad = \quad 0.
\end{aligned}
$$

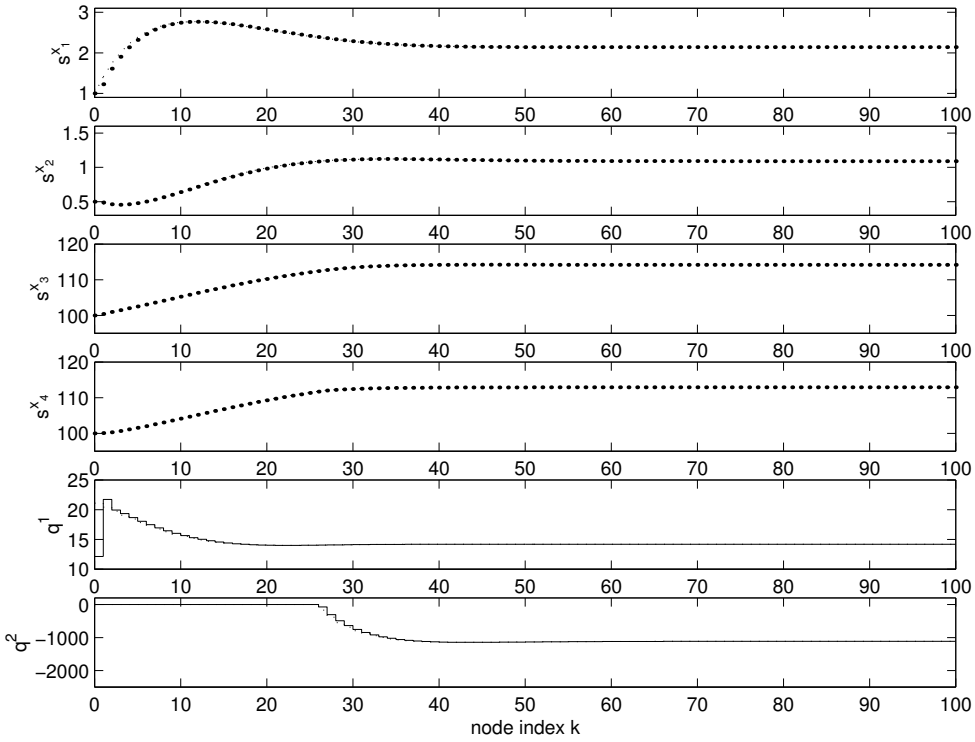


Figure 5.4: Limit point w^* of the real-time iteration approach (with Gauss-Newton iterations) in Example 5.2, which is very similar to the exact off-line solution (dotted, cf. Fig. 2.3).

Therefore, the iterations converge towards a limit point $y^* \in D_0$. To show that this point is feasible, note that at some problem P^{k_0} no more components can be fixed (to be specific, $k_0 = N$ in the real-time iterations for the multiple shooting method), so that $R^k = R^{k_0}$, $\forall\, k \geq k_0$. Therefore

$$0 = \lim_{k\to\infty} -J^k(y^k)^{-1}R^k(y^k) = \lim_{k\to\infty} -J^{k_0}(y^k)^{-1}R^{k_0}(y^k) = -J^{k_0}(y^*)^{-1}R^{k_0}(y^*),$$

so that $R^{k_0}(y^*) = 0$ which implies $G(q^*, s^*) = 0$. □

Example 5.2 (Continuous Stirred Tank Reactor)
Let us again consider Example 5.1. The limit point $w^* = (q^*, s^*)$ is shown in Fig. 5.4; a comparison of the convergence rates for the Newton and Gauss-Newton method is shown in Fig. 5.5.

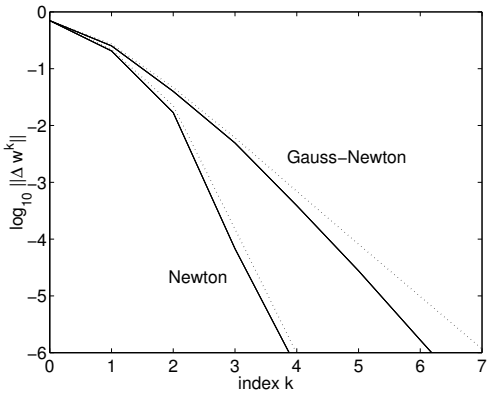


Figure 5.5: Convergence rates for Newton type methods, for the real-time iterations of Example 5.2. Comparison of Newton and Gauss-Newton method. The dotted lines indicate the convergence rates for the off-line solution (cf. Example 5.1).

5.3 Comparison of On-Line and Off-Line Solutions

We will now investigate the error that we make by using the real-time iteration scheme, compared to the exact off-line solution of P^0. We denote the off-line solution now by y_0^* to distinguish it from the limit point y^* of the real-time iterations. We will also compare y^* with the exact solutions y_k^* of the optimization problems P^k for $k \geq 1$. Note that not only the limit point y^* depends on the initial guess y^0 and the chosen Newton type method, but also the exact solutions y_k^*, because the optimization problems P^k are generated on-line.

Several results are established; first, we bound the distances $\|y^* - y_k^*\|$ in the space of KKT points $y = (q, s, \lambda)$ by the size of the first step Δy^0. Secondly, we show how the first step Δy^0 itself is bounded, if the initial guess y^0 was the solution of a neighboring optimization problem. Finally, we will investigate how much optimality is lost with respect to the objective function.

5.3.1 Distance to Optimal Solutions

Theorem 5.7 (Distance to Off-Line Solution)
If the sufficient conditions for off-line convergence of Theorem 5.3 are satisfied, the distance between the limit point y^* of the real-time iterations as defined in Eq. (5.30) *and the solution y_0^* of the off-line optimization problem P^0 can be bounded by*

$$\|y^* - y_0^*\| \leq \frac{2\delta_0 \|\Delta y^0\|}{1 - \delta_0}, \quad \delta_0 = \kappa + \frac{\omega}{2}\|\Delta y^0\|. \tag{5.32}$$

Proof: We make use of the fact that the iterates for the solution of the off-line problem as in Theorem 5.3 and the real-time iterations (y^k) coincide on y^0 and y^1 before they separate.

As they have the same contraction constants $\kappa < 1$ and ω, also δ_0 is identical for both. Using the property (5.15) from the proof of Theorem 5.1 for the Newton type iterates

$$\|y^{k+m} - y^k\| \leq \frac{1}{1-\delta_0}\|\Delta y^k\| \leq \frac{\delta_0^k}{1-\delta_0}\|\Delta y^0\|, \quad \forall\, k, m \geq 0,$$

we deduce that

$$\|y^* - y^1\| = \lim_{m\to\infty} \|y^{1+m} - y^1\| \leq \frac{\delta_0}{1-\delta_0}\|\Delta y^0\|.$$

For the off-line solution y_0^* of P^0, the same inequality holds, so that

$$\|y^* - y_0^*\| \leq \|y^* - y^1\| + \|y_0^* - y^1\| \leq \frac{2\delta_0}{1-\delta_0}\|\Delta y^0\|.$$

□

An interesting corollary of this theorem is the following:

Corollary 5.8 (Shrinking Distance to Optimal Solutions)
The distance between the limit point y^* of the real-time iterations as defined in Eq. (5.30) and the rigorous solution y_k^* of the k-th on-line optimization problem P^k can be bounded by

$$\|y^* - y_k^*\| \leq \frac{2\delta_k\|\Delta y^k\|}{1-\delta_k} \leq \frac{2\delta_0^{k+1}\|\Delta y^0\|}{1-\delta_0}.$$

This means that the limit point y^* of the real-time iterations is close to the rigorous solution of a problem P^k, if $k < N$ is chosen large enough (note that for $k \geq N$, $y_k^* = y^*$ anyway, as the problem does not change anymore). Note that the iterates converge in typical application problems much faster than the horizon shrinks (e.g. Example 5.2). This allows to conclude that the real-time iterates practically provide an optimal solution y_k^* of a problem P^k with a relatively small $k < N$, i.e., for a problem with a slightly shortened horizon only.

5.3.2 Size of First Step after Initial Value Embedding

To know more about the distance between the two points y^* and y_0^*, it is necessary to find a bound on the first step Δy^0. Let us therefore go back to the formulation for the off-line optimization problem $P^0 = P(x_0)$ that was given in Sec. 2.2, which keeps the initial value constraint separate:

$$P^0 : \min_{q,s} F(q,s) \quad \text{subject to} \quad \left\{ \begin{array}{rcl} s_0^x - x_0 & = & 0, \\ \tilde{G}(q,s) & = & 0, \end{array} \right.$$

The optimality residual vector therefore has the structure

$$R_{x_0}(y) := \begin{pmatrix} \nabla_q \mathcal{L} \\ \nabla_s \mathcal{L} \\ s_0^x - x_0 \\ \tilde{G}(q,s) \end{pmatrix}.$$

Note that the derivative $\frac{\partial R}{\partial y}$ and the derivative approximation J do *not* depend on the value of the parameter x_0. We will now establish a bound on the first step Δy^0 after the initial value embedding, if the iterations are started at an initial guess y^0 that is itself the result of a neighboring optimization problem (cf. Theorem 3.6 on the first order prediction by an exact Hessian SQP, and the initial value embedding idea in Sec. 4.2).

Lemma 5.9 (Bound on First Step)
Let us assume that $\bar{y}_0^*$ is the solution of an optimization problem $P_0(\bar{x}_0)$, and that $x_0 = \bar{x}_0 + \epsilon$. Then the first step Δy^0 of the iterations for the solution of problem $P^0 = P_0(x_0)$ when starting at $y_0 := \bar{y}_0^*$ can be bounded by

$$\|\Delta y^0\| \leq \beta \|\epsilon\|,$$

where β is defined as in Theorem 5.3.

Proof: We make use of the fact that $R_{\bar{x}_0}(y^0) = R_{\bar{x}_0}(\bar{y}_0^*) = 0$ and calculate Δy^0 directly:

$$\begin{aligned} \Delta y^0 &= -J(y^0)^{-1} R_{x_0}(y^0) \\ &= -J(y^0)^{-1} \left(R_{\bar{x}_0}(y^0) + \begin{pmatrix} 0 \\ 0 \\ -\epsilon \\ 0 \end{pmatrix} \right) = J(y^0)^{-1} \begin{pmatrix} 0 \\ 0 \\ \epsilon \\ 0 \end{pmatrix}. \end{aligned}$$

The proof is completed by using $\|J(y^0)^{-1}\| \leq \beta$ as shown in the proof of Theorem 5.3. □

As an immediate consequence of this lemma and of Theorem 5.7, we obtain the following:

Corollary 5.10 (Distance after Initial Disturbance)
The distance between the rigorous solution y_0^* of the optimization problem $P_0(x_0)$ and the limit point y^* of the real-time iterations, when started at the solution $\bar{y}_0^*$ of a neighboring optimization problem $P_0(\bar{x}_0)$, is – for a general Newton type method – of first order in the size of the disturbance $\epsilon = x_0 - \bar{x}_0$

$$\|y^* - y_0^*\| \leq 2 \frac{\kappa + \frac{\omega}{2}\beta\|\epsilon\|}{1 - (\kappa + \frac{\omega}{2}\beta\|\epsilon\|)} \beta \|\epsilon\|,$$

and – for an exact Newton method – of second order in the size of the disturbance

$$\|y^* - y_0^*\| \leq \frac{\omega}{1 - \frac{\omega}{2}\beta\|\epsilon\|} \beta^2 \|\epsilon\|^2.$$

5.3.3 Bounds on the Loss of Optimality

Now that we know how far the limit point y^* is from the optimal solution, we can also investigate how much optimality is lost, in terms of the objective function $F(q^*, s^*)$.

Theorem 5.11 (Loss of Optimality)
Let us assume that the sufficient conditions for off-line convergence of Theorem 5.3 are satisfied. Let us also assume that the exact derivative matrix $\frac{\partial R}{\partial y}$ is bounded on D_0:

$$\left\| \frac{\partial R}{\partial y}(y) \right\| \leq B_R, \quad \forall y \in D_0.$$

Denoting the limit point of the real-time iterations by $y^* = (q^*, s^*, \lambda^*)$, and the optimal off-line solution by $y_0^* = (q_0^*, s_0^*, \lambda_0^*)$, the loss of optimality can be bounded by

$$F(q^*, s^*) - F(q_0^*, s_0^*) \leq \frac{1}{2} B_R \|y^* - y_0^*\|^2. \tag{5.33}$$

Proof: First note that not only the point (q_0^*, s_0^*), but also the point (q^*, s^*) is feasible according to Theorem 5.6, i.e., $G(q_0^*, s_0^*) = G(q^*, s^*) = 0$. Therefore, we can compare the values of the Lagrangian function $\mathcal{L}(q, s, \lambda) = F(q, s) - \lambda^T G(q, s)$ that coincide with the objective in both points.

$$\begin{aligned}
\mathcal{L}(y^*) - \mathcal{L}(y_0^*) &= \int_0^1 \tfrac{\partial \mathcal{L}}{\partial y}(y_0^* + t_1(y^* - y_0^*))\,(y^* - y_0^*)\,dt_1 \\
&= \int_0^1 R(y_0^* + t_1(y^* - y_0^*))^T\,(y^* - y_0^*)\,dt_1 \\
&= \int_0^1 \Big(\int_0^{t_1} \tfrac{\partial R}{\partial y}(y_0^* + t_2(y^* - y_0^*))(y^* - y_0^*)\,dt_2 \Big)^T (y^* - y_0^*)\,dt_1 \\
&= (y^* - y_0^*)^T \Big(\int_0^1 \int_0^{t_1} \tfrac{\partial R}{\partial y}(y_0^* + t_2(y^* - y_0^*))\,dt_2\,dt_1 \Big)^T (y^* - y_0^*)
\end{aligned}$$

where we have used the fact that $R(y_0^*) = 0$. We conclude that

$$\|\mathcal{L}(y^*) - \mathcal{L}(y_0^*)\| \leq \frac{1}{2} B_r \|y^* - y_0^*\|^2.$$

□

This theorem together with Corollary 5.10 implies the following:

Corollary 5.12 (Loss of Optimality after Initial Disturbance)
The loss of optimality due to the real-time iterations for the approximate solution of $P_0(x_0)$ is of second order in the size of an initial disturbance ϵ as in Corollary 5.10 for a general Newton type method:

$$F(q^*, s^*) - F(q_0^*, s_0^*) \leq 2B_r \left(\frac{\kappa + \frac{\omega}{2}\beta\|\epsilon\|}{1 - (\kappa + \frac{\omega}{2}\beta\|\epsilon\|)} \beta \right)^2 \|\epsilon\|^2, \tag{5.34}$$

and – for an exact Newton method – of fourth order in the size of the disturbance:

$$F(q^*, s^*) - F(q_0^*, s_0^*) \leq \frac{B_r \omega^2 \beta^4}{2\left(1 - \frac{\omega}{2}\beta\|\epsilon\|\right)^2} \|\epsilon\|^4. \tag{5.35}$$

Chapter 6

A Close Look at one Real-Time Iteration

In this chapter we describe in detail what computations are necessary to perform one real-time iteration, and we show how these computations can be performed efficiently. Starting with the current iterate of the variables (w, λ, μ), we describe how to finally arrive at the solution $(\Delta w, \tilde{\lambda}, \tilde{\mu})$ of the QP (3.10), that allows to generate the next iterate. Though most parts of the algorithm are well known, we present all details here, to be able to show what is meant by the separation into preparation and feedback phase, which is important for the real-time iterations. The feedback phase comprises only a small fraction of the overall computations, which can be found in Subsections 6.5.2 and 6.5.2 for two alternative QP solution approaches.

We will start the chapter by briefly investigating the structure of the nonlinear programming problem in Sec. 6.1, and show how this structure leads to a favourable structure of the QP that has to be generated and solved in each cycle. In our approach, *QP generation* and *QP solution* are intertwined, so that we cannot clearly separate these two steps. In Sec. 6.2 we show that only a so called *partially reduced* QP has to be generated if some solution steps are performed in advance, and in Sec. 6.3 we will explain how the remaining sensitivities can be computed efficiently. We closely follow the lines of Leineweber [Lei99], who developed the employed partial reduction strategy.

An newly developed Gauss-Newton approach to obtain an excellent approximation of the Hessian in the presence of integral least squares terms is presented in Sec. 6.4.

We present two alternative approaches to solve the partially reduced QP: in Sec. 6.5 we describe the so called condensing technique, which we actually used for the presented numerical examples, and which condenses the large, but structured QP into a small, but unstructured QP, which is then solved by standard techniques. The alternative approach presented in Sec. 6.6 does directly attack the large structured QP by a dynamic programming approach that leads to a Riccati recursion. Both methods allow to perform the most expensive steps before the actual value of x_0 is known, thus allowing to prepare an "immediate feedback". Finally, we give a summary of the necessary computation steps

per real-time iteration, and show that the algorithm can be interpreted as a successive generation of approximated optimal feedback control laws.

6.1 Problem Structure

An important feature of the direct multiple shooting method is the sparse structure of the large scale NLP (2.10). Its Lagrangian function $\mathcal{L}(w, \lambda, \mu)$ can be written as

$$\begin{aligned}\mathcal{L}(w,\lambda,\mu) \;:=\; & \textstyle\sum_{i=0}^{N-1} L_i(s_i^x, s_i^z, q_i) + E(s_N^x, s_N^z) \\ & - \textstyle\sum_{i=0}^{N-1} {\lambda_{i+1}^x}^T (x_i(\tau_{i+1}) - s_{i+1}^x) \\ & - \textstyle\sum_{i=0}^{N} \lambda_i^{zT} g(s_i^x, s_i^z, q_i) \quad - \lambda_0^{xT}(x_0 - s_0^x) - \lambda_r^T r^{\mathrm{e}}(s_N^x, s_N^z) \\ & - \mu_r^T r^{\mathrm{i}}(s_N^x, s_N^z) - \textstyle\sum_{i=0}^{N} \mu_i^T h(s_i^x, s_i^z, q_i),\end{aligned}$$

with $\lambda = (\lambda_0^x, \ldots, \lambda_N^x, \lambda_0^z, \ldots, \lambda_N^z, \lambda_r)$ and $\mu = (\mu_r, \mu_0, \ldots, \mu_N)$. This Lagrangian function is *partially separable*: Let us reorder the vector $w = (w_0, \ldots, w_N)$ with $w_i = (s_i^x, s_i^z, q_i)$.[1] Then it can be seen that the Hessian matrix $\nabla_w^2 \mathcal{L}$ is block diagonal with non-zero blocks A_i that correspond each to the variables w_i only (Bock and Plitt, [BP84]), i.e.

$$\nabla_w^2 \mathcal{L} = \begin{pmatrix} A_0 & & & \\ & \ddots & & \\ & & A_{N-1} & \\ & & & A_N \end{pmatrix}.$$

The unreduced QP that could be formulated at a current iterate w looks as follows:

$$\begin{aligned}\min_{\Delta w_0, \ldots, \Delta w_N} \quad & \tfrac{1}{2} \textstyle\sum_{i=0}^{N} \Delta w_i^T A_i \Delta w_i + \sum_{i=0}^{N-1} \nabla_{w_i} L_i(s_i^x, s_i^z, q_i)^T \Delta w_i \\ & + \nabla_{(s_N^x, s_N^z)} E(s_N^x, s_N^z)^T \Delta(s_N^x, s_N^z)\end{aligned} \tag{6.1a}$$

subject to

$$s_{i+1}^x - x_i(\tau_{i+1}) + \Delta s_{i+1}^x - \frac{\partial x_i(\tau_{i+1})}{\partial w_i} \Delta w_i = 0, \quad i = 0, \ldots, N-1, \tag{6.1b}$$

$$g(s_i^x, s_i^z, q_i) + \frac{\partial g}{\partial s_i^x} \Delta s_i^x + \frac{\partial g}{\partial s_i^z} \Delta s_i^z + \frac{\partial g}{\partial q_i} \Delta q_i = 0, \quad i = 0, \ldots, N, \tag{6.1c}$$

$$s_0^x - x_0 + \Delta s_0^x = 0, \tag{6.1d}$$

$$r^{\mathrm{e}}(s_N^x, s_N^z) + \frac{\partial r^{\mathrm{e}}}{\partial (s_N^x, s_N^z)} \Delta(s_N^x, s_N^z) = 0, \tag{6.1e}$$

$$r^{\mathrm{i}}(s_N^x, s_N^z) + \frac{\partial r^{\mathrm{i}}}{\partial (s_N^x, s_N^z)} \Delta(s_N^x, s_N^z) \geq 0, \tag{6.1f}$$

$$h(s_i^x, s_i^z, q_i) + \frac{\partial h}{\partial w_i} \Delta w_i \geq 0, \quad i = 0, \ldots, N. \tag{6.1g}$$

$$\Delta q_N - \Delta q_{N-1} = 0 \tag{6.1h}$$

[1] We recall here that $q_N := q_{N-1}$ is only introduced for notational convenience and has to be eliminated again. Due to the linearity of the constraint $q_N = q_{N-1}$ it does not affect the Hessian matrix.

It is a crucial feature of our algorithm that this QP is *never* generated directly. Instead, following the partial reduction approach developed by Leineweber [Lei99], first only the linearized consistency conditions (6.1c) are generated that allow to eliminate Δs_i^z from the QP, as will be described in the following section.

6.2 The Partial Reduction Technique

The partial reduction approach starts as follows: once the linearized consistency conditions (6.1c)

$$g(s_i^x, s_i^z, q_i) + \left(\frac{\partial g}{\partial s_i^x} \middle| \frac{\partial g}{\partial s_i^z} \middle| \frac{\partial g}{\partial q_i}\right) \Delta w_i = 0, \quad i = 0, \ldots, N,$$

are generated, the (usually sparse) systems

$$\left(\frac{\partial g}{\partial s_i^z}\right) \left(\begin{array}{c|c|c} d_i^z & D_i^{s^x} & D_i^q \end{array} \right) = - \left(g(s_i^x, s_i^z, q_i) \middle| \frac{\partial g}{\partial s_i^x} \middle| \frac{\partial g}{\partial q_i}\right), \quad i = 0, \ldots, N$$

are resolved.[2] The matrix $\left(\frac{\partial g}{\partial s_i^z}\right)$ is always invertible due to the index-one assumption for the DAE system. The solution $\left(\begin{array}{c|c|c} d_i^z & D_i^{s^x} & D_i^q \end{array} \right)$ of this system allows to construct the vector and matrix

$$d_i := \begin{pmatrix} 0 \\ d_i^z \\ 0 \end{pmatrix} \quad \text{and} \quad D_i := \begin{pmatrix} \mathbb{I} & 0 \\ D_i^{s^x} & D_i^q \\ 0 & \mathbb{I} \end{pmatrix},$$

that are called the *range space* and *null space* component of the linearized consistency conditions, because

$$\frac{\partial g(s_i^x, s_i^z, q_i)}{\partial (s_i^x, s_i^z, q_i)} d_i = -g(s_i^x, s_i^z, q_i) \quad \text{and} \quad \frac{\partial g(s_i^x, s_i^z, q_i)}{\partial (s_i^x, s_i^z, q_i)} D_i = 0.$$

It is straightforward to see that

$$\Delta w_i := d_i + D_i \begin{pmatrix} \Delta s_i^x \\ \Delta q_i \end{pmatrix} \tag{6.2}$$

satisfies the linearized consistency conditions (6.1c) for arbitrary values of Δs_i^x and Δq_i. It is therefore possible to formulate an equivalent, *reduced* QP, where the variables Δs_i^z are completely eliminated. For this aim let us define

$$\begin{pmatrix} Q_i & S_i^T \\ S_i & R_i \end{pmatrix} := D_i^T A_i D_i, \quad i = 0, \ldots, N,$$

$$\begin{pmatrix} g_i^x \\ g_i^q \end{pmatrix} := D_i^T \nabla_{w_i} L_i + D_i^T A_i d_i, \quad i = 0, \ldots, N-1, \tag{6.3a}$$

$$\begin{pmatrix} g_N^x \\ g_N^q \end{pmatrix} := D_N^T \nabla_{w_N} E(s_N^x, s_N^z) + D_N^T A_N d_N, \tag{6.3b}$$

[2] We employ an advanced direct sparse solver, the Harwell subroutine MA48 by Duff and Reid [DR96].

for the reduced objective, as well as

$$c_{i+1} := s^x_{i+1} - x_i(\tau_{i+1}) - \frac{\partial x_i(\tau_{i+1})}{\partial s^z_i} d^z_i, \quad (X_i|Y_i) := \frac{\partial x_i(\tau_{i+1})}{\partial w_i} D_i, \tag{6.4}$$
$$h_i := h(s^x_i, s^z_i, q_i) + \frac{\partial h}{\partial s^z_i} d^z_i, \quad (H^x_i|H^q_i) := \frac{\partial h}{\partial w_i} D_i, \quad i = 0, \dots, N-1,$$

and

$$r^{\mathrm{e}} := r^{\mathrm{e}}(s^x_N, s^z_N) + \frac{\partial r^{\mathrm{e}}}{\partial s^z_N} d^z_N, \quad (R^{\mathrm{e},x}|R^{\mathrm{e},q}) := \frac{\partial r^{\mathrm{e}}}{\partial w_N} D_N,$$
$$r^{\mathrm{i}} := r^{\mathrm{i}}(s^x_N, s^z_N) + \frac{\partial r^{\mathrm{i}}}{\partial s^z_N} d^z_N, \quad (R^{\mathrm{i},x}|R^{\mathrm{i},q}) := \frac{\partial r^{\mathrm{i}}}{\partial w_N} D_N,$$

for the constraints, so that we can formulate the following reduced QP that is equivalent to (6.1)

$$\min_{\substack{\Delta s^x_0, \dots, \Delta s^x_N \\ \Delta q_0, \dots, \Delta q_N}} \quad \sum_{i=0}^{N} \{ \tfrac{1}{2} \Delta s^{xT}_i Q_i \Delta s^x_i + \tfrac{1}{2} \Delta q^T_i R_i \Delta q_i + \Delta q^T_i S_i \Delta s^x_i + g^{xT}_i \Delta s^x_i + g^{qT}_i \Delta q_i \} \tag{6.5a}$$

subject to

$$c_{i+1} + \Delta s^x_{i+1} - X_i \Delta s^x_i - Y_i \Delta q_i = 0, \quad i = 0, \dots, N-1, \tag{6.5b}$$
$$s^x_0 - x_0 + \Delta s^x_0 = 0, \tag{6.5c}$$
$$r^{\mathrm{e}} + R^{\mathrm{e},x} \Delta s^x_N + R^{\mathrm{e},q} \Delta q_N = 0, \tag{6.5d}$$
$$r^{\mathrm{i}} + R^{\mathrm{i},x} \Delta s^x_N + R^{\mathrm{i},q} \Delta q_N \geq 0, \tag{6.5e}$$
$$h_i + H^x_i \Delta s^x_i + H^q_i \Delta q_i \geq 0, \quad i = 0, \dots, N, \tag{6.5f}$$
$$\Delta q_N - \Delta q_{N-1} = 0. \tag{6.5g}$$

In partial reduction approaches the full space Hessian blocks A_i are never computed. Therefore, the terms $D_i^T A_i d_i$ are usually dropped in the definitions (6.3), causing only a minor change, as d_i are proportional to $g(s^x_i, s^z_i, q_i)$, which are expected to be close to zero near a solution. However, in Sec. 6.4 we present a newly developed approach to compute efficiently approximations of both, the reduced Hessian $D_i^T A_i D_i$ and the gradient contribution $D_i^T A_i d_i$, that is based on a Gauss-Newton approach for least squares integrals. But let us first describe how the linearized continuity conditions (6.5b) of the partially reduced QP can be generated efficiently.

6.3 Efficient Sensitivity Computation

On each multiple shooting interval $[\tau_i, \tau_{i+1}]$, the relaxed initial value problems (2.3)-(2.5)

$$\begin{aligned} B(\cdot)\cdot\dot{x}_i(\tau) &= T\, f(x_i(\tau), z_i(\tau), q_i) \\ 0 &= g(x_i(\tau), z_i(\tau), q_i) - \exp\left(-\beta\frac{\tau-\tau_i}{\tau_{i+1}\ -\ \tau_i}\right) g(s_i^x, s_i^z, q_i) \\ x_i(\tau_i) &= s_i^x \end{aligned}$$

have to be solved to yield the solution trajectories $x_i(\tau)$ and $z_i(\tau)$. These trajectories depend on the initial values s_i^x, s_i^z of differential and algebraic states, and on the control parameters q_i. In a naive implementation, we would also have to compute the derivatives of the final value $x_i(\tau_{i+1})$ with respect to these quantities. As mentioned above, a crucial feature of Leineweber's partial reduction approach to multiple shooting for DAE [Lei99] is that the full derivative matrices

$$\frac{\partial x_i(\tau_{i+1})}{\partial(s_i^x, s_i^z, q_i)}$$

are never calculated, but instead directly the *directional derivatives*

$$\left(\ k_i \ \middle|\ X_i \ \middle|\ Y_i\ \right) := \frac{\partial x_i(\tau_{i+1})}{\partial(s_i^x, s_i^z, q_i)} \left(\begin{array}{c|c|c} 0 & \mathbb{I} & 0 \\ d_i^z & D_i^{s^x} & D_i^q \\ 0 & 0 & \mathbb{I} \end{array}\right)$$

that are actually needed to formulate the partially reduced QP (6.5).[3] This saves a considerable amount of computational effort for problems with a large share of algebraic variables. Before we describe this approach in detail, a remark is in order about how to generally approach the problem of computing derivatives of a DAE solver output.

Remark on External and Internal Numerical Differentiation

One straightforward approach that is simple to implement is to start an existing DAE solver several times with perturbed initial values and control parameters, and to subtract the perturbed outputs $x_i(\tau_{i+1})$ to compute an approximation of the desired matrix $(k_i|X_i|Y_i)$ by finite-differences (see e.g. Rosen and Luus [RL91]). This approach, which may be called *External Numerical Differentiation* (END), has serious drawbacks, as the output of a modern, adaptive DAE solver is usually a *discontinuous* function of the initial values and control parameters. If the inputs for the DAE solver are varied continuously, the output $x_i(\tau_{i+1})$ usually jumps discontinuously, with jumps that have to be expected to be as big as the integrator tolerance permits (see e.g. Gear and Vu [GV83]). If the perturbed trajectories are chosen close to each other, as it is required in finite-difference schemes to yield a good approximation of the derivative, these discontinuities can outweigh the desired

[3] The vectors $k_i = \frac{\partial x_i(\tau_{i+1})}{\partial s_i^z} d_i^z$ are needed to generate c_{i+1} according to Eq. (6.4).

derivative information, if the integrator accuracy is not chosen extraordinarily high; if such an accuracy is feasible at all, this will cause excessive computation times.

An approach which avoids the drawbacks of END is the so called *Internal Numerical Differentiation* (IND) as described by Bock [Boc81]. The idea is to freeze the discretization scheme for the neighboring trajectories, so that the output becomes a differentiable function of the inputs. This allows to perform the DAE solution even with low accuracy, without jeopardizing the accuracy of the derivative approximation. The frozen discretization scheme is usually adapted to the nominal trajectory.

In addition, much effort can be saved if the perturbed trajectories are computed simultaneously, as many matrix evaluations and factorizations then need to be performed only once for all trajectories.

A related approach that may be interpreted as the "analytical limit of IND" [Boc83] is to solve the sensitivity equations along the nominal system trajectory. We will briefly describe how this approach can be used to compute directional derivatives.

6.3.1 Directional Derivatives

Let us for notational convenience assume that the DAE is explicit, i.e., that $B(\cdot) = \mathbb{I}$, and let us also assume that $T = 1$, so that the initial value problem (2.3)-(2.5) can be written as

$$\begin{aligned} \dot{x}_i(\tau) &= f(x_i(\tau), z_i(\tau), q_i), \\ 0 &= g(x_i(\tau), z_i(\tau), q_i) - e^{-\beta \frac{\tau - \tau_i}{\tau_{i+1} - \tau_i}} g(s_i^x, s_i^z, q_i), \\ x_i(\tau_i) &= s_i^x. \end{aligned}$$

Differentiation of this system with respect to the initial values and control parameters (s_i^x, s_i^z, q_i) and a multiplication from the right by the matrix $(d_i | D_i)$ yields a linear matrix DAE. Defining the matrix functions

$$\left(\begin{array}{c|c|c} k_i(\tau) & X_i(\tau) & Y_i(\tau) \end{array} \right) := \frac{\partial x_i(\tau)}{\partial (s_i^x, s_i^z, q_i)} \left(\begin{array}{c|c} d_i & D_i \end{array} \right),$$

$$\left(\begin{array}{c|c|c} k_i^z(\tau) & X_i^z(\tau) & Y_i^z(\tau) \end{array} \right) := \frac{\partial z_i(\tau)}{\partial (s_i^x, s_i^z, q_i)} \left(\begin{array}{c|c} d_i & D_i \end{array} \right),$$

this matrix DAE can be written as

$$\begin{aligned} \frac{d}{d\tau} \left(k_i(\tau) \middle| X_i(\tau) \middle| Y_i(\tau) \right) &= \frac{\partial f(\cdot)}{\partial (x, z, u)} \left(\begin{array}{c|c|c} k_i(\tau) & X_i(\tau) & Y_i(\tau) \\ k_i^z(\tau) & X_i^z(\tau) & Y_i^z(\tau) \\ 0 & 0 & \mathbb{I} \end{array} \right), \\ 0 &= \frac{\partial g(\cdot)}{\partial (x, z, u)} \left(\begin{array}{c|c|c} k_i(\tau) & X_i(\tau) & Y_i(\tau) \\ k_i^z(\tau) & X_i^z(\tau) & Y_i^z(\tau) \\ 0 & 0 & \mathbb{I} \end{array} \right) \\ &\quad - e^{-\beta \frac{\tau - \tau_i}{\tau_{i+1} - \tau_i}} \left(\begin{array}{c|c|c} -g(s_i^x, s_i^z, q_i) & 0 & 0 \end{array} \right), \\ \left(\begin{array}{c|c|c} k_i(\tau_i) & X_i(\tau_i) & Y_i(\tau_i) \end{array} \right) &= \left(\begin{array}{c|c|c} 0 & \mathbb{I} & 0 \end{array} \right). \end{aligned}$$

The consistent initial value for the algebraic matrix $\left(\; k_i^z \;\middle|\; X_i^z \;\middle|\; Y_i^z \; \right)$ is

$$\left(\; k_i^z(\tau_i) \;\middle|\; X_i^z(\tau_i) \;\middle|\; Y_i^z(\tau_i) \; \right) = \left(\; d_i^z \;\middle|\; D_i^{s^x} \;\middle|\; D_i^q \; \right).$$

This linear matrix DAE can be solved simultaneously with the original initial value problem (2.3)-(2.5), as it is done e.g. in the version of the advanced BDF integrator DAESOL (Bauer [Bau00]) that we used for most computations that are presented in this thesis. The final values are then used to define

$$\left(\; k_i \;\middle|\; X_i \;\middle|\; Y_i \; \right) := \left(\; k_i(\tau_{i+1}) \;\middle|\; X_i(\tau_{i+1}) \;\middle|\; Y_i(\tau_{i+1}) \; \right).$$

Computation of the Reduced Objective Gradient

We have so far not discussed how to compute the reduced objective gradients $D_i^T \nabla_{w_i} L_i$ that are needed to compute

$$g_i := \begin{pmatrix} g_i^x \\ g_i^q \end{pmatrix}$$

in (6.3a), i.e., how to compute the directional derivatives of the objective integrals

$$L_i(s_i^x, s_i^z, q_i) = \int_{\tau_i}^{\tau_{i+1}} L(x_i(\tau), z_i(\tau), q_i))\, d\tau.$$

The objective integrals can be computed by introducing an additional differential state x^{L}, and solving the following initial value problem together with the original initial value problem:

$$\begin{aligned} \dot{x}_i^{\mathrm{L}}(\tau) &= L(x_i(\tau), z_i(\tau), q_i)), \quad \text{for} \quad \tau \in [\tau_i, \tau_{i+1}], \\ \dot{x}_i^{\mathrm{L}}(\tau_i) &= 0. \end{aligned}$$

The directional derivatives can then be computed as above.

Numerical Calculation of the Exact Hessian Matrix

Leineweber [Lei99] has developed a scheme to compute a finite-difference approximation of the exact Hessian matrix blocks A_i, which is so far only applicable to systems described by ordinary differential equations (ODE). His approach generalizes the idea of Internal Numerical Differentiation to second order derivatives, by solving the first order sensitivity equations several times for perturbed initial values, with a fixed discretization scheme. We have employed this method in some examples for comparison with our newly developed Gauss-Newton approach that is described in the following section.

6.4 A Gauss-Newton Method for Integral Least Squares Terms

In the case of a Lagrange term L that has least squares form, i.e., if

$$L(x_i(\tau), z_i(\tau), q_i) = \|l(x_i(\tau), z_i(\tau), q_i)\|_2^2$$

with a vector valued function $l(\cdot)$, there exists a possibility to obtain a cheap approximation of the Hessian blocks A_i by an extension of the Gauss-Newton approach to least squares integrals. This approximation is good if the residual $l(\cdot)$ and if the multipliers λ, and μ are close to zero (cf. the discussion in Sec. 5.1.2).

To derive an expression for the Gauss-Newton approximation of the full Hessian let us neglect the constraint contributions and regard only the objective contribution of the Hessian that is

$$\nabla^2_{(s_i^x, s_i^z, q_i)} \int_{\tau_i}^{\tau_{i+1}} \|l(x_i(\tau), z_i(\tau), q_i)\|_2^2 \, d\tau.$$

A Gauss-Newton approximation of the Hessian can be obtained by differentiating twice under the integral and dropping terms that contain $l(x_i(\tau), z_i(\tau), q_i)$:

$$A_i := 2 \int_{\tau_i}^{\tau_{i+1}} J_i(\tau)^T J_i(\tau) \, d\tau, \tag{6.7}$$

where

$$J_i(\tau) := \left(\frac{\partial l(x_i(\tau), z_i(\tau), q_i)}{\partial (x, z, u)} \right) \left(\begin{array}{c|c|c} \frac{\partial x_i(\tau)}{\partial s_i^x} & \frac{\partial x_i(\tau)}{\partial s_i^z} & \frac{\partial x_i(\tau)}{\partial q_i} \\ \frac{\partial z_i(\tau)}{\partial s_i^x} & \frac{\partial z_i(\tau)}{\partial s_i^z} & \frac{\partial z_i(\tau)}{\partial q_i} \\ 0 & 0 & \mathbb{I} \end{array} \right).$$

6.4.1 A Partially Reduced Hessian Approximation

If we are interested only in the Gauss-Newton approximation $D_i^T A_i D_i$ of the reduced Hessian , we can multiply Eq. (6.7) from the left and the right with D_i^T and D_i:

$$D_i^T A_i D_i = 2 \int_{\tau_i}^{\tau_{i+1}} D_i^T J_i(\tau)^T J_i(\tau) D_i \, d\tau.$$

Fortunately the matrix products $J_i(\tau) D_i$ are cheaply available, if directional derivatives are calculated as described in the previous section. Using the notation of that section, $J_i(\tau) D_i$ can be seen to have the simple form

$$J_i(\tau) D_i = \tilde{J}_i(\tau) := \left(\frac{\partial l(x_i(\tau), z_i(\tau), q_i)}{\partial (x, z, u)} \right) \left(\begin{array}{c|c} X_i(\tau) & Y_i(\tau) \\ X_i^z(\tau) & Y_i^z(\tau) \\ 0 & \mathbb{I} \end{array} \right).$$

The partially reduced objective gradient

$$g_i = 2D_i^T \left(\nabla_{(s_i^x, s_i^z, q_i)} \int_{\tau_i}^{\tau_{i+1}} \|l(x_i(\tau), z_i(\tau), q_i))\|_2^2 \, d\tau \right) + D_i^T A_i d_i.$$

as defined in Eq. (6.3a) can also be calculated exactly, without ever computing the full Hessian approximation. For the exact computation of the reduced obejctive gradient (6.3) we also need the terms $D_i^T A_i d_i$. A multiplication of Eq. (6.7) from the left and the right with D_i^T and d_i yields

$$D_i^T A_i d_i = 2 \int_{\tau_i}^{\tau_{i+1}} \tilde{J}_i(\tau)^T \left(\frac{\partial l(x_i(\tau), z_i(\tau), q_i)}{\partial (x, z)} \right) \begin{pmatrix} k_i(\tau) \\ k_i^z(\tau) \end{pmatrix} \, d\tau.$$

so that

$$g_i := 2 \int_{\tau_i}^{\tau_{i+1}} \tilde{J}_i(\tau)^T \left(l(x_i(\tau), z_i(\tau), q_i) + \left(\frac{\partial l(x_i(\tau), z_i(\tau), q_i)}{\partial (x, z)} \right) \begin{pmatrix} k_i(\tau) \\ k_i^z(\tau) \end{pmatrix} \right) \, d\tau.$$

The matrix $\tilde{J}_i(\tau)$ can be computed simultaneously with the DAE solution. The integral can be calculated by using a suitable integration formula. Note that the evaluation of the integrand is very cheap compared to the computations necessary for the DAE solution. Furthermore, if an interpolation of the sensitivity matrices is employed in the DAE solver, the integrand can be evaluated at arbitrary points on the interval, *without the necessity to stop the integration routine* (cf. Bock and Schlöder [BS81]); these evaluation points are in particular independent of the stepsizes of the DAE solver.

We have implemented this extension of the Gauss-Newton method, which delivers the Hessian approximation at virtually no additional costs, in the current version of the optimal control package MUSCOD-II, in conjunction with the implicit DAE solver DAESOL [BBS99, Bau00].

Remark: In previous Gauss-Newton approaches to NMPC, only least squares terms at discrete time points had been formulated (cf. de Oliveira and Biegler [OB95b] for the sequential approach, and Santos et al. [SOB95] for the direct multiple shooting method), which leads to an unnecessary overhead especially on long prediction intervals with constant controls.

6.5 QP Solution by a Condensing Approach

After we have discussed how the partially reduced QP (6.5) can be generated, we will in this and the following section present two alternative strategies to solve such a QP.

The so called *condensing* approach reduces the QP further to yield a smaller QP in the variables $\Delta q_0, \ldots, \Delta q_{N-1}$ only. In the real-time context, the algorithm proceeds in two steps: first, it uses the linearized continuity conditions (6.5b) to eliminate $\Delta s_1^x, \ldots \Delta s_N^x$ from the QP (6.5). We will also eliminate Δq_N using (6.5g). The resulting QP is called the

condensed QP. In a second step, the initial value constraint (6.5c) will be used to eliminate Δs_0^x, so that a fully reduced QP in the variables $\Delta q := (\Delta q_0, \ldots, q_{N-1})$ only needs to be solved by a standard QP solver. Finally, the solution of the fully reduced QP is expanded to yield the solution in variable and multiplier space of the partially reduced QP.

6.5.1 First Condensing Step

For the first condensing step, let us reorder the variables of the partially reduced QP and summarize them into a partitioned vector

$$\begin{pmatrix} \Delta w_1 \\ \Delta w_2 \end{pmatrix}, \quad \text{with} \quad \Delta w_1 := \begin{pmatrix} \Delta s_1^x \\ \vdots \\ \Delta s_N^x \\ \Delta q_N \end{pmatrix}, \quad \text{and} \quad \Delta w_2 := \begin{pmatrix} \Delta s_0^x \\ \Delta q_0 \\ \vdots \\ \Delta q_{N-1} \end{pmatrix}.$$

By introducing

$$b_1 := \begin{pmatrix} c_1 \\ c_2 \\ c_3 \\ \vdots \\ c_N \\ 0 \end{pmatrix}, \quad B_{11} := \begin{pmatrix} \mathbb{I} & & & & & \\ -X_1 & \mathbb{I} & & & & \\ & -X_2 & \mathbb{I} & & & \\ & & \ddots & \ddots & & \\ & & & -X_{N-1} & \mathbb{I} & \\ & & & & & \mathbb{I} \end{pmatrix},$$

and

$$B_{12} := \begin{pmatrix} -X_0 & -Y_0 & & & & \\ & & -Y_1 & & & \\ & & & -Y_2 & & \\ & & & & \ddots & \\ & & & & & -Y_{N-1} \\ & & & & & -\mathbb{I} \end{pmatrix},$$

the continuity conditions (6.5b) and (6.5f) can be written as

$$b_1 + B_{11}\Delta w_1 + B_{12}\Delta w_2 = 0,$$

and the uncondensed QP (6.5) can be summarized as

$$\begin{aligned} \min_{\Delta w_1, \Delta w_2} \quad & \tfrac{1}{2}\Delta w_1^T A_{11}\Delta w_1 \;+\; \Delta w_1^T A_{12}\Delta w_2 \\ & + \tfrac{1}{2}\Delta w_2^T A_{22}\Delta w_2 \;+\; a_1^T\Delta w_1 \;+\; a_2^T\Delta w_2 \end{aligned} \tag{6.8}$$

subject to

$$\begin{aligned} b_1 + B_{11}\Delta w_1 + B_{12}\Delta w_2 &= 0 \\ b_2 + B_{21}\Delta w_1 + B_{22}\Delta w_2 &= 0 \\ c + C_1\Delta w_1 + C_2\Delta w_2 &\geq 0. \end{aligned}$$

The idea of the condensing approach is to exploit the invertibility of B_{11} to eliminate Δw_1 by

$$\Delta w_1 = -B_{11}^{-1}(B_{12}\Delta w_2 + b_1) =: M\Delta w_2 + m \tag{6.9}$$

and to replace the above QP by a so called *condensed QP*:

$$\min_{\Delta w_2} \frac{1}{2}\Delta w_2^T \tilde{A}\Delta w_2 + \tilde{a}^T\Delta w_2 \qquad \text{s.t.} \quad \begin{cases} \tilde{b} + \tilde{B}\Delta w_2 &= 0 \\ \tilde{c} + \tilde{C}\Delta w_2 &\geq 0 \end{cases} \tag{6.10a}$$

with

$$\begin{aligned} \tilde{A} &= M^T A_{11} M + M^T A_{12} + A_{12}^T M + A_{22}, \\ \tilde{a} &= M^T A_{11} m + A_{12}^T m + M^T a_1 + a_2, \\ \tilde{b} &= b_2 + B_{21} m, \\ \tilde{B} &= B_{21} M + B_{22}, \\ \tilde{c} &= c + C_1 m, \qquad \text{and} \\ \tilde{C} &= C_1 M + C_2. \end{aligned}$$

The generation of the condensed QP can be achieved efficiently by recursive techniques that have been introduced by Bock and Plitt [Pli81, BP84]. They are described in Appendix E.

6.5.2 Second Condensing Step and Immediate Feedback

In the real-time context it is important to note that all computations of the first condensing step can be performed before the actual value of x_0 is known, allowing to prepare an "Immediate Feedback". So let us have a close look at the condensed QP (6.10a). Since $\Delta w_2 = (\Delta s_0^x, \Delta q)$, it has the following structure

$$\begin{aligned}\min_{\Delta s_0^x, \Delta q} \quad & \tfrac{1}{2}\Delta s_0^{xT}\tilde{A}_{\mathrm{ss}}\Delta s_0^x + \tfrac{1}{2}\Delta q^T \tilde{A}_{\mathrm{qq}}\Delta q \\ & + \Delta s_0^{xT}\tilde{A}_{\mathrm{sq}}\Delta q + \tilde{a}_{\mathrm{s}}^T \Delta s_0^x + \tilde{a}_{\mathrm{q}}^T \Delta q \end{aligned} \tag{6.10b}$$

subject to

$$\begin{aligned} s_0^x - x_0 + \Delta s_0^x &= 0 \\ \tilde{b}_{\mathrm{r}} + \tilde{B}_{\mathrm{rs}}\Delta s_0^x + \tilde{B}_{\mathrm{rq}}\Delta q &= 0 \\ \tilde{c} + \tilde{C}_{\mathrm{s}}\Delta s_0^x + \tilde{C}_{\mathrm{q}}\Delta q &\geq 0. \end{aligned}$$

At the moment when x_0 is known, the fully reduced QP can be formulated:

$$\min_{\Delta q} \quad \tfrac{1}{2}\Delta q^T \tilde{A}_{\mathrm{qq}}\Delta q + \left((x_0 - s_0^x)^T \tilde{A}_{\mathrm{sq}} + \tilde{a}_{\mathrm{q}}^T\right)\Delta q \tag{6.11}$$

subject to

$$\begin{aligned} \left(\tilde{b}_{\mathrm{r}} + \tilde{B}_{\mathrm{rs}}(x_0 - s_0^x)\right) + \tilde{B}_{\mathrm{rq}}\Delta q &= 0 \\ \left(\tilde{c} + \tilde{C}_{\mathrm{s}}(x_0 - s_0^x)\right) + \tilde{C}_{\mathrm{q}}\Delta q &\geq 0. \end{aligned}$$

This dense QP can be solved by a standard QP solver. It is of rather small size compared to the original uncondensed QP (6.8), and bears nearly no sparsity. We usually employ QPSOL[4] by Gill, Murray, Saunders, and Wright [GMSW83], a routine that makes use of an active set strategy and is able to cope with indefinite Hessian matrices. Note that in principle even large parts of the fully reduced QP (6.11) can be precomputed before x_0 is available, if matrix factorizations based on the active set for $x_0 = s_0^x$ are calculated in advance, as proposed in [BDLS00].

The solution of the fully reduced QP are the optimal values for $\Delta q_0, \ldots, \Delta q_{N-1}$. The value of Δq_0 plays a crucial role in the real-time context, as it is this control that is given directly to the real system as an immediate feedback.

Remark: The fact that an active set strategy is used to determine the active set carries some danger in the real-time context, as it is well known that the worst case complexity of such an algorithm can be exponential in the number of variables (cf. Klee and Minty [KM72]). Experience shows, however, that the computational burden of this dense QP solution is bounded in practice. In typical applications of our real-time algorithms it is considerably smaller than the effort needed for the first condensing step, which itself needs only a small share of the overall time of a full real-time iteration cycle. A theoretically appealing alternative to active set strategies is provided by Interior-Point Methods (IPM),

[4] QPSOL is available as a NAG routine under the name E04NAF.

that have polynomial run time bounds. For an introduction into IPM algorithms and their application to quadratic programs we refer e.g. to Nocedal and Wright [NW99] or Wright [Wri97].

6.5.3 Expansion of the QP Solution

The expansion of the QP solution passes through the two condensing steps in reverse order: first the fully reduced QP solution is expanded to the condensed QP solution, and secondly, the condensed QP solution is expanded to the full solution of the uncondensed QP (6.8).

First Expansion Step

The solution $(\Delta q, \tilde{\lambda}_r, \tilde{\mu})$ of the fully reduced QP (6.11) can trivially be expanded to yield the solution $(\Delta w_2, \tilde{\lambda}_2, \tilde{\mu})$ of the condensed QP (6.10a) (resp. (6.10b)) with

$$\Delta w_2 = \begin{pmatrix} \Delta s_0^x \\ \Delta q \end{pmatrix}, \quad \text{and} \quad \tilde{\lambda}_2 = \begin{pmatrix} \tilde{\lambda}_0^x \\ \tilde{\lambda}_r \end{pmatrix}$$

by computing

$$\Delta s_0^x = x_0 - s_0^x, \quad \text{and} \quad \tilde{\lambda}_0^x = \tilde{A}_{\mathrm{ss}} \Delta s_0^x + \tilde{A}_{\mathrm{sq}} \Delta q + \tilde{a}_{\mathrm{s}} - \tilde{B}_{\mathrm{rs}}^T \tilde{\lambda}_r - \tilde{C}_{\mathrm{s}}^T \tilde{\mu}.$$

That $(\Delta w_2, \tilde{\lambda}_2, \tilde{\mu})$ is a solution of the condensed QP can be seen by comparing the KKT conditions of (6.10b) with those of (6.11).

Expansion of the Condensed QP Solution

Similarly, the solution $(\Delta w_2, \tilde{\lambda}_2, \tilde{\mu})$ of the condensed QP (6.10a) can further be expanded to the full solution $(\Delta w_1, \Delta w_2, \tilde{\lambda}_1, \tilde{\lambda}_2, \tilde{\mu})$ of the uncondensed QP (6.8) by computing

$$\Delta w_1 = M \Delta w_2 + m$$

and

$$\tilde{\lambda}_1 = B_{11}^{-T} (A_{11} \Delta w_1 + A_{12} \Delta w_2 + a_1 - B_{21}^T \tilde{\lambda}_2 - C_1^T \tilde{\mu}). \tag{6.12}$$

To justify Eq (6.12) let us formulate the stationarity condition of the Lagrange gradient of the uncondensed QP (6.8) as follows:

$$\begin{aligned} A_{11} \Delta w_1 + A_{12} \Delta w_2 + a_1 - B_{11}^T \tilde{\lambda}_1 - B_{21}^T \tilde{\lambda}_2 - C_1^T \tilde{\mu} &= 0, \\ A_{22} \Delta w_2 + A_{12}^T \Delta w_1 + a_2 - B_{12}^T \tilde{\lambda}_1 - B_{22}^T \tilde{\lambda}_2 - C_2^T \tilde{\mu} &= 0. \end{aligned}$$

The first condition is equivalent to Eq. (6.12), whereas the second can be seen to be satisfied, if Lagrange stationarity with respect to Δw_2 is attained in the condensed QP (6.10a):

$$\begin{aligned} 0 &= \tilde{A}\Delta w_2 + \tilde{a} - \tilde{B}^T\tilde{\lambda}_2 - \tilde{C}^T\tilde{\mu} \\ &= (M^T A_{11} M + M^T A_{12} + A_{12}^T M + A_{22})\Delta w_2 + M^T A_{11} m + A_{12}^T m \\ &\quad + M^T a_1 + a_2 - (B_{21}M + B_{22})^T\tilde{\lambda}_2 - (C_1 M + C_2)^T\tilde{\mu} \\ &= A_{22}\Delta w_2 + A_{12}^T\Delta w_1 + a_2 - B_{22}^T\tilde{\lambda}_2 - C_2^T\tilde{\mu} \\ &\quad + M^T\{A_{11}\Delta w_1 + A_{12}\Delta w_2 + a_1 - B_{21}^T\tilde{\lambda}_2 - C_1^T\tilde{\mu}\} \\ &= A_{22}\Delta w_2 + A_{12}^T\Delta w_1 + a_2 - B_{12}^T\tilde{\lambda}_1 - B_{22}^T\tilde{\lambda}_2 - C_2^T\tilde{\mu}. \end{aligned}$$

The expansion step can efficiently be performed by a backwards recursion that is e.g. described by Leineweber [Lei99]. We mention here that the partial reduction approach does *not* allow to recapture the multipliers λ_i^z of the algebraic consistency conditions, because this would require knowledge of derivatives that are not computed. Fortunately, these multipliers are of minor importance in our real-time iteration scheme, as the current multiplier values only enter the next QP formulation through the Hessian approximation. If the extended Gauss-Newton approach is used, the multiplier values do not matter at all in the QP formulation.

6.6 A Riccati Recursion Approach

A second basic strategy to attack the solution of the partially reduced QP (6.5), that leads to a Riccati recursion scheme, can best be presented in the framework of dynamic programming. We will here only introduce the underlying idea, and refer the interested reader to Steinbach [Ste95] or Rao et al. [RWR98] for a more detailed description of the approach. For ease of presentation, we restrict our attention to QP problems (6.5) without final state and inequality constraints (6.5d)-(6.5f). We will also assume that R_N, S_N, and g_N^q are zero, so that the last control Δq_N can directly be eliminated from the problem, i.e., we consider the QP

$$\begin{aligned} \min_{\substack{\Delta s_0^x, \dots, \Delta s_N^x \\ \Delta q_0, \dots, \Delta q_{N-1}}} \quad & \sum_{i=0}^{N-1} \{\tfrac{1}{2}\Delta q_i^T R_i \Delta q_i + \Delta q_i^T S_i \Delta s_i^x + g_i^{xT}\Delta s_i^x + g_i^{qT}\Delta q_i \\ & \quad + \tfrac{1}{2}\Delta s_i^{xT} Q_i \Delta s_i^x\} + \tfrac{1}{2}\Delta s_N^{xT} Q_N \Delta s_N^x + g_N^{xT}\Delta s_N^x \end{aligned} \tag{6.14}$$

subject to

$$\begin{aligned} c_{i+1} + \Delta s_{i+1}^x - X_i\Delta s_i^x - Y_i\Delta q_i = 0, \quad i = 0, \dots, N-1, \\ s_0^x - x_0 + \Delta s_0^x = 0. \end{aligned}$$

The idea of the recursive algorithm to solve the above QP can be summarized as follows: starting with the cost function

$$\Pi_N(\Delta s_N^x) := \frac{1}{2}\Delta s_N^{x\,T} Q_N \Delta s_N^x + g_N^{x\,T}\Delta s_N^x \tag{6.15}$$

of the final node, we construct the so called optimal cost-to-go function $\Pi_{N-1}(\Delta s^x_{N-1})$ of the previous stage, by choosing for each value Δs^x_{N-1} the control Δq_{N-1} that optimizes the added costs to go to the final stage, i.e., the sum of the stage costs and the final stage costs Π_N. This procedure is repeated for Π_{N-2} down to Π_0. At each step the following small optimization problem

$$\begin{aligned}\Pi_i(\Delta s^x_i) := \min_{\Delta s^x_{i+1}, \Delta q_i} \frac{1}{2}\Delta s_i^{xT} Q_i \Delta s^x_i + \frac{1}{2}\Delta q_i^T R_i \Delta q_i + \Delta q_i^T S_i \Delta s^x_i \\ + g_i^{xT}\Delta s^x_i + g_i^{qT}\Delta q_i + \Pi_{i+1}(\Delta s^x_{i+1}) \qquad (6.16)\\ \text{subject to} \\ c_{i+1} + \Delta s^x_{i+1} - X_i \Delta s^x_i - Y_i \Delta q_i = 0\end{aligned}$$

is solved. It turns out that the cost-to-go functions $\Pi_i(\Delta s^x_i)$ remain quadratic functions, a fact that makes the dynamic programming approach so efficient. Let us therefore write

$$\Pi_i(\Delta s^x_i) = \frac{1}{2}\Delta s_i^{xT} P_i \Delta s^x_i + p_i^T \Delta s^x_i + \pi_i, \quad \text{for} \quad i = 0, \ldots, N.$$

The algorithm that we propose for the real-time solution of the QP consists of three steps, first a backwards recursion that prepares the second step (the immediate feedback), and finally a forward recursion which recovers the full QP solution.

6.6.1 Backwards Recursion

The backwards recursion is started by defining Π_N according to Eq. (6.15), i.e.,

$$P_N := Q_N, \quad p_N := g^x_N, \quad \text{and} \quad \pi_N = 0.$$

For the recursion step, let us assume that the optimal cost-to-go function $\Pi_{i+1}(\Delta s^x_{i+1})$ has already been computed, i.e., that the matrix P_{i+1}, the vector p_{i+1} and the scalar π_{i+1} are known. The QP (6.16) can be solved as follows: first we eliminate

$$\Delta s^x_{i+1} = -c_{i+1} + X_i \Delta s^x_i + Y_i \Delta q_i \qquad (6.17)$$

in the objective function

$$\begin{aligned}F_i(\Delta s^x_i, \Delta q_i, \Delta s^x_{i+1}) := \; & \tfrac{1}{2}\Delta s_i^{xT} Q_i \Delta s^x_i + \tfrac{1}{2}\Delta q_i^T R_i \Delta q_i + \Delta q_i^T S_i \Delta s^x_i + g_i^{xT}\Delta s^x_i + g_i^{qT}\Delta q_i \\ & + \tfrac{1}{2}\Delta {s^x_{i+1}}^T P_{i+1} \Delta s^x_{i+1} + p_{i+1}^T \Delta s^x_{i+1} + \pi_{i+1}\end{aligned}$$

that becomes

$$\begin{aligned}F_i(\cdot) = \; & \tfrac{1}{2}\Delta q_i^T (R_i + Y_i^T P_{i+1} Y_i)\Delta q_i \\ & + ((S_i + Y_i^T P_{i+1} X_i)\Delta s^x_i + g^q_i - Y_i^T P_{i+1} c_{i+1} + Y_i^T p_{i+1})^T \Delta q_i \\ & + \tfrac{1}{2}\Delta s_i^{xT}(Q_i + X_i^T P_{i+1} X_i)\Delta s^x_i + g_i^{xT}\Delta s^x_i + c_{i+1}^T P_{i+1} c_{i+1} \\ & - c_{i+1}^T P_{i+1} X_i \Delta s^x_i - p_{i+1}^T c_{i+1} + p_{i+1}^T X_i \Delta s^x_i + \pi_{i+1}.\end{aligned}$$

The minimum of this function with respect to Δq_i is attained at

$$\begin{aligned} \Delta q_i &= -(R_i + Y_i^T P_{i+1} Y_i)^{-1} (S_i + Y_i^T P_{i+1} X_i)\ \Delta s_i^x \\ &\quad -(R_i + Y_i^T P_{i+1} Y_i)^{-1} (g_i^q - Y_i^T P_{i+1} c_{i+1} + Y_i^T p_{i+1}) \\ &=: -K_i \Delta s_i^x - k_i, \end{aligned} \tag{6.18}$$

which inserted into the objective function F_i gives the optimal cost-to-go according to (6.16) as

$$\Pi_i(\Delta s_i^x) = \frac{1}{2} \Delta s_i^{xT} P_i \Delta s_i^x + p_i^T \Delta s_{i+1}^x + \pi_i$$

with

$$\begin{aligned} P_i := \ & Q_i + X_i^T P_{i+1} X_i \\ & -(S_i + Y_i^T P_{i+1} X_i)^T (R_i + Y_i^T P_{i+1} Y_i)^{-1} (S_i + Y_i^T P_{i+1} X_i), \end{aligned}$$

$$\begin{aligned} p_i := \ & g_i^x + X_i^T p_{i+1} - X_i^T P_{i+1} c_{i+1} \\ & -(S_i + Y_i^T P_{i+1} X_i)^T (R_i + Y_i^T P_{i+1} Y_i)^{-1} (g_i^q - Y_i^T P_{i+1} c_{i+1} + Y_i^T p_{i+1}), \end{aligned}$$

$$\begin{aligned} \pi_i := \ & \pi_{i+1} + c_{i+1}^T P_{i+1} c_{i+1} - p_{i+1}^T c_{i+1} \ - (g_i^q - Y_i^T P_{i+1} c_{i+1} + Y_i^T p_{i+1})^T \cdot \\ & (R_i + Y_i^T P_{i+1} Y_i)^{-1} (g_i^q - Y_i^T P_{i+1} c_{i+1} + Y_i^T p_{i+1}). \end{aligned}$$

The values π_{i+1} are irrelevant for the determination of Δs_{i+1}^x and Δq_i; therefore they are usually omitted. The matrix recursion formula for P_i is also known as the discrete-time Riccati matrix equation for time-varying systems.

The only quantities that have to be stored for subsequent use in the forward recursion are the matrices $K_0, \ldots, K_{N-1}$ and $P_0, \ldots, P_N$, and the vectors $k_0, \ldots, k_{N-1}$ and $p_0, \ldots, p_N$.

6.6.2 Immediate Feedback

The complete backwards recursion can be performed before the actual value of x_0 is known. Then, at the moment when x_0 is known, the control response Δq_0 can be quickly determined according to Eq. (6.18),

$$\Delta q_0 = -K_0 (x_0 - s_0^x) - k_0,$$

and can immediately be given to the plant. The control response can be interpreted as the first part of the forward recursion that will be described in the following. However, we stress the fact that it is only the above matrix vector multiplication and vector addition that needs to be performed to deliver the part of the QP solution, Δq_0, that is actually needed for the approximate optimal feedback control response.

This computation requires only $n_u \times n_x + n_u$ floating point operations which can certainly be considered an immediate feedback, when compared to the complete real-time iteration cycle, which needs many orders of magnitude higher computational effort.

6.6.3 Forward Recursion

Starting with a known value of $\Delta s_0^x = x_0 - s_0^x$, Eqs. (6.18) and (6.17) are alternately used to compute

$$\Delta q_i = -K_i \Delta s_i^x - k_i, \quad \text{for} \quad i = 0, \ldots, N-1,$$

and

$$\Delta s_{i+1}^x = -c_{i+1} + X_i \Delta s_i^x + Y_i \Delta q_i \quad \text{for} \quad i = 0, \ldots, N-1.$$

The QP multipliers $\tilde{\lambda}_i^x$ for $i = 0, \ldots, N$ are computed as follows

$$\tilde{\lambda}_i^x = \frac{\partial \Pi_i(\Delta s_i^x)}{\partial \Delta s_i^x} = P_i \Delta s_i^x + p_i.$$

6.6.4 Comparison of Condensing and Riccati Recursion

The Riccati recursion scheme allows to solve the QP with a numerical effort of $O(N)$, which is in sharp contrast to the condensing approach, which in turn is of $O(N^2)$ for the condensing itself, and even $O(N^3)$ for the solution of the dense QP, if we disregard active set changes. For the practical applications that we have encountered so far, however, we have employed the condensing approach. This was motivated by the following observations:

- Active set changes during the QP solution are rather expensive in the Riccati approach, as each active set change would require a full backwards and forward recursion. In practical implementations, the Riccati recursion is therefore usually implemented in conjunction with an interior-point method (IPM) to treat the inequalities (cf. [Ste95, Wri96, RWR98]). But even the IPM approach requires some complete recursions until the QP solution is found, and is therefore not strictly in line with our idea of an immediate feedback, that takes active set changes into account.

- Furthermore, practical experience shows that the sensitivity computation dominates by far the overall computational effort during each real-time iteration cycle for typical application problems which have large state dimensions n_x and a small number N of multiple shooting intervals, when the condensing approach is employed. This reduces the practical benefits of alternative QP solution procedures.

Though the condensing approach works well in current applications, we want to point out that it has its limits, especially for long horizon lenghts N, and that a solution scheme that employs the Riccati recursion with an interior-point method, as e.g. developed by Steinbach [Ste95] for the multiple shooting method, promises to offer advantages in the real-time context and deserves further investigation.

6.7 Division into Preparation and Feedback Phase

We will now summarize the version of the real-time algorithm that we used for most numerical tests in this thesis. It makes use of the newly developed Gauss-Newton approach to obtain the Hessian approximation, and employs the condensing strategy to solve the partially reduced QP. Though we first present the necessary computations in the same order as in the above presentation, we will give a second ordering of the steps that allows to interpret the algorithm as the successive generation of approximated optimal feedback control laws.

6.7.1 Five Computation Steps

During each real-time iteration the following steps have to be performed:

1. Partial reduction: Linearize the consistency conditions and resolve the linear system to eliminate the Δs_i^z as a linear function of Δs_i^x and Δq_i, as described in Sec. 6.2

2. DAE solution and derivative generation: Linearize the continuity conditions by solving the relaxed initial value problems and computing directional derivatives with respect to Δs_i^x and Δq_i following the scheme of Sec. 6.3. Simultaneously, compute the gradient of the objective function, and the Hessian approximation according to the Gauss-Newton approach described in Sec. 6.4. Linearize also the remaining point constraints.

3. First condensing step: Using the linearized continuity conditions, eliminate the variables $\Delta s_1^x, \ldots \Delta s_N^x$. Project the objective gradient onto the space of the remaining variables Δs_0^x, $\Delta q_0, \ldots, \Delta q_{N-1}$, and also the Hessian and the linearized point constraints.

4. Step generation: at the moment that x_0 is known, perform the second condensing step and solve the fully reduced QP with an efficient dense QP solver using an active set strategy. The solution yields the final values of $\Delta q_0, \ldots, \Delta q_{N-1}$. The value $q_0 + \Delta q_0$ can immediately be given as a control to the real-system.

5. Expansion: Expand the fully reduced QP solution to yield the full QP solution $(\Delta w, \tilde{\lambda}, \tilde{\mu})$. Based on this QP solution, pass over to the next SQP iterate and go back to step 1.

6.7.2 The Off-Line Steps in a Rotated Order

It is an important feature of the above cycle that the value x_0 needs only to be known before step 4 can be performed. In our real-time implementation, we isolate step 4 and rotate the order of the above steps, to yield the following scheme:

I) Feedback phase: After observation of the current value x_0 perform *only step 4* and apply the resulting value of $q_0 + \Delta q_0$ immediately to the real process. Maintain the new control value during some process duration δ which is sufficiently long to perform all calculations of one cycle.

II) Preparation phase: During this period δ first expand the outcome of step 4 to the full QP solution (expansion step 5), then compute the new iterate $w^{k+1} = w^k + \Delta w^k$, and based on this *new* iterate, perform the steps 1, 2 and 3 to prepare the feedback response for the following step. Go back to I.

The feedback phase itself is typically orders of magnitude shorter than the preparation phase (cf. Fig 7.7). Thus, our algorithm can be interpreted as the successive generation of immediate feedback laws (cf. Sec. 4.4.2) that take state and control inequality constraints on the complete horizon into account. Experience with the investigated large scale examples shows that the active set does not change much from one cycle to the next so that the computation time for the feedback is bounded and very small in practice.

Chapter 7

Control of a Distillation Column

As an application example for the proposed real-time iteration schemes we consider the control of a high purity binary distillation column. We have performed a variety of closed-loop experiments at a pilot plant distillation column that is located at the *Institut für Systemdynamik und Regelungstechnik (ISR)* of the University of Stuttgart. All experiments were carried out in collaboration with Dr. Ilknur Uslu, Stefan Schwarzkopf, and Rolf Findeisen. Financial support by the *Deutsche Forschungsgemeinschaft (DFG)* within the *DFG-Schwerpunktprogramm* "Real-Time Optimization of Large Systems" is gratefully acknowledged.

In first numerical tests, the feasibility of the real-time optimization scheme could be shown, with computation times in the range of seconds for a 164th order model [DBS+01], and the practical applicability was confirmed in a first series of closed-loop experiments [DUF+01]; however, the observed closed-loop performance suffered from oscillations that were due to time delays in the real plant, that have *not* been captured by the 164th order distillation model. We therefore improved the model by including hydrodynamic effects that have been responsible for the time delays, resulting in a considerably stiffer and larger system model. We will in this chapter present new numerical and experimental results that have been obtained with this improved system model. Parts of the presentation, especially of the experimental setup, follow the lines of a previous paper [DUF+01], from which originate also the Figures 7.1, 7.8, and 7.9.

7.1 The Distillation Column

The distillation column is used for the separation of a binary mixture of Methanol and n-Propanol. It has a diameter of 0.10 m and a height of 7 m and consists of 40 bubble cap trays. The overhead vapor is totally condensed in a water cooled condenser which is open to atmosphere. The reboiler is heated electrically. A flowsheet of the distillation system is shown in Fig. 7.1. The preheated feed stream enters the column at the feed tray as saturated liquid. It can be switched automatically between two feed tanks in order to introduce well defined disturbances in the feed concentration. In the considered

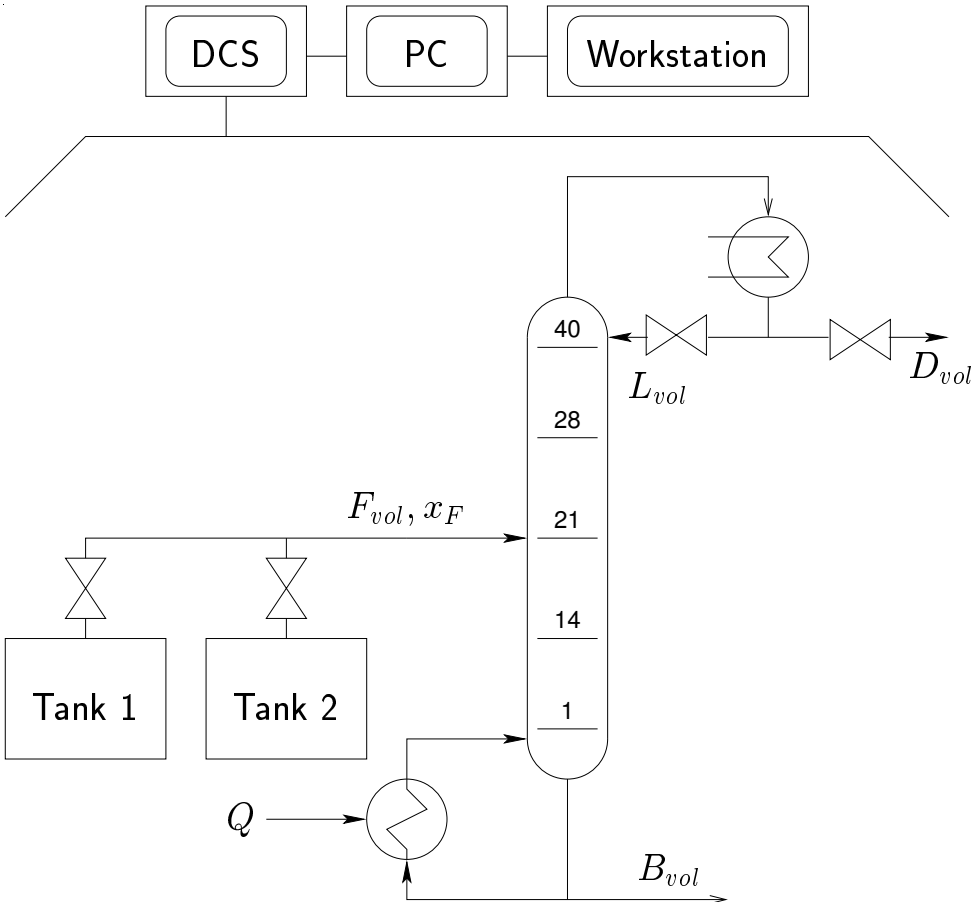


Figure 7.1: Flowsheet of the distillation column

configuration, the process inputs that are available for control purposes are the heat input to the reboiler, Q, and the reflux flow rate L_{vol}. Control aim is to maintain high purity specifications for the distillate and bottom product streams D_{vol} and B_{vol}.

The column is controlled by a distributed control system (DCS), that is used for the lower level control and data acquisition. Basic control loops for the levels, the flow rates, and the heat input are realized on the DCS system. To implement more advanced control schemes the DCS is connected to a PC from and to which direct access from UNIX workstations is possible.

7.1.1 The DAE Model

We will refer to the $N = 40$ trays by $\ell = 1, 2, \ldots, N$, counting from the bottom to the top, with $\ell = N_F = 20$ being the feed tray. For notational convenience, let us refer with $\ell = 0$

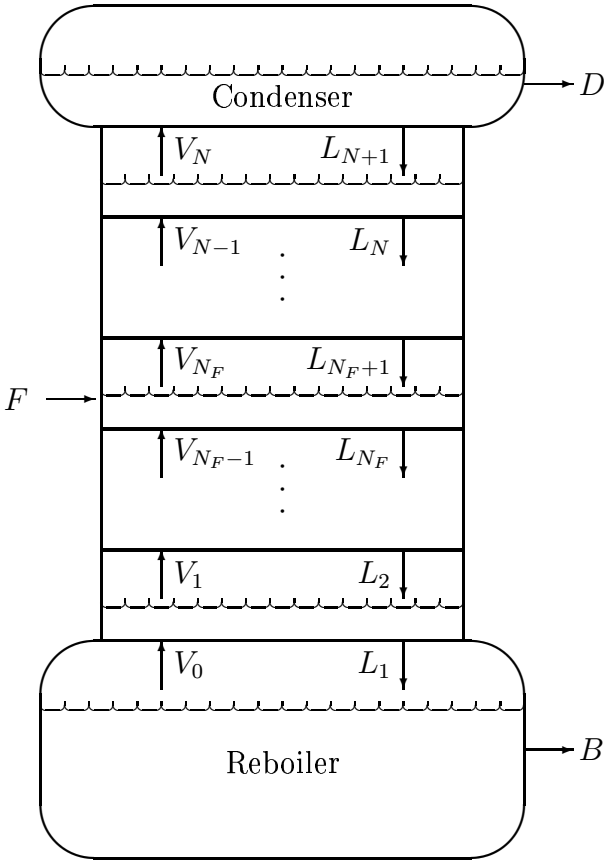


Figure 7.2: Molar flows in the distillation column.

to the reboiler and to the condenser by $\ell = N + 1$. The corresponding temperatures are denoted by $T_0, \ldots, T_{N+1}$.

As we treat a binary distillation we have only two components, Methanol and n-Propanol. Let us denote the liquid Methanol concentrations of reboiler, trays and condenser by X_ℓ for $\ell = 0, 1, 2, \ldots, N + 1$. The concentration $X_{\text{n-Prop.},\ell}$ of n-Propanol is determined by the closing condition, so that we will substitute $X_{\text{n-Prop.},\ell} := 1 - X_{1,\ell}$ directly.

The molar vapor and liquid fluxes out of each tray are denoted by V_ℓ and L_ℓ for $\ell = 1, 2, \ldots, N$. The molar vapor flux out of the reboiler is denoted by V_0 and the liquid bottom product stream by B. Similarly, L_{N+1} denotes the molar liquid reflux out of the condenser into the top tray while D denotes the distillate stream out of the condenser. The molar feed stream entering at tray $\ell = N_F$ is denoted by F; it is assumed to be liquid. All molar flows in the distillation column are depicted in Fig. 7.2.

The molar concentrations of the liquid fluxes are equal to the tray concentrations X_ℓ, whereas the vapor fluxes' molar concentrations are denoted by Y_ℓ for $\ell = 0, 1, \ldots, N$.

We assume that the pressures P_ℓ of reboiler, trays and condenser are constant, as well as the volume holdups n_0^v and n_{N+1}^v of reboiler and condenser. The liquid volume holdups n_ℓ^v of the trays may vary. All volume holdups are related to the molar holdups n_ℓ by

$$n_\ell^v = V^m(X_\ell, T_\ell)\, n_\ell \quad \text{for } \ell = 0, \ldots, N+1,$$

The molar volumes $V^m(X_\ell, T_\ell)$ of the liquid mixture are specified in Appendix B.

To determine the (constant) pressures we assume that the condenser pressure is fixed to the outside pressure, i.e., $P_{N+1} = P_{\text{top}}$ whereas the pressures P_ℓ on the trays and the reboiler are calculated under the assumption of constant pressure drop from tray to tray, i.e.,

$$P_\ell = P_{\ell+1} + \Delta P_\ell \quad \ell = N, N-1, \ldots, 2, 1, 0.$$

The tray temperatures T_ℓ are implicitly defined by the assumption that the sum of the partial pressures equals the total pressure on each tray, i.e.,

$$P_\ell - P_1^s(T_\ell)X_\ell - P_2^s(T_\ell)(1 - X_\ell) = 0, \quad \ell = 0, 1, \ldots, N+1, \tag{7.1}$$

where the partial pressures $P_k^s(T_\ell)$ are computed according to the Antoine Equation, as specified in Appendix B.

The derivative of the temperature with respect to time, $\dot{T}_\ell$, is given by the implicit function theorem:

$$\dot{T}_\ell = -\frac{(P_1^s(T_\ell) - P_2^s(T_\ell))\dot{X}_\ell}{\frac{\partial P_1^s}{\partial T_\ell}X_\ell + \frac{\partial P_2^s}{\partial T_\ell}(1 - X_\ell)}.$$

To account for non-ideality of the trays and other unmodelled effects we have introduced the tray efficiencies α_ℓ for $\ell = 1, 2, \ldots, N$ to calculate the composition Y_ℓ of the vapor flow out of tray ℓ as a linear combination of the ideal vapor composition on the tray and the incoming vapor composition from the tray below, i.e.,

$$Y_\ell = \alpha_\ell \frac{P_1^s(T_\ell)}{P_\ell} X_\ell + (1 - \alpha_\ell)\, Y_{\ell-1}, \quad \ell = 1, \ldots, N,$$

starting with $Y_0 = \frac{P_1^s(T_0)}{P_0} X_0$. The concentration of the liquid outflow D at the top of the column is equal to the condenser concentration X_{N+1}.

Mass balances: The differential equations that determine the change of the molar holdups n_ℓ of the trays are given by the mass conservation for $\ell = 1, 2, \ldots, N_F - 1, N_F + 1, \ldots, N$

$$\dot{n}_\ell = V_{\ell-1} - V_\ell + L_{\ell+1} - L_\ell, \tag{7.2}$$

and for $\ell = N_F$ by

$$\dot{n}_{N_F} = V_{N_F-1} - V_{N_F} + L_{N_F+1} - L_{N_F} + F, \tag{7.3}$$

where F is the molar inflow on the feed tray, that can be determined from the volume feed flow F_{vol}, the Methanol concentration X_F in the feed and its temperature T_F via

$$F_{\text{vol}} = V^m(X_F, T_F)F.$$

Mass conservation in reboiler and condenser are given by

$$\dot{n}_0 = -V_0 + L_1 - B, \tag{7.4}$$

and

$$\dot{n}_{N+1} = V_N - D - L_{N+1}. \tag{7.5}$$

The assumption that reboiler and condenser volume n_0^v and n_{N+1}^v are fixed leads to two further equations for $\ell = 0, N+1$

$$0 = \dot{n}_\ell^v = V^m(X_\ell, T_\ell)\dot{n}_\ell + \frac{\partial V^m}{\partial(X,T)}(\dot{X}_\ell, \dot{T}_\ell)^T\, n_\ell, \tag{7.6}$$

that allow to eliminate $\dot{n}_0$ and $\dot{n}_{N+1}$. Therefore, n_0 and n_{N+1} are chosen to be *no* differential variables.

The liquid reflux stream L_{vol} from the condenser is controlled and allows to determine L_{N+1} via

$$L_{\text{vol}} = V^m(X_{N+1}, T_C)L_{N+1},$$

where T_C is the temperature of the condensate.

The componentwise mass conservation in the reboiler requires

$$\dot{X}_0 n_0 + X_0\dot{n}_0 = -V_0Y_0 + L_1X_1 - BX_0, \tag{7.7}$$

on the trays $\ell = 1, 2, \ldots, N_F - 1, N_F + 1, \ldots, N$,

$$\dot{X}_\ell n_\ell + X_\ell\dot{n}_\ell = V_{\ell-1}Y_{\ell-1} - V_\ell Y_\ell + L_{\ell+1}X_{\ell+1} - L_\ell X_\ell, \tag{7.8}$$

on the feed tray

$$\begin{aligned} \dot{X}_{N_F} n_{N_F} + X_{N_F}\dot{n}_{N_F} \;=\; & V_{N_F-1}Y_{N_F-1} - V_{N_F}Y_{N_F} \\ & + L_{N_F+1}X_{N_F+1} - L_{N_F}X_{N_F} + FX_F, \end{aligned} \tag{7.9}$$

and in the condenser

$$\dot{X}_{N+1}n_{N+1} + X_{N+1}\dot{n}_{N+1} = V_NY_N - DX_{N+1} - L_{N+1}X_{N+1}. \tag{7.10}$$

Enthalpy balances: With the liquid and vapor stream enthalpies abbreviated as $h_\ell^L := h^L(X_\ell, T_\ell)$ and $h_\ell^V := h^V(Y_\ell, T_\ell, P_\ell)$ for $\ell = 0, \ldots, N$ (see Appendix B), we can formulate the enthalpy balance in the reboiler that allows to determine the vapor stream V_0:

$$\dot{n}_0 h_0^L + n_0 \left(\frac{\partial h_0^L}{\partial X_0} \dot{X}_0 + \frac{\partial h_0^L}{\partial T_0} \dot{T}_0 \right) = Q - Q_{\text{loss}} - V_0 h_0^V + L_1 h_1^L - B h_0^L. \tag{7.11}$$

Here Q is the applied heat input, and with Q_{loss} we account for possible heat losses in the reboiler. The enthalpy balances for the trays $\ell = 1, 2, \ldots, N_F - 1, N_F + 1, \ldots, N - 1$ are

$$\begin{aligned} \dot{n}_\ell h_\ell^L + n_\ell \left(\tfrac{\partial h_\ell^L}{\partial X_\ell} \dot{X}_\ell + \tfrac{\partial h_\ell^L}{\partial T_\ell} \dot{T}_\ell \right) &= V_{\ell-1} h_{\ell-1}^V - V_\ell h_\ell^V + L_{\ell+1} h_{\ell+1}^L \\ &\quad - L_\ell h_\ell^L, \end{aligned} \tag{7.12}$$

and for the feed tray

$$\begin{aligned} &\dot{n}_{N_F} h_{N_F}^L + n_{N_F} \left(\tfrac{\partial h_{N_F}^L}{\partial X_{N_F}} \dot{X}_{N_F} + \tfrac{\partial h_{N_F}^L}{\partial T_{N_F}} \dot{T}_{N_F} \right) \\ &= \quad V_{N_F-1} h_{N_F-1}^V - V_{N_F} h_{N_F}^V + L_{N_F+1} h_{N_F+1}^L - L_{N_F} h_{N_F}^L \\ &\quad + F h^L(X_F, T_F, P_F). \end{aligned} \tag{7.13}$$

As the liquid reflux L_{N+1} of the condensate is at a temperature T_C, the enthalpy balance on tray N reads

$$\begin{aligned} &\dot{n}_N h_N^L + n_N \left(\tfrac{\partial h_N^L}{\partial X_N} \dot{X}_N + \tfrac{\partial h_N^L}{\partial T_N} \dot{T}_N \right) \\ &= V_{N-1} h_{N-1}^V - V_N h_N^V + L_{N+1} h^L(X_{N+1}, T_C, P_{N+1}) - L_N h_N^L. \end{aligned} \tag{7.14}$$

Hydrodynamics: To determine the liquid outflow L_ℓ of each tray, we use a heuristic scheme that is based on the so called "Francis weir formula". It requires only two parameters per tray, one is a reference volume n_ℓ^{ref}, the second is denoted by W_ℓ. The formula postulates that

$$L_\ell \, V^m(X_\ell, T_\ell) = W_\ell (n_\ell^v - n_\ell^{\text{ref}})^{\frac{3}{2}}, \quad \ell = 1, \ldots, N, \tag{7.15}$$

and can be derived by an analysis of the gravity flow over a horizontal weir with vertical walls, that is given in Appendix B.

Summarizing the DAE

We can subsume all system states in two vectors x and z which denote the differential and the algebraic state vectors, respectively.

The (molar) Methanol concentrations in reboiler, on the 40 trays, and in the condenser X_ℓ for $\ell = 0, 1, \ldots, N+1$ are the first 42 components of the differential state vector x, and the molar tray holdups n_ℓ for $\ell = 1, \ldots, N$ are the second 40 components.

The liquid and vapor (molar) fluxes L_ℓ and V_ℓ ($\ell = 1, 2, \ldots, N$) out of the 40 trays as well as the 42 temperatures T_ℓ ($\ell = 0, 1, 2, \ldots, N+1$) of reboiler,

trays and condenser form the 122 components of the algebraic state vector $z = (L_1, \ldots, L_N, V_1, \ldots, V_N, T_0, \ldots, T_{N+1})^T$.[1] Note that many algebraic variables that can easily be eliminated (as e.g. h_ℓ^L, $P_k^s(T_\ell)$, V_0, etc.) do not count as algebraic variables in this formulation.

The two components of the control vector $u = (L_{\text{vol}}, Q)^T$ are the volumetric reflux flow L_{vol}, and the heat input Q, that determines implicitly the molar vapor flux V_0 out of the reboiler. All remaining system parameters, i.e., n_0^v, P_{top}, n_{N+1}^v, $\Delta P_{0,\ldots,N}$, $n_{1,\ldots,N}^{\text{ref}}$, $\alpha_{1,\ldots,N}$, F_{vol}, X_F, $W_{1,\ldots,N}$, T_F, Q_{loss}, and T_C, can be subsumed in a vector p.

The equations (7.7)–(7.10) and (7.2)–(7.3) are the 82 differential equations f, and (7.15), (7.12)–(7.14), and (7.1) form the 122 algebraic equations g of the system.

After substituting $\dot{n}_\ell$ in Eqs. (7.7)–(7.10) and dividing these equations by n_ℓ, we can summarize the DAE system, which has index one, in the following form:

$$\dot{x}(t) = f(x(t), z(t), u(t), p) \tag{7.16}$$

$$0 = g(x(t), z(t), u(t), p). \tag{7.17}$$

The employed values for the parameters p have been estimated and are listed in Table 7.1 in Section 7.2. A complete reference to the material property functions $V^m(x,T)$, $P_k^s(T)$, $h^L(X,T)$, and $h^V(Y,T,P)$ is given in Appendix B.

7.2 Determination of the System Parameters

In the actual application, the performance of NMPC crucially depends on the quality of the model. Considering this fact, steady state and open-loop dynamic experiments have been performed. To obtain measurements of the dynamic behaviour of the column step changes in the feed rate F_{vol} and composition X_F, the reflux rate L_{vol}, and heat input Q were performed. Measurements of *all* 42 temperatures $T_0, \ldots, T_{N+1}$ were taken to obtain a least squares fit of the simulated to the observed behaviour. The additional assumptions for this fit are that the tray efficiencies are constant on each of the two column sections, i.e., $\alpha_1 = \ldots = \alpha_{N_F}$ and $\alpha_{N_F+1} = \ldots = \alpha_N$, that the pressure losses are constant on both sections: $\Delta P_0 = \ldots = \Delta P_{N_F-1}$ and $\Delta P_{N_F} = \ldots = \Delta P_N$, and that the volumetric reference tray holdups coincide: $n_1^{\text{ref}} = \ldots = n_N^{\text{ref}}$. Reboiler and condenser holdup are difficult to determine from temperature measurements, as they both contain very pure liquids during reasonable operating conditions and which have constant boiling temperatures. Conversely, these two volumes do not matter much for the NMPC performance. They were determined according to user knowlegde.

[1] The equilibrium temperature of the condenser mixture may help to define the temperature of the reflux by $T_C := T_{N+1}$ when T_C is not specified. Otherwise, this last algebraic variable could be eliminated without changing the dynamics.

Symbol	Value	Symbol	Value
n_0^v	8.5 l	P_{top}	939 h Pa
n_{N+1}^v	0.17 l	ΔP_{strip}	2.5 h Pa
$n_{\text{tray}}^{\text{ref}}$	0.155 l	ΔP_{rect}	1.9 h Pa
α_{strip}	62 %	T_F	71°C
α_{rect}	35 %	T_C	47.2°C
W_{tray}	0.166 $\text{l}^{-\frac{1}{2}}$ s^{-1}	F_{vol}	14.0 l h^{-1}
Q_{loss}	0.51 kW	X_F	0.32

Table 7.1: Constant system parameters

The 10 parameters that could be adjusted to dynamic experimental data were:

$$\begin{aligned}
&\alpha_{\text{strip}} := \alpha_{1,\dots,N_F},\\
&\alpha_{\text{rect}} := \alpha_{N_F+1,\dots,N},\\
&P_{\text{top}},\\
&\Delta P_{\text{strip}} := \Delta P_{0,\dots,N_F-1},\\
&\Delta P_{\text{rect}} := \Delta P_{N_F,\dots,N},\\
&Q_{\text{loss}},\\
&T_F,\\
&T_C\\
&n_{\text{tray}}^{\text{ref}} := n_{1,\dots,N}^{\text{ref}}, \text{ and}\\
&W_{\text{tray}} := W_{1,\dots,N}.
\end{aligned} \tag{7.18}$$

During the test series, these parameters have been adjusted several times using static and dynamic experiments, exploiting both, engineering intuition and advanced software tools. The finally estimated system parameters are listed in Table 7.1.

7.2.1 Static System Parameters

The estimation of the first eight of the parameters (7.18), that we call the *static* system parameters, can in principle be performed using steady state data only. Denoting the measured steady state temperature averages of a steady state experiment by the vector $T^{\text{m}} := (T_0^{\text{m}}, \dots, T_{N+1}^{\text{m}})^T$ and the averaged steady state controls by $u^{\text{m}} := (L_{\text{vol}}^{\text{m}}, Q^{\text{m}})^T$, and introducing the projection matrix T that extracts the temperatures from the algebraic system state, so that $Tz = (T_0, \dots, T_{N+1})^T$, we can formulate the following least squares problem:

$$\min_{x_S, z_S, p} \|Tz_S - T^{\text{m}}\|_Q^2 \tag{7.19}$$

subject to

$$\begin{aligned}
f(x_S, z_S, u^{\text{m}}, p) &= 0,\\
g(x_S, z_S, u^{\text{m}}, p) &= 0,
\end{aligned}$$

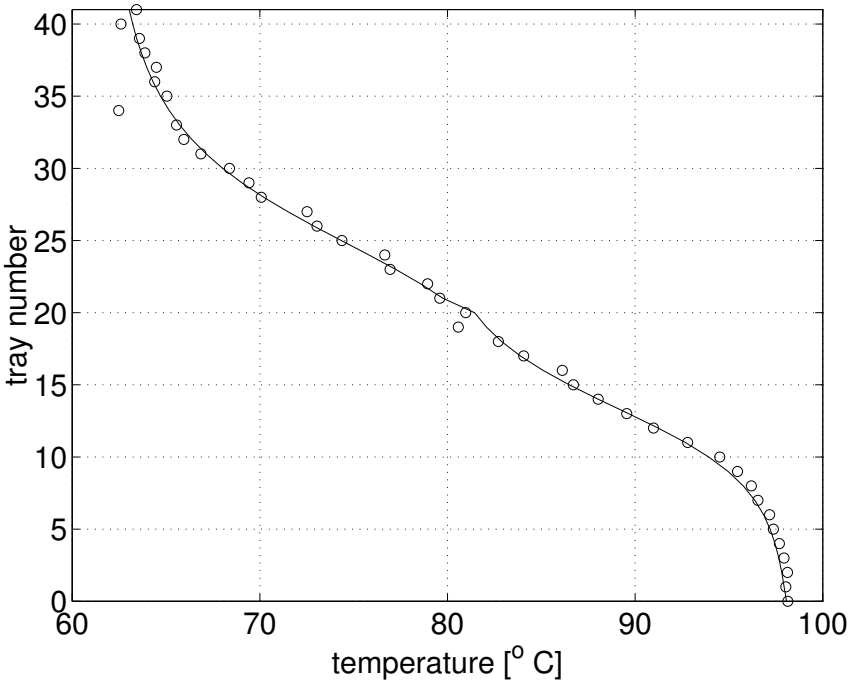


Figure 7.3: Comparison of temperature measurements and estimated steady state temperature profile (solid line).

where the constraints ensure that only steady states are allowed. The positive definite weighting matrix Q would ideally be the inverse of the covariance matrix of the temperature measurements, that can be expected to be diagonal with equal entries. For the NMPC performance tests, however, we have explicitly given more weight to the controlled temperatures, T_{28} and T_{14}, by a factor of ten, to avoid steady state offset due to model-plant mismatch.

7.2.2 Dynamic System Parameters

The last two parameters from the set (7.18), $n_{\text{tray}}^{\text{ref}}$ and W_{tray}, can only be estimated by dynamic experiments. They have been determined by the solution of a nonlinear least squares fit of the dynamic model to the measurement data. Let us for this aim define the time dependent temperature measurement function $T^{\text{m}}(t)$ and the measured control trajectory $u^{\text{m}}(t)$, on a horizon $[0, T]$. Then the estimation problem can be formulated as:

$$\min_{x(\cdot),z(\cdot),p} \int_0^T \|Tz(t) - T^{\text{m}}(t)\|_Q^2 \, dt \tag{7.20}$$

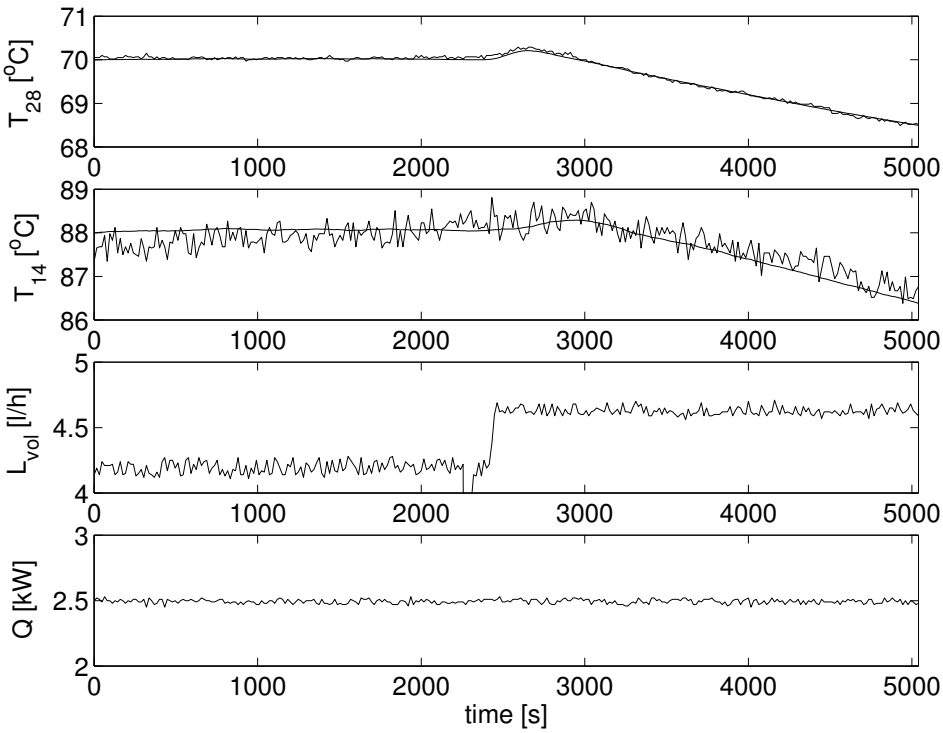


Figure 7.4: Comparison of measured (noisy) and simulated trajectories (smooth) of the temperatures T_{28} and T_{14}, for a small step change in the reflux L_{vol}.

subject to

$$\begin{aligned} \dot{x}(t) - f(x(t), z(t), u^{\text{m}}(t), p) &= 0,\ t \in [0, T], \\ g(x(t), z(t), u^{\text{m}}(t), p) &= 0,\ t \in [0, T]. \end{aligned}$$

If the dynamic experiment starts in steady state, we add the steady state constraint

$$\begin{aligned} f(x(0), z(0), u^{\text{m}}(0), p) &= 0, \\ g(x(0), z(0), u^{\text{m}}(0), p) &= 0. \end{aligned}$$

Though specifically tailored parameter estimation algorithms based on the multiple shooting method exist for the solution of this type of problem (see, e.g., Bock et al. [Boc87, BES88]), we have solved the least squares problems with our current implementation of the Gauss-Newton approach in the software package MUSCOD-II, as described in Sec. 6.4 (with a piecewise polynomial representation of the temperature measurement data). This approach has the practical advantage of being able to perform both, parameter estimation and dynamic optimization, in the same modelling environment, and thus reduces the risk of transcription errors. The finally employed parameter values for $n_{\text{tray}}^{\text{ref}}$ and W_{tray} have been determined by Bürner in a MUSCOD-II/MATLAB environment [Bür01].

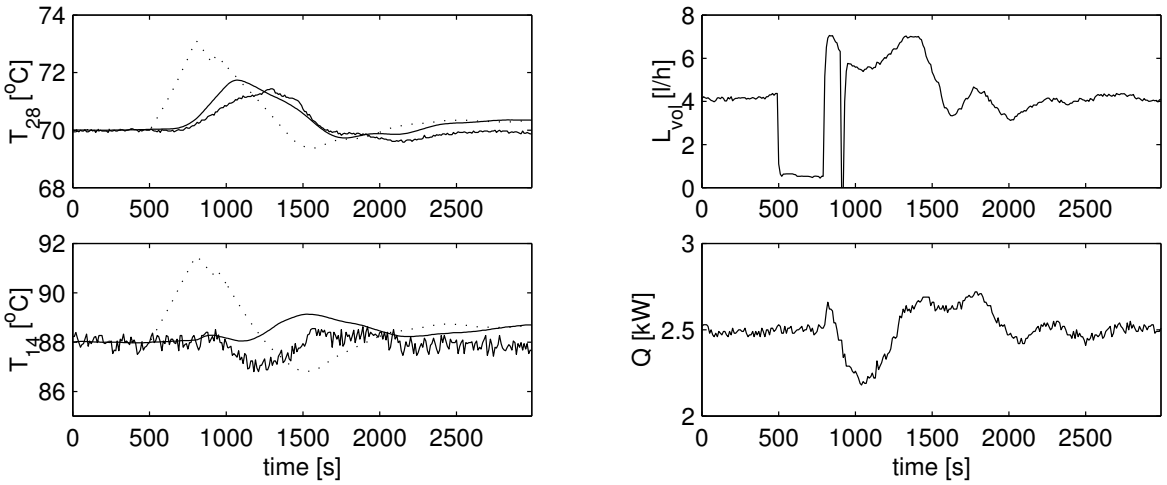


Figure 7.5: Comparison of measured (noisy) and simulated trajectories (smooth), for an a posteriori test with a control scenario which involves large but short reflux variations (right). The dotted lines show for comparison a simulation with an equilibrium model, which does not capture hydrodynamic effects.

In Fig. 7.4, simulated and measured profiles for the temperatures T_{28} and T_{14} are shown, for an experiment involving a step change in the reflux L_{vol}, and starting at the nominal operating conditions. The compared temperature profiles show that the medium time scale dynamics are captured well by the model.

An a posteriori test of the model can be seen on the right hand side of Fig. 7.5, where a simulation was performed using the same control profiles as in a closed-loop experiment (cf. Fig. 7.14), with very large steps in L_{vol}. The time horizon is shorter and the comparison shows that the model does roughly capture short time scale effects that are due to hydrodynamics, in contrast to an equilibrium model, that cannot reproduce the corresponding delays (dotted line) (cf. [DUF+01]).

7.3 Optimal Control Problem Formulation

The control aim is to maintain the specifications on the product purities X_0 and X_{N+1} in reboiler and condenser despite disturbances.

As usual in distillation control, the concentrations X_0 and X_{N+1} are not controlled directly – instead, an inferential control scheme which controls the deviation of the concentrations on tray 14 and 28 from a given setpoint is used. These two concentrations are much more sensitive to changes in the inputs of the system than the product concentrations; if they are kept constant, the product purities are safely maintained for a large range of process conditions. As concentrations are difficult to measure, we consider instead the

tray *temperatures*, which correspond directly to the concentrations via the Antoine equation. In the following we will use the projection $\tilde{T}z := (T_{28}, T_{14})^T$ to extract the controlled temperatures from the vector z, and define $\tilde{T}_{\text{ref}} := \left(T_{28}^{\text{ref}}, T_{14}^{\text{ref}} \right)^T = (\, 70\,^\circ\text{C},\ 88\,^\circ\text{C}\,)^T$ for the desired setpoints.

7.3.1 Steady State Determination

Alternative A (Algebraic Constraints): A desired steady state x_S, z_S, and the corresponding control u_S could in principle be determined, for given parameters p, as a solution of the steady state equation

$$\begin{aligned} f(x_S, z_S, u_S, p) &= 0, \\ g(x_S, z_S, u_S, p) &= 0, \\ \tilde{T}z_S - \tilde{T}_{\text{ref}} &= 0. \end{aligned}$$

Here the last equation restricts the steady state to satisfy the inferential control aim of keeping the temperatures at the fixed reference values. The necessary degrees of freedom are provided by the two components of the steady state controls u_S. This approach was used in the first series of numerical and experimental tests [DBS+01, DUF+01].

Alternative B (End Point Constraint): In the practical computations in this thesis, however, we have adopted an alternative approach to determine the desired steady state: to this end note that the steady state x_S, z_S for given p and u_S could equally be determined by an integration of the model DAE over a sufficiently long time horizon with constant controls u_S, yielding x_S, z_S as final values, which are practically independent of the initial values. The requirement that the steady state should satisfy $\tilde{T}z_S = \tilde{T}_{\text{ref}}$ can then be formulated as a final state constraint that implicitly determines u_S. We employ this second approach to determine u_S by using an additional long prediction interval at the end of the control horizon in the problem formulation. Note that the use of this approach does not cause additional numerical effort if a prediction horizon is employed anyway; on the contrary, this formulation avoids introducing additional variables x_S, z_S into the NLP.

7.3.2 The Optimal Control Problem

Objective Function: The open-loop objective is formulated as the integral of a least squares term

$$L(z, u, u_S) := \|\tilde{T}z - \tilde{T}_{\text{ref}}\|_2^2 \quad + \quad \|R(u - u_S)\|_2^2\,, \tag{7.21}$$

where the second term is introduced for regularization, with a small diagonal weighting matrix

$$R = \begin{pmatrix} 0.05\,^\circ\text{C h l}^{-1} & 0 \\ 0 & 0.05\,^\circ\text{C kW}^{-1} \end{pmatrix}.$$

Prediction Interval: To ensure nominal stability of the closed-loop system, an additional prediction interval $[t_0 + T_c, t_0 + T_p]$ is appended to the control horizon $[t_0, t_0 + T_c]$, with the controls fixed to the setpoint values u_S. The objective contribution of this interval provides an upper bound of the neglected future costs that are due after the end of the control horizon, if its length is sufficiently large (cf. Sec. 1.4.1). A length of $T_p - T_c = 36000$ seconds has been considered to be sufficient in all performed experiments. Note that the optimized system state $x(t_0 + T_c)$ at the *start* of this interval (i.e., at the end of the control horizon) is in practice already very close to the desired steady state value x_S.

Problem Formulation

The resulting optimal control problem is formulated as follows:

$$\min_{u(\cdot),x(\cdot),p,u_S} \int_{t_0}^{t_0+T_p} \left\{ \left\| \tilde{T} z(t) - \tilde{T}_{\text{ref}} \right\|_2^2 \quad + \quad \| R(u(t) - u_S) \|_2^2 \right\} \, dt \tag{7.22}$$

subject to the model DAE

$$\begin{aligned} \dot{x}(t) &= f(x(t), z(t), u(t), p) \\ 0 &= g(x(t), z(t), u(t), p) \end{aligned} \qquad \text{for } t \in [t_0, t_0 + T_p].$$

Initial values for the differential states and values for the system parameters are prescribed:

$$\begin{aligned} x(t_0) &= x_0, \\ p &= p_0. \end{aligned}$$

State and control inequality constraints are formulated by

$$h(x(t), z(t), u(t), p) \geq 0 \quad t \in [t_0, t_0 + T_p],$$

where $h := (D, B)^T$ is the function calculating the fluxes D and B out of condenser and reboiler according to the model equations which cannot become negative. This implicitly provides upper limits to the controls.

The steady state control u_S is determined implicitly by the requirements that u is constant on the long prediction interval

$$u(t) = u_S \qquad \text{for } t \in [t_0 + T_c, t_0 + T_p],$$

and by the final state constraint

$$\tilde{T} z(t_0 + T_p) - \tilde{T}_{\text{ref}} = 0.$$

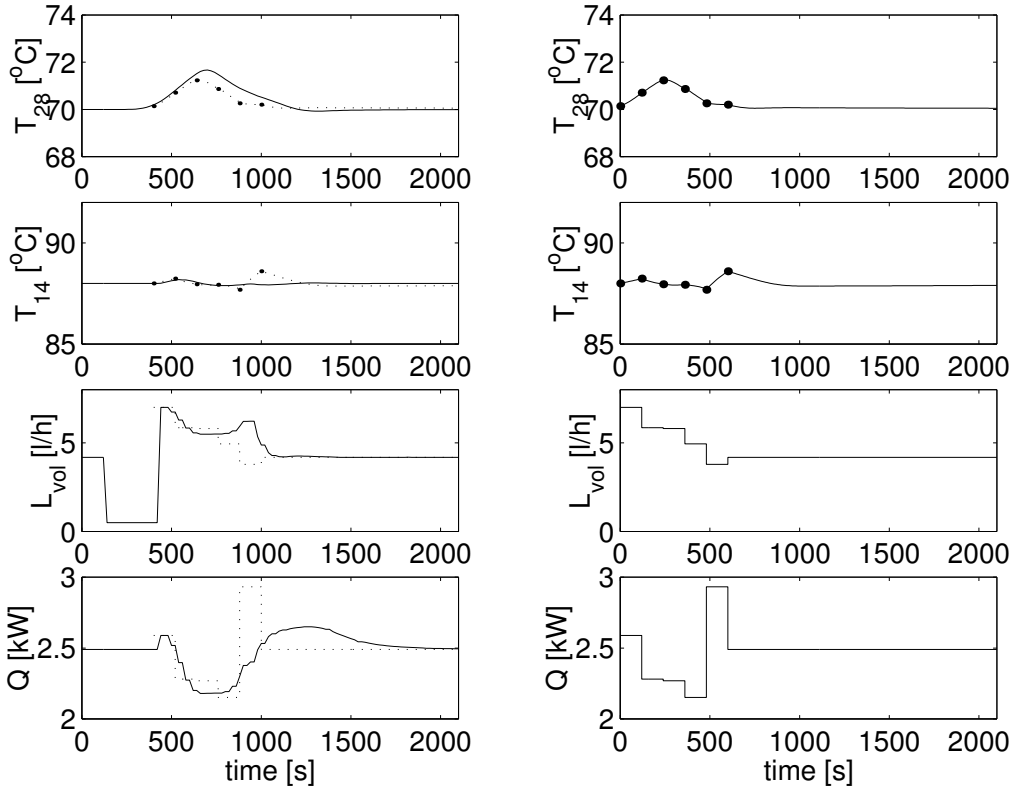


Figure 7.6: Numerical simulation of a closed-loop response after a reflux breakdown of five minutes and snapshot of a predicted trajectory (dotted). The right hand side shows the first 2100 seconds of the predicted trajectory, for the solution of the optimization problem at time t=400 seconds. The remaining 34500 seconds of the prediction horizon are not shown.

7.3.3 Numerical Realization

The length T_c of the control horizon and the control discretization have to be chosen such that the computation time for one real-time iteration does not exceed the relevant time scale of the system or of the disturbances. Based on numerical experiments on the available computer (AMD Athlon processor with 1009 MHz) and on the requirement that one real-time iteration should not exceed 20 seconds, we found that T_c=600 seconds with 5 control intervals each of 120 seconds length is a good choice. For a visualization of the control horizon, see the right hand side of Fig. 7.6, which shows an example solution profile.

As the control interval length is 6 times longer than the desired sampling time, the initialization strategy for subsequent real-time iterations was chosen to be the warm start strategy (cf. Sec. 4.4.2). For the Hessian approximation we have chosen the Gauss-Newton approach for least squares integrals that is described in Sec. 6.4.

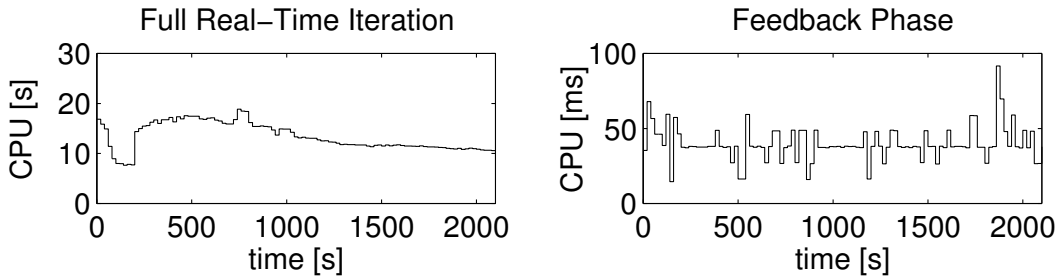


Figure 7.7: CPU times for the full real-time iteration cycles and time that is needed in the feedback phase, for the numerical experiment in Fig. 7.6.

As a first numerical test of the closed-loop algorithm we consider the following scenario: starting at the nominal operating conditions, a reflux breakdown happens and leaves the control inputs fixed to L_{vol}=0.5 l h^{-1} and Q=2.5 kW for a duration of five minutes. After these five minutes the plant can again be controlled by the NMPC scheme. The optimizer works all the time, even if the feedback is not given to the simulated plant. The closed-loop behaviour is shown in Fig. 7.6.

The necessary CPU times for each real-time iteration, as well as the recorded response times are shown in Fig. 7.7. Note that the response times are roughly two orders of magnitude smaller than the CPU time for one iteration.

7.4 Experimental Setup

As said above, we have tested the described NMPC scheme on the pilot plant distillation column for various scenarios. For comparison, we also performed closed-loop experiments with a conventional controller, namely a Proportional Integral (PI) control scheme. We describe in this section how the two schemes were practically set up.

7.4.1 NMPC Controller Setup

On-Line State Estimation

To obtain an estimate of the 82 differential system states and of the model parameter X_F by measurements of the three temperatures T_{14}, T_{21} and T_{28} only, we have implemented a variant of an Extended Kalman Filter (EKF).

In contrast to an ordinary EKF our estimator can incorporate additional knowledge about the possible range of states and parameters in form of bounds. This is especially useful as the tray concentrations need to be constrained to be in the interval $[0, 1]$ from physical reasoning. The algorithm is described in Appendix A. A comparison of esti-

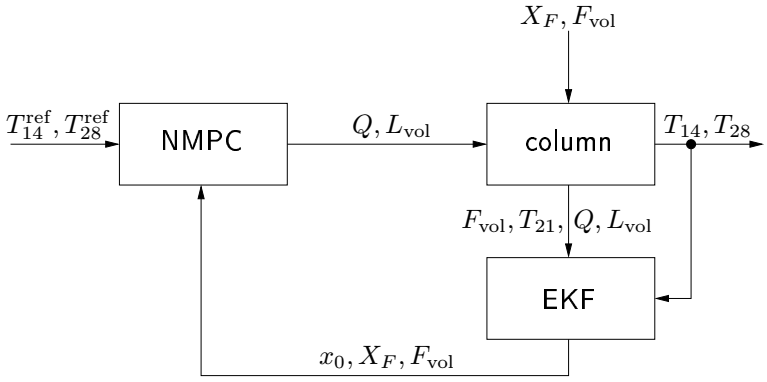


Figure 7.8: Closed-loop NMPC setup

mated and measured temperature profiles can be found in Fig. 7.15 – note that only the temperatures T_{14}, T_{21} and T_{28} are available to the state estimator.

The described EKF type algorithm is currently extended by Bürner to a moving horizon estimator [Bür01]. This new algorithm – so far with a horizon length of 10 seconds only – was already employed for one of the closed-loop experiments, which involved a step change in X_F (cf. Fig. 7.12). The performance in the estimation of X_F can be seen in Fig. 7.13. The large estimation offset is due to model-plant mismatch.

Coupling with the Process Control System

As mentioned above, the distillation column is controlled by a lower level distributed control system (DCS), which is connected to a PC (cf. Fig. 7.1). Access to this PC from UNIX workstations is possible via ftp, so that all higher level algorithms, in particular the state estimator and the real-time iteration scheme, could be implemented on a powerful LINUX workstation with an AMD Athlon processor (1009 MHz). With the given equipment it was only possible to obtain measurements and to write the computed control inputs to the DCS every 10 seconds, i.e., a sampling time of 10 seconds was used for the state estimator. The real-time iteration scheme was implemented in a self-synchronizing way (cf. Sec. 4.4.2), which made it robust against CPU load changes due to other users; its adaptive sampling time did in practice never exceed 20 seconds (cf. Fig. 7.16).

The three processes – data acquisition, state estimation and real-time optimization – were running independently and communicating only via input and output files, in such a way that a breakdown of one component did not cause an immediate breakdown of the others. Missing new inputs were simply replaced by old values. This construction made the whole system sufficiently stable against unexplained delays in the data transfer between the UNIX workstation and the PC. Figure 7.8 shows the overall controller/plant/estimator setup.

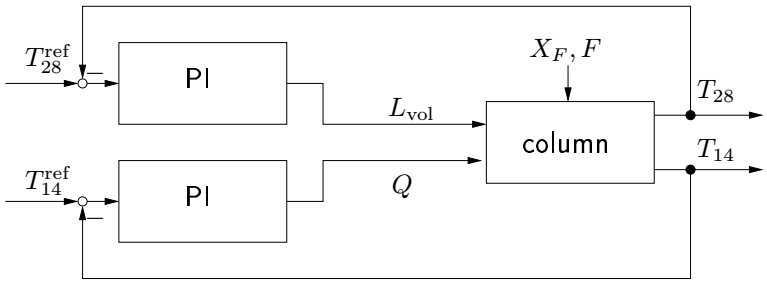


Figure 7.9: Closed-loop PI setup

7.4.2 PI Controller Setup

To be able to assess the performance of the proposed NMPC scheme, we also carried out experiments with an existing PI controller that is usually employed for the control of the column. This conventional control scheme consists of two decoupled single-input/single-output PI loops, one of which uses the heat input Q to control the temperature T_{14}, the other using the reflux L_{vol} to control the temperature T_{28}.

The controlled variables are, as in the NMPC case, the temperatures T_{14} and T_{28}. The manipulated variables are the heat input Q to the boiler (corresponding to the liquid flow V_0 out of the boiler) and the reflux flow L_{vol}. The PI setup is shown in Fig. 7.9.

7.5 Experimental Results

We have tested the NMPC scheme and the PI control scheme on various scenarios. As scenarios we used step changes in the feed flow rate (F_{vol}); a step change in the feed composition (X_F); a short reflux breakdown of five minutes (L_{vol}); and a large disturbance scenario where the column was driven with too much heat input and too low reflux flow for over ten minutes.

7.5.1 Feed Flow Change

Figure 7.10 shows the controlled outputs (T_{28} and T_{14}) and input responses (L_{vol} and Q) where the feed flow rate F_{vol} is changed by -10 % at time $t = 1000$ seconds. The plots on the left hand side show the results of the NMPC scheme and those on the right hand side belong to the PI controller. It can be seen that the performance of the NMPC scheme is better than that of the PI controller, both with respect to the size of the oscillation, mainly in T_{28}, and with respect to the attenuation time: 1000 seconds after the feedflow change the system is more or less in the new steady state, whereas the PI closed-loop system is still oscillating 3000 seconds after the load change. In Fig. 7.11 we show a second step change in F_{vol}. Starting from the steady state for a feedflow that is reduced by -10 % from its nominal value, we increase it at time $t = 1000$ sec. by 20 %, to + 10 % of the

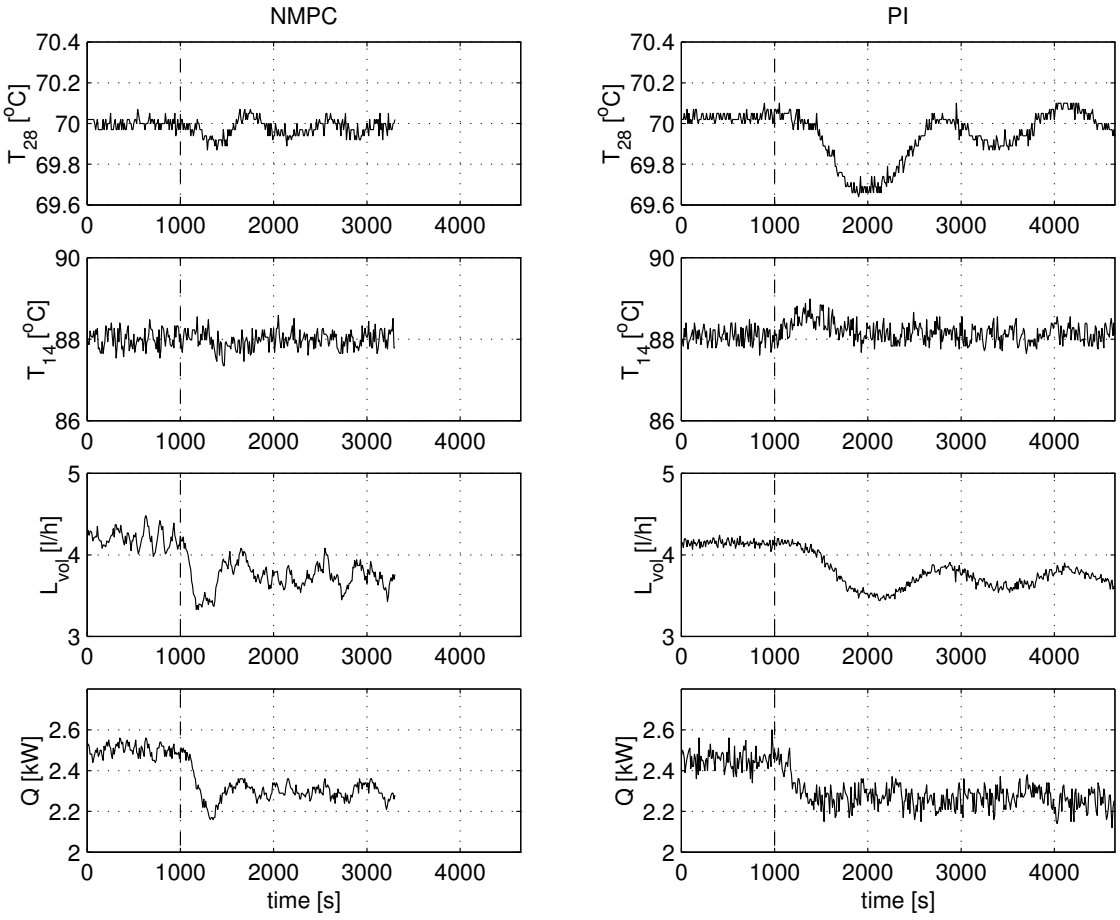


Figure 7.10: Feed flow change: Comparison of real-time iteration NMPC with a conventional PI controller, for a step reduction of 10 % in the feed flow F_{vol}.

nominal value. Again, the NMPC performance is considerably better, having completed the transition 1000 seconds after the feed flow change, and with a maximum deviation in T_{28} of 0.3°C. This is in sharp contrast to the PI performance, which has a maximum deviation of 0.8°C, and which did not even complete the transition to the new steady state 3500 seconds after the step change.

7.5.2 Feed Concentration Change

For the next test, a step change in the feed composition is considered; X_F is decreased from 0.320 to 0.272 at t=1000 sec. In Fig. 7.12, the NMPC closed-loop response is compared to that of the PI controller; here the NMPC controller shows no superior performance to the PI. The steady state offset can be explained by the fact that the NMPC controller does *not* have an integral term (as the PI controller), that is able to remove steady state offset

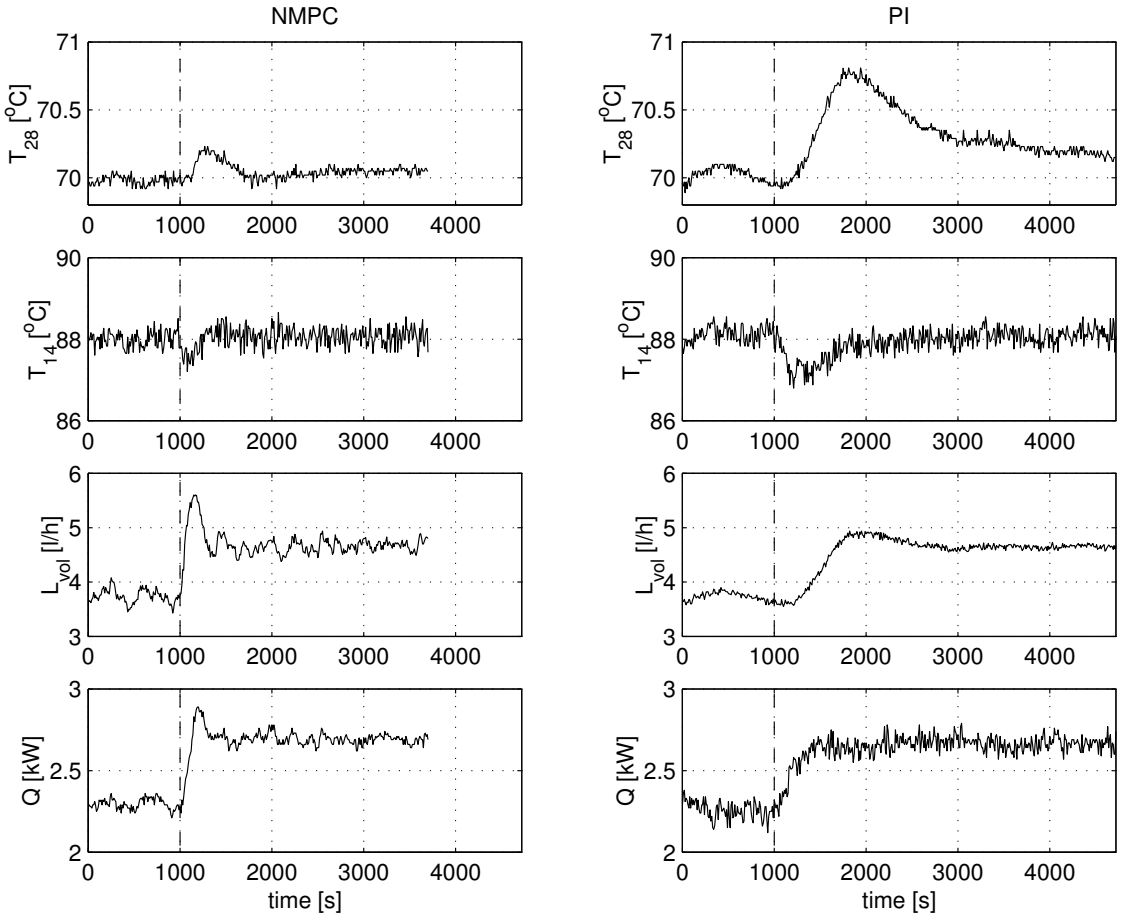


Figure 7.11: Feed flow change: Comparison of real-time iteration NMPC with a conventional PI controller, for a feed flow step change by 20 % (from -10% to +10 % of the nominal value).

in the presence of model-plant mismatch, which has been increased due to the change in X_F. Note that the NMPC performance depends crucially on the quality of the on-line state and parameter estimates, as the jump in X_F has to be detected correctly to yield an appropriate response. For a comparison of estimated and real values of X_F, see Fig. 7.13. It can be seen that it took roughly 600 seconds to detect the composition change completely.

7.5.3 Short Reflux Breakdown

In the previous two cases the disturbing effects of load changes (in the feed flow and composition) on the controlled temperatures T_{28} and T_{14} are relatively small. In order to have a larger disturbance effect we simulated a short reflux flow breakdown, i.e., we fixed

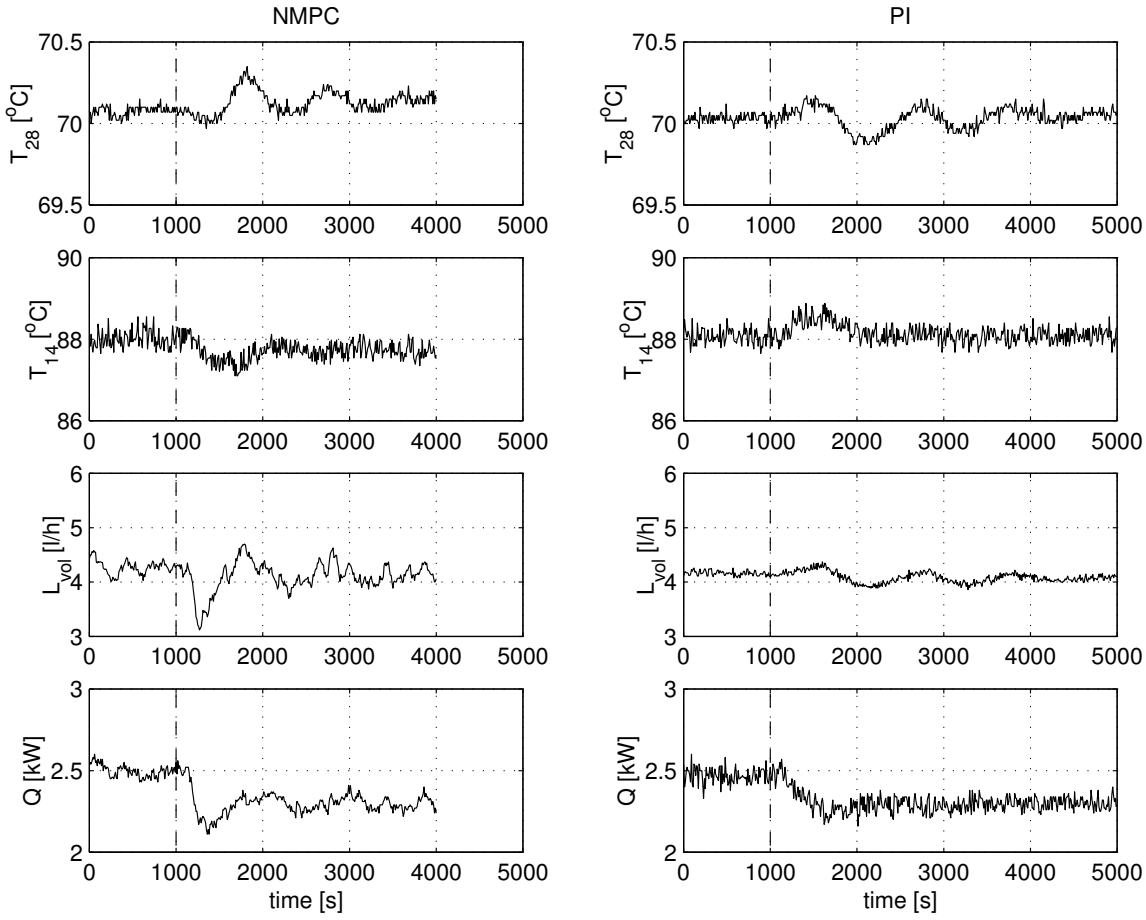


Figure 7.12: Feed concentration change: Comparison of real-time iteration NMPC with a conventional PI controller, for a feed concentration change from $X_F = 0.32$ to $X_F = 0.275$.

the inputs for five minutes, setting the reflux to a very small value of $L_{\text{vol}} = 0.5$ l/h. At time $t = 1000$ sec., the reflux breakdown is assumed to be over, so that feedback can again be applied to the column. The closed-loop responses of NMPC and PI controllers are shown in Fig. 7.14 (this result can also be found in [DUF+01]). Note that both controllers start with the same system state at $t = 1000$ sec.; the PI performance is worse than the NMPC scheme, as T_{28} is increasing up to 72°C, whereas it only increases to 71.3°C for the NMPC scheme, and the disturbance effects last until 3000 sec. after the disturbance, compared to less than 2000 sec. for NMPC.

Valve Saturation: The micro reflux breakdown that happens for both scenarios in the feedback phase is due to *valve saturation*, i.e., due to the fact that the filling level of the reflux drum was shortly becoming too low because more reflux L was desired than the amount of vapor flow V_N entering the condenser (cf. Fig. 7.2). This causes an automatic

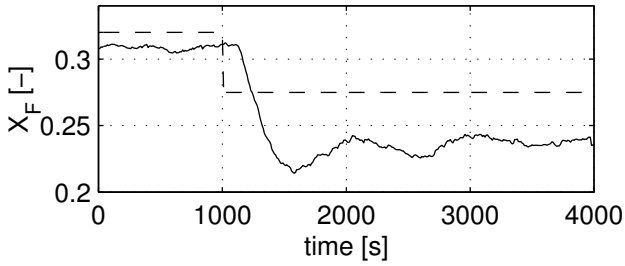


Figure 7.13: On-line estimate of X_F (solid), compared to the real value (dashed), in the scenario of Fig. 7.12.

stop of the reflux pump. In the NMPC scheme, this phenomenon should have been avoided by the constraint $D \geq 0$, i.e., the requirement that the distillate outflow remains non-negative (note that in the model the condenser hold up is assumed constant). However, due to model-plant mismatch, the constraint was violated in the real-plant even though it was satisfied in the model prediction. To account for the uncertainty, we have sharpened the constraint to $D \geq 0.2 \cdot 10^{-5} \frac{\text{kmol}}{\text{sec}}$ in the following experiments, to provide a security margin of 10 % of the nominal value of D. For the PI controllers, there is no easy way to circumvent valve saturation in the presence of large disturbances; therefore we did not perform the large disturbance scenario with the PI controllers.

7.5.4 Large Disturbance Scenario

To have even larger disturbance effects, we consider the following scenario: starting with a steady state for an increased feed flow rate (by 20 %), we reduce at time $t = 700$ seconds simultaneously the feedflow (back to its nominal value) and the reflux, from $L_{\text{vol}} = 5.3 \frac{\text{l}}{\text{h}}$ down to $L_{\text{vol}} = 2 \frac{\text{l}}{\text{h}}$, while maintaining the heating power constant at its (high) value $Q = 2.9\,\text{kW}$. These inputs, that are maintained constant for 800 seconds, heat the column up and move the temperature profile far away from the nominal operationg conditions, as can be seen in the right hand side of Fig. 7.15, where the distorted temperature profile at time $t = 1500$ is shown. Only at this time the NMPC feedback is switched on. The closed-loop response can be seen on the left hand side in Fig. 7.15. While Q jumps immediately down to its minimum value of 1.5kW, L_{vol} is *not* increased to its maximum value, as would from first sight be the best thing to cool the column. However, this would have resulted in valve saturation, as discussed above; it was the constraint $D \geq 0.2 \cdot 10^{-5} \frac{\text{kmol}}{\text{sec}}$ that caused this interesting feature of the closed-loop behaviour.

7.5.5 Brief Discussion

We have seen that the proposed real-time iteration NMPC control scheme is not only feasible for a practical large scale application, but that it results in a good performance when

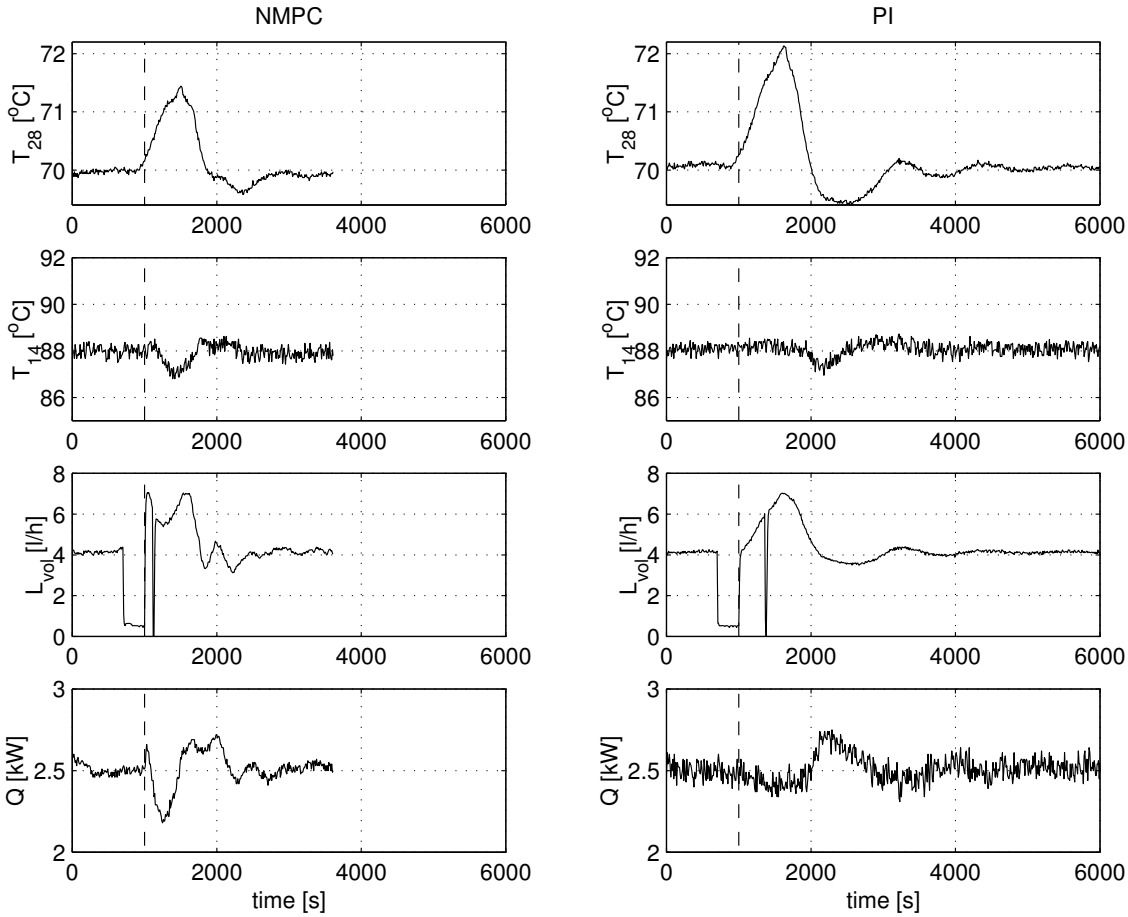


Figure 7.14: Reflux breakdown: Comparison of real-time iteration NMPC with a conventional PI controller, after a reflux breakdown of 5 minutes. At time t=1000 s both closed-loop scenarios start at the same state.

no estimation difficulties exist. The poorest performance occurred in the feed composition change scenario, where the state estimator was not able to track the system parameter X_F instantly. On the other hand, the NMPC scheme shows good performance when constraints play a role, which are difficult to handle with a PI control scheme. Especially the closed-loop response of the large disturbance scenario shows interesting features and deserves further analysis. We will have a closer look at the observed real-time performance, and we will also compare the experimentally observed trajectory with computations that have been performed a posteriori.

Observed Computation Times: Let us first have a look on the computation times under experimental conditions. We measure not only the overall time for each real-time iteration, but also the response time, i.e., the time between the moment that the current

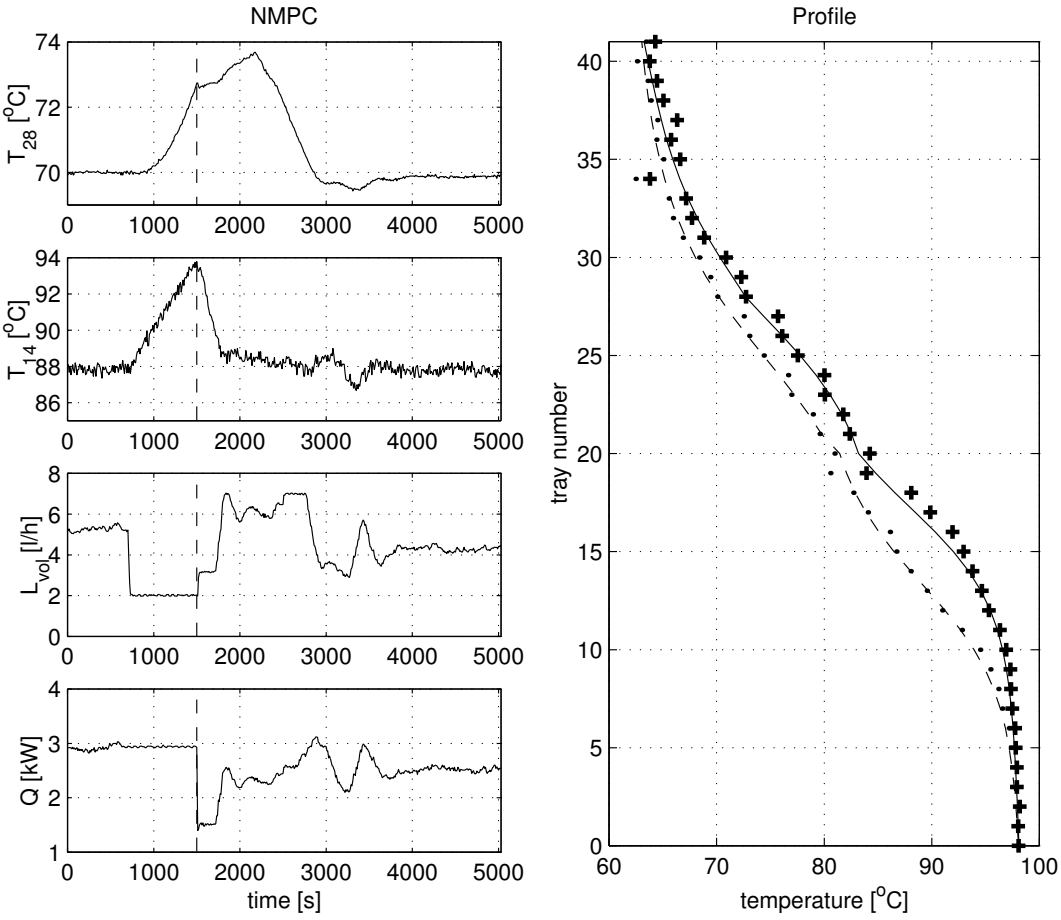


Figure 7.15: Large disturbance scenario. Left: Closed-loop response. Feedback starts only at time $t = 1500$ seconds. The temperature profile at this time is shown on the right hand side (+), together with the estimated profile (solid) and compared to the nominal profile (dots/dashed).

observed state x_0 is given to the optimizer, and the moment that the control response is available for the data transfer to the column. Both time measurements were done externally (from a MATLAB environment), i.e., they are not CPU times in the strict sense, but the overall times that the computations required under the given CPU load conditions. The observed times can be seen in Fig. 7.16. Note that due to the fact that the communication sampling rate was technically restricted to be not shorter than 10 seconds, the immediate response may in our realization have taken up to 10 seconds until it arrives at the distillation column, depending on the phase difference of the (self-synchronizing) optimizer and the data transfer system.

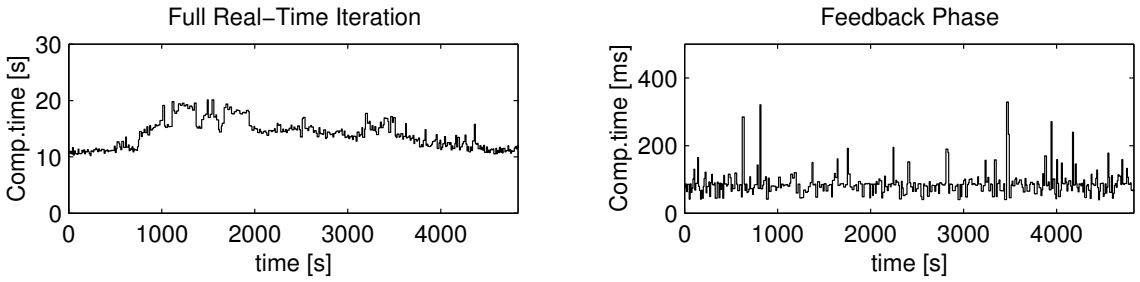


Figure 7.16: Overall computation times for the full real-time iteration cycles and the time that is needed in the feedback phase, for the large disturbance experiment (cf. Fig. 7.15).

Prediction Horizon Contents: To analyse the working of the NMPC controller better, we will have a look at the *predicted* control trajectory at the time point $t = 1500$ seconds, and compare it to the final closed-loop response, in Fig. 7.17. It can be seen that the predicted and real trajectory differ significantly. This is mainly caused by the fact that the control horizon of 600 seconds is too short to capture all necessary control actions for a disturbance as large as the one considered.

We will compare the experimentally observed performance with a *simulated* closed-loop trajectory, and with the optimal solution, according to the model.

Closed-Loop Simulation: It is interesting to test how similar the experimental result in Fig. 7.15 is to a closed-loop *simulation*, where noise effects and model-plant mismatch do not play a role. We have therefore taken the (estimated) system state at time $t = 1500$, to start a closed-loop simulation, using the same controller setup as before, but under the assumption that the plant is identical to the model. The result of this simulation can be seen in the central column in Fig. 7.18.

Optimal Solution: For completeness, the experimental and simulated closed-loop trajectories are compared with a theoretical off-line result, namely with the optimal open-loop trajectory, that can be seen on the right column of Fig. 7.18. It can be seen that the experimental and simulated closed-loop trajectories show considerable similarity with the theoretically optimal solution.

The computation of the optimal trajectory with the off-line multiple shooting method required 23 major SQP iterations with a CPU time of 3356 seconds (AMD Athlon processor with 1009 MHz), where the control horizon was chosen to consist of 45 multiple shooting intervals, each of 30 seconds length. Note that the computation time for this problem is in the same order as the whole process duration.

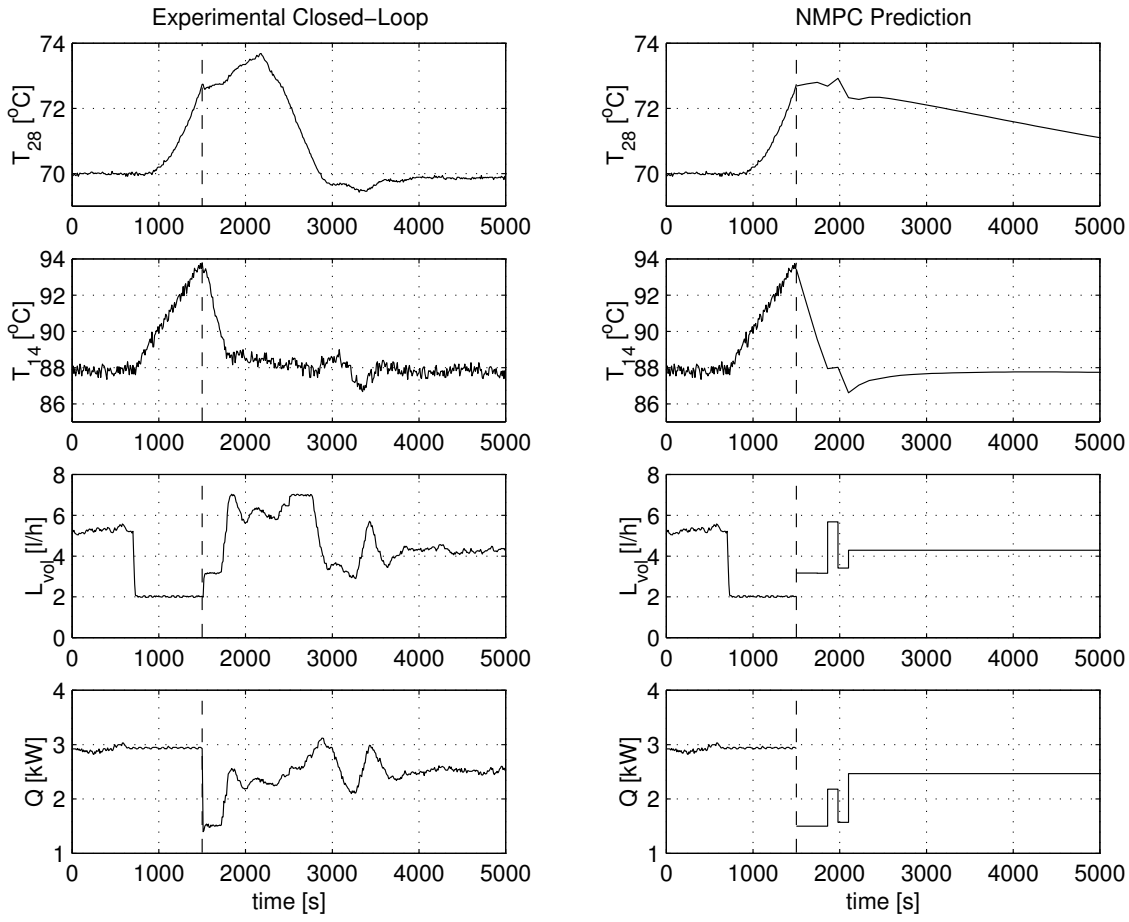


Figure 7.17: Comparison of experimental closed-loop trajectory (left) with the NMPC prediction horizon at time $t = 1500$ seconds (right).

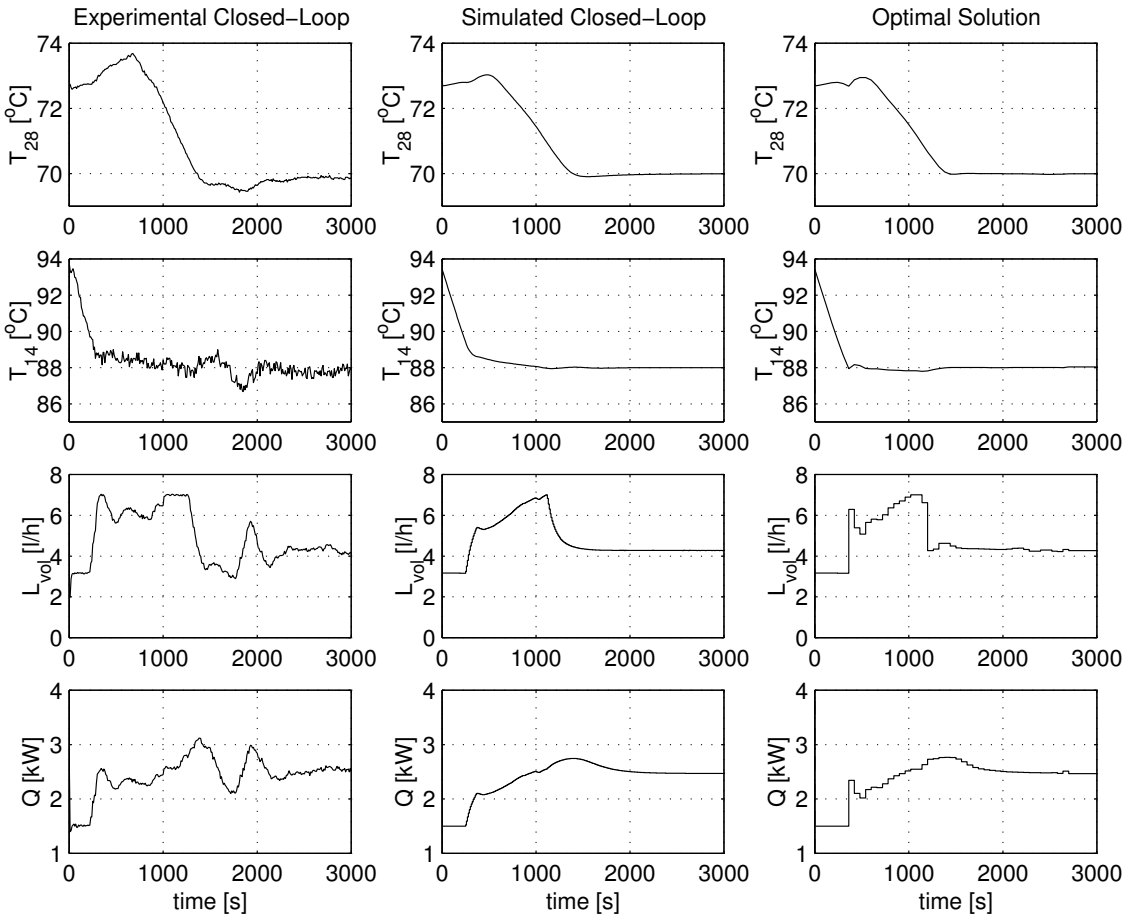


Figure 7.18: Comparison of experimental closed-loop trajectory (left) with simulated closed-loop (center) and theoretical optimal trajectory (right).

Chapter 8

Control of a Looping Kite

In order to demonstrate the versatility of the proposed real-time iteration scheme we present here the control of an airborne kite as a periodic control example. The kite is held by two lines which allow to control the lateral angle of the kite, see Fig. 8.1. By pulling one line the kite will turn in the direction of the line being pulled. This allows an experienced kite pilot to fly loops or similar figures. The aim of our automatic control is to make the kite fly a figure that may be called a "lying eight", with a cycle time of 8 seconds (see Fig. 8.2). The corresponding orbit is not open-loop stable, so that feedback has to be applied during the flight – we will show simulation results where our proposed real-time iteration scheme was used to control the kite, with a sampling time of one second.

8.1 The Dual Line Kite Model

The movement of the kite at the sky can be modelled by Newton's laws of motion and a suitable model for the aerodynamic force. Most difficulty lies in the determination of suitable coordinate systems: we will first describe the kite's motion in polar coordinates, and secondly determine the direction of the aerodynamic forces.

8.1.1 Newton's Laws of Motion in Polar Coordinates

The position $p \in \mathbb{R}^3$ of the kite can be modelled in 3-dimensional Euclidean space, choosing the position of the kite pilot as the origin, and the third component p_3 to be the height of the kite above the ground. With m denoting the mass of the kite and $F \in \mathbb{R}^3$ the total force acting on the kite, Newton's law of motion reads

$$\ddot{p} = \frac{d^2 p}{dt^2} = \frac{F}{m}.$$

Let us introduce polar coordinates θ, ϕ, r:

$$p = \begin{pmatrix} p_1 \\ p_2 \\ p_3 \end{pmatrix} = \begin{pmatrix} r\sin(\theta)\cos(\phi) \\ r\sin(\theta)\sin(\phi) \\ r\cos(\theta) \end{pmatrix}.$$

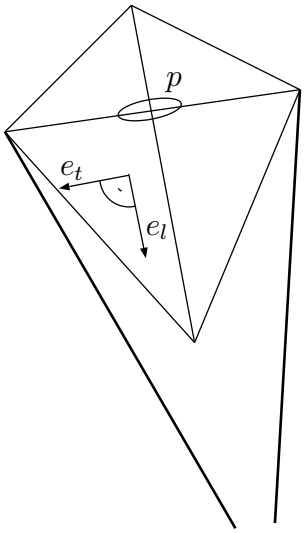


Figure 8.1: A picture of the kite.

Note that the distance r between pilot and kite is usually constant during flight, and θ is the angle that the lines form with the vertical. In these coordinates, $\ddot{p}$ looks as follows

$$\begin{aligned} \ddot{p} &= \frac{d}{dt}\left(\frac{\partial p}{\partial \theta}\dot{\theta} + \frac{\partial p}{\partial \phi}\dot{\phi} + \frac{\partial p}{\partial r}\dot{r}\right) \\ &= \frac{\partial p}{\partial \theta}\ddot{\theta} + \frac{\partial p}{\partial \phi}\ddot{\phi} + \frac{\partial p}{\partial r}\ddot{r} + \frac{\partial^2 p}{\partial \theta^2}\dot{\theta}^2 + \frac{\partial^2 p}{\partial \phi^2}\dot{\phi}^2 + \frac{\partial^2 p}{\partial r^2}\dot{r}^2 \\ &+ 2\frac{\partial^2 p}{\partial \phi \partial \theta}\dot{\phi}\dot{\theta} + 2\frac{\partial^2 p}{\partial r \partial \theta}\dot{r}\dot{\theta} + 2\frac{\partial^2 p}{\partial r \partial \phi}\dot{r}\dot{\phi}. \end{aligned} \tag{8.1}$$

Let us introduce a local right handed coordinate system with the three basis vectors

$$e_\theta = \begin{pmatrix} \cos(\theta)\cos(\phi) \\ \cos(\theta)\sin(\phi) \\ -\sin(\theta) \end{pmatrix}, \quad e_\phi = \begin{pmatrix} -\sin(\phi) \\ \cos(\phi) \\ 0 \end{pmatrix}, \quad \text{and} \quad e_r = \begin{pmatrix} \sin(\theta)\cos(\phi) \\ \sin(\theta)\sin(\phi) \\ \cos(\theta) \end{pmatrix}.$$

In this coordinate system, the partial derivatives of p with respect to θ, ϕ, r become

$$\frac{\partial p}{\partial \theta} = r e_\theta, \quad \frac{\partial p}{\partial \phi} = r\sin(\theta) e_\phi, \quad \text{and} \quad \frac{\partial p}{\partial r} = e_r,$$

and

$$\frac{\partial^2 p}{\partial \theta^2} = -r e_r, \quad \frac{\partial^2 p}{\partial \phi^2} = -r\sin^2(\theta) e_r - r\sin(\theta)\cos(\theta) e_\theta, \quad \text{and} \quad \frac{\partial^2 p}{\partial r^2} = 0,$$

as well as

$$\frac{\partial^2 p}{\partial \phi \partial \theta} = r\cos(\theta) e_\phi, \quad \frac{\partial^2 p}{\partial r \partial \theta} = e_\theta, \quad \text{and} \quad \frac{\partial^2 p}{\partial r \partial \phi} = \sin(\theta) e_\phi.$$

Eq. (8.1) can therefore be written as:

$$\begin{aligned}\ddot{p} &= e_\theta \left(r\ddot{\theta} - r\sin(\theta)\cos(\theta)\dot{\phi}^2 + 2\dot{r}\dot{\theta}\right) \\ &+ e_\phi \left(r\sin(\theta)\ddot{\phi} + 2r\cos(\phi)\dot{\phi}\dot{\theta} + 2\sin(\theta)\dot{r}\dot{\phi}\right) \\ &+ e_r \left(\ddot{r} - r\dot{\theta}^2 - r\sin^2(\theta)\dot{\phi}^2\right).\end{aligned}$$

Defining

$$F_\theta := F \cdot e_\theta, \quad F_\phi := F \cdot e_\phi, \quad \text{and} \quad F_r := F \cdot e_r,$$

we can write Newton's laws of motion in the form

$$\begin{aligned} r\ddot{\theta} - r\sin(\theta)\cos(\theta)\dot{\phi}^2 + 2\dot{r}\dot{\theta} &= \frac{F_\theta}{m}, \\ r\sin(\theta)\ddot{\phi} + 2r\cos(\theta)\dot{\phi}\dot{\theta} + 2\sin(\theta)\dot{r}\dot{\phi} &= \frac{F_\phi}{m}, \\ \ddot{r} - r\dot{\theta}^2 - r\sin^2(\theta)\dot{\phi}^2 &= \frac{F_r}{m}. \end{aligned} \tag{8.2}$$

If the length of the lines, denoted by r, is kept constant, all terms involving time derivatives of r will drop out. Furthermore, the last equation (8.2) will become redundant, as the force in the radial direction will be augmented by a constraint force contribution F_c, so that Eq. (8.2) is automatically satisfied when the augmented force $F_r' := F_r - F_c$ replaces F_r, with $F_c = F_r + r\dot{\theta}^2 + r\sin^2(\theta)\dot{\phi}^2$. In this case the equations of motion[1] simplify to

$$\ddot{\theta} = \frac{F_\theta}{rm} + \sin(\theta)\cos(\theta)\dot{\phi}^2, \tag{8.3}$$

$$\ddot{\phi} = \frac{F_\phi}{rm} - 2\cot(\theta)\dot{\phi}\dot{\theta}. \tag{8.4}$$

In our model, the force vector $F = F^{\text{gra}} + F^{\text{aer}}$ consists of two contributions, the gravitational force F^{gra} and the aerodynamic force F^{aer}. In cartesian coordinates, $F^{\text{gra}} = (0, 0, -mg)^T$ with $g = 9.81$ m s^{-2} being the earth's gravitational acceleration. In local coordinates we therefore have

$$F_\theta = F_\theta^{\text{gra}} + F_\theta^{\text{aer}} = \sin(\theta)mg + F_\theta^{\text{aer}} \quad \text{and} \quad F_\phi = F_\phi^{\text{aer}}.$$

It remains to derive an expression for the aerodynamic force F^{aer}.

8.1.2 Kite Orientation and the Aerodynamic Force

To model the aerodynamic force that is acting on the kite, we first assume that the kite's trailing edge is always pulled by the tail into the direction of the effective wind, as seen

[1] Note that the validity of these equations requires that $F_c = F_r + r\dot{\theta}^2 + r\sin^2(\theta)\dot{\phi}^2 \geq 0$, as a line can only pull, not push.

Name	Symbol	Value
line length	r	50 m
kite mass	m	1 kg
wind velocity	v_w	6 m/s
density of air	ρ	1.2 kg/m^3
characteristic area	A	0.5 m^2
lift coefficient	C_l	1.5
drag coefficient	C_d	0.29

Table 8.1: The kite parameters.

from the kite's inertial frame. Under this assumption the kite's longitudinal axis is always in line with the effective wind vector $w_e := w - \dot{p}$, where $w = (v_w, 0, 0)^T$ is the wind as seen from the earth system, and $\dot{p}$ the kite velocity. If we introduce a unit vector e_l pointing from the front towards the trailing edge of the kite (cf. Fig. 8.1), we therefore assume that

$$e_l = \frac{w_e}{\|w_e\|}.$$

The transversal axis of the kite can be described by a perpendicular unit vector e_t that is pointing from the left to the right wing tip. Clearly, it is orthogonal to the longitudinal axis, i.e.,

$$e_t \cdot e_l = \frac{e_t \cdot w_e}{\|w_e\|} = 0. \tag{8.5}$$

The orientation of the transversal axis e_t against the lines' axis (which is given by the vector e_r) can be influenced by the length difference Δl of the two lines. If the distance between the two lines' fixing points on the kite is d, then the vector from the left to the right fixing point is de_t, and the projection of this vector onto the lines' axis should equal Δl (being positive if the right wingtip is farther away from the pilot), i.e., $\Delta l = d\, e_t \cdot e_r$. Let us define the *lateral angle* ψ to be

$$\psi = \arcsin\left(\frac{\Delta l}{d}\right).$$

We will assume that we control this angle ψ directly. It determines the orientation of e_t which has to satisfy:

$$e_t \cdot e_r = \frac{\Delta l}{d} = \sin(\psi). \tag{8.6}$$

A third requirement that e_t should satisfy is that

$$(e_l \times e_t) \cdot e_r = \frac{w_e \times e_t}{\|w_e\|} \cdot e_r > 0, \tag{8.7}$$

which takes account of the fact that the kite is always in the same orientation with respect to the lines.

How to find a vector e_t that satisfies these requirements (8.5)–(8.7)? Using the projection w_e^p of the effective wind vector w_e onto the tangent plane spanned by e_θ and e_ϕ,

$$w_e^p := e_\theta(e_\theta \cdot w_e) + e_\phi(e_\phi \cdot w_e) = w_e - e_r(e_r \cdot w_e),$$

we can define the orthogonal unit vectors

$$e_w := \frac{w_e^p}{\|w_e^p\|} \quad \text{and} \quad e_o := e_r \times e_w,$$

so that (e_w, e_o, e_r) form an orthogonal right-handed coordinate basis. Note that in this basis the effective wind w_e has no component in e_o direction, as

$$w_e = \|w_e^p\| e_w + (w_e \cdot e_r) e_r.$$

We will show that the definition

$$e_t := e_w(-\cos(\psi)\sin(\eta)) + e_o(\cos(\psi)\cos(\eta)) + e_r \sin(\psi)$$

with

$$\eta := \arcsin\left(\frac{w_e \cdot e_r}{\|w_e^p\|}\tan(\psi)\right)$$

satisfies the requirements (8.5)–(8.7). Equation (8.5) can be verified by substitution of the definition of η into

$$e_t \cdot w_e = -\cos(\psi)\sin(\eta)\|w_e^p\| + \sin(\psi)(w_e \cdot e_r) = 0.$$

Eq. (8.6) is trivially satisfied, and Eq. (8.7) can be verified by calculation of

$$\begin{aligned}(w_e \times e_t) \cdot e_r &= (w_e \cdot e_w)\cos(\psi)\cos(\eta) - (w_e \cdot e_o)(-\cos(\psi)\sin(\eta)) \\ &= \|w_e^p\|\cos(\psi)\cos(\eta)\end{aligned}$$

(where we used the fact that $w_e \cdot e_o = 0$). For angles ψ and η in the range from $-\pi/2$ to $\pi/2$ this expression is always positive. The above considerations allow to determine the orientation of the kite depending on the control ψ and the effective wind w_e only. Note that the considerations would break down if the effective wind w_e would be equal to zero, or if

$$\left|\frac{w_e \cdot e_r}{w_e \cdot e_w}\tan(\psi)\right| > 1.$$

The two vectors $e_l \times e_t$ and e_l are the directions of aerodynamic lift and drag, respectively. To compute the magnitudes F_l and F_d of lift and drag we assume that the lift and drag coefficients C_l and C_d are constant, so that we have

$$F_l = \frac{1}{2}\rho\|w_e\|^2 A C_l \quad \text{and} \quad F_d = \frac{1}{2}\rho\|w_e\|^2 A C_d,$$

with ρ being the density of air, and A being the characteristic area of the kite.

Given the directions and magnitudes of lift and drag, we can compute F^{aer} as their sum, yielding

$$F^{\text{aer}} = F_l(e_l \times e_t) + F_d e_l$$

or, in the local coordinate system

$$F_\theta^{\text{aer}} = F_l((e_l \times e_t) \cdot e_\theta) + F_d(e_l \cdot e_\theta) \text{ and } F_\phi^{\text{aer}} = F_l((e_l \times e_t) \cdot e_\phi) + F_d(e_l \cdot e_\phi).$$

The system parameters that have been chosen for the simulation model are listed in Table 8.1. Defining the system state $x := (\theta, \dot{\theta}, \phi, \dot{\phi})^T$ and the control $u := \psi$ we can summarize the system equations (8.3)–(8.4) in the short form

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

with

$$f((\theta, \dot{\theta}, \phi, \dot{\phi})^T, \psi) := \begin{pmatrix} \dot{\theta} \\ \dfrac{F_\theta^{\text{aer}}(\theta, \dot{\theta}, \phi, \dot{\phi}, \psi)}{rm} + \sin(\theta)\dfrac{g}{r} + \sin(\theta)\cos(\theta)\dot{\phi}^2 \\ \dot{\phi} \\ \dfrac{F_\phi^{\text{aer}}(\theta, \dot{\theta}, \phi, \dot{\phi}, \psi)}{rm} - 2\cot(\theta)\dot{\phi}\dot{\theta} \end{pmatrix}.$$

8.2 A Periodic Orbit

Using the above system model, a periodic orbit was determined that can be characterized as a "lying eight" and which is depicted as a $\phi - \theta$–plot in Fig. 8.2, and as a time plot in Fig. 8.3. The wind is assumed to blow in the direction of the p_1-axis ($\theta = 90^o$ and $\phi = 0^o$). The periodic solution was computed using the off-line variant of MUSCOD-II, imposing periodicity conditions with period $T = 8$ seconds and suitable state bounds and a suitable objective function in order to yield a solution that was considered to be a meaningful reference orbit. Note that the control ψ (see Fig. 8.3) is positive when the kite shall turn in a clockwise direction, as seen from the pilot's viewpoint, and negative for an anti-clockwise direction. We will denote the periodic reference solution by $x_r(t)$ and $u_r(t)$. This solution is defined for all $t \in (-\infty, \infty)$ and satisfies the periodicity condition $x_r(t + T) = x_r(t)$ and $u_r(t + T) = u_r(t)$.

It is interesting to note that small errors accumulate very quickly so that the uncontrolled system will not stay in the periodic orbit very long during a numerical simulation (see Fig. 8.4). This observation can be confirmed by investigating the asymptotic stability properties of the periodic orbit.

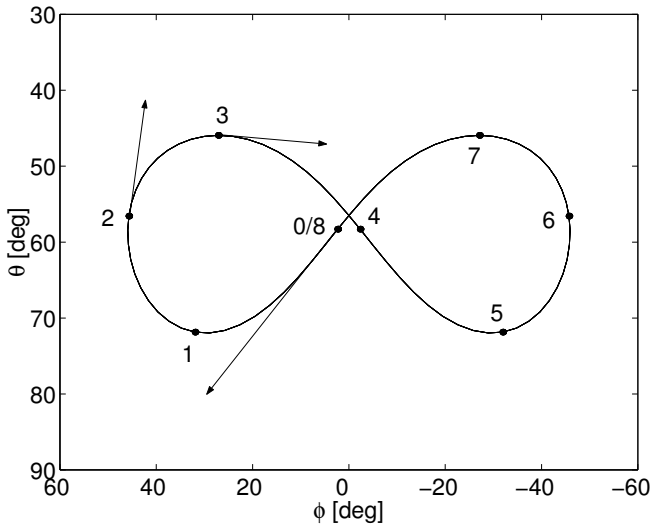


Figure 8.2: Periodic orbit plotted in the $\phi - \theta-$plane, as seen by the kite pilot. The dots separate intervals of one second.

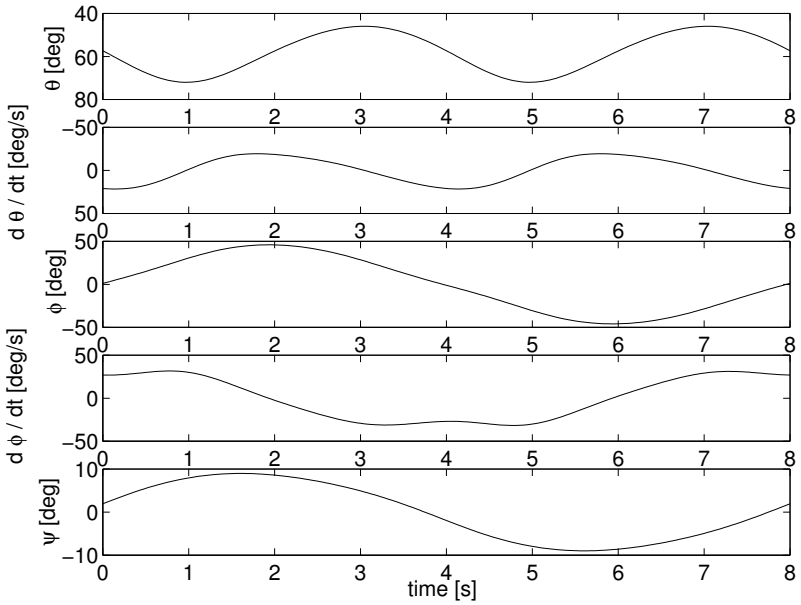


Figure 8.3: Periodic orbit: system states and control ψ plotted for one period $T = 8$ s . Note that θ and $\dot{\theta}$ oscillate with double frequency.

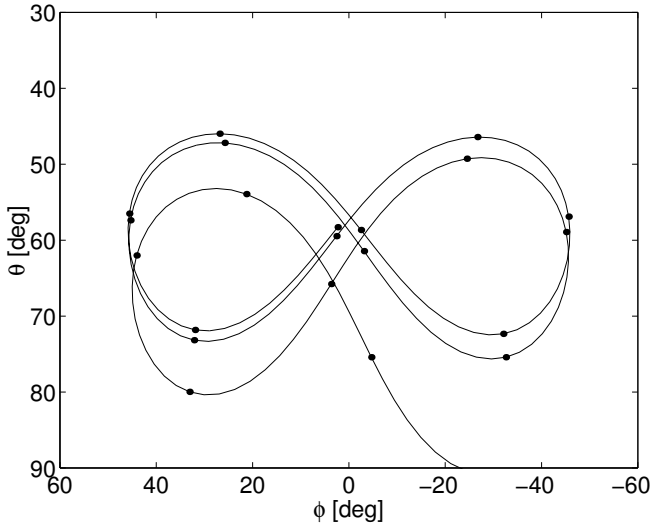


Figure 8.4: Open-loop control applied to the undisturbed system.

8.2.1 Stability Analysis of the Open-Loop System

To determine the asymptotic stability properties of the open-loop system along the periodic orbit, let us consider an initial value problem for the open-loop system on the interval $[0, T]$ corresponding to one period:

$$\begin{aligned} \dot{x}(t) &= f(x(t), u_r(t)), \quad \forall t \in [0, T], \\ x(0) &= x_0. \end{aligned}$$

The solution trajectories $x(t)$ can be regarded as functions of the initial value x_0. Note that for $x_0 = x_r(0)$ the solution is identical to the reference trajectory $x_r(t)$. The sensitivity matrices

$$W(t) := \frac{\partial x(t)}{\partial x_0}(x_r(0)), \quad t \in [0, T],$$

can therefore be obtained as the solution of the matrix initial value problem:

$$\begin{aligned} \dot{W}(t) &= \frac{\partial f}{\partial x}(x_r(t), u_r(t)) \cdot W(t) \quad \forall t \in [0, T], \\ W(0) &= \mathbb{I}_{n_x}. \end{aligned}$$

The final value $W(T)$ is called the *monodromy matrix*. It characterizes the sensitivity of the final state of each period with respect to the initial value. Asymptotically stable periodic orbits are characterized by a monodromy matrix whose eigenvalues (also called "Floquet Multipliers") all have a modulus smaller than one, which means that initial disturbances are damped out during the cycles. For a proof see e.g. Amann [Ama83].

A numerical computation of $W(T)$ for the kite model along the chosen periodic orbit yields

$$W(T) = \begin{pmatrix} 3.0182 & 2.4014 & 0.9587 & -0.1307 \\ 3.3399 & 2.5500 & 0.0054 & -0.3935 \\ -2.7170 & -1.8596 & 0.8436 & 0.5072 \\ -2.8961 & -2.0491 & 0.5601 & 0.4640 \end{pmatrix},$$

which has the eigenvalue spectrum

$$\sigma\left(W(T)\right) = \{\ 5.29, \quad 1.53, \quad 6.16 \cdot 10^{-2}, \quad 4.17 \cdot 10^{-7}\ \},$$

containing two eigenvalues that have a modulus bigger than one. This confirms that the system is asymptotically unstable in the periodic reference orbit.

8.3 The Optimal Control Problem

Given an initial state x_{t_0} at time t_0, an optimal control problem can be formulated that takes account of the objective to keep the system close to the reference orbit. For this aim we define a Lagrange term

$$L(x,u,t) := (x - x_r(t))^T Q (x - x_r(t)) + (u - u_r(t))^T R (u - u_r(t))$$

with diagonal weighting matrices

$$Q := \cdot \begin{pmatrix} 1.2 & 0 & 0 & 0 \\ 0 & 3.0\mathrm{s}^2 & 0 & 0 \\ 0 & 0 & 3.0 & 0 \\ 0 & 0 & 0 & 3.0\mathrm{s}^2 \end{pmatrix} 10^{-4}\mathrm{deg}^{-2}\mathrm{s}^{-1} \ \text{ and } \ R := 1.0 \cdot 10^{-2}\mathrm{deg}^{-2}\mathrm{s}^{-1}.$$

A hard constraint is given by the fact that we do not want the kite to crash onto the ground ($\theta = 90$ degrees), and for security, we require a path constraint function

$$h(x,u) := \left(75\,\mathrm{deg} - \theta\right)$$

to be positive. Using these definitions, we formulate the following optimal control problem on the moving horizon $[t_0, t_0 + 2T]$:

$$\begin{aligned} &\min_{u(\cdot),x(\cdot)} \int_{t_0}^{t_0+2T} L(x(t),u(t),t)\,dt && (8.8)\\ &\text{subject to} \\ &\qquad \dot{x}(t) = f(x(t),u(t)), && \forall t \in [t_0, t_0+2T], \\ &\qquad x(t_0) = x_{t_0}, \\ &h(x(t),u(t)) \geq 0, && \forall t \in [t_0, t_0+2T]. \end{aligned}$$

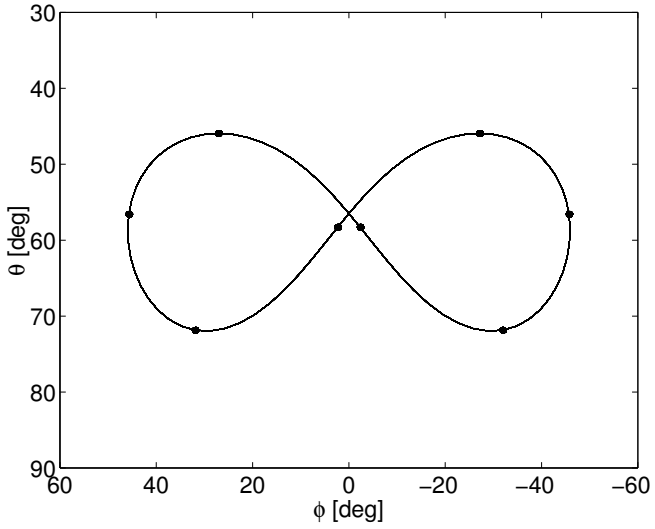


Figure 8.5: Closed-loop control applied to the undisturbed system, simulation of 100 periods. Numerical errors are attenuated by very small control responses (with $u(t) - u_r(t)$ in the order of 10^{-2} degree) and do not accumulate.

8.4 Closed-Loop Simulations

In the multiple shooting discretization the multiple shooting intervals were chosen to be each of one second length, thus allowing eight control corrections per period T. The Hessian matrix was approximated using the Gauss-Newton approach for integral least squares terms described in Sec. 6.4. The initialization of subsequent optimization problems was achieved with a shift strategy where the new final interval was initialized by an integration using the nominal open-loop control $u_r(t)$, cf. Sec. 4.4.1.

As a first test of the algorithm we try to control the undisturbed system, and the result of a simulation of 100 periods is depicted in Fig. 8.5. It can be seen that the reference orbit is perfectly tracked. The dots separate intervals of one second length and correspond to the sampling times.

For a second test we give the kite a slight "kick" at time $t = 1.0$ seconds that leads to a disturbance in the angular velocity $\dot{\theta}$. It changes from -1 deg/s to $+5$ deg/s. The closed-loop response is depicted in Fig. 8.6 as a $\phi - \theta-$plot.

As a third test we give the kite a moderate "kick" at time $t = 3.5$ seconds that lets the angular velocity $\dot{\theta}$ change from 12 deg/s to 25 deg/s. The closed-loop response is depicted in Fig. 8.7. For a comparison we also show the open-loop response to this disturbance in Fig. 8.8, which results in a crash 5 seconds after the disturbance.

In a fourth test we "kick" the kite strongly at time $t = 4.0$ seconds so that the angular velocity $\dot{\theta}$ changes abruptly from 20 deg/s to -7 deg/s. The closed-loop response is depicted in Fig. 8.9.

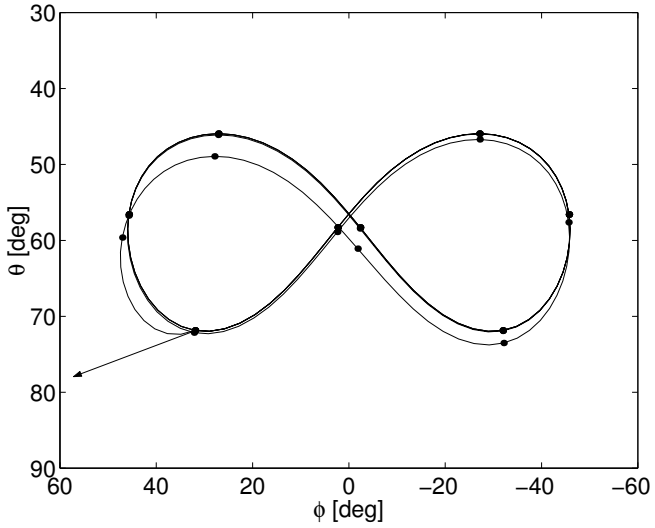


Figure 8.6: Closed-loop response to a small disturbance in $\dot{\theta}$ that changes from -1 deg/s to $+5$ deg/s at time $t = 1.0$ seconds. After one period the disturbance is nearly attenuated.

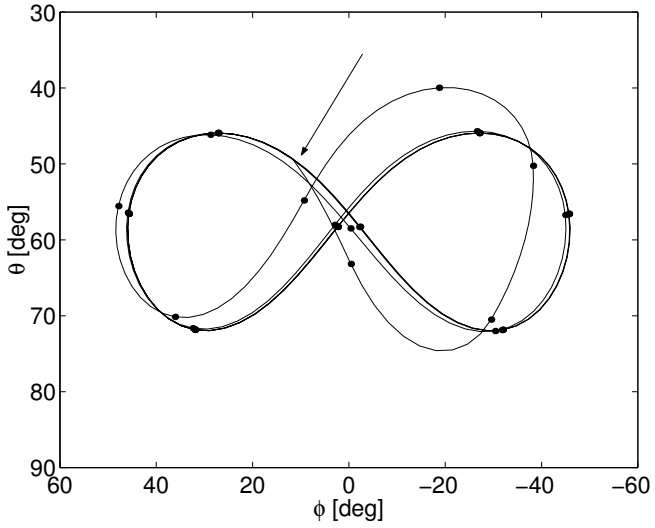


Figure 8.7: Closed-loop control response to a moderate disturbance in $\dot{\theta}$ that changes from 12 deg/s to 25 deg/s at time $t = 3.5$ seconds. After 1.5 periods the disturbance is attenuated.

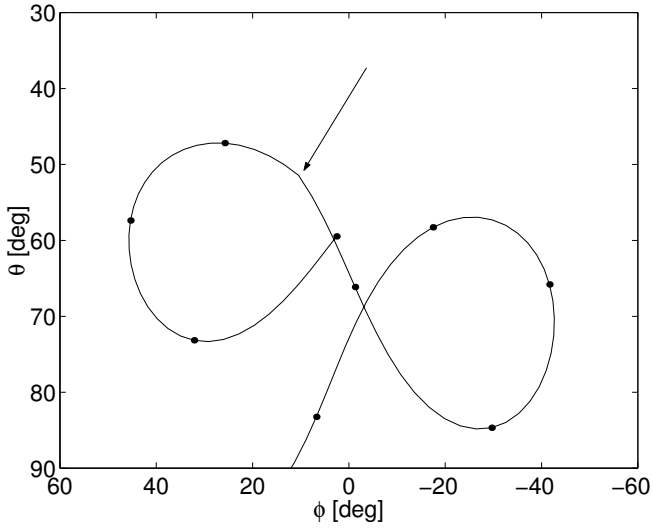


Figure 8.8: Open-loop response to the same disturbance as in Fig. 8.7, at time $t = 3.5$ seconds. Five seconds after the disturbance the kite crashes onto the ground (θ=90 degrees).

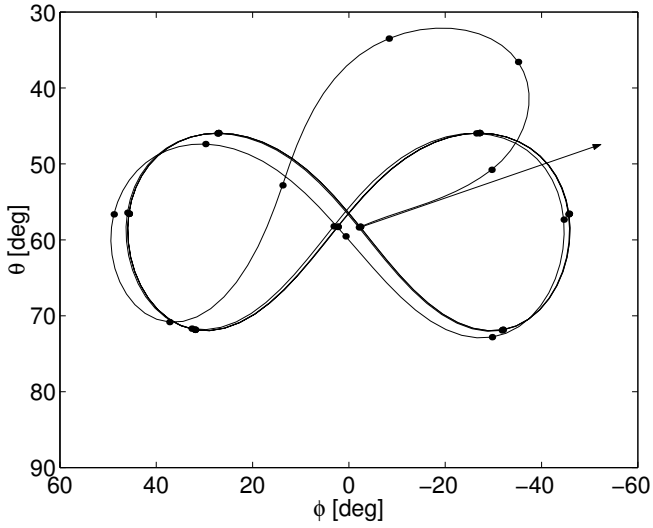


Figure 8.9: Closed-loop response to a strong disturbance in $\dot{\theta}$ that changes from 20 deg/s to a value of -7 deg/s at time $t = 4.0$ seconds. After two periods the disturbance is completely attenuated.

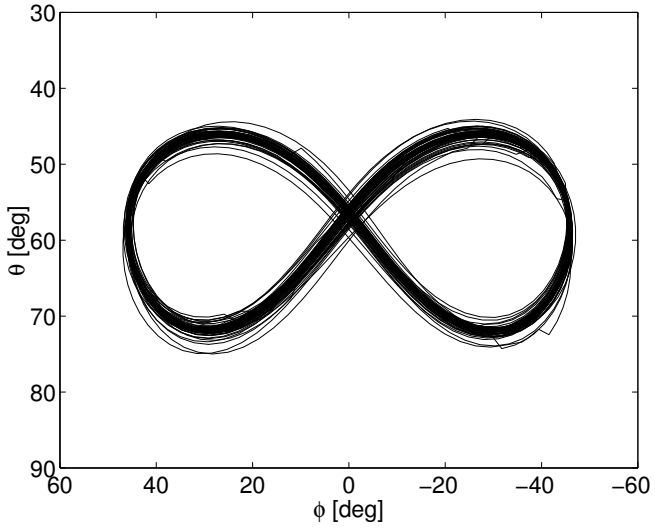


Figure 8.10: Closed-loop trajectory for the weak disturbance test, simulated over 100 periods.

As a last test we apply random noise of various magnitude to the system: disturbances happen with probability $p = 10\%$ s^{-1}, and they simultaneously disturb all 4 components of the system state, with independent magnitudes that are characterized by the standard deviations

$$s_\theta = 0.9\,\text{deg}, \quad s_{\dot\theta} = 0.9\,\text{deg s}^{-1}, \quad s_\phi = 0.6\,\text{deg}, \quad \text{and} \quad s_{\dot\phi} = 0.6\,\text{deg s}^{-1}$$

for the weak disturbance test, and

$$s_\theta = 4.5\,\text{deg}, \quad s_{\dot\theta} = 4.5\,\text{deg s}^{-1}, \quad s_\phi = 3\,\text{deg}, \quad \text{and} \quad s_{\dot\phi} = 3\,\text{deg s}^{-1}$$

for the strong disturbance test. For each scenario, we have carried out simulations for 100 periods (i.e., for 800 seconds). The resulting $\phi - \theta-$plots can be seen in Fig. 8.10 for the weak disturbance scenario, and in Fig. 8.11 for the strong disturbance scenario. While the weak scenario shows how nicely the closed-loop system behaves even in the presence of moderate disturbances, the strong disturbance scenario is certainly at the limits of the applicability of the chosen control approach, as the disturbances sometimes push the system state out of the state bounds specified in the optimization problem ($\theta \leq 75$ degrees). The resulting infeasibility of the optimization problems was cushioned by the relaxation strategy of the QP solver. However, this does not give any guarantee for the working of our approach in the presence of severe disturbances. Instead, a scheme employing soft constraint formulations should be employed.

The computation time for each real-time iteratiion cycle did not exceed the sampling time of one second in all simulations and averaged to 0.45 seconds with a standard deviation

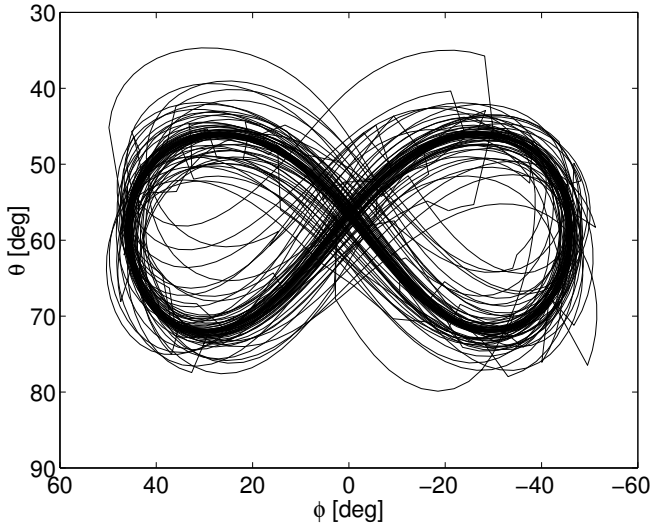


Figure 8.11: Closed-loop trajectory for the strong disturbance test, simulated over 100 periods.

of 0.02 seconds (on a Compaq Alpha XP1000 workstation). The immediate feedback took in average one tenth of this value, 0.05 seconds.

Conclusions and Outlook

Summary and Conclusions

We have presented a new numerical method for the real-time optimization of constrained nonlinear processes and have demonstrated its practical applicability in an experimental case study, the nonlinear model predictive control of a distillation column which is described by a large scale stiff DAE model. Sampling times in the range of seconds could be realized. The theoretical contraction properties of the algorithm have been investigated and computable bounds on the loss of optimality with respect to a rigorous solution could be established.

Description of the Method (Chapters 1 – 4 and 6)

The proposed approach is based on the **direct multiple shooting method** (Chap. 2) that allows to combine the use of efficient state-of-the-art DAE solvers with the advantages of a simultaneous strategy, like the possibility to treat unstable system models. In particular, the presented algorithm is characterized by the following, newly developed features:

- The initialization of subsequent optimization problems with an **initial value embedding strategy** delivers, for an exact Hessian SQP, a first-order predictor for the solution of new problems even in the presence of active set changes (Chap. 3). For general Newton type methods, the initial value embedding still delivers an excellent predictor (Chap. 5, Sec. 5.3).

- Dovetailing of the solution iterations with the process development in a **real-time iteration scheme** allows to reduce sampling times to a minimum, but maintains all advantages of a fully nonlinear treatment of the optimization problems (Chap. 4).

- A separation of the computations in each real-time iteration into a **preparation phase** and a **feedback phase** is realized (Chap. 6). The feedback phase is typically orders of magnitude shorter than the preparation phase, and allows to deliver a linearized feedback that takes all linearized constraints into account. This feedback is equivalent to *linear* MPC schemes, using a system linearization along the current optimal reference trajectory. The delay of one sampling time that is present in all previous NMPC schemes is avoided.

- A **Gauss-Newton approach for least squares integrals** allows to compute an excellent Hessian approximation at negligible computational costs. The Gauss-Newton Hessian is computed simultanously with the sensitivity computation *without the necessity to stop the integration routine* (Sec. 6.4). This is especially useful on long prediction intervals with constant controls.

Contractivity of the Real-Time Iteration Scheme (Chapter 5)

Contractivity of the scheme is proven for real-time iterations on shrinking horizons, and the outcome of the iterates is compared to the corresponding exact solution on the full horizon. The scenario assumes that plant and optimization model coincide, but that one unpredicted disturbance happens at the start of the considered time period.

- The real-time iteration scheme is **contracting** under the same conditions as the off-line method (Theorem 5.6). This means: if the full horizon optimization problem and a given initialization satisfy the sufficient conditions for local convergence of the off-line Newton type method, then the real-time iteration scheme is contracting.

- The iterates **approach the optimal solution on the remaining horizon**, with a velocity that depends on the contraction rate (Corollary 5.8). Due to the excellent contraction properties of the direct multiple shooting method, this means that after a few iterations the real-time solution is practically identical to the exact solution on the remaining horizon.

- We establish a **bound on the loss of optimality** with respect to the optimal solution on the full horizon (Theorem 5.11). This bound limits possible losses on the first intervals, before the iterates approach the optimal solution on the remaining horizon.

- If the algorithm was initialized at a neighboring solution, as it typically happens in practice, the **loss of optimality is of fourth order** in the size of the initial disturbance for an **exact Newton method** (Corollary 5.12).

Application Tests of the Scheme (Chapters 7 and 8)

Experimental Control of a Distillation Column: The algorithm is successsfully applied to a nontrivial process control example, namely the NMPC of a pilot plant distillation column situated at the *Institut für Systemdynamik und Regelungstechnik*, Stuttgart. A model for the column is developed, considering enthalpy balances and hydrodynamics, which results in a stiff DAE with 82 differential and 122 algebraic state variables. Model parameters are fitted to experimental dynamic data.

The optimization problem is formulated using the integrated squared deviation of two controlled temperatures as objective, and employing a control horizon with 5 sampling intervals of 120 seconds each, and a prediction horizon of 36 000 seconds. The computation

times for each real-time iteration are below 20 seconds, and the realized feedback times are under practical conditions below 400 milliseconds. The experimentally observed closed-loop behaviour shows good performance, especially for large disturbances.

The study proves that NMPC using large scale process models is feasible under practical conditions, when the real-time iteration scheme is used (Chap. 7).

Simulated Control of a Looping Kite: In a second example, the real-time iteration approach is applied to a simulated unstable periodic process, an airborne kite. The newly developed kite model consists of four differential states and one control. The control aim is to let the kite fly an unstable periodic trajectory with a period of eight seconds. The real-time iteration scheme is able to successfully stabilize the system for all investigated disturbance scenarios, meeting the real-time requirement of at maximum one second per iteration. Simulation results show excellent robustness of the real-time optimization algorithm even in the presence of large disturbances. (Chap. 8).

Outlook

Within this thesis, we have demonstrated the practical feasibility of NMPC using large scale detailed process models. Several future developments of numerical methods for NMPC come to mind, which may extend its area of applicability.

Parallelization

A parallelization of the developed algorithm, which has already been achieved for the off-line direct multiple shooting method, promises to reduce computation times considerably. The method is particularly well suited for parallel computation, since the most expensive part of the algorithm, the integrations and sensitivity computations, are decoupled on different multiple shooting intervals and can be performed in parallel [GB94, BS01]. For the distillation model developed in this thesis, processor efficiencies in the range of 80 % for 8 nodes have been observed. Only minor modifications have to be made to adapt the existing parallel version of the off-line method to the real-time iteration context.

Reduced Approach

Another interesting future development is to employ a reduction approach that exploits the initial value constraint and the continuity conditions for an efficient derivative computation in the multiple shooting method. The approach, that is originally due to Schlöder [Sch88], has long been successfully applied to large scale parameter estimation problems (see e.g. Dieses [Die01] for recent developments and applications). An application of the approach to the described Gauss-Newton method for optimal control is possible and promises large savings in computation times in the sensitivity generation, thus allowing to further reduce sampling times. The reduced approach is fully compatible with most algorithmic ideas of

this thesis, especially with the initial value embedding and the dovetailing of the solution iterations with the process development. However, the separation into preparation and feedback phase cannot be realized as easily as before, as some parts of the DAE sensitivity calculation can only be performed *after* the initial value x_0 is known.

The approach would be especially efficient for models with large differential state dimensions and a relatively small number of control parameters.

On-Line Parameter and State Estimation

In the application of NMPC techniques, an important requirement is knowledge of the system state and of the current values of the system parameters. Moving horizon strategies to attack this task have been formulated (see e.g. Rao and Rawlings [RR00]), but the field of numerical methods for the real-time solution of the resulting optimal control problems still needs considerable development. A transfer of the real-time iteration scheme to this type of problem promises to deliver a powerful method for the on-line solution of moving horizon state estimation problems, and is currently under investigation (cf. Bürner [Bür01]).

Periodic Control

In the last numerical example we showed the feasibility of an NMPC approach designed to control an unstable periodic system. Given the existing optimization scheme, the stabilizing periodic feedback law was easily obtained by a straightforward periodic formulation of the least squares objective. In the area of periodic control, the use of NMPC techniques may allow new periodic process designs that have so far been avoided, and an application of the developed numerical methods to this problem class deserves further investigation.

Appendix A

An Extended Kalman Filter Variant

We will here describe the variant of the Extended Kalman Filter (EKF) that was used for the state estimation in the experimental tests of Chap. 7. For an introduction into current developments in the area of nonlinear state estimation we refer e.g. to Muske and Edgar [ME96] or to Rao and Rawlings [RR00]. We also refer to an overview article by Binder et al. [BBB+01] that discusses some aspects of state estimation, and to the work of Bürner on numerical methods of moving horizon state estimation [Bür01].

In contrast to a standard Extended Kalman Filter (EKF), our variant is able to treat bounds on the system state, a feature that can be crucial for the practical applicability of the algoritm. We will first formulate the on-line estimation problem in Sec. A.1 and introduce the EKF type algorithm in Sec. A.2, and afterwards motivate it by heuristic arguments in Sec. A.3.

A.1 Problem Formulation

Transformation into Discrete Time

Let us consider the system development on intervals of fixed length δ only, which correspond to the sampling rate of measurements. Given initial values x_k (that may also comprise constant system parameters, cf. Sec. 1.1) and controls u_k, the system DAE can be solved on the interval $[t_k, t_k + \delta]$

$$\begin{aligned} B(\cdot)\,\dot{x}(t) &= f(x(t), z(t), u_k), \\ 0 &= g(x(t), z(t), u_k), \\ x(t_k) &= x_k. \end{aligned} \tag{A.1}$$

In the following, we are interested only in the values $x(t_k+\delta)$ and $z(t_k)$ of this solution. Let us denote them by $X_k(x_k)$ and $Z_k(x_k)$, where the constant control values u_k are accounted for by the index k. As some of the states can be measured, let us also introduce the measurement function

$$h_k(x_k) := H_x x_k + H_z Z_k(x_k)$$

with constant matrices H_x and H_z. For the distillation model with temperature measurements we have set $H_x = 0$, and chosen H_z such that it just extracts the measured temperatures from the algebraic state vector.

The undisturbed system development $\{y_k, x_k\}_{k=0}^{\infty}$ with an initial value x_0 under a given control sequence $\{u_k\}_{k=0}^{\infty}$ can then be described by the equations

$$\begin{aligned} x_{k+1} &= X_k(x_k), \quad \text{for} \quad k = 0, 1, \dots, \\ y_k &= h_k(x_k). \end{aligned} \tag{A.2}$$

Stochastic Formulation

The necessity to estimate the system state arises because the real system does not coincide with the model. To account for this, we model the discrete time system as a stochastic system, and we also assume that the measurements are distorted by noise. Let us therefore regard the discrete time stochastic system and measurement model

$$\begin{aligned} x_{k+1} &= X_k(x_k) + w_k, \\ y_k &= h_k(x_k) + v_k. \end{aligned} \tag{A.3}$$

The state disturbance and measurement noise sequences $\{w_k\}_{k=0}^{\infty}$ and $\{v_k\}_{k=0}^{\infty}$ are assumed to be independent and identically distributed, both with zero mean and known (positive definite) covariance matrices

$$\Sigma_w := \mathbb{E}\{w_k w_k^T\} \quad \text{and} \quad \Sigma_v := \mathbb{E}\{v_k v_k^T\}.$$

The notation $\mathbb{E}\{\cdot\}$ denotes the expectation values. From the real system behaviour at a sampling time k, only the measurement sequence $\{y_i\}_{i=0}^{k}$ is available. Additional knowledge exists in form of state bounds that require that

$$x_{\mathrm{LB}} \le x_i \le x_{\mathrm{UB}} \quad \text{for} \quad i = 0, \dots, k.$$

The problem is to infer the system state x_k from this given information.

The Idea of Kalman Filtering

The Extended Kalman Filter (EKF) for nonlinear systems proceeds in principle as the linear Kalman filter [Kal60, Son90], but is based on subsequent linearizations of the system model at the best available estimate. The idea of the Kalman filter is to compare each measurement with the prediction of the model, and to correct the estimated state $\hat{x}$ according to the deviation. The weight of past measurement information is kept in a weighting matrix P.

A.2 The EKF Type Algorithm

Given a current estimate $\hat{x}_k \in \mathbb{R}^{n_x}$, a nonsingular square weighting matrix P_k (of the same dimension $\mathbb{R}^{n_x \times n_x}$ as Σ_w) and a measurement $y_k \in \mathbb{R}^{n_y}$ at time k, the recursive algorithm computes the matrix P_{k+1} and the vector $\hat{x}_{k+1}$ as follows:

1. Compute $h := h_k(\hat{x}_k)$ and $H := \frac{\partial h_k(\hat{x}_k)}{\partial \hat{x}_k}$.

2. Compute a QR decomposition

$$\begin{pmatrix} P_k \\ \Sigma_v^{-\frac{1}{2}} H \end{pmatrix} =: \hat{Q}\hat{R} \tag{A.4}$$

with $\hat{R}$ upper triangular and of full rank (note that this is always possible as P_k is nonsingular). Obtain a corrected differential state value

$$x' := \hat{x}_k - \hat{R}^{-1}\hat{Q}^T \begin{pmatrix} 0 \\ \Sigma_v^{-\frac{1}{2}}(h - y_k) \end{pmatrix}. \tag{A.5}$$

3. To avoid a violation of upper and lower bounds (that may make the DAE solution impossible), solve

$$\min_x \|\hat{R}(x - x')\|_2^2 \quad \text{subject to} \quad x_{\mathrm{LB}} \le x \le x_{\mathrm{UB}}.$$

Denote the solution by $\tilde{x}$. Once the active set and $\tilde{x}$ are known, define a matrix $Q_1 = (e_1, e_2, \ldots, e_{n_a})$ consisting of n_a unit vectors e_i corresponding to the n_a components of the active set, so that the equivalent problem

$$\min_x \|\hat{R}(x - x')\|_2^2 \quad \text{subject to} \quad Q_1^T(x - \tilde{x}) = 0, \tag{A.6}$$

can be formulated (which has the solution $\tilde{x}$ itself). Denote by Q_2 the orthonormal complement to Q_1, so that $(Q_1|Q_2)$ is an orthonormal (permutation) matrix. Perform another QR factorization $\hat{R}Q_2 =: Q'R'$ to yield the invertible matrix $R' \in \mathbb{R}^{(n_x-a)\times(n_x-a)}$. This is the only step that is additional to a standard EKF scheme, and it can be justified heuristically. If no bounds are active, $\tilde{x} = x'$ and $R' = \hat{R}$.

4. Compute $\hat{x}_{k+1} := X_k(\tilde{x})$ and $G := \frac{\partial X_k(\tilde{x})}{\partial \tilde{x}}$.

5. Compute a complete QR decomposition

$$\begin{pmatrix} R' \\ -\Sigma_w^{-\frac{1}{2}} G Q_2 \end{pmatrix} =: \left(\bar{Q} \,\middle|\, \tilde{Q} \right) \begin{pmatrix} \bar{R} \\ 0 \end{pmatrix}$$

with $\bar{R}$ non-singular.

6. Compute

$$P_{k+1} := \tilde{Q}^T \begin{pmatrix} 0 \\ \Sigma_w^{-\frac{1}{2}} \end{pmatrix}.$$

Note that our algorithm produces predictive estimates $\hat{x}_{k+1}$ with knowledge of the k-th measurement y_k (and the control value u_k) only.

Derivative Generation

The function $X_k(x_k)$ and its derivative $\frac{\partial X_k}{\partial x_k}$ can efficiently be computed by a DAE solver using the principle of internal numerical differentiation (IND) (cf. Sec. 6.3). We use the DAE solver DAESOL [Bau00]. The computation of $h_k(x_k) = H_x x_k + H_z Z_k(x_k)$ requires the determination of consistent algebraic variables $Z_k(x_k)$ that satisfy $g(x_k, Z_k(x_k), u_k) = 0$. This is achieved in our implementation by a (damped) Newton's method which causes very little computational effort compared to the DAE solution. By the implicit function theorem, the derivative $\frac{\partial h_k}{\partial x_k}$ can be evaluated to be

$$\frac{\partial h_k}{\partial x_k} = H_x + H_z \left(\frac{\partial g}{\partial z}\right)^{-1} \frac{\partial g}{\partial x}.$$

A.3 Heuristic Motivation

The idea behind the EKF algorithm is based on dynamic programming arguments. Let us define a function

$$F(x, \bar{x}) := \begin{pmatrix} P_k(x - \hat{x}_k) \\ \Sigma_v^{-\frac{1}{2}}(h_k(x) - y_k) \\ \Sigma_w^{-\frac{1}{2}}(\bar{x} - X_k(x)) \end{pmatrix}$$

that represents the costs on stage k, given a past state estimate $\hat{x}_k$ and a weighting matrix P_k. The idea is to approximately summarize the optimal value

$$\min_x \|F(x, \bar{x})\|_2^2 \text{ subject to } x_{\text{LB}} \leq x \leq x_{\text{UB}}, \tag{A.7}$$

that depends on the state $\bar{x}$, in a quadratic function

$$\|P_{k+1}(\bar{x} - \hat{x}_{k+1})\|_2^2 + \text{ const }.$$

To obtain this approximation, we will linearize the system, as only then it is possible to summarize the optimal stage costs in a quadratic function (using the discrete time Kalman filter idea).

The linearization of problem (A.7) does not only concern the function F, as usual in EKF algorithms, but also the constraints, which have to be converted into appropriate equality constraints to make the problem truly linear. The procedure of the previous section can be regarded as a dovetailing of the problem linearization and the linear Kalman filter algorithm.

We linearize the problem during the solution procedure, as described in the previous section: let us linearize $h_k(x)$ at the point $\hat{x}_k$ to yield the approximation $h + H(x - \hat{x}_k)$ (step 1), then let us choose a point $\tilde{x}$ (the outcome of the QP solution, steps 2 and 3) at which

we linearize $X_k(x)$ to yield the linearization $\hat{x}_{k+1}+G(x-\tilde{x}) = \hat{x}_{k+1}+G(\hat{x}_k-\tilde{x})+G(x-\hat{x}_k)$ (step 4). We can therefore approximate $F(x,\bar{x})$ by the linear function

$$\begin{aligned}&\tilde{f} + (\tilde{F}_x|\tilde{F}_{\bar{x}}) \begin{pmatrix} x-\hat{x}_k \\ \bar{x}-\hat{x}_{k+1} \end{pmatrix} \\ = \; & \begin{pmatrix} 0 \\ \Sigma_v^{-\frac{1}{2}}(h-y_k) \\ -\Sigma_w^{-\frac{1}{2}}G(\hat{x}_k-\tilde{x}) \end{pmatrix} + \left(\begin{array}{c|c} P_k & 0 \\ \Sigma_v^{-\frac{1}{2}}H & 0 \\ -\Sigma_w^{-\frac{1}{2}}G & \Sigma_w^{-\frac{1}{2}} \end{array}\right) \begin{pmatrix} x-\hat{x}_k \\ \bar{x}-\hat{x}_{k+1} \end{pmatrix}.\end{aligned}$$

Fixing also the active set we transform the inequality constraints into equalities

$$Q_1^T(x-\tilde{x}) = 0,$$

so that the linearization of problem (A.7) can be written as

$$\min_x \left\| \tilde{f} + (\tilde{F}_x|\tilde{F}_{\bar{x}}) \begin{pmatrix} x-\hat{x}_k \\ \bar{x}-\hat{x}_{k+1} \end{pmatrix} \right\|_2^2 \text{ subject to } Q_1^T(x-\tilde{x}) = 0,$$

or, equivalently, as an unconstrained problem, where we directly substitute $x = \tilde{x} + Q_2 y$ (using the orthonormal complement Q_2 of Q_1):

$$\min_y \left\| \bar{f} + (\tilde{F}_x Q_2|\tilde{F}_{\bar{x}}) \begin{pmatrix} y \\ \bar{x}-\hat{x}_{k+1} \end{pmatrix} \right\|_2^2, \tag{A.8}$$

with

$$\bar{f} := \tilde{f} + \tilde{F}_x(\tilde{x}-\hat{x}_k) = \begin{pmatrix} 0 \\ \Sigma_v^{-\frac{1}{2}}(h-y_k) \\ 0 \end{pmatrix} + \begin{pmatrix} P_k \\ \Sigma_v^{-\frac{1}{2}}H \\ 0 \end{pmatrix} (\tilde{x}-\hat{x}_k). \tag{A.9}$$

Our EKF type algorithm computes a QR factorization of $(\tilde{F}_x Q_2|\tilde{F}_{\bar{x}})$, as

$$\begin{aligned}(\tilde{F}_x Q_2|\tilde{F}_{\bar{x}}) &= \begin{pmatrix} P_k Q_2 & 0 \\ \Sigma_v^{-\frac{1}{2}}HQ_2 & 0 \\ -\Sigma_w^{-\frac{1}{2}}GQ_2 & \Sigma_w^{-\frac{1}{2}} \end{pmatrix} = \begin{pmatrix} \hat{Q} & 0 \\ 0 & \mathbb{I} \end{pmatrix} \begin{pmatrix} \hat{R}Q_2 & 0 \\ -\Sigma_w^{-\frac{1}{2}}GQ_2 & \Sigma_w^{-\frac{1}{2}} \end{pmatrix} \\ &= \begin{pmatrix} \hat{Q} & 0 \\ 0 & \mathbb{I} \end{pmatrix} \begin{pmatrix} Q' & 0 \\ 0 & \mathbb{I} \end{pmatrix} \begin{pmatrix} R' & 0 \\ -\Sigma_w^{-\frac{1}{2}}GQ_2 & \Sigma_w^{-\frac{1}{2}} \end{pmatrix} \\ &= \begin{pmatrix} \hat{Q} & 0 \\ 0 & \mathbb{I} \end{pmatrix} \begin{pmatrix} Q' & 0 \\ 0 & \mathbb{I} \end{pmatrix} (\bar{Q}|\tilde{Q}) \begin{pmatrix} \bar{R} & R'' \\ 0 & P_{k+1} \end{pmatrix}\end{aligned}$$

with $R'' := \bar{Q}^T \begin{pmatrix} 0 \\ \Sigma_w^{-\frac{1}{2}} \end{pmatrix}$. The linear problem (A.8) is therefore equivalent to:

$$\min_y \left\| \begin{pmatrix} \bar{Q}^T \\ \tilde{Q}^T \end{pmatrix} \begin{pmatrix} Q'^T & 0 \\ 0 & \mathbb{I} \end{pmatrix} \begin{pmatrix} \hat{Q}^T & 0 \\ 0 & \mathbb{I} \end{pmatrix} \bar{f} + \begin{pmatrix} \bar{R} & R'' \\ 0 & P_{k+1} \end{pmatrix} \begin{pmatrix} y \\ \bar{x}-\hat{x}_{k+1} \end{pmatrix} \right\|_2^2.$$

The optimal solution of this linearized problem can be summarized as

$$\|P_{k+1}(\bar{x}-\hat{x}_{k+1})\|_2^2.$$

if

$$\tilde{Q}^T \begin{pmatrix} Q'^T & 0 \\ 0 & \mathbb{I} \end{pmatrix} \begin{pmatrix} \hat{Q}^T & 0 \\ 0 & \mathbb{I} \end{pmatrix} \bar{f} = 0.$$

To see that this is indeed the case, note that

$$\begin{pmatrix} \hat{Q}^T & 0 \\ 0 & \mathbb{I} \end{pmatrix} \bar{f} = \begin{pmatrix} \hat{R}(\tilde{x} - x') \\ 0 \end{pmatrix}$$

due to (A.4), (A.5) and (A.9), and that

$$Q'^T \hat{R}(\tilde{x} - x') = R'^{-T}(\hat{R}Q_2)^T \hat{R}(\tilde{x} - x')$$

must be zero, because $y = 0$ is solution of the following optimization problem, that is equivalent to (A.6) with $x = \tilde{x} + Q_2 y$:

$$\min_y \|\hat{R}(\tilde{x} - x') + \hat{R}Q_2 y\|_2^2.$$

Note that at the solution $y = 0$ the gradient of the objective with respect to y is $2(\hat{R}Q_2)^T \hat{R}(\tilde{x} - x')$, which is zero due to the necessary optimality conditions.

Appendix B

Details of the Distillation Model

Physical Property Functions

Molar Volume $V^m(X,T)$

The molar volume $V^m(x,T)$ of the liquid tray content is calculated as a linear combination of the molar volumes of the undiluted components, i.e.,

$$V^m(X,T) := XV_1^m(T) + (1-X)V_2^m(T).$$

with $V_k^m(T)$ calculated according to

$$V_k^m(T) := \frac{1}{a_k}\exp_{b_k}(1+\exp_{(1-T/c_k)}(d_k)).$$

The molar volume coefficients a_k, b_k, c_k, d_k b are given in Table B.1.

Partial Pressures $P_k^s(T)$

The partial pressures $P_k^s(T)$ of the undiluted components are determined by the Antoine equation

$$P_k^s(T) := \exp\left(A_k - \frac{B_k}{T+C_k}\right) \quad k = 1,2.$$

The employed Antoine coefficients are given in Table B.2.

Component k	Molar volume coefficients a_k [kmol l^{-1}]	b_k	c_k [K]	d_k
1 (Methanol)	2.288	0.2685	512.4	0.2453
2 (n-Propanol)	1.235	0.27136	536.4	0.2400

Table B.1: The molar volume coefficients

Component k	Antoine coefficients A_k	B_k [K]	C_k [K]
1 (Methanol)	23.48	3626.6	-34.29
2 (n-Propanol)	22.437	3166.4	-80.15

Table B.2: The Antoine coefficients

	Enthalpy coefficients					
k	$h_{1,k}$ [K^{-1}]	$h_{2,k}$ [K^{-2}]	$h_{3,k}$ [K^{-3}]	T_k^c [K]	P_k^c [Pa]	Ω_k
1	18.31	$1.713\ 10^{-2}$	$6.399\ 10^{-5}$	512.6	$8.096\ 10^{6}$	0.557
2	31.92	$4.49\ 10^{-2}$	$9.663\ 10^{-5}$	536.7	$5.166\ 10^{6}$	0.612

Table B.3: The enthalpy coefficients

The Enthalpies $h^L(X,T)$ and $h^V(Y,T,P)$

The vapour and liquid stream enthalpies $h^L(X,T)$ and $h^V(Y,T,P)$ are given by

$$h^L(X,T) := Xh_1^L(T) + (1-X)h_2^L(T)$$

and

$$h^V(Y,T,P) := Yh_1^V(T,P) + (1-Y)h_2^V(T,P).$$

The pure liquid enthalpies $h_k^L(T)$ are determined according to

$$h_k^L(T) := C\ \left\{h_{1,k}(T-T_0) + h_{2,k}(T-T_0)^2 + h_{3,k}(T-T_0)^3\right\}$$

with $T_0 = 273.15$ K and $C = 4.186$ J mol^{-1}, and the pure vapour enthalpies $h_k^V(T,P)$ according to

$$h_k^V(T,P) := h_k^L(T) \quad + \quad RT_k^c\sqrt{1-\frac{P}{P_k^c}\left(\frac{T}{T_k^c}\right)^{-3}} \left\{a - b\frac{T}{T_k^c} + c\left(\frac{T}{T_k^c}\right)^7 + \Omega_k\left(d - e\frac{T}{T_k^c} + f\left(\frac{T}{T_k^c}\right)^7\right)\right\}$$

with $R = 8.3147$ J mol^{-1} K^{-1}, $a = 6.09648$, $b = 1.28862$, $c = 1.016$, $d = 15.6875$, $e = 13.4721$, and $f = 2.615$.

The employed coefficients are given in Table B.3.

Derivation of the Francis Weir Formula

The Francis weir formula that was introduced in Eq. (7.15), gives a relationship between the volumetric flowrate L_{vol} and the volume holdup n^v of an idealized tray by

$$L_{\text{vol}} = W(n^v - n^{\text{ref}})^{\frac{3}{2}},$$

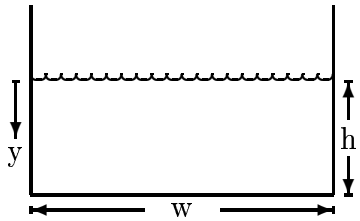


Figure B.1: Cross section of the liquid flow out of a weir.

where the flow constant W and the reference volume n^{ref} are constant. For a derivation, let us regard the gravity flow over a horizontal weir of width w with vertical walls, where the water level is at height h over the lower edge of the weir, as depicted in Figure B.1. The liquid level h can be determined from the excess volume on the tray, if its ground area A is known:

$$h = \frac{n^v - n^{\text{ref}}}{A}.$$

Introducing a coordinate y that starts at the liquid surface and measures the depth, we can determine the (horizontal) water velocity $v(y)$ due to gravity by Bernoulli's equation

$$\frac{1}{2}\rho v^2 = \rho \mathrm{g} y,$$

where ρ is the mass density and g the gravity constant. Note that $v(y) = \sqrt{2\mathrm{g}y}$ is independent of the liquid's density ρ. The overall outflow rate L_{vol} can now be determined by an integration over y from the top level ($y = 0$) down to the weir's upper edge ($y = h$):

$$\begin{aligned} L_{\text{vol}} &= \int_0^h v(y) w dy = w\sqrt{2\mathrm{g}} \int_0^h y^{\frac{1}{2}} dy = w\sqrt{2\mathrm{g}}\tfrac{2}{3} h^{\frac{3}{2}} \\ &= w\sqrt{2\mathrm{g}}\tfrac{2}{3} A^{-\frac{3}{2}} \left(n^v - n^{\text{ref}}\right)^{\frac{3}{2}} =: W \left(n^v - n^{\text{ref}}\right)^{\frac{3}{2}}. \end{aligned}$$

The real values of the flow width w depend on the tray geometry (see e.g. Lockett [Loc86]). However, since we know that the geometry of the bubble cap trays in the pilot plant distillation column is different from ideal trays, we use the Francis weir formula as a heuristic scheme only, and estimate the two parameters W and n^{ref} by using dynamic experimental data.

Appendix C

Proof of Theorem 3.4

In this appendix we will give a proof of Theorem 3.4 from Sec. 3.2. A similar proof of the theorem can be found in [GVJ90, Theorem 3.3.4 and Corollary 3.3.1 (2)].

To prove the theorem, let us subdivide the weakly active constraints $H^{\text{w.act}}$ into those components $H^{\text{w.act},+}$ with $\delta\mu_*^{\text{w.act},+} > 0$ and those $H^{\text{w.act},0}$ with $\delta\mu_*^{\text{w.act},0} = 0$ in the solution of QP (3.7), i.e., we write

$$H^{\text{w.act}}(t,w) =: \begin{pmatrix} H^{\text{w.act},+} \\ H^{\text{w.act},0} \end{pmatrix}(t,w).$$

Let us introduce the function

$$\bar{G}(t,w) := \begin{pmatrix} G \\ H^{\text{s.act}} \\ H^{\text{w.act},+} \end{pmatrix}(t,w).$$

We will see that this is the function of active constraints for all $P(t)$ on $t \in (0,\epsilon)$, and furthermore, that all these constraints are *strongly* active on $(0,\epsilon)$. Let us therefore consider the family of equality constrained problems

$$\min_{w \in \mathbb{R}^{n_w}} F(t,w) \quad \text{subject to} \quad \bar{G}(t,w) = 0,$$

with Lagrangian function $\bar{\mathcal{L}}(t,w,\bar{\lambda}) := F(t,w) - \bar{\lambda}^T\bar{G}(t,w)$. The system of necessary optimality conditions 3.3 for these problems can be stated as

$$\nabla_{(w,\bar{\lambda})}\bar{\mathcal{L}}(t,w^*(t),\bar{\lambda}^*(t)) = \begin{pmatrix} \nabla_w\bar{\mathcal{L}}(t,w^*(t),\bar{\lambda}^*(t)) \\ -\bar{G}(w^*(t)) \end{pmatrix} = 0. \tag{C.1}$$

A tentative total differentiation of these conditions with respect to t yields

$$\frac{\partial}{\partial t}\nabla_{(w,\bar{\lambda})}\bar{\mathcal{L}}(t,w^*(t),\bar{\lambda}^*(t)) \quad + \quad \nabla^2_{(w,\bar{\lambda})}\bar{\mathcal{L}}(t,w^*(t),\bar{\lambda}^*(t)) \quad \frac{\partial}{\partial t}\begin{pmatrix} w^*(t) \\ \bar{\lambda}^*(t) \end{pmatrix} = 0.$$

The matrix

$$\nabla^2_{(w,\bar\lambda)}\bar{\mathcal{L}}(t,w^*(t),\bar\lambda^*(t)) = \begin{pmatrix} \nabla^2_w\bar{\mathcal{L}}(t,w^*(t),\bar\lambda^*(t)) & -\nabla_w\bar G(t,w^*(t)) \\ -\nabla_w\bar G(t,w^*(t))^T & 0 \end{pmatrix}$$

is the Karush-Kuhn-Tucker (KKT) matrix of the nonlinear problem. For $t = 0$ it is invertible as can be proven with the help of Lemma 3.2. To apply the lemma, we set $A := \nabla^2_w\bar{\mathcal{L}}(0,w^*(0),\bar\lambda^*(0))$ and $B := -\nabla_w\bar G(0,w^*(0))^T$, and note that $\nabla_w\bar G^T$ has full rank due to the regularity of $w^*(0)$, and that $A = \nabla^2_w\bar{\mathcal{L}} = \nabla^2_w\mathcal{L}$ because the multipliers of the weakly active and inactive constraints are zero. Matrix A is positive definite on the null space $\mathcal{N}^B$ of $B = \nabla_w\bar G^T$, because $\mathcal{N}^B$ is a subspace of the null space $\mathcal{N}^{\mathrm{s}}$ of the linearized strongly active constraints $\nabla_w\tilde G^{\mathrm{s}T}$, and A is positive definite on $\mathcal{N}^{\mathrm{s}}$ due to the sufficient conditions 3.3.

By the implicit function theorem, the invertibility of the KKT matrix at $t = 0$ ensures that there exists for a sufficiently small $\epsilon > 0$ a curve $\bar v : (-\epsilon,\epsilon) \to \mathbb{R}^{n_w}\times\mathbb{R}^{n_{\bar G}}$, $t \mapsto \begin{pmatrix} w^*(t) \\ \bar\lambda^*(t)\end{pmatrix}$ of points satisfying condition (C.1), with continuous derivative

$$\frac{\partial}{\partial t}\begin{pmatrix} w^*(t) \\ \bar\lambda^*(t)\end{pmatrix} = -\Big(\nabla^2_{(w,\bar\lambda)}\bar{\mathcal{L}}(t,w^*(t),\bar\lambda^*(t))\Big)^{-1}\frac{\partial}{\partial t}\nabla_{(w,\bar\lambda)}\bar{\mathcal{L}}(t,w^*(t),\bar\lambda^*(t)). \tag{C.2}$$

Let us enlarge this curve $\bar v$ in a straightforward way to yield a curve $\tilde v : (-\epsilon,\epsilon) \to \mathbb{R}^{n_w} \times \mathbb{R}^{n_G} \times \mathbb{R}^{n_H}$,

$$t \mapsto \begin{pmatrix} w^* \\ \lambda^* \\ \begin{pmatrix}\mu^*\end{pmatrix}\end{pmatrix} = \begin{pmatrix} w^* \\ \lambda^* \\ \mu^{*\mathrm{s.act}} \\ \mu^{*\mathrm{w.act},+} \\ \mu^{*\mathrm{w.act},0} \\ \mu^{*\mathrm{inact}}\end{pmatrix}(t) := \begin{pmatrix} w^*(t) \\ \begin{pmatrix}\bar\lambda^*(t)\end{pmatrix} \\ 0 \\ 0\end{pmatrix}.$$

By comparing the necessary optimality conditions 3.3 for the quadratic programming problem (3.7) with Eq. (C.2) for $t = 0$ it can be verified that

$$\frac{\partial}{\partial t}\begin{pmatrix} w^* \\ \lambda^* \\ \mu^{*\mathrm{s.act}} \\ \mu^{*\mathrm{w.act},+} \\ \mu^{*\mathrm{w.act},0} \\ \mu^{*\mathrm{inact}}\end{pmatrix}(0) = \frac{\partial}{\partial t}\begin{pmatrix} w^* \\ \begin{pmatrix}\bar\lambda^*\end{pmatrix} \\ 0 \\ 0\end{pmatrix}(0) = \begin{pmatrix} \delta w_* \\ \delta\lambda_* \\ \delta\mu_*^{\mathrm{s.act}} \\ \delta\mu_*^{\mathrm{w.act},+} \\ 0 \\ 0\end{pmatrix}.$$

We will show that the restriction of this curve to the interval $t \in [0,\epsilon)$ is the desired curve v of strictly optimal KKT points of the corresponding problems $P(t)$. For this aim we show that the points $w^*(t)$, $t \in [0,\epsilon)$, are feasible and that the multipliers $\mu^*(t)$ remain non-negative for $t \in [0,\epsilon)$.

First, by continuity of the function H^{inact} it is clear that

$$H^{\text{inact}}(t, w^*(t)) > 0, \quad \forall t \in [0, \epsilon)$$

if ϵ is chosen sufficiently small. The total derivative of the "zero" part $H^{\text{w.act},0}$ of the weakly active constraints with respect to t is the vector

$$D := \frac{d}{dt} H^{\text{w.act},0}(0, w^*(0)) = \frac{\partial H^{\text{w.act},0}}{\partial t} + \left(\nabla_w H^{\text{w.act},0}\right)^T \delta w_* > 0,$$

whose components are positive due to the strict complementarity assumption for the solution of the quadratic programming problem (3.7). Therefore,

$$H^{\text{w.act},0}(t, w^*(t)) = Dt + O(t^2) \geq 0, \quad \forall t \in [0, \epsilon),$$

if ϵ is chosen sufficiently small. Taking into account that all other constraints are contained in the vector $\bar{G}$, and exploiting the fact that $\bar{G}(w^*(t)) = 0$ along the curve, we can conclude that $w^*(t)$ are feasible points for all $t \in [0, \epsilon)$.

Conversely, let us check that the multipliers $\mu^*(t)$ remain non-negative for $t \in [0, \epsilon)$. From continuity we can conclude that $\mu^{*\text{s.act}} > 0, t \in [0, \epsilon)$, and from $\delta\mu_*^{\text{w.act},+} > 0$ we conclude that

$$\mu^{*\text{w.act},+} = \delta\mu_*^{\text{w.act},+} t + O(t^2) \geq 0, \quad \forall t \in [0, \epsilon).$$

The multipliers $\mu^{*\text{w.act},0}$ and $\mu^{*\text{inact}}$ are identical to zero on the curve. Therefore, the points $(w^*(t), \lambda^*(t), \mu^*(t))$ are KKT points for $t \in [0, \epsilon)$.

Furthermore, we can ensure by continuity of the first and second order partial derivatives $\nabla_{w,\lambda,\mu}\mathcal{L}(t, w^*(t), \lambda^*(t), \mu^*(t))$ and $\nabla^2_{w,\lambda,\mu}\mathcal{L}(t, w^*(t), \lambda^*(t), \mu^*(t))$ that the two remaining conditions of Theorem 3.3 (regularity and positive definiteness on the linearized strongly active constraints), are satisfied at all points on the curve v, by choosing ϵ suitably small. Note that the set $\bar{G}$ of strongly active constraints on the curve $(w^*(t), \lambda^*(t), \mu^*(t))$, $t \in (0, \epsilon)$, comprises always the set $\tilde{G}^{\text{s}}$ of strongly active constraints at the point $(w^*(0), \lambda^*(0), \mu^*(0))$. As the Hessian is positive definite on the null space of the linearized constraints $\tilde{G}^{\text{s}}$, it is also positive definite on the null space of the linearized strongly active constraints at a point $(w^*(t), \lambda^*(t), \mu^*(t))$, which is a subspace. □

Appendix D

Proof of Theorem 5.3

We will prove Theorem 5.3 in two steps: first it is shown that the assumptions of Theorem 5.1 are met and that the iterates therefore converge towards a KKT point y^*, and secondly it is shown that this point also satisfies the sufficient conditions of optimality as stated in Theorem 3.3.

Using the inversion formula (5.17)

$$J(y)^{-1} = C_1(y)A_r(y)^{-1}C_1(y)^T + C_2(y)$$

from Lemma 5.2 and the bounds (5.20a), (5.20b), and (5.20c), a bound on the norm of the inverse of J^{-1} on the domain D can be established:

$$\left\|J(y_1)^{-1}\right\| \leq \beta_{C_1}\beta_A\beta_{C_1} + \beta_{C_2} = \beta < \infty, \quad \forall\, y_1 \in D.$$

From continuity of $J(y)$, $J(y)^{-1}$ is continuous on D. Using the definition (5.7) of J and the full form of $\frac{\partial R}{\partial y}$ as shown in Eq. (5.6), we can conclude with assumption (5.20e) that

$$\left\|J(y_2) - \frac{\partial R}{\partial y}(y_2)\right\| = \left\|A(y_2) - \frac{\partial^2 \mathcal{L}}{\partial (q,s)^2}(y_2)\right\| \leq \frac{\kappa}{\beta}, \quad \forall\, y_2 \in D,$$

and therefore that the first condition (5.13a) of Theorem 5.1 is satisfied:

$$\left\|J(y_1)^{-1}\left(J(y_2) - \frac{\partial R}{\partial y}(y_2)\right)\right\| \leq \kappa < 1, \quad \forall\, y_1, y_2 \in D.$$

Assumption (5.20d) ensures that condition (5.13b) of Theorem 5.1 is also satisfied:

$$\left\|J(y_1)^{-1}\left(J(y_2) - J(y_3)\right)\right\| \leq \omega\|y_2 - y_3\|, \quad \forall\, y_1, y_2, y_3 \in D.$$

This allows to apply Theorem 5.1 to conclude that the iterates converge towards a point $y^* \in D_0 \subset D$ which satisfies $R(y^*) = 0$.

To prove that this point y^* is not only a regular KKT point, but also satisfies the sufficient conditions of optimality according to Theorem 3.3, it suffices to show that the

Hessian matrix $\nabla^2_{(q,s)}\mathcal{L}(q^*, s^*, \lambda^*)$ is positive definite on the null space of the linearized constraints $\nabla_{(q,s)}G(q^*, s^*)$. For this scope first note that the null space of the linearized constraints is spanned by the matrix

$$\begin{pmatrix} \mathbb{I} \\ -\left(\frac{\partial g}{\partial s}\right)^{-1}\frac{\partial g}{\partial q} \end{pmatrix},$$

and therefore it only needs to be shown that the reduced *exact* Hessian

$$A_{\mathrm{re}}(y^*) := \left(\mathbb{I} \,\middle|\, -\frac{\partial g}{\partial q}^T \left(\frac{\partial g}{\partial s}\right)^{-T} \right) \begin{pmatrix} \frac{\partial^2 \mathcal{L}}{\partial q^2} & \frac{\partial^2 \mathcal{L}}{\partial q \partial s}^T \\ \frac{\partial^2 \mathcal{L}}{\partial q \partial s} & \frac{\partial^2 \mathcal{L}}{\partial s^2} \end{pmatrix} \begin{pmatrix} \mathbb{I} \\ -\left(\frac{\partial g}{\partial s}\right)^{-1}\frac{\partial g}{\partial q} \end{pmatrix}$$

is positive definite. To show this, let us introduce the homotopy $A_\alpha : [0,1] \to \mathbb{R}^{(n_q \times n_q)}$

$$A_\alpha := (1-\alpha)A_r(y^*) + \alpha A_{\mathrm{re}}(y^*),$$

and note that

$$\left\| \begin{pmatrix} \mathbb{I} \\ -\left(\frac{\partial g}{\partial s}\right)^{-1}\frac{\partial g}{\partial q} \end{pmatrix} \right\| \leq \|C_1\| \leq \beta_{C_1},$$

so that

$$\begin{aligned} \|A_\alpha - A_r\| &= \left(\mathbb{I} \,\middle|\, -\frac{\partial g}{\partial q}^T \left(\frac{\partial g}{\partial s}\right)^{-T} \right) \alpha \left(\frac{\partial^2 \mathcal{L}}{\partial (q,s)^2} - A \right) \begin{pmatrix} \mathbb{I} \\ -\left(\frac{\partial g}{\partial s}\right)^{-1}\frac{\partial g}{\partial q} \end{pmatrix} \\ &\leq \beta_{C_1}\, \alpha\, \frac{\kappa}{\beta_{C_1}\beta_A\beta_{C_1} + \beta_{C_2}} \beta_{C_1} \leq \frac{\alpha\kappa}{\beta_A}. \end{aligned}$$

A_α is invertible for all $\alpha \in [0,1]$, as its inverse can be written

$$A_\alpha^{-1} = \left(A_r - (A_r - A_r A_r^{-1} A_\alpha)\right)^{-1} = \left(\mathbb{I} - (\mathbb{I} - A_r^{-1}A_\alpha)\right)^{-1} A_r^{-1}$$

and

$$\begin{aligned} \|\mathbb{I} - A_r^{-1}A_\alpha\| &= \|A_r^{-1}(A_r - A_\alpha)\| \\ &\leq \|A_r^{-1}\|\|A_r - A_\alpha\| \\ &\leq \beta_A \frac{\alpha\kappa}{\beta_A} = \alpha\kappa \leq \kappa < 1. \end{aligned}$$

As $A_0 = A_r$ is positive definite and A_α remains invertible for all $\alpha \in [0,1]$, none of the eigenvalues of A_α can become negative on the way from $\alpha = 0$ to $\alpha = 1$, so that in particular $A_1 = A_{\mathrm{re}}$ is positive definite. □

Appendix E

The Recursive Condensing Technique

For the first step of the condensing approach that was introduced in Sec. 6.5, some matrix products and sums have to be computed that involve the block sparse matrices B_{11}, B_{12}, B_{21}, B_{22}, A_{11}, A_{12}, and A_{22}, as defined in Sec. 6.5. We will show how the sparsity can be exploited to perform these computations efficiently.

The matrix $M := -B_{11}^{-1}B_{12}$ and the vector $m := -B_{11}^{-1}b_1$ can be calculated as follows. Computing

$$X_{0|0} := \mathbb{I}, \quad X_{i+1|0} := X_i X_{i|0}, \quad i = 0, \ldots, N-1,$$

and for $j = 0, \ldots, N-1$

$$Y_{j+1|j} := Y_j, \quad Y_{i+1|j} := X_i Y_{i|j}, \quad i = j+1, \ldots, N-1,$$

as well as $m_0 := 0, \quad m_{i+1} := X_i m_i - c_i, \quad i = 0, \ldots, N-1$, the matrix M and the vector m can be written as

$$M := \begin{pmatrix} X_{1|0} & Y_{1|0} & & & & \\ X_{2|0} & Y_{2|0} & Y_{2|1} & & & \\ X_{3|0} & Y_{3|0} & Y_{3|1} & Y_{3|2} & & \\ \vdots & \vdots & \vdots & \ddots & \ddots & \\ X_{N|0} & Y_{N|0} & Y_{N|1} & Y_{N|2} & \cdots & Y_{N|N-1} \\ 0 & 0 & 0 & 0 & 0 & \mathbb{I} \end{pmatrix} \quad \text{and} \quad m := \begin{pmatrix} m_1 \\ m_2 \\ m_3 \\ \vdots \\ m_N \\ 0 \end{pmatrix}.$$

The condensed Hessian matrix $\tilde{A} := M^T A_{11} M + M^T A_{12} + A_{12}^T M + A_{22}$ of the condensed QP can be efficiently computed if the structure of A and M is exploited. Computing

$$\begin{aligned} \tilde{A}_{\mathrm{ss}} &:= \textstyle\sum_{i=0}^{N} X_{i|0}^T Q_i X_{i|0}, \\ \tilde{A}_{\mathrm{s},j} &:= X_{j|0}^T S_j^T + \textstyle\sum_{k=j+1}^{N} X_{k|0}^T Q_k Y_{k|j}, \quad \text{for} \quad j = 0, \ldots, N, \\ \tilde{A}_{i,i} &:= R_i + \textstyle\sum_{k=i+1}^{N} Y_{k|i}^T Q_k Y_{k|i}, \quad \text{for} \quad i = 0, \ldots, N, \\ \tilde{A}_{i,j} &:= Y_{j|i}^T S_j^T + \textstyle\sum_{k=j+1}^{N} Y_{k|i}^T Q_k Y_{k|j}, \quad \text{for} \quad 0 \le i < j \le N, \\ \tilde{A}'_{i,N-1} &:= \tilde{A}_{i,N-1} + \tilde{A}_{i,N}, \quad \text{for} \quad i = 0, \ldots, N-2, \end{aligned}$$

and $\tilde{A}'_{\mathrm{s},N-1} := \tilde{A}_{\mathrm{s},N-1} + \tilde{A}_{\mathrm{s},N}$ as well as $\tilde{A}'_{N-1,N-1} := \tilde{A}_{N-1,N-1} + \tilde{A}_{N-1,N} + \tilde{A}^T_{N-1,N} + \tilde{A}_{N,N}$, we can define

$$\tilde{A} := \begin{pmatrix} \tilde{A}_{\mathrm{ss}} & \tilde{A}_{\mathrm{s},0} & \cdots & \tilde{A}_{\mathrm{s},N-2} & \tilde{A}'_{\mathrm{s},N-1} \\ \tilde{A}^T_{\mathrm{s},0} & \tilde{A}_{0,0} & \cdots & \tilde{A}_{0,N-2} & \tilde{A}'_{0,N-1} \\ \vdots & & \ddots & & \vdots \\ \tilde{A}^T_{\mathrm{s},N-2} & \tilde{A}^T_{0,N-2} & \cdots & \tilde{A}_{N-2,N-2} & \tilde{A}'_{N-2,N-1} \\ \tilde{A}'^T_{\mathrm{s},N-1} & \tilde{A}'^T_{0,N-1} & \cdots & \tilde{A}'^T_{N-2,N-1} & \tilde{A}'_{N-1,N-1} \end{pmatrix}.$$

Similarly, the condensed objective gradient $\tilde{a} = M^T A_{11} m + A_{12}^T m + M^T a_1 + a_2 = (\tilde{a}_{\mathrm{s}}, \tilde{a}_0, \ldots, \tilde{a}_{N-2}, \tilde{a}'_{N-1})$ can be calculated with

$$\tilde{a}^T_{\mathrm{s}} := g_0^{xT} + \sum_{i=1}^{N} \left(m_i^T Q_i + g_i^{xT} \right) X_{i|0},$$

and for $j = 0, \ldots, N$

$$\tilde{a}^T_j := g_j^{qT} + m_j^T S_j^T + \sum_{k=j+1}^{N} \left(m_i^T Q_i + g_i^{xT} \right) Q_k Y_{k|j},$$

and

$$\tilde{a}'_{N-1} := \tilde{a}_{N-1} + \tilde{a}_N.$$

The two remaining condensed constraint functions

$$\tilde{b} + \tilde{B}\Delta w_2 := \begin{pmatrix} \tilde{b}_{\mathrm{s}} \\ \tilde{b}_{\mathrm{r}} \end{pmatrix} + \begin{pmatrix} \tilde{B}_{\mathrm{s}} \\ \tilde{B}_{\mathrm{r}} \end{pmatrix} \Delta w_2,$$

$$\tilde{c} + \tilde{C}\Delta w_2 := \begin{pmatrix} \tilde{c}_{\mathrm{r}} \\ \tilde{c}_0 \\ \vdots \\ \tilde{c}_N \end{pmatrix} + \begin{pmatrix} \tilde{C}_{\mathrm{r}} \\ \tilde{C}_0 \\ \vdots \\ \tilde{C}_N \end{pmatrix} \Delta w_2,$$

are built according to

$$\begin{aligned}
&\tilde{b}_{\mathrm{s}} := s_0^x - x_0, \quad \tilde{B}_{\mathrm{s}} := -\mathbb{I}, \\
&\tilde{b}_{\mathrm{r}} := r^{\mathrm{e}} + R^{\mathrm{e},x} m_N, \quad \tilde{B}_{\mathrm{r}} := R^{\mathrm{e},x} \left(X_{N|0} | Y_{N|0} | \ldots | Y_{N|N-1} \right) + (0 | \ldots | 0 | R^{\mathrm{e},q}), \\
&\tilde{c}_{\mathrm{r}} := r^{\mathrm{i}} + R^{\mathrm{i},x} m_N, \quad \tilde{C}_{\mathrm{r}} := R^{\mathrm{i},x} \left(X_{N|0} | Y_{N|0} | \ldots | Y_{N|N-1} \right) + (0 | \ldots | 0 | R^{\mathrm{i},q}), \\
&\tilde{c}_i = h_i + H_i^x m_i, \; \tilde{C}_i := \left(H_i^x X_{i|0} | H_i^x Y_{i|0} | \ldots | H_i^x Y_{i|i-1} | H_i^q | 0 | \ldots \right), \\
&\tilde{c}_N = h_N + H_N^x m_N, \\
&\tilde{C}_N := \left(H_N^x X_{N|0} | H_N^x Y_{N|0} | \ldots | H_N^x Y_{N|N-2} | H_N^x Y_{N|N-1} + H_N^q \right).
\end{aligned}$$

Bibliography

[ABQ+99] F. Allgöwer, T. A. Badgwell, J. S. 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, 1999.

[ADM95] O. Abel, T. Daszkowski, and W. Marquardt. Vergleich konventioneller und prädiktiver Regelung am Beispiel eines absatzweise betriebenen Reaktors. Technical Report 1995-07, Lehrstuhl für Prozesstechnik, RWTH Aachen, 1995.

[AFN+00] F. Allgöwer, R. Findeisen, Z. Nagy, M. Diehl, H.G. Bock, and J.P. Schlöder. Efficient nonlinear model predictive control for large scale constrained processes. In *Proceedings of the Sixth International Conference on Methods and Models in Automation and Robotics*, pages 43–54. Miedzyzdroje, Poland, 2000.

[Ama83] Herbert Amann. *Gewöhnliche Differentialgleichungen.* de Gruyter, Berlin; New York, 1983.

[Bau00] I. Bauer. *Numerische Verfahren zur Lösung von Anfangswertaufgaben und zur Generierung von ersten und zweiten Ableitungen mit Anwendungen bei Optimierungsaufgaben in Chemie und Verfahrenstechnik.* PhD thesis, University of Heidelberg, 2000.

[BBB+01] T. Binder, L. Blank, H.G. Bock, R. Bulirsch, W. Dahmen, M. Diehl, T. Kronseder, W.Marquardt, J.P. Schlöder, and O. v.Stryk. Introduction to model based optimization of chemical processes on moving horizons. In M. Groetschel, S.O. Krumke, and J. Rambau, editors, *Online Optimization of Large Scale Systems: State of the Art.* Springer, 2001.

[BBLS99] H.G. Bock, I. Bauer, D.B. Leineweber, and J.P. Schlöder. Direct multiple shooting methods for control and optimization of dae in chemical engineering. In F. Keil, W. Mackens, H. Voß, and J. Werther, editors, *Scientific Computing in Chemical Engineering II*, volume 2, pages 2–18, Berlin, 1999. Springer.

[BBS99] I. Bauer, H. G. Bock, and J. P. Schlöder. DAESOL – a BDF-code for the numerical solution of differential algebraic equations. Internal report, IWR, SFB 359, Universität Heidelberg, 1999.

[BDLS00] H.G. Bock, M. Diehl, D. Leineweber, and J.P. Schlöder. A direct multiple shooting method for real-time optimization of nonlinear DAE processes. In F. Allgöwer and A. Zheng, editors, *Nonlinear Predictive Control*, volume 26 of *Progress in Systems Theory*, pages 246–267, Basel, 2000. Birkhäuser.

[BDS+00] H.G. Bock, M. Diehl, J.P. Schlöder, F. Allgöwer, R. Findeisen, and Z. Nagy. Real-time optimization and nonlinear model predictive control of processes governed by differential-algebraic equations. In *ADCHEM2000 - International Symposium on Advanced Control of Chemical Processes*, volume 2, pages 695–703, Pisa, 2000.

[Bel57] R.E. Bellman. *Dynamic Programming.* Princeton University Press, 1957.

[Ber95a] D.P. Bertsekas. *Dynamic Programming and Optimal Control*, volume 1. Athena Scientific, Belmont, MA, 1995.

[Ber95b] D.P. Bertsekas. *Dynamic Programming and Optimal Control*, volume 2. Athena Scientific, Belmont, MA, 1995.

[BES88] H.G. Bock, E. Eich, and J.P. Schlöder. Numerical solution of constrained least squares boundary value problems in differential-algebraic equations. In K. Strehmel, editor, *Numerical Treatment of Differential Equations.* Teubner, Leipzig, 1988.

[BH69] A. E. Bryson and Y.-C. Ho. *Applied Optimal Control.* Ginn and Company, Waltham, Massachusetts, 1969.

[Bie84] L. T. Biegler. Solution of dynamic optimization problems by successive quadratic programming and orthogonal collocation. *Comput. Chem. Engng.*, 8:243–248, 1984.

[Bie00] L. Biegler. Efficient solution of dynamic optimization and NMPC problems. In F. Allgöwer and A. Zheng, editors, *Nonlinear Predictive Control*, volume 26 of *Progress in Systems Theory*, pages 219–244, Basel, 2000. Birkhäuser.

[Boc81] H.G. Bock. Numerical treatment of inverse problems in chemical reaction kinetics. In K. H. Ebert, P. Deuflhard, and W. Jäger, editors, *Modelling of Chemical Reaction Systems*, volume 18 of *Springer Series in Chemical Physics.* Springer, Heidelberg, 1981.

[Boc83] H. G. Bock. Recent advances in parameter identification techniques for ODE. In P. Deuflhard and E. Hairer, editors, *Numerical Treatment of Inverse Problems in Differential and Integral Equations.* Birkhäuser, Boston, 1983.

[Boc87] H.G. Bock. *Randwertproblemmethoden zur Parameteridentifizierung in Systemen nichtlinearer Differentialgleichungen*, volume 183 of *Bonner Mathematische Schriften.* University of Bonn, Bonn, 1987.

[BP84] H.G. Bock and K.J. Plitt. A multiple shooting algorithm for direct solution of optimal control problems. In *Proc. 9th IFAC World Congress Budapest*. Pergamon Press, 1984.

[BR91] L. T. Biegler and J. B. Rawlings. Optimization approaches to nonlinear model predictive control. In Y. Arkun and W. H. Ray, editors, *Chemical Process Control – CPC IV*, pages 543–571, Austin, Texas, 1991. The CACHE Corp.

[BS79] M. S. Bazaara and C. M. Shetty. *Nonlinear Programming: Theory and Applications*. Wiley, New York, 1979.

[BS81] H.G. Bock and J.P. Schlöder. *Numerical solution of retarded differential equations with state-dependent time lags*, volume 61. Z. angew. Math. Mech., 1981.

[BS96] D.P. Bertsekas and S.E. Shreve. *Stochastic Optimal Control: The Discrete Time Case*. Athena Scientific, Belmont, MA, 1996.

[BS01] H.G. Bock and J.P. Schlöder. An efficient multiple shooting based reduced sqp strategy for large-scale dynamic process optimization. *Comp. Chem. Eng.*, 2001. in preparation.

[BT96] D.P. Bertsekas and J.N. Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, Belmont, MA, 1996.

[Bür01] Tobias Bürner. Algorithmen zur on-line Parameter- und Zustandsschätzung bei DAE Systemen. Master's thesis, University of Heidelberg, 2001.

[CA98] H. Chen and F. Allgöwer. A quasi-infinite horizon nonlinear model predictive control scheme with guaranteed stability. *Automatica*, 34(10):1205–1218, 1998.

[CB89] J.E. Cuthrell and L.T. Biegler. Simultaneous optimization and solution methods for batch reactor profiles. *Comp. & Chem. Eng.*, 13(1/2):49–62, 1989.

[Che97] H. Chen. *Stability and Robustness Considerations in Nonlinear Model Predictive Control*. Fortschr.-Ber. VDI Reihe 8 Nr. 674. VDI Verlag, Düsseldorf, 1997.

[CKA95] H. Chen, A. Kremling, and F. Allgöwer. Nonlinear predictive control of a benchmark CSTR. In *Proc. 3rd European Control Conference ECC'95*, pages 3247–3252, Rome, 1995.

[CSA97] H. Chen, C.W. Scherer, and F. Allgöwer. A game theoretic approach to nonlinear robust receding horizon control of constrained systems. In *Proc. Amer. Contr. Conf.*, pages 3073–3077, Albuquerque, 1997.

[DBLS99] M. Diehl, H.G. Bock, D.B. Leineweber, and J.P. Schlöder. Efficient direct multiple shooting in nonlinear model predictive control. In F. Keil, W. Mackens, H. Voß, and J. Werther, editors, *Scientific Computing in Chemical Engineering II*, volume 2, pages 218–227, Berlin, 1999. Springer.

[DBS+01] M. Diehl, H.G. Bock, J.P. Schloeder, R. Findeisen, Z. Nagy, and F. Allgoewer. Real-time optimization and nonlinear model predictive control of processes governed by differential-algebraic equations. *J. Proc. Contr.*, 2001.

[Die98] M. Diehl. A direct multiple shooting method for the optimization and control of chemical processes. Master's thesis, University of Heidelberg, 1998.

[Die01] Angelika Dieses. Numerical methods for optimization problems in water flow and reactive solute transport processes of xenobiotics in soils. Technical Report SFB Preprint 2001-07, University of Heidelberg, 2001. Ph.D. Thesis.

[DMS96] G. De Nicolao, L. Magni, and R. Scattolini. Stabilizing nonlinear receding horizon control via a nonquadratic terminal state penalty. In *Symposium on Control, Optimization and Supervision, CESA'96 IMACS Multiconference*, pages 185–187, Lille, 1996.

[DMS00] G. De Nicolao, L. Magni, and R. Scattolini. Stability and robustness of nonlinear receding horizon control. In F. Allgöwer and A. Zheng, editors, *Nonlinear Predictive Control*, volume 26 of *Progress in Systems Theory*, pages 3–23, Basel, 2000. Birkhäuser.

[DR96] I. S. Duff and J. K. Reid. The design of MA48: A code for the direct solution of sparse unsymmetric linear systems of equations. *ACM Trans. Math. Softw.*, 22:187–226, 1996.

[DUF+01] M. Diehl, I. Uslu, R. Findeisen, S. Schwarzkopf, F. Allgöwer, H.G. Bock, T. Bürner, E.D. Gilles, A. Kienle, J.P. Schlöder, and E. Stein. Real-time optimization of large scale process models: Nonlinear model predictive control of a high purity distillation column. In M. Groetschel, S.O. Krumke, and J. Rambau, editors, *Online Optimization of Large Scale Systems: State of the Art*. Springer, 2001.

[DV93] J.J. Downs and E.F. Vogel. A plant-wide industrial process control problem. *Comp. & Chem. Eng.*, 17:245–255, 1993.

[EKKvS99] G. Engl, A. Kröner, T. Kronseder, and O. von Stryk. Numerical simulation and optimal control of air separation plants. In Chr. Zenger H.-J. Bungartz, F. Durst, editor, *High Performance Scientific and Engineering Computing*, volume 8 of *Lecture Notes in Computational Science and Engineering*, pages 221–231. Springer, 1999.

[FA00] R. Findeisen and F. Allgöwer. Nonlinear model predictive control for index–one DAE systems. In F. Allgöwer and A. Zheng, editors, *Nonlinear Predictive Control*, pages 145–162. Birkhäuser, 2000.

[FAD+00] R. Findeisen, F. Allgöwer, M. Diehl, H.G. Bock, J.P. Schlöder, and Z. Nagy. Efficient nonlinear model predictive control. submitted, 2000.

[Fle87] R. Fletcher. *Practical Methods of Optimization.* Wiley, Chichester, 2 edition, 1987.

[GB94] J. V. Gallitzendörfer and H. G. Bock. Parallel algorithms for optimization boundary value problems in dae. In H. Langendörfer, editor, *Praxisorientierte Parallelverarbeitung.* Hanser, München, 1994.

[GMSW83] P. E. Gill, W. Murray, M. A. Saunders, and M. H. Wright. User's guide for SOL/QPSOL: a fortran package for quadratic programming. Technical Report SOL 83-7, Systems Optimization Laboratory, Department of Operations Research, Stanford University, 1983.

[GPM89] C. E. García, D. M. Prett, and M. Morari. Model predictive control: Theory and practice – a survey. *Automatica*, 25:335ff, 1989.

[GV83] C. W. Gear and T. Vu. Smooth numerical solutions of ordinary differential equations. In P. Deuflhard and E. Hairer, editors, *Numerical Treatment of Inverse Problems in Differential and Integral Equations.* Birkhäuser, Boston, 1983.

[GVJ90] J. Guddat, F. Guerra Vasquez, and H. Th. Jongen. *Parametric Optimization: Singularities, Pathfollowing and Jumps.* B.G.Teubner; John Wiley & Sons, 1990.

[Han76] S. P. Han. Superlinearly convergent variable-metric algorithms for general nonlinear programming problems. *Math. Progr.*, 11:263–282, 1976.

[Hin98] H. Hinsberger. *Ein direktes Mehrzielverfahren zur Lösung von Optimalsteuerungsproblemen mit großen, differential-algebraischen Gleichungssystemen und Anwendungen aus der Verfahrenstechnik.* PhD thesis, Technical University of Clausthal, 1998.

[HMP96] H. Hinsberger, S. Miesbach, and H. J. Pesch. Optimal temperature control of semibatch polymerization reactors. In F. Keil, H. VoßW. Mackens, and J. Werther, editors, *Scientific Computing in Chemical Engineering*, Heidelberg, 1996. Springer.

[HR71] G. A. Hicks and W. H. Ray. Approximation methods for optimal control systems. *Can. J. Chem. Engng*, 49:522–528, 1971.

[JT61] P.D. Joseph and J.T. Tou. On linear control theory. *Trans. AIEE*, 80(18), 1961.

[Kal60] R.E. Kalman. A new approach to linear filtering and prediction problems. *Trans. ASME, J. Basic Engineering*, pages 35–45, March 1960.

[Kar39] W. Karush. Minima of functions of several variables with inequalities as side conditions. Master's thesis, Department of Mathematics, University of Chicago, 1939.

[KE85] P. Krämer-Eis. *Ein Mehrzielverfahren zur numerischen Berechnung optimaler Feedback-Steuerungen bei beschränkten nichtlinearen Steuerungsproblemen*, volume 166 of *Bonner Mathematische Schriften*. University of Bonn, 1985.

[KE93] K.-U. Klatt and S. Engell. Rührkesselreaktor mit Parallel- und Folgereaktion. In S. Engell, editor, *Nichtlineare Regelung – Methoden, Werkzeuge, Anwendungen. VDI-Berichte Nr. 1026*, pages 101–108. VDI-Verlag, Düsseldorf, 1993.

[KEB87] P. Krämer-Eis and H.G. Bock. Numerical treatment of state and control constraints in the computation of feedback laws for nonlinear control problems. In P. Deuflhard et al., editor, *Large Scale Scientific Computing*, pages 287–306. Birkhäuser, 1987.

[KM72] V. Klee and G.J. Minty. How good is the simplex algorithm? In O. Shisha, editor, *Inequalities*, volume III, pages 159–175. Academic Press, New York, 1972.

[KP90a] B. Kugelmann and H.J. Pesch. New general guidance method in constrained optimal control, part 1: Numerical method. *J. Optimization Theory and Application*, 67(3):421–435, 1990.

[KP90b] B. Kugelmann and H.J. Pesch. New general guidance method in constrained optimal control, part 2: Application to space shuttle guidance. *J. Optimization Theory and Application*, 67(3):437–446, 1990.

[Kra85] D. Kraft. On converting optimal control problems into nonlinear programming problems. In K. Schittkowski, editor, *Computational Mathematical Programming*, volume F15 of *NATO ASI*, pages 261–280. Springer, 1985.

[KT51] H. W. Kuhn and A. W. Tucker. Nonlinear programming. In J. Neyman, editor, *Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability*, Berkeley, 1951. University of California Press.

[KvSB01] T. Kronseder, O. von Stryk, and R. Bulirsch. Towards nonlinear model based predictive optimal control of large-scale process models with application to air

separation plants. In J. Rambau M. Groetschel, S.O. Krumke, editor, *Online Optimization of Large Scale Systems: State of the Art*. Springer, 2001.

[LB89] W.C. Li and L.T. Biegler. Multistep, newton-type control strategies for constrained nonlinear processes. *Chem. Eng. Res. Des.*, 67:562–577, 1989.

[LBS97] D.B. Leineweber, H.G. Bock, and J.P. Schlöder. Fast direct methods for real-time optimization of chemical processes. In *Proc. 15th IMACS World Congress on Scientific Computation, Modelling and Applied Mathematics Berlin*, Berlin, 1997. Wissenschaft- und Technik-Verlag.

[Lei96] D. B. Leineweber. The theory of muscod in a nutshell. IWR-Preprint 96-19, University of Heidelberg, 1996.

[Lei99] D.B. Leineweber. *Efficient reduced SQP methods for the optimization of chemical processes described by large sparse DAE models*, volume 613 of *Fortschr.-Ber. VDI Reihe 3, Verfahrenstechnik*. VDI Verlag, Düsseldorf, 1999.

[LEL92] M.J. Liebman, T.F. Edgar, and L.S. Lasdon. Efficient data reconciliation and estimation for dynamic processes using nonlinear programming techniques. *Comp. & Chem. Eng.*, 16(10/11):963–986, 1992.

[LM68] E.B. Lee and L. Markus. *Foundations of Optimal Control Theory*. Wiley, New York, 1968.

[LMG94] J. H. Lee, M. Morari, and C.E. García. State-space interpretation of model predictive control. *Automatica*, 30(4):707–717, 1994.

[Loc86] M.J. Lockett. *Distillation Tray fundamentals*. Cambridge University Press, Cambridge, 1986.

[May96] 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 – CPC V*, pages 217–231. American Institute of Chemical Engineers, 1996.

[May00] D.Q. Mayne. Nonlinear model predictive control: Challenges and opportunities. In F. Allgöwer and A. Zheng, editors, *Nonlinear Predictive Control*, volume 26 of *Progress in Systems Theory*, pages 23–44, Basel, 2000. Birkhäuser.

[ME96] K.R. Muske and T.F. Edgar. Nonlinear state estimation. In M.A. Henson and D.E. Seborg, editors, *Nonlinear Process Control*, pages 311–370. Prentice Hall, 1996.

[MHAM96] A. M'hamdi, A. Helbig, O. Abel, and W. Marquardt. Newton-type receding horizon control and state estimation. In *Proc. 13rd IFAC World Congress*, pages 121–126, San Francisco, 1996.

[MM90] D.Q. Mayne and H. Michalska. Receding horizon control of nonlinear systems. *IEEE Trans. Automat. Contr.*, 35(7):814–824, 1990.

[MM93] H. Michalska and D.Q. Mayne. Robust receding horizon control of constrained nonlinear systems. *IEEE Trans. Automat. Contr.*, AC-38(11):1623–1633, 1993.

[Mor87] M. Morari. Robust process control. *Chem. Eng. Res. Des.*, 65:462–479, 1987.

[NFD+00] Z. Nagy, R. Findeisen, M. Diehl, F. Allgöwer, H.G. Bock, S. Agachi, J.P. Schlöder, and D.B. Leineweber. Real-time feasibility of nonlinear predictive control for large scale processes – a case study. In *Proc. of ACC 2000*, Chicago, 2000. in press.

[NW99] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, 1999.

[OB95a] N. M.C. de Oliveira and L. T. Biegler. Newton-type algorithms for nonlinear process control. algorithm and stability results. *Automatica*, 31(2):281–286, 1995.

[OB95b] N.M.C. de Oliveira and L.T. Biegler. An extension of Newton-type algorithms for nonlinear process control. *Automatica*, 31(2), 1995.

[OM94] S. de Oliveira and M. Morari. Robust model predictive control for nonlinear systems. In *Proc. 33rd IEEE Conf. Decision Contr.*, pages 3561–3566, Lake Buena Vista, FL, December 1994. IEEE.

[PB93] C.C. Pantelides and P.I. Barton. Equation oriented dynamic simulation: Current status and future perspectives. *Comp. Chem. Eng.*, 17S:S263–S285, 1993.

[Pes78] H.J. Pesch. *Numerische Berechnung optimaler Flugbahnkorrekturen in Echtzeitrechnung*. PhD thesis, TU München, 1978.

[Pli81] K.J. Plitt. Ein superlinear konvergentes Mehrzielverfahren zur direkten Berechnung beschränkter optimaler Steuerungen. Master's thesis, University of Bonn, 1981.

[Pow78] M. J. D. Powell. A fast algorithm for nonlinearly constrained optimization calculations. In G. A. Watson, editor, *Numerical Analysis, Dundee 1977*, volume 630 of *Lecture Notes in Mathematics*, Berlin, 1978. Springer.

[PRG+97] L. Petzold, J.B. Rosen, P.E. Gill, L.O. Jay, and K. Park. Numerical optimal control of parabolic PDEs using DASOPT. In Biegler, Coleman, Conn, and Santosa, editors, *Large Scale Optimization with Applications, Part II*. Springer, 1997.

[PSV94] C. C. Pantelides, R. W. H. Sargent, and V. S. Vassiliadis. Optimal control of multistage systems described by high-index differential-algebraic equations. In R. Bulirsch and D. Kraft, editors, *Computational Optimal Control*. Birkhäuser, Basel, 1994.

[QB96] 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, *Fifth International Conference on Chemical Process Control – CPC V*, pages 232–256. American Institute of Chemical Engineers, 1996.

[QB00] S.J. Qin and T.A. Badgwell. An overview of nonlinear model predictive control applications. In F. Allgöwer and A. Zheng, editors, *Nonlinear Predictive Control*, volume 26 of *Progress in Systems Theory*, pages 370–392, Basel, 2000. Birkhäuser.

[RBP+99] R. Ross, V. Bansal, J.D. Perkins, E.N. Pistikopoulos, J.M.G. van Schijndel, and G.L.M. Koot. Optimal design and control of an industrial distillation system. *Comput. Chem. Eng.*, 23(SS):S875–S878, 1999.

[RL91] O. Rosen and R. Luus. Evaluation of gradients for piecewise constant optimal control. *Comput. Chem. Engng*, 15:273–281, 1991.

[RMM94] J.B. Rawlings, E.S. Meadows, and K.R. Muske. Nonlinear model predictive control: A tutorial and survey. In *Proc. Int. Symp. Adv. Control of Chemical Processes, ADCHEM*, Kyoto, Japan, 1994.

[Rob74] S. M. Robinson. Perturbed kuhn-tucker points and rates of convergence for a class of nonlinear programming algorithms. *Math. Progr.*, 7:1–16, 1974.

[RR00] C.V. Rao and J.B. Rawlings. Nonlinear moving horizon state estimation. In F. Allgöwer and A. Zheng, editors, *Nonlinear Predictive Control*, pages 45–69. Birkhäuser, 2000.

[RWR98] C.V. Rao, S.J. Wright, and J.B. Rawlings. Application of interior-point methods to model predictive control. *J. Opt. Theory and Appl.*, 99:723–757, 1998.

[SAC+00] L. O. Santos, P. Afonso, J. Castro, N. M.C. de Oliveira, and L. T. Biegler. On-line implementation of nonlinear MPC: An experimental case study. In *ADCHEM2000 - International Symposium on Advanced Control of Chemical Processes*, volume 2, pages 731–736, Pisa, 2000.

[San00] L.O. Santos. *Multivariable Predictive Control of Nonlinear Chemical Processes*. PhD thesis, Universidade de Coimbra, 2000.

[SBS98] V.H. Schulz, H.G. Bock, and M.C. Steinbach. Exploiting invariants in the numerical solution of multipoint boundary value problems for daes. *SIAM J. Sci. Comp.*, 19:440–467, 1998.

[Sch88] J.P. Schlöder. *Numerische Methoden zur Behandlung hochdimensionaler Aufgaben der Parameteridentifizierung*, volume 187 of *Bonner Mathematische Schriften*. University of Bonn, Bonn, 1988.

[Sch96] V.H. Schulz. *Reduced SQP methods for large-scale optimal control problems in DAE with application to path planning problems for satellite mounted robots*, volume 96-12 of *IWR-Preprint*. University of Heidelberg, 1996. Ph.D. thesis.

[Sim56] H.A. Simon. Dynamic programming under uncertainty with a quadratic criterion function. *Econometrica*, 24:74–81, 1956.

[SOB95] L. O. Santos, N. M.C de Oliveira, and L. T. Biegler. Reliable and efficient optimization strategies for nonlinear model predictive control. In *Proc. Fourth IFAC Symposium DYCORD+'95*, pages 33–38, Oxford, 1995. Elsevier Science.

[Son90] E.D. Sontag. *Mathematical Control Theory*. Springer-Verlag, New York, 1990.

[Sor99] E. Sorensen. A cylic operating policy for batch distillation - theory and practice. *Comput. Chem. Eng.*, 23 (4-5):533–542, 1999.

[SS78] R. W. H. Sargent and G. R. Sullivan. The development of an efficient optimal control package. In J. Stoer, editor, *Proceedings of the 8th IFIP Conference on Optimization Techniques (1977), Part 2*, Heidelberg, 1978. Springer.

[Ste95] M.C. Steinbach. *Fast recursive SQP methods for large-scale optimal control problems*. Ph.d. thesis, University of Heidelberg, 1995.

[TA94] P. Terwiesch and M. Agarwal. On-line corrections of pre-optimized input profiles for batch reactors. *Comp. Chem. Eng.*, 18:433–437, 1994.

[TB95] P. Tanartkit and L. T. Biegler. Stable decomposition for dynamic optimization. *Ind. Eng. Chem. Res.*, 34:1253–1266, 1995.

[TB96] P. Tanartkit and L. T. Biegler. A nested, simultaneous approach for dynamic optimization problems – i. *Comput. Chem. Engng*, 20:735–741, 1996.

[THE75] T.H. Tsang, D.M. Himmelblau, and T.F. Edgar. Optimal control via collocation and non-linear programming. *Int. J. Control*, 1975.

[VSP94a] V.S. Vassiliadis, R.W.H. Sargent, and C.C. Pantelides. Solution of a class of multistage dynamic optimization problems. 1. problems without path constraints. *Ind. Eng. Chem. Res.*, 10(33):2111–2122, 1994.

[VSP94b] V.S. Vassiliadis, R.W.H. Sargent, and C.C. Pantelides. Solution of a class of multistage dynamic optimization problems. 2. problems with path constraints. *Ind. Eng. Chem. Res.*, 10(33):2122–2133, 1994.

[Web95] R. Weber. Prädiktive regelung von nichtlinearen systemen. Semesterarbeit ifa 8781, Institut für Automatik, ETH Zürich, 1995.

[Wil63] R. B. Wilson. *A simplicial algorithm for concave programming.* PhD thesis, Harvard University, 1963.

[Wil65] J. H. Wilkinson. *The Algebraic Eigenvalue Problem.* Claredon Press, Oxford, 1965.

[Wri96] S. J. Wright. Applying new optimization algorithms to model predictive control. In J.C. Kantor, C.E. Garcia, and B. Carnahan, editors, *Fifth International Conference on Chemical Process Control – CPC V*, pages 147–155. American Institute of Chemical Engineers, 1996.

[Wri97] S.J. Wright. *Primal-Dual Interior-Point Methods.* SIAM Publications, Philadelphia, 1997.

[YP93] 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.

[ZDG96] K. Zhou, J.C. Doyle, and K. Glover. *Robust and optimal control.* Prentice-Hall, 1996.