Inaugural-Dissertation

zur

Erlangung der Doktorwürde

der

Naturwissenschaftlich-Mathematischen Gesamtfakultät

der

Ruprecht-Karls-Universität

Heidelberg

vorgelegt von

Dipl.-Math. Leonard Wirsching

geboren in Heidelberg

Tag der mündlichen Prüfung:

.....

Multi-Level Iteration Schemes with Adaptive Level Choice for Nonlinear Model Predictive Control

Leonard Wirsching

Gutachter:

Prof. Dr. Dr. h. c. mult. Hans Georg Bock

.....

Zusammenfassung

In der vorliegenden Arbeit entwickeln wir die Multilevel-Iterationsschemata (MLI), ein numerischer Ansatz für die nichtlineare modellprädiktive Regelung (NMPC) von Prozessen, deren dynamische Modelle durch gewöhnliche Differentialgleichungen beschrieben werden. Der Ansatz basiert auf dem direkten Mehrzielverfahren für die Diskretisierung der Optimalsteuerungsprobleme, die für jede Abtastzeit gelöst werden müssen. Die daraus entstehenden parametrischen nichtlinearen Optimierungsprobleme werden näherungsweise gelöst, indem man in einer Vorbereitungsphase einen verallgemeinerten tangentialen Prädiktor aufstellt. Dieser besteht aus einem parametrischen quadratischen Programm (QP), welches implizit ein stückweise affin-lineares Regelgesetz definiert. In einer Feedback-Phase wird das Regelgesetz dann für die aktuelle Systemschätzung durch die Lösung des quadratischen Programms ausgewertet.

Der in dieser Arbeit entwickelte Ansatz liefert eine signifikante Einsparung im Rechenaufwand durch eine hierarchische Aktualisierung der Matrix- und Vektordaten des verallgemeinerten tangentialen Prädiktors auf vier Ebenen beziehungsweise Levels. Auf dem untersten Level werden keine Updates berechnet und nur die Regelung für den aktuellen Systemzustand berechnet. Das zweite Level umfasst das Update der QP-Beschränkungen und berechnet eine Approximation des QP-Gradienten, auf dem dritten Level wird statt der Approximation der exakte QP-Gradient ausgerechnet und auf dem obersten Level werden alle Vektor- und Matrixdaten des QPs neu ausgewertet. Feedback-Schemata sind dann eine Abfolge von Levelentscheidungen für jede einzelne Abtastzeit und damit eine sukzessive Aktualisierung des stückweise affin-linearen Lenkgesetzes, welches vom verallgemeinerten tangentialen Prädiktor implizit aufgestellt wird.

Wir präsentieren und diskutieren vier Strategien für die Datenkommunikation zwischen den Leveln innerhalb eines Schemas und wir beschreiben, wie Schemata mit vorgegebenen Levelentscheidungen in der Praxis zusammengestellt werden können. Wir zeigen lokale Konvergenztheorie für die Level mit eigenen primal-dualen Variablen bei festem Systemzustand und diskutieren die vorhandene Konvergenztheorie für den Fall eines geregelten Prozesses. Weiterhin skizzieren wir eine Variante der Level, die zusätzliche rechnerische Einsparungen ermöglicht.

Für die adaptive Wahl der Level zur Laufzeit entwickeln wir zwei kontraktionsbasierte Kriterien, um zu entscheiden, ob die aktuell verwendeten Linearisierungen brauchbar bleiben und verwenden diese Kriterien in einem Algorithmus zur Wahl des Levels für die nächste Abtastzeit. Außerdem schlagen wir im Falle eines mitgeführten Zustandsschätzers ein Kriterium vor, welches zusätzliche Entscheidungshilfe bei der Levelwahl für die nächste Abtastzeit bietet. Aufbauend auf dem zweiten Level schlagen wir einen effizienten Algorithmus für suboptimales NMPC vor.

Für die vorgestellten algorithmischen Ansätze beschreiben wir Strukturausnutzung mittels eines angepassten Kondensierungsalgorithmus, skizzieren die Online Active Set Strategie als effizienten Ansatz, um die quadratischen Teilprobleme zu lösen und erweitern diese Methode auf lineare Least-Squares-Probleme, und entwickeln iterative matrixfreie Methoden für eines der kontraktionsbasierten Kriterien, welches den Spektralradius der Iterationsmatrix schätzt.

Wir beschreiben drei Anwendungsbereiche, in denen MLI eine bedeutende Einsparung an Rechenleistung im Vergleich zu aktuellen numerischen Methoden für NMPC bewirkt. Sowohl für MLI-Schemata mit vorgegebener als auch adaptiver Levelwahl führen wir umfangreiche numerische Tests mittels anspruchsvollen nichtlinearen Testproblemen durch und vergleichen die Leistungsfähigkeit von MLI mit dem aktuellen Stand der Technik für NMPC. Die Schemata, die MLI mit adaptiver Levelwahl erzeugt, sind rechnerisch viel billiger bei vergleichbarer Reglerperformanz, und per Konstruktion ist MLI mit adaptiver Levelwahl in der Lage, Feedbacksteuerungen mit einer viel höheren Frequenz zu geben, was für die betrachteten Testprobleme eine deutliche Verbesserung der Reglerperformanz bewirkt.

Zur Durchführung der numerischen Experimente haben wir den vorgeschlagenen Ansatz in einer auf MATLAB® basierenden Software namens MLI implementiert, welches ein Softwarepaket zur effizienten automatischen Generierung von Ableitungen erster und höherer Ordnung für die Lösung des dynamischen Modells sowie für Zielfunktion und Beschränkungen nutzt, Strukturausnutzung durch Condensing durchführt und die parametrischen quadratischen Teilprobleme mittels eines Softwarepaketes mit einer effizienten Implementierung der Online Active Set Strategie löst.

Abstract

In this thesis we develop the Multi-Level Iteration schemes (MLI), a numerical method for Nonlinear Model Predictive Control (NMPC) where the dynamical models are described by ordinary differential equations. The method is based on Direct Multiple Shooting for the discretization of the optimal control problems to be solved in each sample. The arising parametric nonlinear problems are solved approximately by setting up a generalized tangential predictor in a preparation phase. This generalized tangential predictor is given by a quadratic program (QP), which implicitly defines a piecewise affine linear feedback law. The feedback law is then evaluated in a feedback phase by solving the QP for the current state estimate as soon as it becomes known to the controller.

The method developed in this thesis yields significant computational savings by updating the matrix and vector data of the tangential predictor in a hierarchy of four levels. The lowest level performs no updates and just calculates the feedback for a new initial state estimate. The second level updates the QP constraint functions and approximates the QP gradient. The third level updates the QP constraint functions and calculates the exact QP gradient. The fourth level evaluates all matrix and vector data of the QP. Feedback schemes are then assembled by choosing a level for each sample. This yields a successive update of the piecewise affine linear feedback law that is implicitly defined by the generalized tangential predictor.

We present and discuss four strategies for data communication between the levels in a scheme and we describe how schemes with fixed level choices can be assembled in practice. We give local convergence theory for each level type holding its own set of primal-dual variables for fixed initial values, and discuss existing convergence theory for the case of a closed-loop process. We outline a modification of the levels that yields additional computational savings.

For the adaptive choice of the levels at runtime, we develop two contraction-based criteria to decide whether the currently used linearization remains valid and use them in an algorithm to decide which level to employ for the next sample. Furthermore, we propose a criterion applicable to online estimation. The criterion provides additional information for the level decision for the next sample. Focusing on the second lowest level, we propose an efficient algorithm for suboptimal NMPC.

For the presented algorithmic approaches, we describe structure exploitation in the form of tailored condensing, outline the Online Active Set Strategy as an efficient way to solve the quadratic subproblems and extend the method to linear least-squares problems. We develop iterative matrix-free methods for one contraction-based criterion, which estimates the spectral radius of the iteration matrix.

We describe three application fields where MLI provides significant computational sav-

ings compared to state-of-the-art numerical methods for NMPC. For both fixed and adaptive MLI schemes, we carry out extensive numerical testings for challenging nonlinear test problems and compare the performance of MLI to a state-of-the-art numerical method for NMPC. The schemes obtained by adaptive MLI are computationally much cheaper while showing comparable performance. By construction, the adaptive MLI allows giving feedback with a much higher frequency, which significantly improves controller performance for the considered test problems.

To perform the numerical experiments, we have implemented the proposed method within a MATLAB® based software called MLI, which makes use of a software package for the automatic derivative generation of first and higher order for the solution of the dynamic model as well as objective and constraint functions, which performs structure exploitation by condensing, and which efficiently solves the parametric quadratic subproblems by using a software package that provides an implementation of the Online Active Set Strategy.

Acknowledgments

I would like to thank Deutsche Forschungsgesellschaft (DFG) for support within the International Graduiertenkolleg IGK 710 “Complex processes: Modeling, Simulation and Optimization”, the Heidelberg Graduate School of Mathematical and Computational Methods for the Sciences (HGS MathComp) and the projects “Optimierungsbasierte Regelung verfahrenstechnischer Prozesse” part 1 (project ID 5402002) and part 2 (project ID 5400160). I am also grateful for support of the European Commision within the projects “Embedded Optimization for Resource Constrained Platforms” (EMBOCON, Project ID 248940, Funded under FP7-ICT) and “Model-based Optimizing Control - from a vision to industrial reality” (MOBOCON, Project ID 291458, Funded under FP7-IDEAS-ERC). Furthermore I would like to acknowledge support by Bundesministerium für Bildung und Forschung (BMBF) within the Verbundprojekt “Modellbasierte Optimierung von Pharma-Prozessen” (MOPhaPro, Project ID 05M2106).

I would like to express my deep gratitude to my teacher and advisor Hans Georg Bock and to Johannes Schlöder for their constant support, their advice, and their ongoing trust in my capabilities. It was an honour and a highly enriching experience to be able to do this thesis in their workgroup Simulation and Optimization at the Interdisciplinary Center for Scientific Computing, and at the Faculty of Mathematics and Computer Science at Heidelberg University.

I would like to sincerely thank Moritz Diehl for inducting me into the fascinating world of Model Predictive Control and for being a true inspiration, Sebastian Sager for dedicated mentoring and a shared enthusiasm on all variants of chess, and Ekaterina Kostina for giving me much needed leeway in the finishing stage of this thesis.

I cannot thank enough my friend, former officemate, and advisor Andreas Potschka for many inspiring discussions, for his trust in me, for extensive and much appreciated comments and feedback on this thesis, and for his invaluable support and advice in all things scientific and non-scientific. I also want to express my wholehearted thanks to my friends Christian Hoffmann, Christian Kirches, and Tom Kraus for many fruitful and delighting discussions, for their steadfast support, and for many cheerful post-office hours – thank you, dear Gentlemen, it was a true pleasure!

I would like to sincerely thank my friend and former officemate Dörte Beigel and my current officemate Andreas Sommer for a great working atmosphere with many interesting discussions, for sharing work and ideas, for encouragement, and for great collaboration in joint projects.

It was a true pleasure to meet and interact with all the open-minded, friendly, and supportive co-workers in the research group, with plenty of delightful discussions over a cup of coffee or two. In particular I would like to thank Holger Diedam, Hans Joachim

Ferreau, Kathrin Hatz, Peter Kühl, and Simon Lenz.

I am indebted to Jan Albersmeyer, Ruben Garske, and Andreas Meyer for the implementation of SolvIND and Hans Joachim Ferreau, Andreas Potschka, and Christian Kirches for the implementation of qpOASES, two important parts of the software package MLI which I implemented for this thesis.

I am grateful to Felix Lenders who shared with me literature research, administrative information, and young fatherhood experiences.

In the course of working on this thesis, I had the luck to be one of the main contributors to several publications. I want to express my graditude to my co-authors (in alphabetical order): Jan Albersmeyer, Dörte Beigel, Hans Georg Bock, Moritz Diehl, Hans Joachim Ferreau, Janick Frasch, Christian Kirches, Peter Kühl, Sebastian Sager, and Johannes Schlöder.

For helping me through the jungle of administration, I would like to thank Anastasia Valter, Margret Rothfuss, and Dorothea Heukäufer, and Thomas Klöpfer for the valuable support with the IT infrastructure.

I am infinitely indepted and thankful to my parents Verena and Rolf-Dieter. They have sparked and nurtured my interest in mathematics, and have supported and encouraged me with love and help in word and deed throughout my life and in particular during my studies and the research on this thesis. I also want to thank my sister Cordula and my brother Daniel for their love and support. I am also thankful for continued support to my parents-in-law Margit and Klaus.

Last but not least I want to thank my wife Clarissa from the bottom of my heart for her love, her patience, her trust, and her support throughout this thesis. Everything I have committed to paper here is for her and my wonderful daughter Johanna. Full of joy I am looking forward to our next steps into the future!

Contents

1 Introduction

Dynamic processes are abundant in science, engineering, and industry. To give only some examples consider a car moving on the street, a chemical reaction, the metabolism of a cell, a space probe flying to the outer planets, the workflow of a production line, or the course and spread of a disease. In each case we deal with a system with a state that changes with time, possibly subject to inputs and constraints.

To study and understand the process and its time-dependent behavior, we can observe the system and take measurements. However, it is quite often the case that observing the system is difficult, expensive, dangerous, or heavily time-consuming. Also, the question of quantitative description and prediction, and analysis of the system behavior, e.g., under perturbations arises.

A highly successful approach to deal with these questions with an ever growing importance during the whole history of science and in particular since the appearance of computers is the mathematical modeling of dynamic processes [165]. There are plenty ways to model time-dependent systems, depending whether the modeled process is discrete or continuous, and deterministic or probabilistic. In this thesis we will consider dynamic process models formulated as ordinary differential equations.

The typical development cycle for a process model as considered in this thesis is the following: as a starting point we have a dynamic model, in general derived from first principles, and usually depending on parameter values which may be known or unknown. In order to make the dynamic model useful for simulation, we take measurements from the real process and perform *parameter estimation* and thus *model fitting*, cf. [12, 13, 28, 33, 175]. There are techniques that allow to choose the experiments particularly efficiently with regard to the estimation process. These techniques are known as *(optimum) experimental design*, cf. [34, 110, 152, 83, 66]. In case we have several candidate models and seek out the one which best describes the real process, a particular variant of optimum experimental design, namely *model discrimination* [35, 10, 8, 11, 96], allows to generate experiments which best reveal the less suited or false models. The results of this approach of designing and performing experiments and fitting the model to the data may well indicate that the model is still not suitable to describe the real process satisfactorily, and then the whole cycle has to be repeated, starting with improving the dynamic model. For further reading about modeling and modeling languages see, e.g., [134, 63, 82, 139, 48, 100, 62, 26, 61, 164].

Following the successful modeling phase, the dynamic model can then be used to simulate the real process. This helps or even enables to study and understand the behavior of expensive processes such as car crashes, processes with ultra-fast dynamics or dynamics which evolve over years, such as protein foldings or space probe traveling, or dangerous processes like exothermic chemical reactions. One can investigate the process behavior

under various initial and boundary conditions and for various parameter values. One can find critical points and study the process behavior under small perturbations. Modeling and simulation of processes is also known under the collective term *scientific computing* and its huge importance is reflected in its naming as the "third pillar of science" [46], besides theory and experimentation.

However, the interest of scientists and engineers in process models does not end with the ability to simulate the process behavior. In general, the behavior of processes is influenced by quantities which can be manipulated purposefully to obtain a desired process behavior. For example, consider using the steering wheel, the brake, and the gas pedal, and choosing gears while driving a car, or controlling inflow of educts, reactor temperature, and product outflow in a chemical reaction. These quantities can be fixed one-time choices done in advance before the process starts, or time-dependent varying choices applied during runtime. We call the representations of these quantities in the process model *parameters* in the former and *controls* or *control functions* in the latter case. The mathematical representation of states, controls, and parameters allows us to numerically quantify how good a certain choice of controls and parameters is in regard to a specific desired process goal by formulating a suitable functional, e.g., the end time of a process, the energy consumed by a process, the product quality, distance to a specified point or trajectory, and so on. We call this functional the *objective function*. Furthermore, we may want or need to impose *constraints* on the states and controls of the process model for various reasons, e.g., to avoid unphysical states or states that lie outside the area of validity of the model, to model limited control resources, or to keep the states away from safety-critical domains.

The three parts described above – dynamic model, objective function, and constraints – then come together in the discipline of *optimal control*, where we seek control functions and parameters in such a way that the objective function is minimized while the constraints are satisfied. This optimization problem is particularly difficult since the optimization variables, which are the states and controls, are functions and thus infinite-dimensional. It should be noted that optimal control actually refers to two approaches: the first one as described seeking optimal controls as functions over time, depending on the initially chosen configuration of the optimal control problem (initial states, fixed parameters, process start and end time) but independent of intermediate states during run time, and the second one calculating controls or more precisely evaluating control policies for the actual current state of the system, also known as *(optimal) feedback control*.

The big advantage of the second approach, in particular with regard to the applicability of the computed controls to the real process, is the fact that the controller can react to disturbances. In general, even the best models do not perfectly reproduce the state evolution of the real process. Furthermore, there can be, e.g., disturbances due to changes or errors in the process environment, imperfect control realizations, or if the process has a stochastic component. In these cases, the application of classical or *offline* optimal control would lead to suboptimal behavior or even fail to meet the constraints because the computed optimal controls do not take the disturbances into account.

The theory of optimal control has been intensely studied since the 1950s and can nowadays be considered mature, cf. textbooks [38, 169, 171, 130, 20, 21]. In particular, two results have to be highlighted, *Pontryagin's maximum principle* [144] as a necessary condition for the solution of optimal control problems, and the *Hamilton-Jacobi-Bellman equation* [15, 14] as a sufficient condition. The former is the fundamental theorem for the computation of classical offline optimal controls, the latter is the fundamental theorem for the optimal feedback control approach.

In this thesis we focus on the development and investigation of new efficient computational methods for calculating feedback control. The Hamilton-Jacobi-Bellman equation and its discrete counterpart, the Bellman equation, have given rise to elegant early results like the linear-quadratic regulator and the principle of dynamic programming. However, only for few special cases a closed solution is available, and numerical solution approaches, cf. [16], heavily suffer from the curse of dimensionality, thus using the Hamilton-Jacobi-Bellman equation or dynamic programming for modern nonlinear, constrained, medium-to-large-scaled applications is essentially prohibitive. Other approaches like the neighboring extremal control, which is essentially a linear-quadratic regulator set up in a linearization of a reference solution obtained by offline optimal control, are computationally tractable and still subject to research [99, 190, 189, 136, 84], but by construction this approach is valid only in a neighborhood of a specific offline solution.

In this thesis, we focus on the state-of-the-art approach of *(nonlinear) model predictive control* ((N)MPC), which is a hybrid approach to calculate feedback control by repeatedly solving offline optimal control problems on a finite horizon for the current system state and feeding back the first part of the computed controls to the process. Information about the real process enter in model predictive control via the current state, which may or may not be completely available from measurements. To make Model Predictive Control a practically usable approach, it thus must be complemented by techniques to obtain an estimate of the current state, i.e. *online state estimation*. We use in particular the *Moving Horizon Estimation* approach, which is closely related to Model Predictive Control. Model Predictive Control dates back to the 1980s [49, 44, 45] and theory can nowadays be considered mature, cf. textbooks [91, 156]. Furthermore, linear model predictive control has seen more and more applications in industry, cf. [153, 154, 155]. However, NMPC is still subject to intense study [79, 135, 138], in particular with regard to efficient numerical schemes for feedback control computation.

The key to efficient numerics for Model Predictive Control is a combination of efficient treatment of the arising optimal control problems and exploitation of the specific structure as a sequence of closely related parametric optimization problems. Early approaches to numerical optimal control, the so-called *indirect methods*, applied Pontryagin's maximum principle to obtain an expression in states and adjoints for the optimal control and then solving the boundary value problem for states and adjoints which results from the first-order necessary conditions for optimality. This approach is elegant and produces high-accuracy solutions [31, 142, 39], however it has severe drawbacks in practice, since it requires knowledge in advance of the switching structure of the solution [170], an explicit

expression in states and adjoints for the optimal control which has to be derived individually for each problem and is often hard to obtain, and the resulting boundary value problems are in general difficult to solve due to dynamic instability and lack of a-priori information about the adjoint trajectories.

The modern quasi-standard approaches belong to the *direct methods* which discretize the control and state space and thus transform the optimal control problem to possibly large-scale, structured *nonlinear programs* (NLP). The approach pursued in this thesis is the *Direct Multiple Shooting* method [32, 28], which introduces a partition of the time horizon, discretizes the control space, e.g., by piecewise constant controls, solves the dynamic system on each subinterval of the partition, and ensures continuity by adding matching constraints. For the resulting nonlinear problem we then use *sequential quadratic programming* (SQP) [183, 93, 94, 150, 151, 149] to generate the primal-dual iterates. The main advantages of using Direct Multiple Shooting are the possibility to use state-of-the-art integrators for adaptivity and derivative evaluation while keeping the problem size fixed, and the possibility to exploit the block structure in the subproblems to reduce the problem size to the size of single shooting problems while keeping the excellent local convergence properties of Multiple Shooting problems. An alternative direct approach is *direct collocation* [157, 23, 95, 40, 41, 25], which discretizes states and controls by lower-order piecewise polynomials and solve the arising large-scale structured nonlinear program with sequential quadratic programming or an interior point solver. Algorithmic approaches and theory for the solution of nonlinear programs can be found in detail in textbooks such as [87, 19, 140, 24].

Efficient numerics for NMPC do, however, comprise more than efficient methods for offline optimal control. In this thesis, we follow an approach that is central to most modern numerical schemes for Model Predictive Control. The idea is to set up in advance a *tangential predictor* in an approximation of the solution, e.g., the (approximated) solution of the last subproblem or the predicted solution of the current problem, and make an update by evaluating the tangential predictor as soon as the actual current state is available. Key factor to this approach is that the expensive calculations to set up the tangential predictor can be done without knowing the current state and thus the computations can be performed uncoupled to the sampling of the current state, and that the evaluation of the tangential predictor is much cheaper and thus leads to minimal feedback delays. In particular, we base our work on the *Real-Time Iteration* approach [51, 55, 58, 54, 57, 115, 174] which sets up the tangential predictor as a quadratic subproblem using the *initial value embedding* and thus is able to handle even active set changes within the update, and reduces the computational work to one SQP iteration per sample. A popular alternative approach is, e.g., the *advanced-step controller* and derivatives [192, 191] which applies the tangential predictor approach to the collocation/interior-point framework. For detailed reviews on various numerical approaches for real-time optimization using tangential predictors, including the Real-Time Iterations and the advanced-step controller, see [56] and more recently [188].

The goal of this thesis is to go beyond and further improve on the Real-Time Itera-

tion scheme by developing, implementing, and testing the *Multi-Level Iteration* (MLI) approach. Preliminary work and work published in the process of making this thesis comprise [30, 185, 107, 85] for algorithmic descriptions and [186, 187, 106, 3, 129] for applications of the approach. At the core of the presented approach is a feedback controller that is given by a generalized tangential predictor. This generalized tangential predictor consists of a quadratic program (QP), which implicitly defines a piecewise affine linear feedback law. The feedback law is evaluated by solving the QP for the current state estimate. The key idea is that since the function and derivative evaluation is the most expensive part per iteration, the data of the feedback controller, i.e., residual vectors, gradient, and first and second order derivative matrices, are updated on four different hierarchically ordered levels. The approach is also motivated by the observation that system dynamics and system linearizations have quite different time scales of validity, with an extreme example given by the linear oscillator with quickly changing trajectories and constant derivatives. The levels work as follows: the lowest level performs no updates and just calculates the piecewise affine linear feedback law for a new initial state estimate. The second level updates the QP constraint functions and approximates the QP gradient. The third level updates the QP constraint functions and calculates the exact QP gradient. The fourth level evaluates all matrix and vector data of the QP. Feedback schemes are then assembled by choosing a level for each sample. This yields a successive update of the piecewise affine linear feedback law that is implicitly defined by the generalized tangential predictor.

We present and discuss four strategies for data communication between the levels in a scheme and we describe how schemes with fixed level choices can be assembled in practice. We give local convergence theory for each level type holding its own set of primal-dual variables for fixed initial values [184, 29, 60], and discuss existing convergence theory for the case of a closed-loop process [58, 54, 57, 174]. We outline a modification of the levels that yields additional computational savings [85]. For the adaptive choice of the levels at runtime, we develop two contraction-based criteria to decide whether the currently used linearization remains valid. The criteria build on work published in [147]. We use them in an algorithm to decide which level to employ for the next sample. Furthermore, we propose a criterion applicable to online estimation. The criterion provides additional information for the level decision for the next sample. Focusing on the second lowest level, we propose an efficient algorithm for suboptimal NMPC. For the presented algorithmic approaches, we describe structure exploitation in the form of tailored condensing, outline the Online Active Set Strategy as an efficient way to solve the quadratic subproblems and extend the method to linear least-squares problems. We develop iterative matrix-free methods for one contraction-based criterion, which estimates the spectral radius of the iteration matrix.

We describe three application fields, namely NMPC on long horizons [106], robust NMPC [117, 116], and dual NMPC [121, 120], where MLI provides significant computational savings compared to state-of-the-art numerical methods for NMPC. For both fixed and adaptive MLI schemes, we carry out extensive numerical testings for challenging non-

linear test problems and compare the performance of MLI to a state-of-the-art numerical method for NMPC. The schemes obtained by adaptive MLI are computationally much cheaper while showing comparable performance. By construction, the adaptive MLI allows giving feedback with a much higher frequency, which significantly improves controller performance for the considered test problems. To perform the numerical experiments, we have implemented the proposed method within a MATLAB® based software called MLI, which makes use of a software package for the automatic derivative generation of first and higher order for the solution of the dynamic model as well as objective and constraint functions [4, 2], which performs structure exploitation by condensing, and which efficiently solves the parametric quadratic subproblems by using a software package that provides an implementation of the Online Active Set Strategy [74, 75, 148].

1.1 Contributions

In the following we give a listed summary of the contributions of this thesis.

- A comprehensive presentation of the numerical framework for the solution of optimization problems constrained by differential equations by Direct Multiple Shooting, including a first-time investigation of details for Direct Multiple Shooting for DAE-constrained problems concerning issues of uniqueness and smoothness of the algebraic states.

- A detailed introduction to the idea of tangential predictors and initial value embedding, including a short synopsis of the real-time iteration scheme for Nonlinear Model Predictive Control and moving horizon estimation.

- Development and investigation of the Multi-Level Iteration schemes, an efficient approach for high-frequency computation of feedback with four computational models for the data update in the implicit feedback update law.

- For one computational model development and investigation of two contraction-based adaptive methods for the decision on matrix data update in the implicit feedback update law.

- Using the contraction-based decision criteria for matrix data update, proposition and discussion of an adaptive online real-time feasible level choice algorithm for the Multi-Level Iteration schemes.

- Development of a criterion for potential model deficiency computed in online estimation and discussion how this information can be used in the adaptive level choice algorithm.

- Proposition of an adaptive algorithm for suboptimal NMPC making use of cheap feasibility-improving iterations.

- Numerical efficient specialization of the online active set strategy for the solution of parametric quadratic programs to parametric linear least-squares problems.
- A new numerical scheme for Dual NMPC with covariance propagation by differential or difference equations.
- Implementation of the presented approach in the software package `MLI` and extensive numerical testing and comparison of fixed and adaptive multi level iteration schemes on several challenging test applications from chemical engineering, mechanical engineering, and population dynamics.

1.2 Thesis overview

This thesis is structured in three parts: theoretical introduction, development of the new method, and applications and numerical results.

In Chapter 1 we introduce and formalize the notion of optimization problems constrained by differential equations. We then review the Direct Multiple Shooting approach, which discretizes the infinite-dimensional optimization problem to obtain a finite-dimensional nonlinear program. We give a brief overview on the theory of nonlinear programming and the sequential quadratic programming approach for the numerical solution of nonlinear programs. Finally, we discuss issues of structure exploitation and derivative calculation which are characteristic for nonlinear programs arising from a Direct Multiple Shooting discretization.

In Chapter 2 we motivate the necessity of feedback control and outline classical feedback approaches which are widely used in modern industrial plants but have severe drawbacks from a theoretical point of view. As state-of-the-art approach, we present Nonlinear Model Predictive Control (NMPC) as a numerically tractable and theoretically founded way to compute feedback which is (nearly) optimal. We briefly touch stability theory for NMPC. As state-of-the-art approach for online state and parameter estimation we present Moving Horizon Estimation (MHE), which is in a way the dual problem to NMPC. This concludes the first part.

We begin the second part in Chapter 3 with a short introduction to a state-of-the-art numerical method for NMPC which is the starting point for our new approach. First, we present the ideas of tangential predictors and initial value embedding. From these ideas we motivate and present the Real-Time Iteration scheme (RTI), which is an efficient algorithmic approach to real-time optimization and estimation. We state numerical details and stability results.

In Chapter 4 we present the Multi-Level Iteration schemes (MLI) as a new and efficient numerical approach for NMPC. We give a short motivation, then describe the different levels. We discuss how to assemble schemes with fixed level choice and how to exchange data between the levels. We give local convergence theory for a special case and reference existing work that applies to a subset of Multi-Level Iteration schemes for the general case. We close the chapter with a note on mixed-level and fractional-level iterations.

In Chapter 5 we deal with MLI with adaptive level choice. We start with preliminary considerations, then we present two approaches for the estimation of the contraction rate, and formulate and discuss an adaptive level choice algorithm which make use of these estimates. For the case that an MHE estimator is used, we present a criterion that may also be used in the decision algorithm for the adaptive level choice. Finally, we outline a feedback approach which, while not fitting exactly in the MLI approach of this thesis, makes use of the contraction rate estimates and is motivated by the stability results for suboptimal NMPC.

In Chapter 6 we consider algorithmic details for several parts of the MLI iterations. First, we consider efficient numerics for solving a sequence of parametric QPs. We then give an extension of the presented Online Active Set Strategy for the solution of parametric linear least-squares problems. Next, we consider a tailored condensing for mixed-level and fractional-level MLI. Finally, we consider efficient iterative methods for the estimation of the spectral radius of the MLI iteration matrix. This concludes the second part.

In Chapter 7 we consider several methodological fields of application for the MLI approach. First, we briefly discuss and point to numerical results for MLI for NMPC on long horizons. Then, we outline an approach for the calculation of robust feedback, and introduce a formulation for a dual NMPC problem to avoid the conservatism of the robust worst-case approach.

In Chapter 8 we present the application of MLI with prescribed level choice to two numerical test cases, the continuous stirred-tank reactor, and a single-track car model with nonlinear tire model. In both cases, NMPC is used to reject a disturbance and track a desired state. We compare the performance of MLI with the established approach of RTI, investigate MLI for various data communication strategies, and show the advantage of higher sampling rates for the considered test cases.

In Chapter 9 we apply MLI with adaptive level choice to two test cases: a chain of masses connected by springs, which is to be brought to rest, and the single-track car problem from Chapter 8. For the chain application, we discuss numerical results for both the postiteration approach and the spectral radius estimation approach. For the single-track car problem, we describe the control scenario and give numerical results for the postiteration approach.

In Chapter 10 we present numerical results for using the MHE criterion described in Chapter 5 as an additional tool for MLI with adaptive level choice by postiterations, and for the application of MLI to solve the problems arising in the Dual NMPC approach. As test case, we use in both cases the Lotka-Volterra model for the dynamics of a predator-prey system. This concludes the third part and this thesis. In Chapter 11 we give conclusions and an outlook to potential future research directions.

Part I

Principles of Model Predictive Control and Moving Horizon Estimation

2 Direct Multiple Shooting for optimization problems constrained by differential equations

In this chapter we introduce and formalize the notion of optimization problems constrained by differential equations. We then review the Direct Multiple Shooting approach, which discretizes the infinite-dimensional optimization problem to obtain a finite-dimensional nonlinear program. We give a brief overview on the theory of nonlinear programming and the sequential quadratic programming approach for the numerical solution of nonlinear programs. Finally, we discuss issues of structure exploitation and derivative calculation which are characteristic for nonlinear programs arising from a Direct Multiple Shooting discretization.

2.1 Optimization problems constrained by differential equations

At a fairly abstract level we can consider an optimization problem as a combination of three basic ingredients: first, a set from which to choose the *decision* or *optimization variables* of the optimization problem, second, an *objective function* that maps each choice of the variables into an ordered set and thus allows to compare different choices of variables, and third, a set of *constraint functions* which describe the subset of the variable set from which we allow to choose, the so-called *feasible set*. It should be noted that the decision variables can be from both finite-dimensional and infinite-dimensional spaces.

Optimization problems constrained by differential equations are characterized by the property that the optimization variables can be split into independent variables, which usually comprise time dependent functions $u : \mathcal{T} \mapsto \mathbb{R}^{n_u}$ called *control functions* and finite-dimensional values $p \in \mathbb{R}^{n_p}$ called *parameters*, and dependent variables, where the dependency is described by a system of ordinary differential equations (ODE) over a time interval (or time horizon) $\mathcal{T} = [t_\mathrm{s}, t_\mathrm{f}]$

$$\dot{x}(t) = f(t, x(t), u(t), p), \tag{2.1}$$

with $f(\cdot)$ the *model equations* or *(differential) right-hand side*. As solution of the differential equations, the dependent variables are functions over time and we call them *(differential) states* $x : \mathcal{T} \mapsto \mathbb{R}^{n_x}$.

If the dependent variables are described by a combination of differential and algebraic

equations (DAE), e.g., by the semi-explicit DAE

$$\dot{x}(t) = f(t, x(t), z(t), u(t), p) \tag{2.2a}$$
$$0 = g(t, x(t), z(t), u(t), p), \tag{2.2b}$$

with $g(\cdot)$ the *algebraic right-hand side*, we distinguish between differential states $x : \mathcal{T} \mapsto \mathbb{R}^{n_x}$ and *algebraic states* $z : \mathcal{T} \mapsto \mathbb{R}^{n_z}$. Theory concerning existence, uniqueness and smoothness of the state variables depending on properties of the right-hand sides and input variables can be found, e.g., in [81, 102] for ODEs and in [89, 119] for DAEs.

Differential equations of the form given in (2.1) are called *first-order* differential equations because the right-hand side describes the behavior of the first time derivative of the solution $x(t)$. Higher-order differential equations can be reformulated as first-order differential equations as pointed out in the following remark and thus can also be treated by the presented approach.

Remark 2.1 (Reformulation of higher-order differential equations)**.** *Dynamic process models formulated as higher-order differential equations, i.e., using the common shorthand notation of* $x^{(k)}(t) \triangleq \frac{d^k x}{dt^k}(t)$,

$$x^{(n)}(t) = f(t, x(t), \ldots, x^{(n-1)}(t), u(t), p), \quad t \in \mathcal{T} = [t_\mathrm{s}, t_\mathrm{f}], \tag{2.3}$$

can be reformulated in the form of (2.1) *by introducing the augmented state*

$$y(t) \triangleq \begin{pmatrix} x_1(t) \\ \vdots \\ x_n(t) \end{pmatrix} \triangleq \begin{pmatrix} x(t) \\ \vdots \\ x^{(n-1)}(t) \end{pmatrix} \tag{2.4}$$

and the augmented right hand side

$$\tilde{f}(t, y(t), u(t), p) \triangleq \begin{pmatrix} x_2(t) \\ \vdots \\ x_n(t) \\ f(t, x_1(t), \ldots, x_n(t), u(t), p) \end{pmatrix}. \tag{2.5}$$

As an example, consider the simple pendulum with friction ($\ddot{x}(t) \triangleq \frac{d^2 x}{dt^2}(t)$*)*

$$\ddot{x}(t) = -b\,\dot{x}(t) - k\,x(t), \quad \textit{with constants } b, k > 0. \tag{2.6}$$

Then, with the augmented state $y(t)$

$$y(t) \triangleq \begin{pmatrix} x_1(t) \\ x_2(t) \end{pmatrix} \triangleq \begin{pmatrix} x(t) \\ \dot{x}(t) \end{pmatrix}$$

we obtain from (2.6) *the first-order model*

$$\dot{y}(t) = \begin{pmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \end{pmatrix} = \begin{pmatrix} x_2(t) \\ -b\,x_2(t) - k\,x_1(t) \end{pmatrix}.$$

In the following we give example classes of optimization problems constrained by differential equations and introduce the formal notation.

2.1.1 Optimal control problems

In the *optimal control problem* (OCP)

$$\begin{aligned}
\underset{x(\cdot),z(\cdot),u(\cdot),p,t_\mathrm{f}}{\text{minimize}} \quad & \int_{t_\mathrm{s}}^{t_\mathrm{f}} l(t,x(t),z(t),u(t),p)\ \mathrm{d}t + m(x(t_\mathrm{f}),z(t_\mathrm{f}),p,t_\mathrm{f}) \\
\text{subject to} \quad & \dot{x}(t) = f(t,x(t),z(t),u(t),p), \\
& 0 = g(t,x(t),z(t),u(t),p), \\
& 0 \leq c^{\mathrm{path}}(t,x(t),z(t),u(t),p), \\
& 0 = r_\mathrm{e}(x(t_\mathrm{s}),z(t_\mathrm{s}),x(t_\mathrm{f}),z(t_\mathrm{f})), \\
& 0 \leq r_\mathrm{i}(x(t_\mathrm{s}),z(t_\mathrm{s}),x(t_\mathrm{f}),z(t_\mathrm{f})),
\end{aligned} \tag{2.7}$$

the optimization variables are the differential and algebraic states $x(t)$ and $z(t)$, the control profiles $u(t)$, the parameter values p, and possibly the final time t_f. The latter can be regarded as an additional parameter, see also the brief discussion on transformations of the problem formulation given below.

The objective function consists of up to two parts, one part that integrates a cost function $l(\cdot)$ depending on time, states, controls, and parameters over the whole optimization horizon and another part that assigns a cost $m(\cdot)$ particularly in the end point t_f, depending on the end states, the parameters, and the end time. The former part is referred to as *Lagrange term* and is used to value accumulated time-variable costs, e.g., deviation of the trajectory from a given set-point or overall fuel/energy/mass consumption, while the latter part is called *Mayer term* and is used to value costs arising only in the end point, e.g., end time or final position/velocity. The combination of the two types of objectives is called a *Bolza* objective. However, these two different objective types exist rather for convenience in the problem formulation since they can easily be transformed into each other, see also the brief discussion on transformations of the problem formulation given below.

The *path constraint* function $c^{\mathrm{path}}(\cdot)$ describes a functional dependence of states, controls, and parameters that has to satisfy the path constraint $0 \leq c^{\mathrm{path}}(\cdot)$ for each time point of the time interval $\mathcal{T}$, which amounts to infinitely many constraints. The path constraints can be as simple as fixed bounds on states and controls but also be more complex and possibly nonlinear functions such as a safety function for an exothermic chemical reaction.

The *boundary conditions* $r_\mathrm{e}(\cdot)$ allow to fix some or all components of the states to given values either at the start point t_s or at the end point t_f. If the states are completely fixed only at t_s, we say that the differential equations have to satisfy an *initial value problem* (IVP).

The *point constraints* $r_\mathrm{i}(\cdot)$ allow to restrict the initial and end states to a certain feasible region. This can be used, e.g., to restrict final positions or velocities to a certain range. Furthermore, end time constraints play an important role in stability theory for optimizing feedback control, see Chapter 3.

Note that for an efficient structure exploitation in the numerical framework that is

used in this work we have to demand that both the boundary conditions and the point constraints are either decoupled or at most linearly coupled. This ensures *partial separability* of the discretized problem. In practice, most often the formulated boundary conditions and point constraints already satisfy this demand, e.g., decoupled conditions for $(x(t_\mathrm{s}), z(t_\mathrm{s}))$ and $(x(t_\mathrm{f}), z(t_\mathrm{f}))$, or linear coupling in periodicity constraints. For the general case of nonlinear couplings there exist reformulations to transform the problem to an equivalent problem with linearly coupled boundary conditions and point constraints. In the following we always assume that the boundary conditions and point constraints are at most linearly coupled.

2.1.2 Estimation problems

Another class are the *parameter estimation problems* (PEP)

$$\begin{aligned} \underset{x(\cdot),z(\cdot),p}{\text{minimize}} \quad & \sum_{j=0}^{M} \left\| \frac{h(t_j, x(t_j), z(t_j), p) - \eta_j}{\sigma_j} \right\|_2^2 \\ \text{subject to} \quad & \dot{x}(t) = f(t, x(t), z(t), \bar{u}(t), p), \\ & 0 = g(t, x(t), z(t), \bar{u}(t), p), \\ & 0 = r_\mathrm{e}(x(t_\mathrm{s}), z(t_\mathrm{s}), x(t_\mathrm{f}), z(t_\mathrm{f})), \end{aligned} \tag{2.8}$$

where we have measurement data η_j that is subject to statistical measurement errors. We try to reconstruct differential and algebraic states $x(t)$ and $z(t)$ and parameter values p by minimizing the normalized squared deviation of the measurement data to a *model function* $h(\cdot)$ of the measurement process. Doing this we assume that there exist "true" parameters $p^\star$ and states $x^\star, z^\star$ so that $h(t_j, x^\star(t_j), z^\star(t_j), p^\star) - \eta_j$ is distributed with mean value zero and standard deviation σ_j. For this problem, the controls $\bar{u}$ are assumed to be known and are not subject to optimization. The given formulation performs *maximum-likelihood estimation* and is an example of *least-squares* (LS) optimization. For background on estimation theory and practice see. e.g., [12, 13, 175]. Some details specific for the problems we consider will also be given in Chapter 3.

2.1.3 Problem reformulations

We consider several problem transformations which allow us to restrict our work to problems with significantly simplified structure compared with OCP (2.7) or PEP (2.8) without loss of generality (w.l.o.g.). The presented transformations as well as further transformations can be found, e.g., in [86].

Transformation to fixed end time

To transform a problem with free end time t_f to a problem with fixed end time we extend the parameters p by an additional parameter $p_t \triangleq (t_\mathrm{f} - t_\mathrm{s})$ and introduce a new time

variable τ and transformed states and controls by

$$\begin{aligned} t(\tau) &\triangleq t_\mathrm{s} + \tau p_t, \quad \tau \in [0,1], \\ \bar{x}(\tau) = x(t(\tau)) &\triangleq x(t_\mathrm{s} + \tau p_t), \\ \bar{z}(\tau) = z(t(\tau)) &\triangleq z(t_\mathrm{s} + \tau p_t), \\ \bar{u}(\tau) = u(t(\tau)) &\triangleq u(t_\mathrm{s} + \tau p_t). \end{aligned}$$

Then we obtain

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}\tau}\bar{x}(\tau) &= \dot{x}(t(\tau)) \cdot \frac{\mathrm{d}t(\tau)}{\mathrm{d}\tau} \\ &= p_t \cdot f(t(\tau), x(t(\tau)), z(t(\tau)), u(t(\tau)), p), \\ &\triangleq \bar{f}(t(\tau), \bar{x}(\tau), \bar{z}(\tau), \bar{u}(\tau), \bar{p}), \end{aligned}$$

with $\bar{p} \triangleq (p, p_t)$. This allows us to assume w.l.o.g. that the time horizon $\mathcal{T}$ is fixed.

Transformation to autonomous problem

If the functions $l(\cdot), m(\cdot), f(\cdot), g(\cdot)$, and $h(\cdot)$ in OCP (2.7) do not depend explicitly on t we call the problem *autonomous*, otherwise *non-autonomous*. To eliminate the explicit dependence on t, we introduce an additional differential state x_t with

$$\dot{x}_t = 1, \quad x_t(t_\mathrm{s}) = t_\mathrm{s}. \tag{2.9}$$

Then we can replace t by $x_t(t)$ and t_f by $x_t(t_\mathrm{f})$ everywhere in OCP (2.7). Of course we can analogously transform PEP (2.8). Thus, w.l.o.g. we can restrict ourselves to autonomous problems.

Transformation to pure Mayer objective

The Lagrange objective can be transformed into a Mayer objective by adding a differential state x_l with

$$\dot{x}_l = l(t, x(t), z(t), u(t), p), \quad x_l(t_\mathrm{s}) = 0. \tag{2.10}$$

Then we obtain the new Mayer objective

$$\bar{m}(\bar{x}(t_\mathrm{f}), z(t_\mathrm{f}), p, t_\mathrm{f}) \triangleq m(x(t_\mathrm{f}), z(t_\mathrm{f}), p, t_\mathrm{f}) + x_l(t_\mathrm{f})$$

with $\bar{x}(t_\mathrm{f}) \triangleq (x(t_\mathrm{f}), x_l(t_\mathrm{f}))$. Thus, w.l.o.g. we can restrict ourselves to problems with pure Mayer objective.

Elimination of parameters

We can eliminate the parameters p from OCP (2.7) or PEP (2.8) by introducing new differential states x_p with

$$\dot{x}_p = 0, \quad x_p(t_\mathrm{s}) = p. \tag{2.11}$$

Then we replace each occurrence of p by x_p. Parameters that are to be determined by optimization are thus transformed into free initial conditions. This transformation allows us to restrict ourselves w.l.o.g. to problems without parameters.

Altogether, the presented transformations allow us to restrict our investigations w.l.o.g. to the OCP problem formulation

$$\begin{aligned} \underset{x(\cdot),z(\cdot),u(\cdot)}{\text{minimize}} \quad & \int_0^{\mathsf{T}} l(x(t),z(t),u(t))\ \mathrm{d}t + m(x(T),z(T)) \\ \text{subject to} \quad & \dot{x}(t) = f(x(t),z(t),u(t)), \\ & 0 = g(x(t),z(t),u(t)), \\ & 0 \le c^{\text{path}}(x(t),z(t),u(t)), \\ & 0 = r_{\text{e}}(x(0),z(0),x(T),z(T)), \\ & 0 \le r_{\text{i}}(x(0),z(0),x(T),z(T)), \end{aligned} \tag{2.12}$$

with a fixed horizon length T. Occasionally, we keep the parameters p in the problem formulation, but we assume them to have known constant values and not enter the optimization problem as degrees of freedom. We also frequently make use of the transformation to a pure Mayer objective formulation. For the estimation problem we keep the parameters as degrees of freedom where suitable, and we also assume the problem to be autonomous and formulated on a fixed horizon $[0,T]$.

Example 1 (Continuous stirred-tank reactor (CSTR))**.**

To illustrate the introduced terms and definitions let us consider the following example of an optimal control problem for a chemical process, the so-called *van der Vusse* reaction. The process model has been introduced by Klatt and Engell [64] and formulated as a benchmark example for Nonlinear Model Predictive Control by Chen and coworkers [43, 42]. In this reaction, a substance A reacts to a substance B and two unwanted by-products C and D in the following way

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

The reaction takes place in a well-stirred reactor which allows to neglect spatial concentration and temperature variations. The reactor has a feed inflow which contains only substance A at a concentration $c_{\text{A},0}$ and a temperature θ_0. The feed flow rate $\dot{V}$ can be controlled. To keep the liquid tank volume constant, the outflow, which contains the substances A, B, C, and D with concentrations $c_{\text{A}}, c_{\text{B}}, c_{\text{C}}$, and c_{D}, is kept at the same rate as the inflow.

A major source of nonlinearity of the process comes from the reaction rates k_i, which depend on the reactor temperature θ by the Arrhenius law

$$k_i(\theta) = k_{i,0} \exp\left(\frac{E_i}{273.15+\theta}\right), \quad i = 1,2,3.$$

Because the reaction is exothermic, cooling is a crucial issue. It is done by a cooling jacket with controlled heat removal rate $\dot{Q}_{\mathrm{K}}$. Thus, the reactor temperature is held down indirectly by controlling the cooling jacket temperature θ_{K}. A schematic display of the process is given in Figure 2.1.

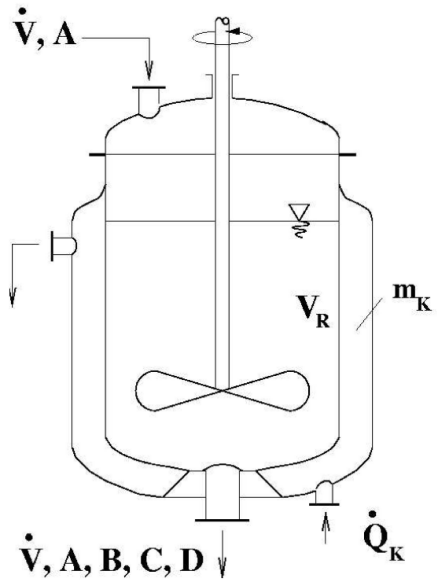


Figure 2.1: Schematic diagram of the CSTR (from [43])

The model equations for the concentrations c_{A} and c_{B} as well as the temperatures θ and θ_{K} are given below. Note that the model does not contain the concentrations c_{C} and c_{D} of the unwanted substances C and D. However, the influence of the respective reactions on the concentrations c_{A} and c_{B} and the temperature θ is of course included in the model. The parameter values in the model equations are given in Table 2.1.

$$\dot{c}_{\mathrm{A}} = \frac{\dot{V}}{V_{\mathrm{R}}}\left(c_{\mathrm{A},0} - c_{\mathrm{A}}\right) - k_1(\theta)c_{\mathrm{A}} - k_3(\theta)c_{\mathrm{A}}^2 \tag{2.13a}$$

$$\dot{c}_{\mathrm{B}} = -\frac{\dot{V}}{V_{\mathrm{R}}}c_{\mathrm{B}} + k_1(\theta)c_{\mathrm{A}} - k_2(\theta)c_{\mathrm{B}} \tag{2.13b}$$

$$\dot{\theta} = \frac{\dot{V}}{V_{\mathrm{R}}}\left(\theta_0 - \theta\right) - \frac{1}{\rho C_{\mathrm{p}}}\left(k_1(\theta)H_1c_{\mathrm{A}} + k_2(\theta)H_2c_{\mathrm{B}} + k_3(\theta)H_3c_{\mathrm{A}}^2\right) + \frac{k_{\mathrm{w}}A_{\mathrm{R}}}{\rho C_{\mathrm{p}}V_{\mathrm{R}}}\left(\theta_{\mathrm{K}} - \theta\right) \tag{2.13c}$$

$$\dot{\theta}_{\mathrm{K}} = \frac{1}{m_{\mathrm{K}}C_{\mathrm{PK}}}\left(\dot{Q}_{\mathrm{K}} + k_{\mathrm{w}}A_{\mathrm{R}}\left(\theta - \theta_{\mathrm{K}}\right)\right) \tag{2.13d}$$

In relation to the notions introduced earlier in this section, the dependent variables in this example are the differential states $x = (c_{\mathrm{A}}, c_{\mathrm{B}}, \theta, \theta_{\mathrm{K}})$, and the independent variables are the controls $u = \left(\frac{\dot{V}}{V_{\mathrm{R}}}, \dot{Q}_{\mathrm{K}}\right)$. The differential equations (2.13) can then be summarized as $\dot{x} = f(x, u)$ as in OCP (2.12) and they describe the dependence of x from u.

Symbol	Value	Symbol	Value
$k_{1,0}$	$1.287 \cdot 10^{12}$ h^{-1}	ρ	0.9342 $\frac{\text{kg}}{\text{l}}$
$k_{2,0}$	$1.287 \cdot 10^{12}$ h^{-1}	C_p	3.01 $\frac{\text{kJ}}{\text{kg K}}$
$k_{3,0}$	$9.043 \cdot 10^{9}$ h^{-1}	k_w	4032 $\frac{\text{kJ}}{\text{h m}^2\text{ K}}$
E_1	-9758.3	A_R	0.215 m^2
E_2	-9758.3	V_R	10 l
E_3	-8560	m_k	5 kg
H_1	4.2 $\frac{\text{kJ}}{\text{mol}}$	C_PK	2.0 $\frac{\text{kJ}}{\text{kg K}}$
H_2	-11.0 $\frac{\text{kJ}}{\text{mol}}$	$c_{\text{A},0}$	5.1 $\frac{\text{mol}}{\text{l}}$
H_3	-41.85 $\frac{\text{kJ}}{\text{mol}}$	θ_0	104.9 °C

Table 2.1: Constant CSTR parameter values

The distinction between constants and parameters is not always unambiguous and may depend on the goal of the optimization. In the example, many of the parameters are of physical nature and thus are true constants. However, the parameters A_R, V_R, and m_k are chosen by design of the reactor and thus could in principle be included in the optimization. Optimization problems which consider such choices are referred to as *design optimization*. In the example, however, we consider the reactor as given and thus the parameters A_R, V_R, and m_k as constant.

The other parameters that are no physical constants are the feed concentration and temperature $c_{\text{A},0}$ and θ_0. They are chosen by the experimenter and may also change values intentionally or unintentionally during the process. In the example, we do not include these parameters as optimization variables, but we consider scenarios where their values change.

To obtain the differential states x uniquely from the controls u, we also need the boundary conditions $0 = r_\text{e}(\cdot)$. In the example, we consider initial conditions of the form

$$0 = x(0) - x_0$$

with given current state x_0.

The objective function reflects the overall goal of the optimization. In this example, we desire to steer the process quickly to a *steady state*, i.e., a state x_s corresponding to controls u_s with

$$f(x_\text{s}, u_\text{s}) = 0.$$

This means, if the process is in the state x_s and we apply the controls u_s, then the process will stay in x_s forever. Obviously, there may be infinitely many steady states. We choose

the following

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

which optimizes the ratio of the steady state concentration of the product B to the feed concentration of the educt A. A suitable objective function to steer the process to $(x_{\mathrm{s}}, u_{\mathrm{s}})$ is then given by the quadratic Lagrange term

$$l(x,u) = (x - x_{\mathrm{s}})^{\intercal}\, Q\, (x - x_{\mathrm{s}}) + (u - u_{\mathrm{s}})^{\intercal}\, R\, (u - u_{\mathrm{s}})$$

with weighting matrices Q and R, chosen as

$$Q = \mathrm{diag}\left(0.2\ \frac{\mathrm{l}^2}{\mathrm{mol}^2}, 1.0\ \frac{\mathrm{l}^2}{\mathrm{mol}^2}, 0.5\ ^{\circ}\mathrm{C}^{-2}, 0.2\ ^{\circ}\mathrm{C}^{-2}\right)$$

and

$$R = \mathrm{diag}\left(0.5\ \mathrm{h}^2, 5.0 \cdot 10^{-7} \frac{\mathrm{h}^2}{\mathrm{kJ}^2}\right).$$

As a typical example for the path constraints $0 \leq c^{\mathrm{path}}(\cdot)$ we require simple bounds on the controls, i.e.,

$$0 \leq \begin{pmatrix} u(t) - \underline{u} \\ \overline{u} - u(t) \end{pmatrix}, \quad t \in [0,T],$$

with

$$\underline{u} = \begin{pmatrix} 3.0\ \mathrm{h}^{-1} \\ -9000\ \frac{\mathrm{kJ}}{\mathrm{h}} \end{pmatrix} \quad \text{and} \quad \overline{u} = \begin{pmatrix} 35.0\ \mathrm{h}^{-1} \\ 0\ \frac{\mathrm{kJ}}{\mathrm{h}} \end{pmatrix}.$$

This completes the description of the optimal control problem in the formulation of OCP (2.12). We will come back to this problem in Chapter 9 to illustrate the algorithmic approaches developed in this thesis.

2.2 Numerical approaches for solving OCPs

Numerical approaches to solve OCPs can be divided mainly into direct and indirect methods. Most direct methods follow the "first discretize, then optimize" approach, which means that the infinite-dimensional problem is transformed to a finite-dimensional *nonlinear program* (NLP) by discretization and then the NLP is solved by iterative methods such as *sequential quadratic programming* (SQP) or *Newton's method*. The term "direct" refers to the fact that in these methods the control variables are kept in the formulation as decision variables. In contrast, the indirect methods use the optimality conditions of the infinite-dimensional problem to eliminate the control variables in terms of state variables and adjoint state variables and so come up with a boundary value problem which is then solved numerically.

We will give details of the *Direct Multiple Shooting* method, see also [32, 143, 126, 127] which is our method of choice for treating differentially constrained optimization problems and mention briefly different other direct and indirect methods.

2.2.1 Direct Multiple Shooting

We start by partitioning the horizon $[0, T]$ into N subintervals $[t_i, t_{i+1}]$, $0 \leq i < N$. Control functions are discretized as

$$u(t) = \varphi_i(t, q_i) \quad \text{for } t \in [t_i, t_{i+1}], \tag{2.14}$$

where $\varphi_i(\cdot)$ are functions with local support parametrized by the finite dimensional vector q_i. Typical examples for $\varphi_i(\cdot)$ are piecewise constant controls or piecewise linear controls.

We introduce additional variables s_i^x, s_i^z which serve as initial values for computing the state trajectories independently on the subintervals $[t_i, t_{i+1}]$:

$$\begin{aligned} \dot{x}_i(t) &= f(x_i(t), z_i(t), \varphi_i(t, q_i)) && \text{(2.15a)} \\ 0 &= g(x_i(t), z_i(t), \varphi_i(t, q_i)) - \alpha_i(t) g(s_i^x, s_i^z, \varphi_i(t_i, q_i)), && \text{(2.15b)} \end{aligned}$$

with $x_i(t_i) = s_i^x$ and $z_i(t_i) = s_i^z$. We use this relaxed formulation in (2.15b) to allow arbitrary initial values s_i^x, s_i^z, which do not have to be consistent with the algebraic equations. A common choice for the scalar damping factor $\alpha_i(t)$ is $\alpha_i(t) = \exp\left(-\bar{\alpha}\frac{t-t_i}{t_{i+1}-t_i}\right), \bar{\alpha} \geq 0$.

To ensure that we compute a feasible solution of the differential equation on the whole interval $[t_\mathrm{s}, t_\mathrm{f}]$ we have to add *matching conditions*

$$0 = s_{i+1}^x - x_i(t_{i+1}; s_i^x, s_i^z, q_i), \ 0 \leq i < N \tag{2.16}$$

to the optimization problem, where $x_i(t; s_i^x, s_i^z, q_i)$ is the solution of the IVP on $[t_i, t_{i+1}]$, depending on s_i^x, s_i^z, q_i. Furthermore, we have to add *consistency conditions*

$$0 = g(s_i^x, s_i^z, \varphi_i(t_i, q_i)), \ 0 \leq i \leq N, \tag{2.17}$$

with $q_N \triangleq q_{N-1}$, which guarantee that we solve the original differential equation in the solution of the optimization problem. This approach of solving the boundary value problem along with the optimization problem is referred to as *simultaneous* approach in contrast to *sequential* approaches such as Direct Single Shooting, see below.

The path constraints are discretized in the time points t_i, i.e., we require

$$0 \leq c^\mathrm{path}(s_i^x, s_i^z, \varphi_i(t_i, q_i)), \quad i = 0, \ldots, N \tag{2.18}$$

to hold, again with $q_N \triangleq q_{N-1}$. This may allow for the violation of the path constraints within the Multiple Shooting intervals but bounds on piecewise constant controls are strictly enforced. Furthermore, techniques for ensuring feasibility for each $t \in \mathcal{T}$ are available, cf. [125, 145, 146].

To sum up, from the Multiple Shooting discretization we obtain the NLP

$$\begin{aligned} \underset{s^x, s^z, q}{\text{minimize}} \quad & \sum_{i=0}^{N-1} L\left(s_i^x, s_i^z, q_i\right) + m\left(s_N^x, s_N^z\right) && && \text{(2.19a)} \\ \text{subject to} \quad & 0 = s_{i+1}^x - x_i(t_{i+1}; s_i^x, s_i^z, q_i), && 0 \leq i < N, && \text{(2.19b)} \\ & 0 = g(s_i^x, s_i^z, \varphi_i(t_i, q_i)), && 0 \leq i \leq N, && \text{(2.19c)} \\ & 0 \leq c^\mathrm{path}(s_i^x, s_i^z, \varphi_i(t_i, q_i)), && 0 \leq i \leq N, && \text{(2.19d)} \\ & 0 = r_\mathrm{e}(s_0^x, s_0^z, s_N^x, s_N^z), && && \text{(2.19e)} \\ & 0 \leq r_\mathrm{i}(s_0^x, s_0^z, s_N^x, s_N^z). && && \text{(2.19f)} \end{aligned}$$

with $s^x = (s_0^x, \ldots, s_N^x)$, $s^z = (s_0^z, \ldots, s_N^z)$, and $q = (q_0, \ldots, q_{N-1}), q_N \triangleq q_{N-1}$ and

$$L\left(s_i^x, s_i^z, q_i\right) \triangleq \int_{t_i}^{t_{i+1}} l(x_i(t; s_i^x, s_i^z, q_i), z_i(t; s_i^x, s_i^z, q_i), \varphi_i(t, q_i))\ \mathrm{d}t.$$

Note that in NLP (2.19) the computationally most expensive parts are the objective terms $L\left(s_i^x, s_i^z, q_i\right)$ and the matching condition terms $x_i(t; s_i^x, s_i^z, q_i)$ since they involve the solution of the dynamic equations.

2.2.2 A remark on DAE problems

Under the assumption of index-1 of DAE (2.2), i.e., invertibility of $\frac{\partial g}{\partial z}(\cdot)$, an efficient Direct Multiple Shooting approach for optimal control problems constrained by DAEs has been developed by Leineweber [125, 126, 127] and coworkers. It is based on the elimination of the algebraic states using the consistency conditions (2.17) and the efficient evaluation of the derivatives of the reduced problem by directional sensitivity calculations.

Some points have to be considered for a correct interpretation of the results when solving DAE-constrained problems by the Multiple Shooting approach. First, if the algebraic equations depend explicitly on the controls, the arising consistency conditions introduce a dependence of s_N^z from the point value $\varphi_N(t_N, q_N)$ which is somewhat arbitrary due to the arbitrary nature of the artificial control parameters q_N. This is particularly critical if the objective explicitly depends on s_N^z. The practical remedy is the above described convention of constraining q_N to the value of q_{N-1} but note that in theory the value of q_N should not influence the solution of the optimization problem at all since it represents control action beyond the interval $\mathcal{T}$. However, if the algebraic equations are index-1 and depend affine-linearly on z, i.e., if $g(x, z, u) = A(x, u)z + \tilde{g}(x, u)$, we can show the following useful result.

Proposition 2.1. *For the index-1 DAE affine-linear in $z(t)$*

$$\dot{x}(t) = f(x(t), z(t), u(t)) \tag{2.20a}$$
$$0 = A(x(t), u(t)) \cdot z(t) + g(x(t), u(t)) \tag{2.20b}$$

consider a Multiple Shooting discretization with grid $\{t_0, \ldots, t_N\}$ and a piecewise constant control discretization (2.14) *with $\varphi_i(t, q_i) \triangleq q_i$. Let*

$$s^{x\star} = (s_0^{x\star}, \ldots, s_N^{x\star}), \quad s^{z\star} = \left(s_0^{z\star}, \ldots, s_{N-1}^{z\star}\right), \quad q^\star = \left(q_0^\star, \ldots, q_{N-1}^\star\right)$$

be a Multiple Shooting solution satisfying the matching conditions (2.16) *and the consistency conditions* (2.17) *for $i = 0, \ldots, N-1$.*

Then, by choosing $q_N^\star \triangleq q_{N-1}^\star$ and $s_N^{z\star}$ such that the consistency conditions (2.17) *are also satisfied for $i = N$ one obtains*

$$z(t_N; s_{N-1}^{x\star}, s_{N-1}^{z\star}, q_{N-1}^\star) = s_N^{z\star},$$

i.e., the algebraic node value $s_N^{z\star}$ is the continuous continuation of the DAE solution on the last interval $[t_{N-1}, t_N]$ for the initial values $s_{N-1}^{x\star}, s_{N-1}^{z\star}, q_{N-1}^\star$.

Proof. As the algebraic solution of (2.20) $z(t; s^{x\star}_{N-1}, s^{z\star}_{N-1}, q^\star_{N-1})$ satisfies for $t = t_N$ the condition

$$\begin{aligned} 0 &= A(x(t_N; s^{x\star}_{N-1}, s^{z\star}_{N-1}, q^\star_{N-1}), q^\star_{N-1}) \cdot z(t_N; s^{x\star}_{N-1}, s^{z\star}_{N-1}, q^\star_{N-1}) \\ &\quad + g(x(t_N; s^{x\star}_{N-1}, s^{z\star}_{N-1}, q^\star_{N-1}), q^\star_{N-1}). \end{aligned}$$

Since $s^{x\star}, s^{z\star}, q^\star$ satisfy the matching conditions, we obtain

$$0 = A(s^{x\star}_N, q^\star_{N-1}) \cdot z(t_N; s^{x\star}_{N-1}, s^{z\star}_{N-1}, q^\star_{N-1}) + g(s^{x\star}_N, q^\star_{N-1}).$$

Furthermore, $s^{x\star}_N, s^{z\star}_N, q^\star_N$ satisfy the consistency condition by choice, hence

$$0 = A(s^{x\star}_N, q^\star_N) \cdot s^{z\star}_N + g(s^{x\star}_N, q^\star_N).$$

The choice $q^\star_N \triangleq q^\star_{N-1}$ together with the invertibility of $A(s^{x\star}_N, q^\star_{N-1})$ due to the index-1 assumption then completes the proof. □

In the general nonlinear case this is not necessarily true, see the discussion below. However, many DAEs in real-world applications have affine-linear algebraic equations, usually not even depending on the controls and thus leading to continuous algebraic states.

Second, if the algebraic equations depend explicitly on the controls, the algebraic states in the solution will in general not be smooth. Rather, the algebraic states inherit their smoothness properties from the controls. This is due to the (local) description of the algebraic states as function of the differential states and the controls by the implicit function theorem. To illustrate this, consider the following example DAE describing the motion of a point fixed on the unit circle

$$\begin{aligned} \dot{x} &= uAx + zx, \\ 0 &= x^\intercal x - 1, \end{aligned}$$

with $x \in \mathbb{R}^2, u, z \in \mathbb{R}$, and $A \in \mathbb{R}^{2\times 2}$ a given matrix. After an index reduction step (differentiation of the algebraic equation w.r.t. t) we obtain

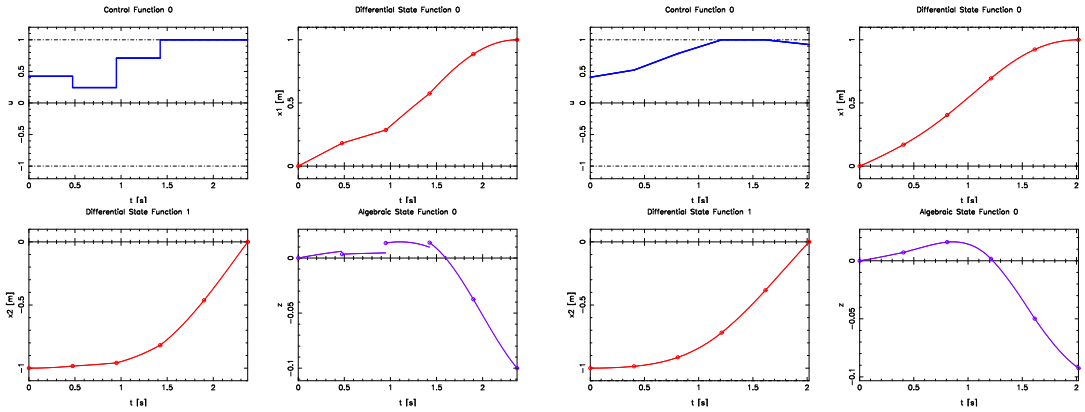
$$\dot{x} = uAx + zx, \tag{2.21a}$$

$$0 = u\,(x^\intercal Ax) + z\,(x^\intercal x)\,, \tag{2.21b}$$

which is an index-1 DAE where the algebraic equation yields the relation

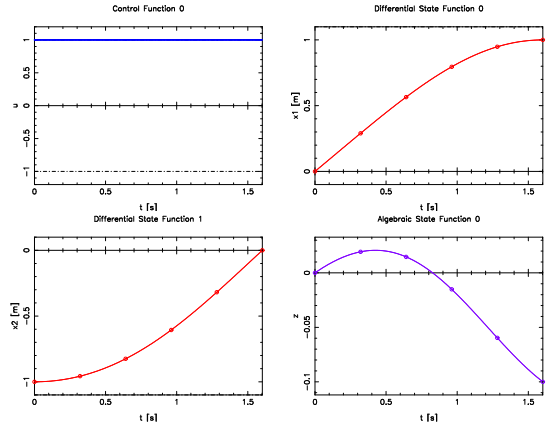
$$z = -u\frac{x^\intercal Ax}{x^\intercal x}.$$

Thus, the smoothness of z will be determined by the smoothness of u, which in turn depends on the chosen discretization. So while we use the same notion of "states" for both differential and algebraic states, we should be aware that they may exhibit quite different smoothness behavior. This effect is depicted in Figure 2.2 for a time optimal control of



(a) Intermediate iterate with piecewise constant controls

(b) Intermediate iterate with continuous piecewise linear controls



(c) Optimal solution for both control discretizations

Figure 2.2: Time optimal control of DAE (2.21). The intermediate iterates (upper row) are both feasible but slightly suboptimal. For the piecewise control discretization (upper left) the algebraic states jump where the control discretization jumps. For the continuous piecewise linear control discretization (upper right) no jumps appear in the algebraic variables. In the solution (lower row) the controls are globally smooth and so are the algebraic and differential states. The iterates and the solution were obtained with the Direct Multiple Shooting software `MUSCOD-II` [126, 127].

DAE (2.21), where the differential states have to be transferred from $x(0) = (0, -1)^\intercal$ to $x(T) = (1, 0)^\intercal$ subject to control bounds $|u| \leq 1$.

Third, uniqueness of the algebraic states cannot be guaranteed even in the index-1 case, since the consistency conditions (2.17) may have more than one solution if they are nonlinear. This means, that depending on the initialization of the values s_i^x, s_i^z, q_i we may end up with different but still consistent solutions to the DAE. Particularly, we cannot expect that the DAE solution obtained by the Direct Multiple Shooting method coincides with the solution of the forward problem if the same obtained controls are applied. This is due to the fact that the algebraic states may possibly switch to another solution branch

at each Multiple Shooting node. To illustrate this, consider the DAE

$$\dot{x} = uz^2, \tag{2.22a}$$
$$0 = z^2 - x, \tag{2.22b}$$

with $x, z, u \in \mathbb{R}$ which is index-1 for $z \neq 0$. If we maximize $x(T)$ subject to $|u| \leq 1$ and $x(0) = x_0 > 0$, the optimal control is clearly $u^\star \equiv 1$. A forward simulation with $u^\star$ gives $x(t) = x_0 \exp(t)$ and either $z(t) = \sqrt{x(t)}$ or $z(t) = -\sqrt{x(t)}$ depending on the choice of the consistent initial value of z for x_0. The solution of a Direct Multiple Shooting algorithm depends strongly on the initialization of the algebraic values s_i^z, as depicted in Figure 2.3.

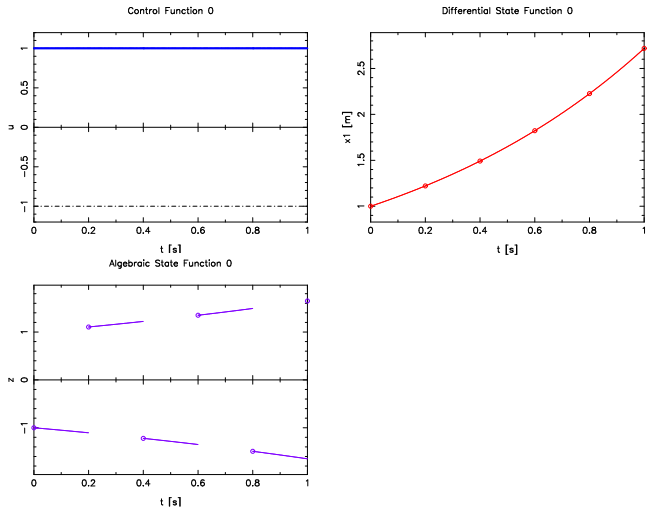


Figure 2.3: Optimal control of DAE (2.22) by Direct Multiple Shooting. The algebraic node values s_i^z were initialized with alternating sign. The solution was obtained with the Direct Multiple Shooting software `MUSCOD-II` [126, 127].

It should be stressed again, that the solutions obtained by Direct Multiple Shooting are perfectly valid, but may appear counterintuitive at some occasions. The discussion above aims at helping to interpret the results correctly in these cases.

The methodology developed in this thesis applies to optimization problems constrained both by ODEs and DAEs with index-1. For the sake of clarity of exposition we will in general not consider DAE constrained problems in the following. However, we will of course mention details specific to DAE constrained optimization problems if necessary.

2.2.3 Other direct approaches

Direct Single Shooting

In Direct Single Shooting, also known as the sequential approach, no additional intermediate starting values s_i^x, s_i^z with $i = 1, \ldots, N$ are introduced. Instead, the states are eliminated from the problem by solving the differential equation over the whole horizon $[0, T]$. Thus, the degrees of freedom are only the control parametrization variables q and

possibly the initial state (s_0^x, s_0^z). The inequalities are required to hold on a chosen time grid, on which the states are evaluated by integration.

Direct Single Shooting is a conceptually simple and appealing approach to solve OCPs, however, it comes with several drawbacks. For example, it may be impossible to integrate over the whole time horizon due to singularities, particularly for highly nonlinear or unstable dynamical systems. Furthermore, even if integration over the whole interval is possible, one usually has to provide good initial values to ensure convergence to a solution. The Direct Single Shooting method is described in, e.g., [163].

Direct Collocation

In Direct Collocation, the state and control trajectories are discretized on a fine grid of *collocation points* and represented by piecewise polynomial ansatzfunctions between the grid points. The differential equations are then required to hold for the polynomials in the grid points and thus yield the *collocation conditions* for the coefficients of the polynomials. The constraints are required to hold in the collocation points.

This approach gives rise to high-dimensional, highly structured NLPs which can be solved efficiently by both *interior-point methods* and sequential quadratic programming. It is well suited for nonlinear and unstable problems. A drawback is the fact that grid adaption for control of the approximation error may lead to changes in the problem dimensions during the solution process. The Direct Collocation method is described, e.g., in the textbook [24]. A popular state-of-the-art interior point solver used regularly for Direct Collocation is implemented in the software `IPOPT`, cf. [180].

2.2.4 Indirect approaches

Dynamic Programming and the Hamilton-Jacobi-Bellman equation

A central result in optimal control theory is *Bellman's principle of optimality of subarcs* which states that if one divides the control horizon in two subintervals, the optimal control on the second subinterval is still optimal for the restriction of the OCP to the second subinterval, provided that the initial states for the restriction are chosen by applying the optimal control on the first subinterval to the original initial states. Using this, one can derive a *partial differential equation* (PDE) for the *value function* which is closely connected to the objective function. This PDE is called the *Hamilton-Jacobi-Bellman (HJB) equation* and its solution yields not only the solution of the OCP but also the whole extremal field over all initial values which allows feedback control. However, for most problems the HJB equation is computationally intractable, so this approach is of limited use for practical application.

Dynamic Programming is the discrete-time counterpart of the HJB equation for the case of a discrete-time dynamical system. Bellman's principle then allows to construct the solution going backwards step by step. While still computationally highly expensive for the general case, it is a much more practical approach to calculate feedback controls

than the HJB equation. An excellent introduction to Dynamic Programming and the HJB equation is provided by [18].

Maximum Principle

From the HJB equation which provides a sufficient condition for the optimal solution of an OCP one can also derive necessary optimality conditions. Provided one has an optimal control and a corresponding optimal state trajectory, the optimal control has to fulfill a certain condition, point-wise minimizing a suitable help function called *Hamiltonian*. This gives rise to a numerical approach for solving OCPs, namely, eliminating the control in favor of the states and adjoint states via the minimization condition. This results in a boundary value problem for the states and adjoint states, which can be solved numerically to obtain candidate solutions for the OCP.

The advantage of this approach is that one obtains theoretical solutions of the optimal control problem (the direct approaches optimize on a pre-chosen subspace of the controls and thus are usually suboptimal) and thus numerical solutions can be computed with very high accuracy. This is sometimes crucial, for example in long-term spaceflight planning. On the downside, the elimination of the control by optimizing the Hamiltonian can be quite difficult. In particular, the presence of constraints may lead to a piecewise description of the optimal control which in turn leads to a boundary value problem with switching, which is a challenging problem class for itself. Since the switching structure has to be known in advance, one often uses a direct method as a first step to determine the switching structure and only then solves the boundary value problem from the maximum principle approach. Furthermore, the boundary value problem itself is usually difficult to solve due to nonlinearity and stability issues.

The Maximum Principle approach is well established and extensively described in, e.g., [38, 171, 18].

2.3 Basic theory of nonlinear programming

We can write the NLP (2.19) obtained by Direct Multiple Shooting generically as

$$\begin{aligned} \underset{w}{\text{minimize}} \quad & b(w) \\ \text{subject to} \quad & c(w) = 0 \\ & d(w) \geq 0 \end{aligned} \tag{2.24}$$

where the optimization variable $w \in \mathbb{R}^n$ comprises the states s^x and s^z, the control parameters q, and the parameters p and $b : \mathbb{R}^n \mapsto \mathbb{R}$, $c = (c_1, \ldots, c_l) : \mathbb{R}^n \mapsto \mathbb{R}^l$, and $d = (d_1, \ldots, d_m) : \mathbb{R}^n \mapsto \mathbb{R}^m$ are the objective function, the equality constraints and the inequality constraints, respectively.

Before we consider more closely how to solve NLP (2.24) let us begin by defining what a solution of an NLP is.

Definition 2.1 (Feasibility and Optimality)**.**

1. *A point $\hat{w} \in \mathbb{R}^n$ is called* feasible *if $c(\hat{w}) = 0$ and $d(\hat{w}) \geq 0$. We then say that the constraints are* satisfied *at $\hat{w}$. Otherwise the point $\hat{w}$ is called* infeasible *and we say that the constraints are* violated *at $\hat{w}$.*

2. *The point $w^\star$ is a* local minimizer *(or* local solution*) of problem (2.24) if $w^\star$ is feasible and if there exists a neighborhood $\mathcal{N}(w^\star)$ such that*

$$b(w^\star) \leq b(w) \quad \text{for all feasible } w \in \mathcal{N}(w^\star) \tag{2.25}$$

If inequality (2.25) is strict for all feasible $w \in \mathcal{N}(w^\star)$, $w \neq w^\star$ we call $w^\star$ a strong *or* strict *local minimizer. Otherwise we call $w^\star$ a* weak *local minimizer. If there exists a neighborhood of $w^\star$ so that $w^\star$ is the only local solution in that neighborhood we call $w^\star$ an* isolated *local minimizer.*

At each feasible point, a part of the inequalities may be satisfied as equalities. We define:

Definition 2.2 (Active Constraints and Active Set)**.**

1. *The constraint $d_j(w) \geq 0$ is said to be* active *at $\hat{w}$ if $d_j(\hat{w}) = 0$. If $d_j(\hat{w}) > 0$ then the constraint is said to be* inactive *at $\hat{w}$.*

2. *For a feasible $\hat{w} \in \mathbb{R}^n$ the set $\mathcal{A}(\hat{w}) \triangleq \{j | d_j(\hat{w}) = 0\}$ is called the* active set *at $\hat{w}$.*

To decide if a point $w^\star$ is a local solution of problem (2.24) we have to compare its objective function value $f(w^\star)$ with the objective function values of feasible points in its neighborhood, according to inequality (2.25). Due to in general nonlinear constraints feasibility can often be retained only on nonlinear paths in $\mathbb{R}^n$, called *feasible arcs*. A (continuously differentiable) arc is a directed (continuously differentiable) curve $\alpha : [0, \epsilon] \mapsto \mathbb{R}^n$, where $\epsilon > 0$. To ensure the existence of such a feasible arc one has to impose additional conditions, so called *constraint qualifications*. We state one of several possible definitions (cf. [78]).

Definition 2.3 (Kuhn-Tucker Constraint Qualification)**.**
The Kuhn-Tucker constraint qualification (KTCQ) with respect to the equality constraints $c(w) = 0$ and the inequality constraints $d(w) \geq 0$ holds at the feasible point $\hat{w}$ if every vector $y \in \mathbb{R}^n$ satisfying

$$\begin{aligned} \nabla c_i(\hat{w})^\intercal y &= 0, \quad i = 1, \ldots, l, \\ \nabla d_j(\hat{w})^\intercal y &\geq 0, \quad j \in \mathcal{A}(\hat{w}), \end{aligned}$$

is tangent to a continuously differentiable arc α emanating from $\hat{w}$ and contained in the feasible region.

There are feasible points where the KTCQ does not hold, Kuhn and Tucker give a simple example in [118]. For a review on various constraint qualifications including the KTCQ and their relations see, e.g., [108].

For linear constraints the KTQC always holds, which is an important observation since in the later discussed SQP method for solving (2.24) we consider linearized constraints in each iteration step. Unfortunately, for nonlinear constraints the KTCQ is almost impossible to test in practice. A sufficient condition which can be computationally verified is the following.

Definition 2.4 (Linear Independence Constraint Qualification)**.**
The Linear Independence Constraint Qualification (LICQ) *holds at* $\hat{w}$*, if* $\nabla c_i(\hat{w})$*,* $i = 1, \ldots, l$ *and* $\nabla d_j(\hat{w})$*,* $j \in \mathcal{A}(\hat{w})$ *are linearly independent. If* $\hat{w}$ *is also feasible we call it a* regular point*.*

For a constructive proof of sufficiency see [78]. Furthermore, the LICQ are the weakest constraint qualification that ensures both the existence and uniqueness of Lagrange multipliers, see [179]. In the following, let $\nabla c(w^\star) \triangleq \left(\frac{\mathrm{d}c}{\mathrm{d}w}(w^\star)\right)^\intercal$ and $\nabla d(w^\star) \triangleq \left(\frac{\mathrm{d}d}{\mathrm{d}w}(w^\star)\right)^\intercal$. We now state first order necessary conditions for a local solution of problem (2.24).

Theorem 2.2 (Karush-Kuhn-Tucker Necessary Conditions)**.**
Assume that the Kuhn-Tucker constraint qualification holds at $w^\star \in \mathbb{R}^n$*. If* $w^\star$ *is a local minimizer of* (2.24) *then there exist multipliers* $\lambda^\star \in \mathbb{R}^l$ *and* $\mu^\star \in \mathbb{R}^m$ *so that* $(w^\star, \lambda^\star, \mu^\star)$ *satisfy the following set of conditions*

$$\begin{aligned}
\nabla_w b(w^\star) - \nabla c(w^\star)\lambda^\star - \nabla d(w^\star)\mu^\star &= 0, && \text{(2.26a)}\\
c(w^\star) &= 0, && \text{(2.26b)}\\
d(w^\star) &\geq 0, && \text{(2.26c)}\\
\mu^\star &\geq 0, && \text{(2.26d)}\\
\mu_j^\star \, d_j(w^\star) &= 0, \quad j = 1, \ldots, m. && \text{(2.26e)}
\end{aligned}$$

The conditions given above are known as Karush-Kuhn-Tucker (KKT) conditions *and a tuple* $(w^\star, \lambda^\star, \mu^\star)$ *satisfying these conditions is called a* KKT point *or* stationary point*.*

A proof of Theorem 2.2 can be found in any textbook on optimization, see for example [78, 140]. Equation (2.26a) can also be rewritten as $\nabla_w \mathcal{L}(w^\star, \lambda^\star, \mu^\star) = 0$ with

$$\begin{aligned}
\mathcal{L}(w, \lambda, \mu) &\triangleq b(w) - \lambda^\intercal \, c(w) - \mu^\intercal \, d(w) && \text{(2.27)}\\
&= b(w) - \sum_{i=1}^{l} \lambda_i \, c_i(w) - \sum_{j=1}^{m} \mu_j \, d_j(w)
\end{aligned}$$

being called the *Lagrange function* and $\lambda \in \mathbb{R}^l$ and $\mu \in \mathbb{R}^m$ being called the *Lagrange multipliers*.

Let us define the following disjunctive decomposition of the active set at a local minimizer $w^\star$ with corresponding multipliers $(\lambda^\star, \mu^\star)$

$$\begin{aligned}
\mathcal{A}^+(w^\star) &\triangleq \left\{ j \in \mathcal{A}(w^\star) | \mu_j^\star > 0 \right\}, && \text{(2.28)}\\
\mathcal{A}^0(w^\star) &\triangleq \left\{ j \in \mathcal{A}(w^\star) | \mu_j^\star = 0 \right\}. && \text{(2.29)}
\end{aligned}$$

In the case of $\mathcal{A}^0(w^\star) = \emptyset$, i.e. $\mathcal{A}(w^\star) = \mathcal{A}^+(w^\star)$, we say that the *strict complementary condition* holds.

For a KKT point, we can in general not distinguish from first-order conditions alone between a minimizer, a maximizer or a saddle point. To do so, we need to check second order conditions which give information on the curvature and thus on the nature of the stationary point. In contrast to the first-order case (with exception of some special cases) there exist both necessary and sufficient second-order conditions for a minimizer. We will restrict ourselves to a sufficient condition which provides strong results from the numerical point of view, e.g., local applicability of numerical methods to the problem, uniqueness of the multipliers and sensitivity information for the optimal objective value. Of course we need to require more restrictive conditions to hold in order to obtain these results (cf. [131]).

Definition 2.5 (Jacobian Uniqueness Condition)**.**
A KKT point $(w^\star, \lambda^\star, \mu^\star)$ satisfies the Jacobian uniqueness condition if the following conditions hold:

- *$w^\star$ is a regular point,*
- *the strict complementary condition holds,*
- *for any vector $y \in \mathbb{R}^n$ satisfying*

$$\begin{aligned} \nabla c_i(w^\star)^\intercal\, y &= 0, \quad i = 1, \dots, l, \\ \nabla d_j(w^\star)^\intercal\, y &= 0, \quad j \in \mathcal{A}(w^\star), \end{aligned}$$

it follows that

$$y^\intercal\, \nabla_w^2 \mathcal{L}(w^\star, \lambda^\star, \mu^\star)\, y > 0.$$

With the help of these conditions we can now state a strong sufficient condition for optimality (cf. [131, 78]).

Proposition 2.3 (Sufficiency and Uniqueness of Multipliers)**.**
If a KKT point $(w^\star, \lambda^\star, \mu^\star)$ satisfies the Jacobian uniqueness condition then $w^\star$ is a strict local minimizer of problem (2.24) *and the Lagrange multipliers $\lambda^\star \in \mathbb{R}^l$ and $\mu^\star \in \mathbb{R}^m$ are uniquely determined. The Jacobian matrix J corresponding to the set of nonlinear equations* (2.26a)*,* (2.26b)*, and* (2.26e)*, which is given by*

$$J = \begin{pmatrix} \nabla_w^2 \mathcal{L}(w, \lambda, \mu) & -\nabla c(w) & -\nabla d(w) \\ \frac{\mathrm{d}g}{\mathrm{d}w}(w) & 0 & 0 \\ \operatorname{diag}(\mu_1, \dots, \mu_m)\ \frac{\mathrm{d}d}{\mathrm{d}w}(w) & 0 & \operatorname{diag}(d_1(w), \dots, d_m(w)) \end{pmatrix},$$

is nonsingular at the point $(w^\star, \lambda^\star, \mu^\star)$.

The non-singularity of J at $(w^\star, \lambda^\star, \mu^\star)$ implies in particular that it is possible to apply Newton's method to solve the equations (2.26a), (2.26b), and (2.26e) given an initial estimate close enough to the solution. Furthermore, if $\nabla_w^2 \mathcal{L}$ is Lipschitz continuous near $(w^\star, \lambda^\star, \mu^\star)$ then the convergence rate is quadratic, see, e.g., [140].

2.4 Sequential quadratic programming

After defining what a (local) solution of NLP (2.24) is and how it is characterized, we now briefly outline how to numerically obtain such a solution. We solve NLP (2.24) using an iterative framework, i.e., we start with an initial guess (w^0, λ^0, μ^0) for the primal variables w and the dual variables (λ, μ) and then generate a sequence (w^k, λ^k, μ^k) that converges to a KKT point. The steps are calculated by building and minimizing a quadratic model of the Lagrangian function subject to the locally linearized equality and inequality constraints

$$\min_{\Delta w} \quad \tfrac{1}{2}\Delta w^\intercal \nabla_w^2 \mathcal{L}(w^k, \lambda^k, \mu^k)\Delta w + \nabla b(w^k)^\intercal \Delta w \tag{2.30a}$$

$$\text{s.t.} \quad 0 = \frac{\mathrm{d}c}{\mathrm{d}w}(w^k)\Delta w + c(w^k), \tag{2.30b}$$

$$0 \leq \frac{\mathrm{d}d}{\mathrm{d}w}(w^k)\Delta w + d(w^k), \tag{2.30c}$$

which belongs to the problem class of *quadratic programs* (QP). After solving QP (2.30) we then iterate

$$w^{k+1} = w^k + \alpha_k \Delta w^k, \quad \lambda^{k+1} = (1-\alpha_k)\lambda^k + \alpha_k \lambda^k_{\mathrm{QP}}, \quad \mu^{k+1} = (1-\alpha_k)\mu^k + \alpha_k \mu^k_{\mathrm{QP}}, \tag{2.31}$$

where $(\Delta w^k, \lambda^k_{\mathrm{QP}}, \mu^k_{\mathrm{QP}})$ is the primal-dual solution of the QP. The step length $\alpha_k \in (0, 1]$ is selected to ensure global convergence either by line search or trust region algorithms, see for example [140]. In this thesis we will not deal with globalization issues but rather consider the full-step case $\alpha_k = 1$. Note that we rather use $\nabla b(w_k)$ instead of $\nabla_w \mathcal{L}(w^k, \lambda^k, \mu^k)$ in the QP objective, as by this formulation the QP multipliers $\lambda^k_{\mathrm{QP}}, \mu^k_{\mathrm{QP}}$ can be directly used as the new NLP multipliers λ^{k+1}, μ^{k+1} in the full-step case.

This whole approach is known as *sequential quadratic programming* (SQP). It is particularly effective for problems with nonlinear constraints. In practice, SQP often uses a positive definite approximation $B_k \approx \nabla_w^2 \mathcal{L}(w^k, \lambda^k, \mu^k)$ of the Hessian of the Lagrange function because it is computationally less expensive and a positive definite QP Hessian guarantees the existence of a unique global minimizer of the QP. Popular choices for B_k comprise for example positive definite approximations with low-rank updates such as the Broyden-Goldfarb-Fletcher-Shanno (BFGS) update, the BFGS update with limited memory (L-BFGS), or Gauss-Newton approximations, cf. [140]. It should be noted that also the constraint Jacobians may be replaced by approximations. In this case, we call the approach described above an *inexact SQP* method. We discuss inexact SQP methods in detail in Chapter 5.

2.4.1 Local convergence theory

Given the active set $\mathcal{A}(w^k)$ of QP (2.30) the step for the primal variables and the dual variables of the active constraints could in principle be obtained by solving the following linear system which represents the KKT conditions of the corresponding equality-constrained

QP:

$$\begin{pmatrix} \nabla_w^2 \mathcal{L}_k & -\nabla c_k & -\nabla d_k \\ \frac{\mathrm{d}c_k}{\mathrm{d}w} & 0 & 0 \\ \frac{\mathrm{d}d_k^{\mathrm{act}}}{\mathrm{d}w} & 0 & 0 \end{pmatrix} \begin{pmatrix} \Delta w^k \\ \lambda_{\mathrm{QP}}^k \\ \mu_{\mathrm{QP}}^{\mathrm{act},k} \end{pmatrix} = - \begin{pmatrix} \nabla_w \mathcal{L}_k \\ c_k \\ d_k^{\mathrm{act}} \end{pmatrix}, \tag{2.32}$$

where subscript k denotes evaluation in (w^k, λ^k, μ^k) and $d^{\mathrm{act}}(w) \triangleq (d_i(w))_{i \in \mathcal{A}(w)}$. Of course, the active set has to be determined during the QP solution process and in general, $\mathcal{A}(w^k)$ will change from iteration to iteration as long as w^k is not close enough to a solution $w^\star$ that satisfies the Jacobian uniqueness condition.

However, it is a key observation for the local convergence analysis of SQP methods that close to a primal-dual solution $(w^\star, \lambda^\star, \mu^\star)$ of NLP (2.24) that satisfies the Jacobian uniqueness condition, the active set of all QP subproblems (2.30) is the same and identical to the active set of the NLP in $w^\star$. Thus, we can locally consider the SQP method as an iteration with steps generated by equality-constrained QP subproblems. The following theorem is due to [158].

Theorem 2.4 (Stability of the Active Set near to a Solution)**.** *Let $w^\star$ be a minimizer of problem* (2.24) *and suppose that the Jacobian uniqueness condition holds at $w^\star$. Then if (w^k, λ^k, μ^k) is sufficiently close to $(w^\star, \lambda^\star, \mu^\star)$, there is a local solution of the subproblem* (2.30) *the active set $\mathcal{A}(w^k)$ of which is the same as the active set $\mathcal{A}(w^\star)$ of the nonlinear program* (2.24) *at $w^\star$.*

Furthermore, we can see that (2.32) is also a step of Newton's method applied to the system of nonlinear equations

$$F(z) \triangleq \begin{pmatrix} \nabla_w \mathcal{L}(z) \\ c(z) \\ d^{\mathrm{act}}(z) \end{pmatrix} \tag{2.33}$$

with the shorthand $z \triangleq (w, \lambda, \mu)$. This allows to analyze the local convergence of SQP methods with the well-understood theory of local convergence of Newton's method and Newton-like methods. We present a variant of the Local Contraction Theorem (see [28]) in the formulation of [147].

Let the set of Newton pairs be defined according to

$$\mathcal{N} \triangleq \left\{ (z, z') \in D \times D | z' = z - M(z) F(z) \right\}$$

with D a neighborhood of $z^\star$ with constant active set according to Theorem 2.4 and let $\| \cdot \|$ denote a norm of $\mathbb{R}^N$. Furthermore, let $J(z) \triangleq \frac{\mathrm{d}F}{\mathrm{d}z}(z)$ be the exact Jacobian of $F(z)$ and $M(z)$ be an approximation of $J^{-1}(z)$. We need two conditions on J and M:

Definition 2.6 (Lipschitz condition: ω-condition)**.** *The Jacobian J together with the approximation M satisfy the ω-condition in D if there exists $\omega < \infty$ such that for all $t \in [0, 1], (z, z') \in \mathcal{N}$*

$$\| M(z')(J(z + t(z' - z)) - J(z))(z - z') \| \leq \omega t \| z - z' \|^2.$$

Definition 2.7 (Compatibility condition: κ-condition)**.** *The approximation M satisfies the κ-condition in D if there exists $\kappa < 1$ such that for all $(z, z') \in \mathcal{N}$*

$$\|M(z')(I - J(z)M(z))F(z)\| \leq \kappa \|z - z'\|.$$

With the constants from the previous two definitions we define

$$c_k \triangleq \kappa + (\omega/2)\|\Delta z^k\|$$

and for $c_0 < 1$ the closed ball

$$D_0 \triangleq \overline{B}(z_0; \|\Delta z_0\|/(1 - c_0)).$$

The following theorem then characterizes the local convergence of a full step (i.e., $\alpha_k = 1$) Newton-type method in a neighborhood of the solution. For a proof see, e.g., [28, 147].

Theorem 2.5 (Local Contraction Theorem)**.** *Let J and M satisfy the ω-κ-conditions in D and let $z^0 \in D$. If $c_0 < 1$ and $D_0 \subset D$, then $z^k \in D_0$ and the sequence (z^k) converges to some $z^\star \in D_0$. The following estimate holds on the step sizes*

$$\|\Delta z^{k+1}\| \leq c_k \|\Delta z^k\| = \kappa \|\Delta z^k\| + (\omega/2)\|\Delta z^k\|^2.$$

Furthermore, the a priori estimate

$$\|z^{k+j} - z^\star\| \leq \frac{(c_k)^j}{1 - c_k}\|\Delta z^k\| \leq \frac{(c_0)^{k+j}}{1 - c_0}\|\Delta z^0\|$$

holds and the limit $z^\star$ satisfies

$$M(z^\star)F(z^\star) = 0.$$

If additionally $M(z)$ is continuous and nonsingular in $z^\star$, then

$$F(z^\star) = 0.$$

Note that the Theorem does not require an exact Hessian $\nabla^2_w \mathcal{L}(w^k, \lambda^k, \mu^k)$ in (2.32) and even allows for approximations of the constraint Jacobians $\frac{\mathrm{d}c}{\mathrm{d}w}(w)$ and $\frac{\mathrm{d}d^{\mathrm{act}}}{\mathrm{d}w}(w)$, see also Chapter 5.

2.5 Structured SQP for Direct Multiple Shooting

When solving an NLP obtained by a Direct Multiple Shooting discretization with SQP there are two main non-standard characteristics: first, the introduction of the shooting nodes leads to a special block-sparse structure in the QPs (2.30) which has to be exploited in an efficient algorithm. Second, since the matching conditions involve the solutions of differential equations, one has to think about calculating efficient and accurate derivatives of these solutions for the SQP constraint linearizations.

2.5.1 Structure exploitation

We start by looking at the exploitation of the structure arising from the Multiple Shooting discretization in the solution of the QP subproblems. Let us write down QP (2.30) in full detail according to NLP (2.19).

$$\begin{aligned}
\underset{\Delta s, \Delta q}{\text{minimize}} \quad & \sum_{i=0}^{N-1} \left[\frac{1}{2} \begin{pmatrix} \Delta s_i \\ \Delta q_i \end{pmatrix}^\mathsf{T} \begin{pmatrix} B_i^{ss} & B_i^{sq} \\ B_i^{qs} & B_i^{qq} \end{pmatrix} \begin{pmatrix} \Delta s_i \\ \Delta q_i \end{pmatrix} + \begin{pmatrix} b_i^s \\ b_i^q \end{pmatrix}^\mathsf{T} \begin{pmatrix} \Delta s_i \\ \Delta q_i \end{pmatrix} \right] && \text{(2.34a)} \\
& + \frac{1}{2} \Delta s_N^\mathsf{T} B^{ss} \Delta s_N + b_N^{s,T} \Delta s_N \\
\text{subject to} \quad & 0 = \Delta s_{i+1} - G_i^s \Delta s_i - G_i^q \Delta q_i + c_i, \quad i = 0, \dots, N-1, && \text{(2.34b)} \\
& 0 \leq H_i^s \Delta s_i + H_i^q \Delta q_i + c_i^{\text{path}}, \quad i = 0, \dots, N-1, && \text{(2.34c)} \\
& 0 = R_{\text{e},0}^s \Delta s_0 + R_{\text{e},N}^s \Delta s_N + r_\text{e}, && \text{(2.34d)} \\
& 0 \leq R_{\text{i},0}^s \Delta s_0 + R_{\text{i},N}^s \Delta s_N + r_\text{i}, && \text{(2.34e)}
\end{aligned}$$

with approximations

$$\begin{aligned}
B_i^{ss} &\approx \frac{\partial^2 \mathcal{L}}{\partial {s_i}^2}(w, \lambda, \mu), \quad i = 0, \dots, N, \\
(B_i^{qs})^\mathsf{T} = B_i^{sq} &\approx \frac{\partial^2 \mathcal{L}}{\partial q_i \partial s_i}(w, \lambda, \mu), \quad i = 0, \dots, N-1, \\
B_i^{qq} &\approx \frac{\partial^2 \mathcal{L}}{\partial {q_i}^2}(w, \lambda, \mu), \quad i = 0, \dots, N-1
\end{aligned}$$

of the Hessian of the Lagrange function, Jacobians

$$\begin{aligned}
G_i^{s|q} &= \frac{\partial x_i}{\partial s_i | q_i}(t_{i+1}; s_i, q_i), \\
H_i^{s|q} &= \frac{\partial c^{\text{path}}}{\partial s_i | q_i}(s_i, \phi_i(t_i, q_i)), \\
R_{\text{e},i}^s &= \frac{\partial r_\text{e}}{\partial s_i}(s_0, s_N), \quad i \in \{0, N\}, \\
R_{\text{i},i}^s &= \frac{\partial r_\text{i}}{\partial s_i}(s_0, s_N), \quad i \in \{0, N\}
\end{aligned}$$

of the ODE solutions, the path constraints, and the boundary equality and inequality conditions, respectively. Furthermore,

$$\begin{aligned}
b_i^{s|q} &= \nabla_{s_i | q_i} L(s_i, q_i), \quad i = 0, \dots, N-1, \\
b_N^s &= \nabla_{s_N} m(s_N), \\
c_i &= s_{i+1} - x(t_{i+1}; s_i, q_i), \quad i = 0, \dots, N-1, \\
c_i^{\text{path}} &= c^{\text{path}}(s_i, \phi_i(t_i, q_i)), \quad i = 0, \dots, N-1, \\
r_\text{e} &= r_\text{e}(s_0, s_N), \\
r_\text{i} &= r_\text{i}(s_0, s_N),
\end{aligned}$$

are objective gradients, matching condition residuals, path constraint residuals, and boundary equality and inequality condition residuals, respectively.

We can use the linearized matching constraints (2.34b) to eliminate the degrees of freedom $\Delta s_1, \ldots, \Delta s_N$ from QP (2.34). This approach is called *condensing* and results in a QP of the same size as the QPs arising from the Single Shooting approach. We obtain the following relation between the eliminated variables and the variables that remain in the QP

$$\begin{pmatrix} \Delta s_1 \\ \Delta s_2 \\ \vdots \\ \Delta s_N \end{pmatrix} = \begin{pmatrix} C_1^s & C_{1,0}^q & & & \\ C_2^s & C_{2,0}^q & C_{2,1}^q & & \\ \vdots & & & \ddots & \\ C_N^s & C_{N,0}^q & \cdots & & C_{N,N-1}^q \end{pmatrix} \begin{pmatrix} \Delta s_0 \\ \Delta q_0 \\ \vdots \\ \Delta q_{N-1} \end{pmatrix} + \begin{pmatrix} c_0' \\ c_1' \\ \vdots \\ c_{N-1}' \end{pmatrix}, \quad (2.35)$$

where

$$\begin{array}{lll} C_1^s = G_0^s, & C_i^s = G_{i-1}^x C_{i-1}^s, & i = 2, \ldots, N, \\ C_{j+1,j}^q = G_j^q, & C_{i,j}^q = G_{i-1}^x C_{i-1,j}^q, & j = 0, \ldots, N-1,\ i = j+2, \ldots, N, \\ c_0' = -c_0, & c_i' = G_{i-1}^x c_{i-1}' - c_i, & i = 1, \ldots N-1, \end{array} \quad (2.36)$$

and for notational simplicity we further define $C_0^s \triangleq \mathbb{I}$, $C_{0,0}^q \triangleq 0$, and $c_{-1}' \triangleq 0$. Inserting (2.35) into QP (2.34) we obtain

$$\begin{aligned} \underset{\Delta s_0, \Delta q}{\text{minimize}} \quad & \frac{1}{2} \begin{pmatrix} \Delta s_0 \\ \Delta q_0 \\ \vdots \\ \Delta q_{N-1} \end{pmatrix}^\mathsf{T} \begin{pmatrix} \tilde{B}^{s_0,s_0} & \tilde{B}_0^{s_0,q} & \cdots & \tilde{B}_{N-1}^{s_0,q} \\ \tilde{B}_0^{q,s_0} & \tilde{B}_{0,0}^{q,q} & \cdots & \tilde{B}_{0,N-1}^{q,q} \\ \vdots & \vdots & \ddots & \vdots \\ \tilde{B}_{N-1}^{q,s_0} & \tilde{B}_{N-1,0}^{q,q} & \cdots & \tilde{B}_{N-1,N-1}^{q,q} \end{pmatrix} \begin{pmatrix} \Delta s_0 \\ \Delta q_0 \\ \vdots \\ \Delta q_{N-1} \end{pmatrix} \quad (2.37a) \\ & + \begin{pmatrix} \tilde{b}^{s_0} \\ \tilde{b}^{q_0} \\ \vdots \\ \tilde{b}^{q_{N-1}} \end{pmatrix}^\mathsf{T} \begin{pmatrix} \Delta s_0 \\ \Delta q_0 \\ \vdots \\ \Delta q_{N-1} \end{pmatrix} \\ \text{subject to} \quad & 0 \leq \tilde{H}_i^s \Delta s_0 + \sum_{j=0}^{i} \tilde{H}_{i,j}^q \Delta q_j + \tilde{c}_i^{\text{path}}, \quad i = 0, \ldots, N-1 \quad (2.37b) \\ & 0 = \tilde{R}_{\text{e},0}^s \Delta s_0 + \sum_{j=0}^{N-1} \tilde{R}_{\text{e},j}^q \Delta q_j + \tilde{r}_\text{e} \quad (2.37c) \\ & 0 \leq \tilde{R}_{\text{i},0}^s \Delta s_0 + \sum_{j=0}^{N-1} \tilde{R}_{\text{i},j}^q \Delta q_j + \tilde{r}_\text{i}, \quad (2.37d) \end{aligned}$$

with the condensed Hessian blocks

$$\begin{aligned}
\tilde{B}^{s_0,s_0} &\triangleq \sum_{i=0}^{N} (C_i^s)^\intercal \, B_i^{ss} \, C_i^s, \\
\left(\tilde{B}_i^{q,s_0}\right)^\intercal = \tilde{B}_i^{s_0,q} &\triangleq (C_i^s)^\intercal \, B_i^{sq} + \sum_{j=i+1}^{N} \left(C_j^s\right)^\intercal B_j^{ss} \, C_{j,i}^q, \; i = 0, \ldots, N-1, \\
\tilde{B}_{i,i}^{q,q} &\triangleq B_i^{qq} + \sum_{j=i+1}^{N} \left(C_{j,i}^q\right)^\intercal B_j^{ss} \, C_{j,i}^q, \; i = 0, \ldots, N-1, \\
\left(\tilde{B}_{j,i}^{q,q}\right)^\intercal = \tilde{B}_{i,j}^{q,q} &\triangleq \left(C_{j,i}^q\right)^\intercal B_j^{sq} + \sum_{k=j+1}^{N} \left(C_{k,i}^q\right)^\intercal B_k^{ss} \, C_{k,j}^q, \; \begin{cases} i = 0, \ldots, N-2, \\ j = i+1, \ldots, N-1, \end{cases}
\end{aligned}$$

the condensed gradients

$$\begin{aligned}
\tilde{b}^{s_0} &\triangleq b_0^s + \sum_{i=1}^{N} (C_i^s)^\intercal \left(B_i^{ss} c_{i-1}' + b_i^s\right), \\
\tilde{b}^{q_i} &\triangleq b_i^q + B_i^{qs} c_{i-1}' + \sum_{j=i+1}^{N} \left(C_{j,i}^q\right)^\intercal \left(b_j^s + B^{ss} c_{j-1}'\right), \; i = 0, \ldots, N-1,
\end{aligned}$$

the condensed path constraint matrices and residuals

$$\begin{aligned}
\tilde{H}_i^s &\triangleq H_i^s C_i^s, \\
\tilde{H}_{i,j}^q &\triangleq H_i^s C_{i,j}^q + \delta_{i,j} H_i^q \\
\tilde{c}_i^{\text{path}} &\triangleq H_i^s c_{i-1}',
\end{aligned}$$

and the condensed boundary equality and inequality condition matrices and residuals

$$\begin{aligned}
\tilde{R}_{\text{e|i},0}^s &\triangleq R_{\text{e|i},0}^s + R_{\text{e|i},N}^s C_N^s, \\
\tilde{R}_{\text{e|i},j}^q &\triangleq R_{\text{e|i},N}^s C_{N,j}^q, \\
\tilde{r}_{\text{e|i}} &\triangleq R_{\text{e|i},N}^s c_{N-1}' + r_{\text{e|i}}.
\end{aligned}$$

After solving QP (2.37) we can recover the variables $(\Delta s_1, \ldots, \Delta s_N)$ by recursively using equations (2.34b). Furthermore, the multipliers for (2.34b) can then be obtained by suitably resolving the Lagrange gradient of QP (2.34), for details see [124, 125].

The presented condensing approach involves products of the sensitivity matrices. Depending on the properties of the dynamic system this may lead to severe ill-conditioning since the condition number of the matrix products is the product of the condition numbers of the individual matrices. An alternative computationally more expensive but stable condensing approach based on QR decompositions is presented in [166]. Furthermore, if the number of control variable exceeds the number of state variables, *complementary condensing* as presented in [104] is computationally more efficient. A condensing approach that is quadratic in the number of Multiple Shooting intervals is presented in [7].

2.5.2 Derivative calculation

The principle of Internal Numerical Differentiation

Besides the special structure discussed above, one of the most distinct features of the Multiple Shooting NLP (2.19) is the fact that part of the constraints, i.e., the matching conditions, are evaluated by complex schemes for numerical integration of differential equations. Thus, the evaluation of these constraints and in particular of their derivatives is usually much more expensive than the evaluation of the other constraints and may well be the dominant part of the overall computational effort per SQP iteration.

The conceptually simplest way to obtain derivatives (or *sensitivities*) of the IVP

$$\dot{x}(t) = f(t, x(t), q), \quad x(t_0) = x_0, \quad t \in [t_0, t_1] \tag{2.38}$$

with respect to initial values x_0 and control parameters q is applying finite differences to the solutions for perturbed initial values and control parameters,

$$(G^x)_i = \frac{\partial x}{\partial (x_0)_i}(t_1; x_0, q) \approx \frac{x(t_1; x_0 + \delta_i e_i, q) - x(t_1; x_0, q)}{h_i}, \ i = 1, \ldots, n_{\mathrm{x}}, \tag{2.39a}$$

$$(G^q)_i = \frac{\partial x}{\partial q_i}(t_1; x_0, q) \approx \frac{x(t_1; x_0, q + \delta_i e_i) - x(t_1; x_0, q)}{h_i}, \ i = 1, \ldots, n_{\mathrm{q}}, \tag{2.39b}$$

with e_i the i-th unit vector of $\mathbb{R}^{n_{\mathrm{x}}}$ and $\mathbb{R}^{n_{\mathrm{q}}}$, respectively. We refer to this approach as *external numerical differentiation* (END).

However, this approach has several severe drawbacks. First, one can see from the approximation error of the differential quotients (2.39a) that one cannot expect a derivative accuracy of more than half of the valid digits of the evaluation of the trajectories, cf. [28]. Second, modern integrator schemes adaptively select parts of the scheme such as step sizes, orders, matrix rebuilds and decompositions, and number of Newton-like iterations to reduce the computational effort while meeting prescribed accuracy demands. The conditional nature of these adaptive decisions leads to the fact that the output of the integrator may depend non-smoothly on the inputs, i.e., initial states, controls, and parameter values. However, END implicitly differentiates these non-differentiable components. Third, for the non-scalar case it may be difficult to choose both evaluation tolerance and disturbance δ if there are components with different growth behavior, which also means that the derivative computation may become unstable for low tolerances.

A better alternative is *internal numerical differentiation* (IND) which differentiates the (adaptively generated) scheme while keeping the adaptive components fixed. This can be done in several ways: first, one may calculate the nominal trajectory, reuse the nominal integration scheme to evaluate trajectories for disturbed initial values and control parameters and then apply finite differences to approximate the sensitivities. This approach is referred to as *varied trajectories*. Second, if analytical derivatives of the right-hand side are available, e.g., by using tools for automatic differentiation (AD), one may calculate the nominal trajectory and reuse the nominal integration scheme to evaluate the variational

differential equations of (2.38)

$$\dot{G}^x = \frac{\partial f}{\partial x}(t, x(t), q)G^x, \quad G^x(t_0) = \mathbb{I}, \quad t \in [t_0, t_1], \tag{2.40a}$$

$$\dot{G}^q = \frac{\partial f}{\partial x}(t, x(t), q)G^q + \frac{\partial f}{\partial q}(t, x(t), q), \quad G^q(t_0) = 0, \quad t \in [t_0, t_1], \tag{2.40b}$$

along the nominal trajectory.

By using IND one obtains more accurate sensitivity approximations for the same nominal integration tolerance compared to END, and the reuse of the nominal integration scheme amounts to significant computational savings. Moreover, the sensitivities computed by IND are *consistent* with the discretization, i.e., they are the derivatives of the discretized solution of the differential equations (2.38). This is important in the optimization context, where the constraint derivatives also enter in the right-hand side of the optimization problem, see (2.32).

Example of IND for an adaptive discretization scheme

To illustrate the principle of IND, we consider explicit single step integration methods which compute discrete approximations

$$\eta_{k+1} = \eta_k + h_k \Phi(\tau_k, \eta_k, q; h_k), \ k = 0, \dots, n-1, \quad \eta_0 = x_0, \tag{2.41}$$

to the solution x of IVP (2.38) in the time points $t_0 = \tau_0, \dots, \tau_n = t_1$ with $\tau_{k+1} = \tau_k + h_k$. The integration method is defined by the *step function* $\Phi(\cdot)$. A well known family of single step methods are the explicit Runge-Kutta (RK) schemes with step function

$$\Phi(\tau, \eta, q; h) \triangleq \sum\nolimits_{i=1}^{s} \gamma_i k_i, \quad k_i \triangleq f\left(\tau + h\alpha_i, \eta + h\sum\nolimits_{j=1}^{i-1} \beta_{ij} k_j, q\right), \tag{2.42}$$

where $s \geq 1$ is the number of *stages* and the coefficients $\alpha, \gamma \in \mathbb{R}^s$, $\beta \in \mathbb{R}^{s \times s}$ are also referred to as *Butcher tableau*.

While the step function defines the actual single step method, the step size h_k can be chosen quite arbitrarily (provided that $\tau_k + h_k \leq t_1$). There are two opposing considerations: on the one hand a small step size leads to a small local error, on the other hand the computational effort for the integration depends linearly on the number of steps taken. Therefore, modern adaptive schemes try to estimate the local error and then choose steps as large as possible while respecting a prescribed local error tolerance. Details for the RK schemes can be found e.g. in [67, 65] and many other related works.

Following a nominal integration one can compute sensitivities complying with the principle of IND by fixing the steps obtained by the adaptive error control and differentiating the scheme, e.g.,

$$\eta^v_{k+1} = \eta^v_k + h_k \frac{\mathrm{d}\Phi}{\mathrm{d}\eta}(\tau_k, \eta_k, q; h_k)\eta^v_k, \ k = 0, \dots, n-1, \quad \eta^v_0 = v, \tag{2.43}$$

for a computation of the forward directional sensitivity $\eta^v_n = \frac{\mathrm{d}\eta_n}{\mathrm{d}x_0} v \approx \frac{\mathrm{d}x(t_1)}{\mathrm{d}x_0} v$ with direction $0 \neq v \in \mathbb{R}^{n_x}$. For the RK schemes one obtains the following step function for the forward

directional sensitivity

$$\begin{aligned}\frac{\mathrm{d}\Phi}{\mathrm{d}\eta}(\tau,\eta,q;h)\eta^v &= \sum_{i=1}^{s}\gamma_i\frac{\mathrm{d}k_i}{\mathrm{d}\eta}\eta^v,\\ \frac{\mathrm{d}k_i}{\mathrm{d}\eta}\eta^v &= \frac{\partial f}{\partial x}(\tau+h\alpha_i,\eta+h\sum\nolimits_{j=1}^{i-1}\beta_{ij}k_j,q)\left(\eta^v+h\sum\nolimits_{j=1}^{i-1}\beta_{ij}\frac{\mathrm{d}k_j}{\mathrm{d}\eta}\eta^v\right).\end{aligned}$$

Because they share common information such as evaluation points of the right-hand side, the forward sensitivity scheme (2.43) is best evaluated simultaneously with the forward scheme (2.41). Clearly, both schemes are consistent: step sizes and evaluation points coincide, and forward directions are propagated by $\frac{\mathrm{d}\Phi}{\mathrm{d}\eta}(\tau,\eta,q;h)\eta^v$ just like state approximations by Φ. Hence the IND principle is satisfied. Analogously one obtains also forward directional sensitivities $\frac{\mathrm{d}\eta_m}{\mathrm{d}q}v$. For the computation of backward directional or *adjoint* sensitivities by RK schemes see, e.g., [184]. For a comprehensive presentation of the *Backward Differentiation Formulas* (BDF) for the integration of stiff systems and DAE systems and the sensitivity generation of arbitrary order by Taylor polynomial arithmetic for these problems see, e.g., [1, 2].

3 Model Predictive Control and Moving Horizon Estimation

In this chapter we motivate the necessity of feedback control and outline some classical feedback approaches which are widely used in modern industrial processes but have severe drawbacks from a theoretical point of view. As state-of-the-art approach, we present Model Predictive Control (MPC) as a numerically tractable and theoretically founded way to compute feedback which is (nearly) optimal. We will briefly touch stability theory for MPC.

Since feedback control approaches in general need full state information, whereas quite often there are only measurements of some states or some functional relations of the states available, the need for state estimation arises, i.e., the reconstruction of the full state information based on the available measurements. Also, if the process behavior depends on parameter values which might accidentally change during runtime, a parameter estimation step is necessary to obtain the new values. As state-of-the-art approach for online state and parameter estimation we present Moving Horizon Estimation (MHE), which is in a way the dual problem to MPC.

3.1 Feedback control

In the last chapter we have considered approaches for the solution of optimal control problems. However, it is in general not advisable to apply the computed optimal control $u^\star(t)$ to the real process in a straightforward manner. The reason for this is that a real process is in general subject to significant disturbances. These reasons can be manifold such as unmodeled aspects of the process behavior, external disturbances from the process environment, inexactly determined parameter values, or intrinsically random behavior of the process. These disturbances may invalidate the computed optimal control because it does not take into account the actual behavior of the process. The optimal control is therefore also called *open-loop* optimal control, see also Figure 3.1.

To take the disturbances into account we have to incorporate knowledge on the behavior of the system by feeding back the actual system state x to the controller. Then we can compute new controls $u^\star(x)$ which depend on the current state and which allow to react to unpredicted system behavior. This approach is known as *closed-loop* or *feedback* control, see also Figure 3.2.

It is quite a common case that only some states or functional dependencies of the states of the process can be measured. Then the state vector x has to be estimated from the measurement vector y. Since the measurements are in general subject to measurement

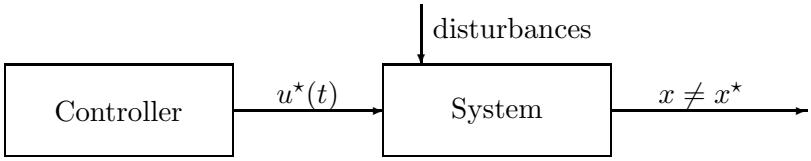


Figure 3.1: Open–loop optimal control ignores actual system behavior. The system output x is in general not the predicted optimal outcome $x^\star$.

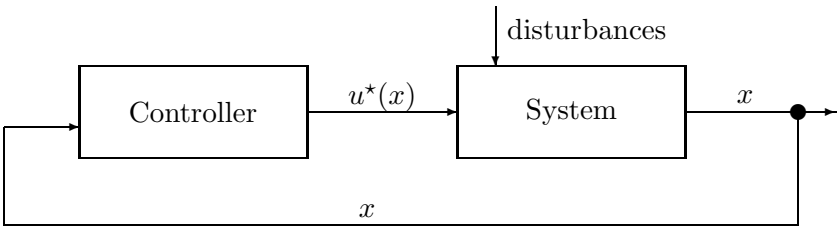


Figure 3.2: Closed–loop optimal control incorporates actual system behavior. The feedback control $u^\star$ depends on the actual system state x.

errors, the estimator will give an estimate $\hat{x}$ for the true state values x. With this estimate, feedback control can be calculated, see also Figure 3.3. Controllers which also take knowledge on the uncertainty of the estimate into account are called *robust* controllers.

3.2 Feedback control approaches

We start by presenting selected control approaches that nowadays still dominate application in industry. The main goal of these early approaches is the stabilization of the process or the regularization to a prescribed set-point.

The proportional-integral-derivative (PID) controller is applied to one-dimensional process variables $x(t) \in \mathbb{R}^1$. It aims to steer $x(t)$ to a set point $\bar{x}(t) \in \mathbb{R}^1$ by setting the manipulated variable $u(t) \in \mathbb{R}^1$ proportional to the error $e(t) \triangleq x(t) - \bar{x}(t)$ and its integral and derivative, namely

$$u(t) = \alpha e(t) + \beta \int_0^t e(\tau)\, \mathrm{d}\tau + \gamma \frac{\mathrm{d}}{\mathrm{d}t} e(t).$$

The proportional term aims to deal with the present error, the integral term weights summed up past errors and the derivative term takes future error into account. PID control is still the most widely used feedback control approach in industry nowadays. For a more detailed discussion of the role of the three parts and the tuning of PID controllers see, e.g.,[9, 128, 177, 176].

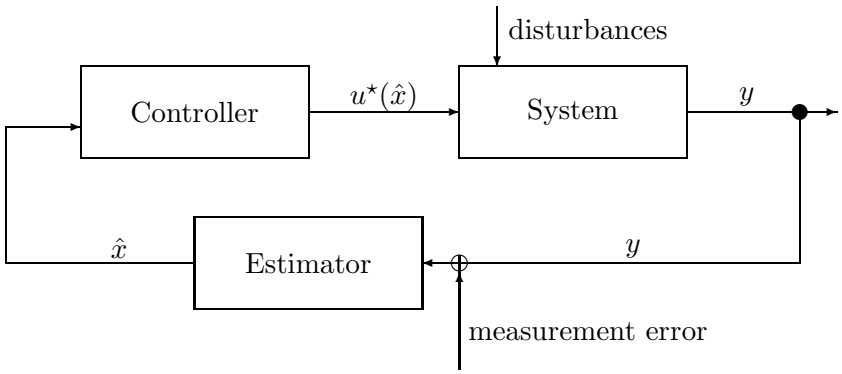


Figure 3.3: Feedback control with state estimation. The controller works with a state estimate $\hat{x}$ obtained from measurements y.

The method of Spectrum Assignment can be considered as a generalization of the PID control to systems with more than one state and control, so called *multiple input multiple output* systems. For the continuous or discrete control system

$$\dot{x}(t) = Ax(t) + Bu(t), \quad \text{or} \quad x_{k+1} = Ax_k + Bu_k,$$

the method aims to find a matrix K so that the feedback $u(t) = Kx(t)$ or $u_k = Kx_k$ stabilizes the feedback systems

$$\dot{x}(t) = (A + BK)\, u(t), \quad \text{or} \quad x_{k+1} = (A + BK)\, x_k.$$

The Pole-Shifting Theorem states that one can obtain an arbitrary spectrum for $A + BK$ by suitably choosing a (unique) matrix K if the matrices A, B satisfy a certain non-degeneracy condition. In particular, one can choose all the eigenvalues to lie in the left half of the complex plane which leads to convergence to $x = 0$, i.e., asymptotic stability of the dynamical system. For the discrete case, one can even choose the eigenvalues to be zero, i.e. $A + BK$ to be nilpotent, which results in bringing x exactly to zero after n_x steps. This is called *deadbeat control*. More details about Spectrum Assignment can be found, e.g., in [169].

The PID and the Spectrum Assignment method allow a whole range of possible feedback laws and do not consider any aspect of optimality for a particular feedback law chosen. This is a major difference to the *linear quadratic regulator* (LQR) which considers the following problem formulation

$$\min_{u(\cdot)} \int_0^T x(t)^\intercal Q\, x(t) + u(t)^\intercal R\, u(t)\, \mathrm{d}t \quad \text{s.t.} \quad \dot{x}(t) = Ax(t) + Bu(t),\ x(0) = x_0,$$

where T can be finite or infinite. There are several approaches to solve this problem exactly, e.g., it is one of the rare problems for which the HJB equation can be solved analytically. For solution approaches and problem modifications see standard textbooks

on optimal control, e.g., [38, 171, 18]. For finite time horizons T one obtains the feedback law as

$$u(t) = K(t)\,x(t) \triangleq -R^{-1}B^\intercal P(t)\,x(t),$$

where $P(t)$ solves the continuous-time *Riccati differential equation*

$$\dot{P}(t) = -A^\intercal P(t) - P(t)A + P(t)BR^{-1}B^\intercal P(t) + Q, \quad P(T) = 0.$$

For infinite time horizons T one obtains the feedback law as

$$u(t) = K\,x(t) \triangleq -R^{-1}B^\intercal P\,x(t),$$

where P solves the *algebraic Riccati equation*

$$0 = A^\intercal P - PA + PBR^{-1}B^\intercal P + Q.$$

The LQR approach can also be used approximately for feedback control of optimal control problems with a general nonlinear objective and governed by nonlinear differential equations. For an open-loop optimal control solution $x^\star(t), u^\star(t)$ of the nonlinear problem, one considers perturbed solutions $x(t), u(t)$ with small perturbations $\Delta x(t) \triangleq x(t) - x^\star(t)$ and $\Delta u(\cdot) \triangleq u(t) - u^\star(t)$. Taking the second variation of the objective and the first variation of the differential equation, one ends up with a linear-quadratic optimization problem in $\Delta x(\cdot), \Delta u(\cdot)$, which then gives an LQR feedback law for $\Delta u(\cdot)$ of the form $\Delta u(t) = K(t)\Delta x(t)$. This approach is called the *neighboring optimal* or *neighboring extremal* control, cf. also [38, 171] for details. Numerical techniques for neighboring optimal control have been developed, e.g., by [141, 111, 112].

The approaches presented so far have several severe drawbacks. For example, most of them apply to linear systems only or require the feedback to keep the trajectory close to a precomputed open-loop solution. Furthermore, none of the presented approaches is able to incorporate inequality constraints which play an important role in real-world problem formulations. To address these issues we employ the state-of-the-art feedback control approach *Model Predictive Control* (MPC) which will be described in some detail in the following.

3.3 The principle of Nonlinear Model Predictive Control

The basic idea of MPC is to generate feedback control by repeatedly solving open-loop optimal control problems on a finite prediction horizon. The MPC approach is depicted in Figure 3.4. The process time is sampled with sampling period δ. We denote the sample index with k. At the sampling times t_0^k the current real process state x_0^k is obtained. Then the optimal control $u^\star(t; x_0^k)$ is determined by solving an optimal control problem on a prediction horizon $[t_0^k, t_0^k + T]$. In this problem, the state prediction is coupled to the real process behavior by setting the initial value of the predicted states to x_0^k. Only the first part $u^\star(t_0^k; x_0^k)$ of the optimal controls is applied to the real process for the sampling time

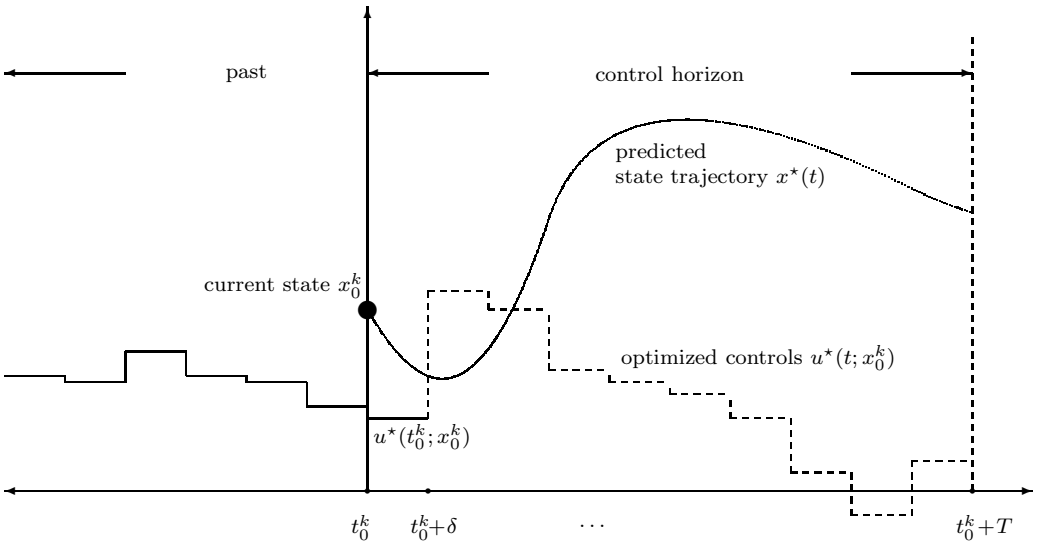


Figure 3.4: Principle of Model Predictive Control. At each sampling time t_0^k an open–loop control problem is solved with the current state x_0^k as initial value and the first part $u^\star(t_0^k; x_0^k)$ of the optimal controls is fed back.

period δ, and at the next sampling time $t_0^{k+1} \triangleq t_0^k + \delta$ a new optimal control problem is solved on the horizon $[t_0^{k+1}, t_0^{k+1} + T]$, i.e. the time horizon is moved forward.

Specifically, the MPC feedback is calculated by solving for each sampling time the following OCP, which is a special case of OCP (2.7)

$$\begin{aligned} \underset{x(\cdot),u(\cdot)}{\text{minimize}} \quad & \int_{t_0^k}^{t_0^k+T} l(x(t),u(t))\ \mathrm{d}t + m(x(T)) \\ \text{subject to} \quad & \dot{x}(t) = f(t,x(t),u(t)), \\ & 0 = x(t_0^k) - x_0^k, \\ & 0 \le c^{\text{path}}(x(t),u(t)), \\ & 0 \le r(x(t_0^k+T)). \end{aligned} \tag{3.1}$$

If the objective function is quadratic and both the dynamic equations and the inequality constraints are affine-linear, we talk about *Linear Model Predictive Control* (LMPC). If the dynamical system or the constraints are nonlinear, or the objective is nonlinear and not quadratic, we talk about *Nonlinear Model Predictive Control* (NMPC).

The possibility of predicting the process behavior by a nonlinear model and pricing controls and the corresponding state trajectories over a certain prediction horizon by a suitably chosen objective function enables the controller to choose much better feedback than, e.g., the PID controller, which takes into account only instantaneous information and does not use information about the process dynamics, or the LQR controller which is limited to linear models and quadratic objectives. Also, due to the flexible choice for the objective function, MPC can be applied to a much wider range of problems such as time-

optimal or economic feedback control. Furthermore, equality and inequality constraints can be added to the problem formulation in a natural way which is not possible in the approaches presented earlier.

Note that the MPC approach as described above is idealized in the sense that in practical applications it is impossible to obtain the real process state x_0^k. Instead, we have measurements of some or all components of the state or of some quantities that depend on the state. These measurements are subject to measurement errors, thus we have to obtain an estimate of the real process state x_0^k. We discuss this topic in detail in Section 3.5.

3.4 Basic theory for Model Predictive Control

We consider basic stability theory for MPC and keep the presentation close to [156]. The MPC stability theory is mostly concerned with feedback control of the following discrete-time dynamical system

$$x_{k+1} = f^{\text{mpc}}(x_k, u_k), \tag{3.2}$$

where we assume that $f^{\text{mpc}}(0,0) = 0$, i.e., the origin is a steady state for the system if zero controls are applied. Aim of the MPC theory is to show that the MPC feedback $u^{\text{mpc}}(x)$ is able to drive the states of the dynamical system to the origin, which means that the origin is asymptotically stable with a region of attraction X for the closed-loop system

$$x_{k+1} = f^{\text{cl}}(x_k) \triangleq f^{\text{mpc}}(x_k, u^{\text{mpc}}(x_k)). \tag{3.3}$$

Recall the following definitions for various kinds of stability and region of attraction:

Definition 3.1 (Stability)**.** *For the dynamical system $x_{k+1} = f^{cl}(x_k)$ denote by $\phi(i;x)$ the solution at time i starting from x. Then the origin is*

- locally stable *if, for each $\epsilon > 0$, there exists a $\delta = \delta(\epsilon) > 0$ such that $\|x\| < \delta$ implies $\|\phi(i;x)\| < \epsilon$ for all $i = 0, 1, \ldots$.*
- *unstable, if it is not locally stable.*
- locally attractive *if there exists $\eta > 0$ such that $\|x\| < \eta$ implies $\|\phi(i;x)\| \to 0$ as $i \to \infty$.*
- globally attractive *if $\|\phi(i;x)\| \to 0$ as $i \to \infty$ for all $x \in \mathbb{R}^{n_x}$.*
- locally asymptotically stable *if it is locally stable and locally attractive.*
- globally asymptotically stable *if it is locally stable and globally attractive.*

A region of attraction *for the asymptotically stable origin is any set of initial values x such that $\|\phi(i;x)\| \to 0$ as $i \to \infty$.*

The problem formulation most often used in theoretical studies of the stability of MPC reads as follows:

$$\begin{aligned} \underset{u}{\text{minimize}} \quad & \sum_{k=0}^{N-1} \ell(x_k, u_k) + V_f(x_N) \\ \text{subject to} \quad & x_{k+1} = f^{\text{mpc}}(x_k, u_k),\ k = 0, \dots, N-1 \\ & x_0 = x, \\ & c^{\text{path}}(x_k, u_k) \in \mathbb{Y},\ k = 0, \dots, N-1, \\ & x_N \in \mathbb{X}_f, \end{aligned} \tag{3.4}$$

with the optimization variables being the control sequence $u = (u_0, \dots, u_{N-1})$. For the solution $u^\star$ of (3.4), the MPC feedback is then defined as $u^{\text{mpc}}(x) \triangleq u_0^\star$. Problem (3.4) looks very similar to Problem (2.19), which arises from the Direct Multiple Shooting discretization of a continuous-time OCP, with the specific choice of a piecewise constant control discretization. However, there is an important difference: in MPC theory, we do not consider the states x_k as degrees of freedom but eliminate them by using the difference equation (3.2). We denote with $\phi(k; x, u)$ the solution of the difference equation at time k starting from x and applying the control sequence $u_0, \dots, u_{k-1}$. With the definitions

$$V_N(x, u) \triangleq \sum_{k=0}^{N-1} \ell(\phi(k; x, u), u_k) + V_f(\phi(N; x, u)) \tag{3.5}$$

and

$$\mathcal{U}_N(x) \triangleq \left\{ u | c^{\text{path}}(\phi(k; x, u), u_k) \in \mathbb{Y} \text{ for } k = 0, \dots, N-1, \quad \phi(N; x, u) \in \mathbb{X}_f \right\} \tag{3.6}$$

we can write Problem (3.4) more compactly as

$$V_N^0(x) \triangleq \underset{u}{\text{minimize}} \left\{ V_N(x, u) | u \in \mathcal{U}_N(x) \right\}. \tag{3.7}$$

The function $V_N^0(x)$ defined in (3.7) is called the *value function* and plays a major role in MPC theory. With the definition of the following helpful class of functions we can then write down a central result of stability theory for dynamical systems.

Definition 3.2 ($\mathcal{K}$ function)**.** *A function $\sigma : \mathbb{R}_+ \mapsto \mathbb{R}_+$*

- *belongs to class $\mathcal{K}$ if it is continuous, zero at zero, and strictly increasing.*
- *belongs to class $\mathcal{K}_\infty$ if it belongs to class $\mathcal{K}$ and is unbounded ($\sigma(s) \to \infty$ as $s \to \infty$).*

A function $\gamma : \mathbb{R} \mapsto \mathbb{R}_+$ belongs to class $\mathcal{PD}$ (is positive definite*) if it is continuous and positive everywhere except at the origin.*

Theorem 3.1. *The origin is asymptotically stable with a region of attraction X for the system $x_{k+1} = f^{cl}(x_k)$ if there exist:*

- *a function $V : \mathbb{R}^{n_x} \mapsto \mathbb{R}_+$,*

- *a* positive invariant *set* X*, i.e.,* $x \in X \Rightarrow f^{cl}(x) \in X$*,*
- *two* $\mathcal{K}_\infty$ *functions* $\alpha_1(\cdot)$ *and* $\alpha_2(\cdot)$ *and a PD function* $\alpha_3(\cdot)$ *satisfying*

$$V(x) \geq \alpha_1(\|x\|) \tag{3.8a}$$
$$V(x) \leq \alpha_2(\|x\|) \tag{3.8b}$$
$$V(f^{cl}(x)) \leq V(x) - \alpha_3(\|x\|) \tag{3.8c}$$

for all $x \in X$*.*

A function $V : \mathbb{R}^{n_x} \mapsto \mathbb{R}_+$ satisfying the conditions (3.8) is called a *Lyapunov function*. In general, the difficulty of this elegant and powerful approach to stability theory is that there is no generic way to construct Lyapunov functions for a given dynamical system. For MPC however, the value function $V_N^0(x)$ is a reasonable candidate, which is also motivated by analogy to the stability theory for the infinite-horizon regulator. The MPC stability theory establishes criteria for $\ell(\cdot), V_f(\cdot)$, and $\mathbb{X}_f$ under which the conditions (3.8) hold for $V_N^0(\cdot)$ and the dynamical system $x_{k+1} = f(x_k, u^{\text{mpc}}(x_k))$. For a detailed derivation see [156].

3.5 Online state and parameter estimation

As we have seen in the last section, the feedback calculation is linked to the real process behavior via the current system state x, which is obtained by taking measurements η from the process. Quite often, we can not measure the full system state x but only some components x_i or some functional dependence $h^{\text{est}}(x)$ of the states. To close the control loop we therefore have to obtain an estimate of the full state from the available measurements.

To estimate the state we will employ a model of both the system dynamics and the measurement process and seek to find an estimate $\hat{x}$ of the true states x that best explains the measurement data η. In general, we will not be able to get the exact system state, i.e., $\hat{x} \neq x$. This is due to the errors which enter the estimation process, namely errors from the modeling of the real process and errors from the measurements.

We will start by considering the straight-forward deterministic approach of *weighted least-squares estimation* taking into account all available measurements from all measurement times and, if available, a-priori information on the initial state. It turns out that this approach also has a statistical interpretation if the measurement error is assumed to be a statistical quantity. While being theoretically desirable, the weighted least-squares estimation approach becomes computationally intractable with an increasing amount of measurement data available. Only for the case of linear system and measurement models, the estimator can be evaluated exactly by recursion. The arising recursion equations are known as the *Kalman filter* or *linear Kalman filter* (LKF) equations. Kalman filtering allows a straightforward extension to nonlinear system and measurement models by local linearization, yielding the *extended Kalman filter* (EKF). Due to its conceptual simplicity the EKF is a popular and widespread state estimator in practical applications. However,

if the linearization is not well suited to describe the nonlinear process then EKF yields bad or even unphysical state estimates. Thus, we turn back to the least-squares estimation approach to motivate and derive the *Moving Horizon Estimation*, which is our preferred state-of-the-art approach for state estimation. Finally, we discuss the extension to combined state and parameter estimation as well as the issue of properly choosing the arrival cost term.

System dynamics model

We model the system states x_k in the k-th sample by

$$x_k = f^{\text{est}}(x_{k-1}, u_{k-1}) + w_{k-1}. \tag{3.9a}$$

The propagation function f^{est} describes the aspects of the process that are explicitly and deterministically modeled. By introducing the additive *process noise* term w_{k-1} we can account for deviations between the mathematical process model $f^{\text{est}}(\cdot)$ and the actual process state, e.g., model-process mismatches or random disturbances.

Measurement model

We model the measurements η_k in the k-th sample by

$$\eta_k = h^{\text{est}}(x_k) + \nu_k, \tag{3.9b}$$

i.e., the measurement consists of a functional dependence h of the system state x_k and a disturbance ν_k which models the measurement error. Quite typically, a subset of the states is measured; in this case we have $h^{\text{est}}(x_k) = H^{\text{est}} x_k$, where the rows of the matrix H^{est} consist of unit vectors corresponding to the measured state components.

State estimation problem

The state estimation problem for sample $k \geq 0$ now can be formulated as follows (cf. [156]): given measurement data $\eta_0, \ldots, \eta_k$ and possibly a-priori state information $\bar{x}_0$, find an initial state x_0 and sequences of process noise $w_0, \ldots, w_{k-1}$ and measurement errors $\nu_0, \ldots, \nu_k$ such that the measurements η_j satisfy (3.9b) where the x_j are generated by (3.9a) starting from x_0. The controls $u_0, \ldots, u_{k-1}$ are assumed to be known. This is reasonable since we use the measurement data up to η_k, which has been obtained from the process using the controls $u_0, \ldots, u_{k-1}$.

Since this problem is over-parametrized and thus admits infinitely many solutions, a reasonable approach is to seek the unique solution which minimizes the sum of the squared norms of the measurement errors and a quadratic term which allows to incorporate the a-priori information. This well-known approach is called the *least-squares estimation*. It is also known as *full information estimation* because the whole available measurement information is used. Note that it is purely deterministic and makes no statistical assumption on the error.

$$\begin{aligned} \underset{x_0,\nu_k}{\text{minimize}} \quad & \|x_0 - \bar{x}_0\|^2_{Q_0} + \sum_{j=0}^{k-1} \|w_j\|^2_Q + \sum_{j=0}^{k} \|\nu_j\|^2_R \\ \text{subject to} \quad & x_{j+1} = f^{\text{est}}(x_j, u_j) + w_j,\ j = 0, \dots, k-1, \\ & \nu_j = \eta_j - h^{\text{est}}(x_j),\ j = 0, \dots, k. \end{aligned} \tag{3.10}$$

We use the notation $\|v\|^2_Q \triangleq v^\intercal Q v$, where Q is a symmetric positive-definite matrix. In the deterministic approach, the most suitable choice for the scaling matrices Q_0, Q, and R is not obvious. However, if the errors are suitably distributed statistical quantities then Problem 3.10 also has a probabilistic interpretation for the right choice of scaling matrices. To see this, we formulate the following assumptions.

Assumption 3.1 (Statistical distribution of error quantities)**.**

- *The initial state x_0 is normally distributed with mean $\bar{x}_0$ and covariance matrix P.*
- *The state disturbances w_j are normally distributed with zero mean and covariance matrix W.*
- *The measurement errors ν_j are normally distributed with zero mean and covariance matrix V.*
- *The initial state, state disturbances, and measurement errors are mutually statistically independent.*

Under Assumption 3.1, the *a-posteriori* density function $p(x_0, \dots, x_k | \eta_0, \dots, \eta_k)$ of the states $x_0, \dots, x_k$ for observed measurements $(\eta_0, \dots, \eta_k)$ has been derived in [47], where the notation $p(x|y)$ denotes the conditional probability density of x given y as usual. We obtain

$$\begin{aligned} & p(x_0, \dots, x_k | \eta_0, \dots, \eta_k) \propto \\ & \qquad p(x_0) \prod_{j=1}^{k} p(x_j | x_{j-1}) \prod_{j=0}^{k} p(\eta_j - h^{\text{est}}(x_j)) \propto \\ & \exp\left(-\frac{1}{2}\|x_0 - \bar{x}_0\|^2_{P^{-1}}\right) \prod_{j=0}^{k-1} \exp\left(-\frac{1}{2}\|w_j\|^2_{W^{-1}}\right) \prod_{j=0}^{k} \exp\left(-\frac{1}{2}\|\nu_j\|^2_{V^{-1}}\right) \end{aligned} \tag{3.11}$$

subject to (3.9a), (3.9b), where $\propto$ denotes a proportional relation between the left-hand term and the right-hand term, i.e., the both terms differ only by a multiplication with a constant.

The mode, i.e., the value of $x = (x_0, \dots, x_k)$ where the a-posteriori probability density distribution attains its maximum, is called the maximum a-posteriori probability (MAP) estimator. It is similar to the Maximum-Likelihood (ML) estimator and for a non-informative prior, i.e., if all possible values of x are equally probable, the both estimators are equivalent.

By taking the logarithm in (3.11) one can easily see that the maximum modal value can be computed by solving Problem 3.10 for the choice of weighting matrices $Q_0 = P^{-1}$, $Q = W^{-1}$, and $R = V^{-1}$.

The least-squares estimation approach is theoretically desirable, and powerful algorithms for the solution of the least-squares problems 3.10 are available. However, in the online optimization context, the growing number of measurement data makes Problem 3.10 increasingly large and therefore ultimately prohibitively expensive to solve. There is an important exception to this for estimations problems with linear dynamical model and linear measurement model. In this case, the exact solution of Problem 3.10 can be computed recursively from the solution of the smaller, previous problem. This approach is called the *Kalman filter* and we give a short synopsis in the following section.

3.6 Kalman filters

For a linear dynamical model and a linear measurement model, the linear Kalman filter equations (cf. [101, 171]) recursively compute for each sample mean and covariance of the state probability distribution, taking into account both the propagation through the dynamical system and the new measurement information.

This approach can be extended straightforwardly to nonlinear dynamical models and measurement models by linearization of the state and measurement model, yielding the *extended Kalman filter*. However, the resulting estimates are not solutions of Problem 3.10 in the nonlinear case and thus the state estimate is not a MAP estimator.

3.6.1 Linear Kalman filter

For the dynamical and measurement model

$$x_{k+1} = Ax_k + Bu_k + w_k \tag{3.12a}$$
$$y_k = Cx_k + v_k \tag{3.12b}$$

let W be the covariance matrix of the state noise w_k and V be the covariance matrix of the measurement noise v_k. Starting with an initial guess $\hat{x}_{0|-1} = x_0$ for the state and $\hat{P}_{0|-1} = P$ for the covariance, we iterate for $k = 0, 1, 2, \ldots$

$$L_k = \hat{P}_{k|k-1}C^\intercal(C\hat{P}_{k|k-1}C^\intercal + V)^{-1}, \tag{3.13a}$$
$$P_{k|k} = \hat{P}_{k|k-1} - L_kC\hat{P}_{k|k-1}, \tag{3.13b}$$
$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + L_k(y_k - C\hat{x}_{k|k-1}), \tag{3.13c}$$

which yields the state estimate $\hat{x}_{k|k}$ and covariance estimate $P_{k|k}$ for sample k. We then prepare the next sample by propagating the state and covariance estimate through the dynamical system by

$$\hat{x}_{k+1|k} = A\hat{x}_{k|k} + Bu_k, \tag{3.14a}$$
$$\hat{P}_{k+1|k} = AP_{k|k}A^\intercal + W. \tag{3.14b}$$

The initial state and covariance guess allows to include a-priori information about the state. If no information is available, a non-informative prior can be chosen by setting, e.g., $x_0 = 0$ and P to a diagonal matrix with very large entries.

3.6.2 Extended Kalman filter

The algorithmic simplicity of the linear Kalman filter makes it highly appealing to extend the approach to nonlinear dynamical and measurement models. In the update step where we incorporate the current measurement information we use a linearization of the measurement model to compute the gain matrix L_k and the covariance estimate $P_{k|k}$ and the nonlinear model to compute the state estimate $\hat{x}_{k|k}$. Analogously, in the propagation step we use the nonlinear model to propagate the state estimate and a linearization to propagate the covariance estimate. The resulting equations closely resemble Equations (3.13) and (3.14)(cf. [171]).

For the dynamical and measurement model

$$\begin{aligned} x_{k+1} &= f^{\text{est}}(x_k, u_k) + w_k, && \text{(3.15a)} \\ y_k &= h^{\text{est}}(x_k) + v_k, && \text{(3.15b)} \end{aligned}$$

let again W be the covariance matrix of the state noise w_k and V be the covariance matrix of the measurement noise v_k. Furthermore, let $A_k = \frac{\mathrm{d}f^{\text{est}}}{\mathrm{d}x}(\hat{x}_{k|k}, u_k)$ and $C_k = \frac{\mathrm{d}h^{\text{est}}}{\mathrm{d}x}(\hat{x}_{k|k-1})$. Starting with an initial guess $\hat{x}_{0|-1} = x_0$ for the state and $\hat{P}_{0|-1} = P$ for the covariance, we iterate for $k = 0, 1, 2, \ldots$

$$\begin{aligned} L_k &= \hat{P}_{k|k-1} C_k^\intercal (C_k \hat{P}_{k|k-1} C_k^\intercal + V)^{-1} && \text{(3.16a)} \\ P_{k|k} &= \hat{P}_{k|k-1} - L_k C_k \hat{P}_{k|k-1} && \text{(3.16b)} \\ \hat{x}_{k|k} &= \hat{x}_{k|k-1} + L_k (y_k - h^{\text{est}}(\hat{x}_{k|k-1})) && \text{(3.16c)} \end{aligned}$$

which yields the state estimate $\hat{x}_{k|k}$ and covariance estimate $P_{k|k}$ for sample k and then prepare the next sample by propagating the state and covariance estimate through the dynamical system by

$$\begin{aligned} \hat{x}_{k+1|k} &= f^{\text{est}}(\hat{x}_{k|k}, u_k) && \text{(3.17a)} \\ \hat{P}_{k+1|k} &= A_k P_{k|k} A_k^\intercal + W && \text{(3.17b)} \end{aligned}$$

The EKF provides an algorithmically simple and computationally cheap approach for state estimation. However, there are several important drawbacks, namely, the linearizations may be bad approximations of the real nonlinear model functions and constraints on the state cannot be incorporated. For a study of the practical implementation problems of the EKF see, e.g., [182]. The problem of bad linearizations can be partly addressed by using the *unscented Kalman filter* (UKF) which makes use of function evaluations rather than linearizations to propagate the covariances, see for example [98, 97] for details. However, the problem of incorporating constraints remains, and both EKF and UKF can even produce unphysical state estimates, see [156].

3.7 Moving Horizon Estimation

The approach described in the following produces estimates that approximate the estimates generated by the full information estimator better. It also allows to easily incorporate constraints to avoid unphysical or otherwise undesired estimates. The basic idea is to come back to Problem 3.10, but instead of taking into account all measurement information we only consider the N most recent measurements. This keeps the computational effort fixed and independent from the sampling time and gives rise to the name *Moving Horizon Estimation* (MHE) (cf. [156]). Older measurement information that lies outside of the estimation horizon is discarded or can be approximately incorporated by using a suitably chosen initial cost term which is called *arrival cost*.

For sampling times $T \leq N$ the horizon is not yet filled completely and so we solve the full least squares estimation problem with T measurements. Since we have an additional measurement in each following sample, we call this approach for the MHE initialization *growing horizon initialization*.

For sampling times $T > N$ the MHE problem reads

$$\begin{aligned} \underset{x_{T-N},w}{\text{minimize}} \quad & \|x_{T-N} - \bar{x}_{T-N}\|^2_{Q_{T-N}} + \sum_{j=T-N}^{T-1} \|w_j\|^2_Q + \sum_{j=T-N}^{T} \|\eta_j - h^{\text{est}}(x_j)\|^2_R \\ \text{subject to} \quad & x_{j+1} = f^{\text{est}}(x_j, u_j) + w_j,\ j = T-N, \ldots, T-1. \end{aligned} \tag{3.18}$$

In Problem 3.18 the tuning parameters $\bar{x}_{T-N}$ and Q_{T-N} are used to incorporate approximately the information from the discarded earlier measurements. Dropping the arrival cost term, i.e. setting $Q_{T-N} = 0$, can be interpreted statistically as having or using no a-priori information of the value of x_{T-N}, i.e. the a-priori estimate $\bar{x}_{T-N}$ has an infinite variance. In practice, discarding the a-priori information could be sensible in some cases, e.g., if a change or disturbance in the system dynamics invalidates old measurement information. However, in general it is quite desirable to use the available information to the best possible extent and thus a proper choice of $\bar{x}_{T-N}$ and Q_{T-N} is important for the performance of the estimator. We briefly discuss a suitable choice later in this section.

From a numerical point of view, Problem 3.18 is an equality-constrained nonlinear least-squares problem which allows a straightforward extension to a general constrained nonlinear least-squares problem by adding inequality constraints, in particular bounds on the variables. This allows to avoid unphysical intermediate variables, and helps guiding the algorithm by cutting off undesired parts of the search space during the iterative solution process. However, it should be noted that active inequality constraints in the solution may invalidate the statistical interpretation of the estimate as MAP estimator.

For the moving horizon estimator, *stability* results have been established. Stability means, that for the class of bounded noise sequences that converge to zero the estimated state will converge to the true state, depending on certain properties of the system. Even for *noninformative prior*, i.e. $Q_{T-N} = 0$, stability can be guaranteed. However, a careful choice of the arrival cost term improves the controller performance significantly (cf. [156]).

In the following, we give numerical details for the choice and update of the arrival cost

term for the MHE implementation used in this work. More details and further reading can be found in [59, 114, 115]. Going from sampling time T to $T+1$ the ideal prior weighting for the new problem on $[t_{T-N+1}, t_{T+1}]$ can be derived by dynamic programming arguments and reads

$$\begin{aligned} F(x_{T-N+1}) = \min_{x_{T-N}} \Big\{ \Big\| x_{T-N} - \bar{x}_{T-N} \Big\|^2_{Q_{T-N}} + \|w_{T-N}\|^2_Q + \|\eta_{T-N} - h^{\text{est}}(x_{T-N})\|^2_R \Big\} \\ \text{s. t. } w_{T-N} = x_{T-N+1} - f^{\text{est}}(x_{T-N}, u_{T-N}). \end{aligned} \tag{3.19}$$

Problem (3.19) is nonlinear and thus writing down a closed-form solution is in general impossible. However, linearization of the nonlinear functions $f^{\text{est}}(\cdot)$ and $h^{\text{est}}(\cdot)$ in the best available estimate $\hat{x}_{T-N}$ of state x_{T-N} at sampling time T yields the analytically solvable approximation in the variables (x_{T-N}, x_{T-N+1})

$$\min_{x_{T-N}} \left\| \begin{matrix} Q_{T-N}^{\frac{1}{2}} \left(x_{T-N} - \bar{x}_{T-N} \right) \\ R^{\frac{1}{2}} (\eta_{T-N} - \bar{h} - H_x \, x_{T-N}) \\ Q^{\frac{1}{2}} \left(x_{T-N+1} - \bar{x} - X_x \, x_{T-N} \right) \end{matrix} \right\|_2^2 \triangleq \left\| A \begin{pmatrix} x_{T-N} \\ x_{T-N+1} \end{pmatrix} + \begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} \right\|_2^2, \tag{3.20}$$

where we use the shorthands

$$\begin{aligned} H_x &= \frac{\mathrm{d}h^{\text{est}}}{\mathrm{d}x}(\hat{x}_{T-N}) \\ X_x &= \frac{\partial f^{\text{est}}}{\partial x}(\hat{x}_{T-N}, u_{T-N}) \\ \bar{h} &= h^{\text{est}}(\hat{x}_{T-N}) - H_x \, \hat{x}_{T-N} \\ \bar{x} &= f^{\text{est}}(\hat{x}_{T-N}, u_{T-N}) - X_x \, \hat{x}_{T-N}. \end{aligned}$$

QR decomposition of

$$A = \begin{pmatrix} Q_{T-N}^{\frac{1}{2}} & 0 \\ -R^{\frac{1}{2}} H_x & 0 \\ -Q^{\frac{1}{2}} X_x & Q^{\frac{1}{2}} \end{pmatrix} \overset{\text{QR}}{=} \bar{Q} \begin{pmatrix} \bar{R}_1 & \bar{R}_{12} \\ 0 & \bar{R}_2 \\ 0 & 0 \end{pmatrix}, \quad \text{and} \begin{pmatrix} r_1 \\ r_2 \\ r_3 \end{pmatrix} \triangleq \bar{Q}^\intercal \begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix}$$

yields the following problem equivalent to (3.20)

$$\min_{x_{T-N}} \left\| \begin{pmatrix} r_1 \\ r_2 \\ r_3 \end{pmatrix} + \begin{pmatrix} \bar{R}_1 & \bar{R}_{12} \\ 0 & \bar{R}_2 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} x_{T-N} \\ x_{T-N+1} \end{pmatrix} \right\|_2^2.$$

This problem has the analytical solution

$$\tilde{F}(x_{T-N+1}) = \|r_3\|_2^2 + \left\| r_2 + \bar{R}_2 \, x_{T-N+1} \right\|_2^2$$

and so we obtain as new prior weighting

$$Q_{T-N+1} = \bar{R}_2^\intercal \bar{R}_2, \quad \bar{x}_{T-N+1} = -\bar{R}_2^{-1} r_2. \tag{3.21}$$

From (3.21) and (3.20) we can see that $Q_{T-N+1} = \bar{R}_2^\intercal \bar{R}_2 \preceq \bar{R}_{12}^\intercal \bar{R}_{12} + \bar{R}_2^\intercal \bar{R}_2 = Q$, and thus for all vectors v the relation $\|v\|^2_{Q_{T-N+1}} \leq \|v\|^2_Q$ holds. Therefore, the accuracy of the knowledge about the a-priori state information $\bar{x}_{T-N+1}$ is naturally bounded by the covariance Q of the state noise. This means that the influence of accumulated information from earlier samples cannot grow over time to dominate the whole estimation problem and thus the estimator can quickly react to sudden changes and disturbances in the process behavior.

Online estimation of parameters Quite often, it is necessary or desirable to estimate not only the current state but also the value of parameters of the process which may change during runtime, e.g., due to external disturbances or malfunctions. We account for those parameters by including them explicitly in the dynamic model (3.9a), and in the measurement (3.9b), i.e., $x_k = f^{\text{est}}(x_{k-1}, u_{k-1}, p) + w_{k-1}$ and $\eta_k = h^{\text{est}}(x_k, p) + \nu_k$. The Kalman filter variants cannot directly handle parameter estimation, however a transformation of parameters to states analogously to the transformation described in (2.11) allows to estimate parameters along with states by Kalman filter variants. The MHE can directly handle combined state and parameter estimation, which is numerically more efficient than using the transformation of parameters to states. For details about the combined state and parameter estimation with MHE see, e.g., [115].

Part II

Efficient numerical methods for Model Predictive Control

4 Real-Time Iteration schemes for NMPC and MHE

The state-of-the-art real-time optimization methods NMPC and MHE solve in theory an optimal control problem in each sample to calculate optimal feedback controls and state and parameter estimates, respectively. In Chapter 1, we discussed the Direct Multiple Shooting method as a reliable and efficient approach to solve these optimal control problems.

The immediate and naive approach is to simply apply in each sample some optimal control solver in a black box approach to the arising optimal control problems. However, while this approach is used, e.g., quite frequently in chemical engineering where process dynamics are comparably slow, it is in general computationally wasteful and puts limits on the size of models that can be used in such a framework. If the dynamics are fast-changing, e.g., as in many mechanical processes, solving the optimal control problem can be prohibitive even on modern computers and hardware. The feedback controls obtained in that way will in general be outdated or even infeasible and a quick reaction to disturbances is not possible.

To obtain efficient algorithms for real-time optimization, we have to make use of the fact that the consecutive optimal control problems are closely related and thus information gained from the preceding problem can be used advantageously in the computations for the current problem. In particular, we will present the ideas of *tangential predictors* (TP) and *initial value embedding* (IVE). These ideas motivate the *Real-Time Iteration scheme* (RTI), which is an efficient algorithmic approach to real-time optimization and estimation. We will state numerical details and well-known stability results.

Note that in this chapter, f and g denote different mathematical objects than in the previous chapters.

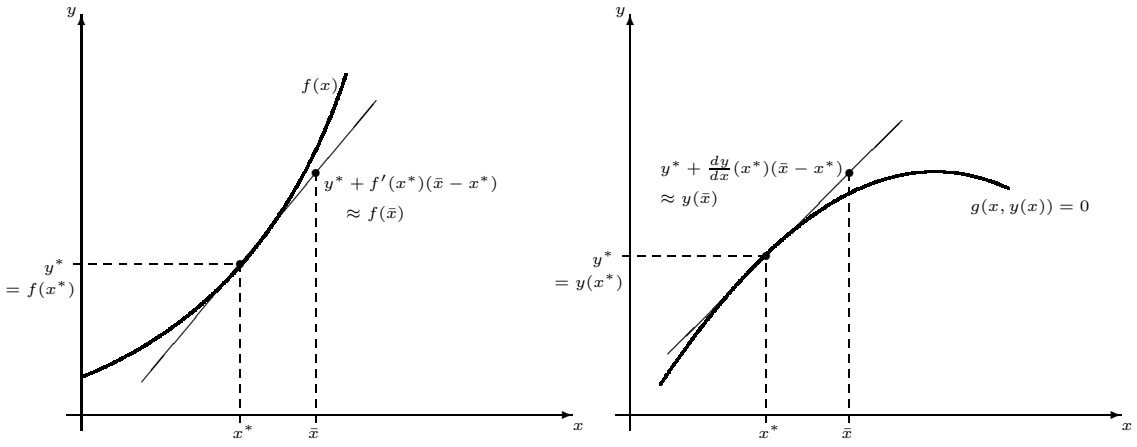
4.1 Tangential predictors and initial value embedding

We start by explaining the idea of tangential predictors in a more general framework and then see how the concept is applied in the NMPC context. Consider a function $f : \mathbb{R}^n \mapsto \mathbb{R}^m$ of a variable $x \in \mathbb{R}^n$ and let x^* be some fixed variable value. If f is sufficiently smooth, Taylor's theorem states that

$$f(x) = f(x^*) + f'(x^*)(x - x^*) + O(\|x - x^*\|^2),$$

see any textbook on analysis, e.g., [159, 80]. That means, we obtain a good, first-order accurate approximation of $f(x)$ for a whole range of x close to x^* by using the *tangential*

predictor $t(x) = f(x^*) + f'(x^*)(x - x^*)$. Now think of f as a function which is highly expensive to evaluate and consider the task that you have to quickly provide good approximations of $f(x)$ as soon as values x are given. Since $f(x^*)$ and $f'(x^*)$ can be computed in advance, evaluating the tangential predictor $t(x)$ solves the task highly efficiently, by applying simply a matrix-vector multiply-add, see Figure 4.1(a).



(a) TP for explicit function

(b) TP for implicit function

Figure 4.1: Principle of tangential predictors (TP)

Next consider a function $g : \mathbb{R}^{n \times m} \mapsto \mathbb{R}^m$ and the equality constraint $g(x, y) = 0$. The implicit function theorem states, that for values (x^*, y^*) satisfying $g(x^*, y^*) = 0$ and under some technical assumptions this constraint defines a function $y : \mathbb{R}^n \mapsto \mathbb{R}^m$, which then satisfies $g(x, y(x)) = 0$ in a neighborhood of (x^*, y^*) and $y(x^*) = y^*$, see also [159, 80]. Again, consider the case that obtaining y from g is a computationally highly expensive task and that we quickly need good approximations of $y(x)$ for given values x. Like in the explicit case, these approximations can be obtained by evaluating the tangential predictor $t(x) = y(x^*) + \frac{dy}{dx}(x^*)(x - x^*)$. What makes this case more difficult is that the derivative $\frac{dy}{dx}(x^*)$ of the implicit function y is needed, which is also provided by the implicit function theorem as

$$\frac{dy}{dx}(x^*) = -\frac{\partial g}{\partial y}(x^*, y^*)^{-1} \frac{\partial g}{\partial x}(x^*, y^*),$$

where we assume that $\frac{\partial g}{\partial y}(x^*, y^*)$ is invertible. Again, we can compute the expensive parts $y(x^*)$ and $\frac{dy}{dx}(x^*)$ in advance and thus obtain the approximations of $y(x)$ quickly and efficiently by a matrix-vector multiply-add, see Figure 4.1(b).

Let us now move to how the concept of tangential predictors can be used for efficient numerics for NMPC. The application of the Direct Multiple Shooting method to the

NMPC optimal control problem as stated in (3.1) yields in each sample the NLP

$$\begin{aligned}
\underset{s,q}{\text{minimize}} \quad & \sum_{i=0}^{N-1} L\left(s_i, q_i\right) + m\left(s_N\right) && && \text{(4.1a)}\\
\text{subject to} \quad & 0 = s_0 - x_0, && && \text{(4.1b)}\\
& 0 = s_{i+1} - x_i(t_{i+1}; s_i, q_i), && 0 \le i < N, && \text{(4.1c)}\\
& 0 \le c^{\text{path}}(s_i, q_i), && 0 \le i \le N-1, && \text{(4.1d)}\\
& 0 \le r(s_N), && && \text{(4.1e)}
\end{aligned}$$

with s_i the Multiple Shooting state variables and q_i the Multiple Shooting control variables. We can see that these NLPs differ from sample to sample only by the value of the parameter x_0 in the constraint (4.1b). NLP (4.1) is thus an example for *parametric programming*, and we can write it more generically as

$$\begin{aligned}
\underset{w}{\text{minimize}} \quad & b(w) && \text{(4.2a)}\\
\text{subject to} \quad & 0 = c(w, x_0), && \text{(4.2b)}\\
& 0 \le d(w), && \text{(4.2c)}
\end{aligned}$$

where we have combined the state and control variables in w. Consider for a moment that there are no inequalities $d(w)$. Then, the first-order KKT conditions, which define the critical points of NLP (4.2), are

$$g(w, \lambda, x_0) \triangleq \begin{pmatrix} \nabla b(w) - \nabla c(w, x_0)\lambda \\ c(w, x_0) \end{pmatrix} = 0, \tag{4.3}$$

with multipliers λ of the equality constraints c. Thus, the optimality conditions g implicitly define the functions (and primal-dual solution candidates) $w(x_0)$ and $\lambda(x_0)$, and we can apply the tangential predictor for implicitly defined functions as discussed above. That means, with a solution w^*, λ^* for an initial value x_0^* we can obtain computationally inexpensive approximations of primal-dual pairs $w(x_0), \lambda(x_0)$ for new initial values x_0 in a neighborhood of x_0^*.

If inequality constraints $d(w)$ are present, the optimality conditions are no longer a set of equations only. However, in [51] it is shown that we can still compute tangential predictors, which are now not given by the solution of a linear equation but by the solution of a QP. The situation is depicted in Figure 4.2. Close to x_0^*, where small changes of x_0 do not lead to a change in the active set of the solution, the tangential predictor works like discussed above in the implicit case. However, even if the new initial value x_0 leads to a change in the active set of the solution, the QP still provides a good tangential predictor because the QP takes active set changes into account by incorporating the linearized inequality constraints.

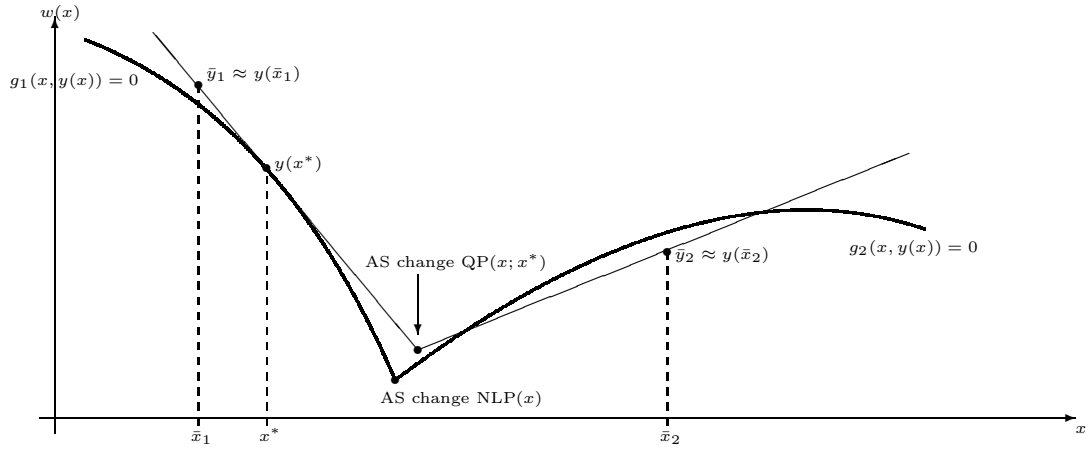


Figure 4.2: Generalized tangential predictor (TP) with active set (AS) change. The TP is given by the parametric quadratic program QP$(x; x^*)$ set up in x^* with parameter x. It approximately detects AS changes in the optimal solution of the parametric nonlinear program NLP(x) with parameter x.

Moreover, if problem (4.2) is augmented to

$$\begin{aligned} \underset{t,w}{\text{minimize}} \quad & b(w) && \text{(4.4a)} \\ \text{subject to} \quad & 0 = t - x_0, && \text{(4.4b)} \\ & 0 = c(w, t), && \text{(4.4c)} \\ & 0 \leq d(w), && \text{(4.4d)} \end{aligned}$$

and $t^*, w^*, \lambda^*, \mu^*$ is a primal-dual solution of the KKT conditions for the initial value $\bar{x_0}$, the QP which gives us the tangential predictor for a new initial value x_0 has the specific form

$$\begin{aligned} \underset{\Delta t, \Delta w}{\text{minimize}} \quad & \frac{1}{2} \begin{pmatrix} \Delta w \\ \Delta t \end{pmatrix}^\mathsf{T} \mathcal{H}(w^*, t^*, \lambda^*, \mu^*) \begin{pmatrix} \Delta w \\ \Delta t \end{pmatrix} + \nabla_w b(w^*)^\mathsf{T} \Delta w && \text{(4.5a)} \\ \text{subject to} \quad & 0 = \Delta t + (t^* - x_0), && \text{(4.5b)} \\ & 0 = \frac{\partial c}{\partial t}(w^*, t^*) \Delta t + \frac{\partial c}{\partial w}(w^*, t^*) \Delta w + c(w^*, t^*), && \text{(4.5c)} \\ & 0 \leq \frac{\mathrm{d}d}{\mathrm{d}w}(w^*) \Delta w + d(w^*), && \text{(4.5d)} \end{aligned}$$

where $\mathcal{H}$ is the Hessian of the Lagrange function with respect to w and t. The tangential predictor is now simply calculated by updating the solution $w^*, t^*, \lambda^*, \mu^*$ with the computed step in the primal and dual variables. It should be noted that the tangential predictor becomes approximate if QP (4.5) is not initialized in the solution $w^*, t^*, \lambda^*, \mu^*$, i.e., if we use an approximation of $\mathcal{H}$ or the constraint Jacobians or if the gradient or constraint residuals are not evaluated exactly.

The augmentation of problem (4.2) by the constraint $0 = t - x_0$ is called the *initial value embedding* (IVE) and the working principle is that introducing the parameter as an

additional NLP variable by this trivial equation adds derivative information with respect to the parameter to the QP. It should be noted that the NLPs (4.1) from a Direct Multiple Shooting discretization of the NMPC feedback generating control problems always exhibit this structure naturally due to the initial value constraint (4.1b) and so no additional variable t has to be introduced, i.e., QP (4.5) is replaced by

$$\begin{aligned}
\underset{\Delta w}{\text{minimize}} \quad & \frac{1}{2}\Delta w^\intercal \mathcal{H}(w^*, \lambda^*, \mu^*)\Delta w + \nabla_w b(w^*)^\intercal \Delta w && \text{(4.6a)}\\
\text{subject to} \quad & 0 = \Delta s_0 + (s_0^* - x_0), && \text{(4.6b)}\\
& 0 = \frac{\mathrm{d}c}{\mathrm{d}w}(w^*)\Delta w + c(w^*), && \text{(4.6c)}\\
& 0 \leq \frac{\mathrm{d}d}{\mathrm{d}w}(w^*)\Delta w + d(w^*). && \text{(4.6d)}
\end{aligned}$$

For further reading and an excellent survey on the topic of tangential predictors in NMPC the reader is referred to [56].

4.2 Real-Time Iteration scheme for Model Predictive Control

In NMPC we have to address two important issues for the feedback calculation: First to enable short sampling intervals and, second, to reduce the delay between obtaining the system state x_0 and feedback of the calculated control response $u(x_0)$ to the system. A high feedback frequency allows to handle time-critical processes, improves performance in the presence of uncertainties and disturbances, and enables the process to operate closer to its bounds and constraints. Classical NMPC, which waits for x_0 and then solves NLP (4.1), e.g., by a structured SQP method, struggles to address both issues. Because there is no general bound on the number of iterations needed to obtain a solution, we have to choose sampling intervals according to worst-case guesses of the solution runtime. Furthermore, while the solver is running, we have to apply outdated controls to the process which may lead to performance losses or even constraint violations.

In order to improve performance, recall the results discussed in the last section. Solving QP (4.6) amounts to taking the first iteration of a full-step exact-Hessian SQP method initialized in the old solution w^*, λ^*, μ^* for the new initial value x_0. Furthermore, due to the fact that x_0 enters only linearly, and only in the linearized IVE constraint (4.6b), all derivatives and almost all constraint evaluations that are needed to set up QP (4.6) can be computed without knowledge of the actual value of x_0. These observations motivate the following real-time optimization approach, which we refer to as *NMPC-RTI*:

- Per sample time, perform only one SQP iteration, initialized in the current iterate w_k, λ_k, μ_k, which is the best available guess for w^*, λ^*, μ^*. This exploits the ideas of tangential predictors and initial value embedding to obtain good feedback control updates as explained above while allowing to choose small sampling times and thus a high frequency of feedback control updates. Note that due to its linearity, the IVE constraint will be satisfied after the first iteration, as well as any bounds on the variables.

- Split the computations into three phases, namely
  1. *Preparation phase*: Set up and presolve QP (4.6) as far as possible without knowledge of the current initial value x_0, i.e., evaluate the Hessian $\mathcal{H}(w_k, \lambda_k, \mu_k)$ or an approximation B_k, the constraint Jacobians $\frac{\mathrm{d}c}{\mathrm{d}w}(w_k)$ and $\frac{\mathrm{d}d}{\mathrm{d}w}(w_k)$, the gradient $\nabla_w b(w_k)$, and the constraint residuals $c(w_k)$ and $d(w_k)$. Note that this step includes solving the dynamic equations as well as calculating sensitivity information, which makes it computationally particularly demanding. Because all QP matrices are available, we can also eliminate the variables $\Delta s_1, \ldots, \Delta s_N$ by condensing and even perform an initial matrix decomposition for the condensed QP.
  2. *Feedback phase*: As x_0 becomes available, evaluate the IVE constraint (4.6b) and solve the condensed QP to obtain the solution step in the primal variables $\Delta s_0, \Delta q$ and the new multipliers. Immediately return the control feedback $q_{0,k+1} = q_{0,k} + \Delta q_{0,k}$ to the process. Thus, the feedback delay is reduced to the solution time of the QP. The affine-linear dependence of this QP on x_0 can particularly be exploited by parametric quadratic programming, see [75, 148, 73] and the discussion in Chapter 7.
  3. *Transition phase*: Recover the eliminated step variables $\Delta s_1, \ldots, \Delta s_N$ and apply the step to obtain the new set of NLP variables $(w_{k+1}, \lambda_{k+1}, \mu_{k+1})$. If enough time is available, further iterations could be performed to improve $(w_{k+1}, \lambda_{k+1}, \mu_{k+1})$ towards (w^*, λ^*, μ^*), otherwise continue with the preparation phase for the next sample.

This approach is called the *Real-Time Iteration scheme* (RTI) and it has been presented and investigated in [51, 55] and many subsequent works. Stability has been shown for the RTI in various settings in [58, 54, 57, 174]. The key reason for the efficiency of the Real-Time Iterations is the decoupling of the computations and the sampling, which is possible due to the IVE, and thus the efficient use of computational time which would be wasted by waiting for the new x_0 in the black-box optimization approach.

The algorithm presented here is referred to as the *warm start* approach in [51], i.e., we use the unmodified w_k, λ_k, μ_k as initializer for the RTI. This adds another interpretation of the RTI as a continuously running SQP method, where only the parameter x_0 is changed during runtime. Alternatively, the initializer could be chosen by shifting the previous solution and appending suitable primal and dual variables at the end. For a discussion of shifted vs. non-shifted initialization see, e.g., [53, 51, 56]. In summary, shifting is only necessary in periodic tracking applications and for non-autonomous systems and non-autonomous cost functions.

4.3 Real-Time Iteration scheme for Moving Horizon Estimation

As discussed in Chapter 3, quite often we do not have the new current state x_0 readily available but have to obtain an estimate $\hat{x}_0$ from measurement data. We presented MHE as

an state-of-the-art approach to online state and parameter estimation. The NLP arising from a Direct Multiple Shooting discretization of the estimation problem for sampling time T reads

$$\underset{s,\omega,p}{\text{minimize}} \quad \left\| \begin{matrix} s_0 - \bar{x}_{T-N} \\ p - \bar{p}_{T-N} \end{matrix} \right\|^2_{Q_{T-N}} + \sum_{j=0}^{N-1} \|\omega_j\|^2_Q + \sum_{j=0}^{N} \|\eta_{T-N+j} - h(s_j, p)\|^2_R \tag{4.7a}$$

$$\text{subject to} \quad \omega_j = s_{j+1} - x_j(t_{j+1}; s_j, q_{T-N+j}, p),\ j = 0, \dots, N-1, \tag{4.7b}$$

$$s_{j,\min} \le s_j \le s_{j,\max},\ j = 0, \dots, N, \tag{4.7c}$$

$$\omega_{j,\min} \le \omega_j \le \omega_{j,\max},\ j = 0, \dots, N-1, \tag{4.7d}$$

$$p_{\min} \le p \le p_{\max}, \tag{4.7e}$$

where the variables are the states $s_0, \dots, s_N$, the state noise $\omega_0, \dots, \omega_{N-1}$ and the parameters p. The online data, which is updated in each sample, are the a priori information $\bar{x}_{T-N}, \bar{p}_{T-N}$ and Q_{T-N} for the arrival cost term, the controls $q_{T-N}, \dots, q_{T-1}$ and the measurements $\eta_{T-N}, \dots, \eta_T$. In principle, the weighting matrices R for the measurements could vary as well, e.g., to model skipped measurements, and then would also be part of the online data.

With variables $w = (s, w, p)$ and online data summarized as D we can write NLP (4.7) generically as

$$\min_w \|r(w; D)\|_2^2 \quad \text{s.t.} \quad c(w; D) = 0, \quad d(w) \ge 0. \tag{4.8}$$

This problem is a constrained nonlinear least-squares problem and thus our preferred numerical solution approach is the generalized Gauss-Newton method which works iteratively and uses in each iteration a linearization of all problem functions in the current iterate w_k. It generates variable increments Δw_k by solving

$$\min_{\Delta w} \quad \left\| r(w_k; D) + \frac{\mathrm{d}r}{\mathrm{d}w}(w_k; D)\Delta w \right\|_2^2 \tag{4.9a}$$

$$\text{s.t.} \quad c(w_k; D) + \frac{\mathrm{d}c}{\mathrm{d}w}(w_k; D)\Delta w = 0, \tag{4.9b}$$

$$d(w_k; D) + \frac{\mathrm{d}d}{\mathrm{d}w}(w_k; D)\Delta w \ge 0. \tag{4.9c}$$

and setting $w_{k+1} = w_k + \Delta w_k$. As before, the structure introduced by the Multiple Shooting discretization can be exploited by condensing to reduce the size of the problem before solving. Problem 4.9 is in principle a QP and can thus be solved by standard QP solvers. However, to address possible ill-conditioning, algorithms that avoid the explicit calculation of the QP Hessian $(\frac{\mathrm{d}r}{\mathrm{d}w})^\intercal \frac{\mathrm{d}r}{\mathrm{d}w}$ are preferable.

A close-up look at the online data shows that the data connected with the arrival cost term, the controls, and the measurements $\eta_{T-N}, \dots, \eta_{T-1}$ are all known before sample time T, and only the most recent measurement η_T is not yet available. This is in close similarity to the current state x_0 in NMPC and thus suggests to use RTI also for MHE. It should be noted that in the MHE case the first iteration will not be a tangential predictor of the solution, regardless whether an exact Hessian is used or not. This is due to two

aspects: first, we do not have an embedding equation for the measurement η_T, so no sensitivity information with respect to η_T is available in Problem 4.9, and second, the problems change at least slightly from sample to sample due to the update of the arrival cost term.

However, we can see from (4.7) that the most recent measurement η_T only enters linearly in $r(w; D)$ and thus in the objective gradient of Problem 4.9. This means that, as in the case of NMPC, we can compute all derivative matrices, constraint residuals, and almost the complete gradient without the knowledge of η_T. Thus, we can apply the approach of RTI and divide up the calculations into preparation phase, feedback phase, and transition phase. We refer to the following algorithm as *MHE-RTI*.

1. *Preparation phase*: Set up and presolve the linear least-squares problem (4.9) as far as possible without knowledge of the measurement η_T, i.e., evaluate the matrices $\frac{\mathrm{d}r}{\mathrm{d}w}(w_k; D)$, $\frac{\mathrm{d}c}{\mathrm{d}w}(w_k; D)$, and $\frac{\mathrm{d}d}{\mathrm{d}w}(w_k; D)$, the constraint residuals $c(w_k; D)$ and $d(w_k; D)$, and as far as possible the least-squares residual $r(w_k; D)$. Because all matrices are available, we can also eliminate the variables $\Delta s_1, \ldots, \Delta s_N$ by condensing and even perform an initial matrix decomposition for the condensed linear least-squares problem.

2. *Feedback phase*: As η_T becomes available, complete the evaluation of $r(w_k; D)$ and solve the condensed linear least-squares problem to obtain the solution step in the variables $\Delta s_0, \Delta\omega, \Delta p$. Recover the step variables $\Delta s_1, \ldots, \Delta s_N$. Immediately return the state estimate $s_{N,k+1} = s_{N,k} + \Delta s_{N,k}$ and the parameter estimate $p_{k+1} = p_k + \Delta p_k$ to the NMPC-RTI controller.

3. *Transition phase*: Apply the step to obtain the new w_{k+1}. Update the arrival cost term as explained in Chapter 3. Perform a shift of the variables s and ω by discarding s_0 and ω_0 and choosing new values s_N and ω_{N-1}, e.g., $s_N \triangleq x(t_N; s_{N-1}, q_{N-1}, p)$ and $\omega_{N-1} \triangleq 0$.

For an implementation of the combined and parallel cycle of NMPC-RTI and MHE-RTI we have to consider the following synchronization requirements:

- The feedback phase of NMPC-RTI requires a completed feedback phase of MHE-RTI because only then the estimate $\hat{x}_0$ is available.

- The transition phase and the following preparation phase of MHE-RTI requires a completed feedback phase of NMPC-RTI because only then the new control q_{N-1} is available.

For more details on the MHE-RTI and its successful application to problems from process engineering see, e.g., [59, 113, 114, 115].

5 Multi-Level Iteration schemes

In this chapter we present the *Multi-Level Iteration schemes* (MLI) as a new and efficient numerical approach for NMPC. We give a short motivation, then describe the different levels. We discuss how to assemble schemes with fixed level choice and how to exchange data between the levels. We give local convergence theory for a special case and reference previous work for the general case which applies to a subset of iteration schemes. We close the chapter with a note on mixed-level and fractional-level iterations.

5.1 Multi-Level Iteration approach

In the previous chapter, we presented the RTI scheme as an efficient numerical approach to both NMPC and MHE. The key features are the separation of the calculations into preparation phase, feedback phase, and transition phase and that only one nonlinear iteration is performed per sample. The computationally dominant tasks for each iteration are the function and derivative evaluation, condensing, and the QP solution. The former two are part of the preparation phase, the latter is part of the feedback phase.

The ratio of computational effort between these three tasks depends of course on several factors such as system size and nonlinearity, horizon length, number of shooting intervals, and number of constraints. However, as a rule of thumb, for typical applications from process engineering, function and derivative evaluation will most often take by far the bulk of computational effort per sample, followed by condensing and QP solution.

The *Multi-Level Iteration schemes* (MLI) presented in this work aim to reduce the computational effort spent for function and derivative evaluation and thus allow higher sampling rates. This is done by generating feedback by up to four different controller levels, i.e., feedback-generating QPs that update their respective data with decreasing complexity, ranging from full Real-Time Iterations to no updates at all. MLI schemes are then combined from the four different levels with a policy how data is communicated between the levels.

The motivation for the MLI is the observation that quite often the timescales of the process dynamics and their linearizations are different, i.e., while the process dynamics may change quickly, the linearizations may be valid for a much longer timeframe. As an illustrating example consider the linear harmonic oscillator, see (2.6). Depending on the frequency the trajectory of the oscillator may change rapidly with time. However, the linearization is constant and thus remains valid without change for all times. Based on these considerations, MLI tries to avoid computing unnecessary linearizations which are the computationally most expensive task per iteration.

MLI was first proposed in [30]. Applications of MLI to mechanical and chemical processes were reported in [186, 187, 185, 3, 129]. The Euler steps and the explicit feedback law have been first published in [107]. The application of MLI to NMPC with long horizons has been investigated in [106]. The mixed-level and fractional-level MLI schemes have been first published in [85]. The following presentation of these topics leans on and cites in parts from the above mentioned publications.

5.2 Description of the MLI levels

In principle, all four levels follow the idea of RTI with a separation of the computations into preparation phase, feedback phase, and transition phase, and performing a single iteration per sample for feedback calculation. In the feedback phase of sample k, the currently active level solves a QP

$$\begin{aligned} \underset{\Delta w}{\text{minimize}} \quad & \frac{1}{2}\Delta w^\intercal B_k \Delta w + b_k^\intercal \Delta w && \text{(5.1a)} \\ \text{subject to} \quad & 0 = \Delta s_0 + (s_0 - x_0^k), && \text{(5.1b)} \\ & 0 = C_k \Delta w + c_k, && \text{(5.1c)} \\ & 0 \leq D_k \Delta w + d_k, && \text{(5.1d)} \end{aligned}$$

however, the levels update their data $B_k, C_k, D_k, b_k, c_k, d_k$ differently. As a consequence, they have different computational workload in the preparation phase. In the following, we describe the four levels in order of decreasing computational complexity.

5.2.1 Full linearization iterations (level-D)

Level-D iterations are essentially Real-Time Iterations, i.e., full SQP iterations, cf. Section 4.2. Level-D holds its own set of primal and dual variables $(w_k^\mathrm{D}, \lambda_k^\mathrm{D}, \mu_k^\mathrm{D})$ and in each iteration, the constraints $c_k = c(w_k^\mathrm{D}), d_k = d(w_k^\mathrm{D})$, the objective gradient $b_k = \nabla b(w_k^\mathrm{D})$, and the constraint Jacobians $C_k = \frac{\mathrm{d}c}{\mathrm{d}w}(w_k^\mathrm{D}), D_k = \frac{\mathrm{d}d}{\mathrm{d}w}(w_k^\mathrm{D})$ are evaluated and a new Hessian (approximation) $B_k = B(w_k^\mathrm{D}, \lambda_k^\mathrm{D}, \mu_k^\mathrm{D})$ is calculated. For the special case of a least-squares objective, i.e., $b(w) = \frac{1}{2}\|r(w)\|_2^2$, the Gauss-Newton approximation $B_k = J(w_k^\mathrm{D})^\intercal J(w_k^\mathrm{D})$ with $J(w) = \frac{\mathrm{d}r}{\mathrm{d}w}(w)$ is a favorable and recommended choice. After solving QP (5.1), the control feedback $q_{0,k}^\mathrm{D} + \Delta q_{0,k}^\mathrm{D}$ is given to the process and the primal and dual variables are updated by

$$w_{k+1}^\mathrm{D} = w_k^\mathrm{D} + \Delta w_k^\mathrm{D}, \quad \lambda_{k+1}^\mathrm{D} = \lambda_k^\mathrm{QP}, \quad \mu_{k+1}^\mathrm{D} = \mu_k^\mathrm{QP}, \tag{5.2}$$

where $\Delta w_k^\mathrm{D}, \lambda_k^\mathrm{QP}, \mu_k^\mathrm{QP}$ is the primal-dual QP solution.

The computationally most expensive tasks in each level-D iteration are the evaluation of the full constraint Jacobians, in particular the linearization of the system dynamics, and the calculation of the Hessian (approximation). Because all matrices and vectors are updated, a full condensing step is required. If a parametric QP solver is used, an initial matrix decomposition for the condensed QP has to be performed.

5.2.2 Optimality iterations (level-C)

In level-C iterations, the evaluation of the constraint Jacobians, which comprise the computationally expensive system dynamics linearization, is avoided. Level-C holds its own variables $(w_k^{\mathrm{C}}, \lambda_k^{\mathrm{C}}, \mu_k^{\mathrm{C}})$ and approximations B_k, C_k, D_k of the Hessian and the constraint Jacobians. For the special case of a least-squares objective, level-C may hold an approximation J_k of the Jacobian of the objective residual r instead of B_k and use $B_k = J_k^\intercal J_k$ as Hessian approximation. In each iteration the constraints $c_k = c(w_k^{\mathrm{D}}), d_k = d(w_k^{\mathrm{D}})$ are evaluated. Instead of the standard gradient, the *modified gradient*

$$\begin{aligned} b_k &= \nabla b(w_k^{\mathrm{C}}) + \left(C_k^\intercal - \frac{dc}{dw}(w_k^{\mathrm{C}})^\intercal \right) \lambda_k^{\mathrm{C}} + \left(D_k^\intercal - \frac{dd}{dw}(w_k^{\mathrm{C}})^\intercal \right) \mu_k^{\mathrm{C}} \\ &= \nabla \mathcal{L}(w_k^{\mathrm{C}}, \lambda_k^{\mathrm{C}}, \mu_k^{\mathrm{C}}) + C_k^\intercal \lambda_k^{\mathrm{C}} + D_k^\intercal \mu_k^{\mathrm{C}} \end{aligned} \tag{5.3}$$

is computed. Note that although the exact Jacobians enter the modified gradient, they do so only by a matrix-vector product which can be efficiently computed by adjoint IND and the reverse mode of automatic differentiation, cf. [90].

After solving QP (5.1), the control feedback $q_{0,k}^{\mathrm{C}} + \Delta q_{0,k}^{\mathrm{C}}$ is given to the process and the primal and dual variables are updated by

$$w_{k+1}^{\mathrm{C}} = w_k^{\mathrm{C}} + \Delta w_k^{\mathrm{C}}, \quad \lambda_{k+1}^{\mathrm{C}} = \lambda_k^{\mathrm{QP}}, \quad \mu_{k+1}^{\mathrm{C}} = \mu_k^{\mathrm{QP}}, \tag{5.4}$$

where $\Delta w_k^{\mathrm{C}}, \lambda_k^{\mathrm{QP}}, \mu_k^{\mathrm{QP}}$ is the primal-dual QP solution.

In this work, we consider the case that the approximations B_k or J_k, C_k, and D_k are fixed matrices, often provided by a level-D iteration. In this case, the dominant computational cost is often the evaluation of the Lagrange gradient $\nabla \mathcal{L}(w_k^{\mathrm{C}}, \lambda_k^{\mathrm{C}}, \mu_k^{\mathrm{C}})$. It can be obtained at no more than five times the combined cost of the objective function and constraint evaluation, which is in general dominated by the solution of the dynamical system, cf. [90].

It should be noted that the bound on the evaluation cost given above refers to an evaluation of the objective and constraints for a given fixed evaluation scheme. However, the integrator has to determine this scheme first by applying the adaptive choices for error control. This can be computationally so expensive that the evaluation of the scheme afterwards is significantly cheaper and thus the gradient evaluation cost may be only a small multiple of or even cheaper than the constraint evaluation cost including the adaptive choices.

Condensing has to be applied only to the updated vector data, which reduces the complexity to $\mathcal{O}(N)$, with N the number of Multiple Shooting nodes. If a parametric QP solver is used, the final matrix decomposition of the preceding QP can be reused for initialization.

5.2.3 Feasibility iterations (level-B)

In level-B iterations, the exact evaluation of any new derivative information is avoided. Level-B holds its own variables $(w_k^{\mathrm{B}}, \lambda_k^{\mathrm{B}}, \mu_k^{\mathrm{B}})$ and approximations B_k, C_k, D_k of the Hessian

and the constraint Jacobians. Furthermore, level-B holds a fixed reference gradient $\bar{b}$ and a fixed set of reference variables $\bar{w}$, which are in general provided by level-D or level-C iterations. For the special case of a least-squares objective, level-B may hold an approximation J_k of the Jacobian of the objective residual r instead of B_k and use $B_k = J_k^\intercal J_k$ as Hessian approximation. In each iteration the constraints $c_k = c(w_k^{\mathrm{D}}), d_k = d(w_k^{\mathrm{D}})$ are evaluated. An approximation of the gradient is computed by

$$b_k = \bar{b} + B_k \left(w_k^{\mathrm{B}} - \bar{w}\right) \tag{5.5}$$

for the case of an economic objective function, and by

$$b_k = J_k^\intercal r(w_k^{\mathrm{B}}) \tag{5.6}$$

for the case of a least-squares objective function with Gauss-Newton Hessian approximation $B_k = J_k^\intercal J_k$.

After solving QP (5.1), the control feedback $q_{0,k}^{\mathrm{B}} + \Delta q_{0,k}^{\mathrm{B}}$ is given to the process and the primal and dual variables are updated by

$$w_{k+1}^{\mathrm{B}} = w_k^{\mathrm{B}} + \Delta w_k^{\mathrm{B}}, \quad \lambda_{k+1}^{\mathrm{B}} = \lambda_k^{\mathrm{QP}}, \quad \mu_{k+1}^{\mathrm{B}} = \mu_k^{\mathrm{QP}}, \tag{5.7}$$

where $\Delta w_k^{\mathrm{B}}, \lambda_k^{\mathrm{QP}}, \mu_k^{\mathrm{QP}}$ is the primal-dual QP solution.

In this thesis, we consider the case that the approximations B_k or J_k, C_k, and D_k are fixed matrices, mostly provided by a level-D or level-C iteration. In this case, the dominant computational cost is the evaluation of the constraints, in particular the solution of the dynamical system. Condensing has to be applied only to the updated vector data, which reduces the complexity to $\mathcal{O}(N)$, with N the number of Multiple Shooting nodes.

As discussed later in this chapter, in general level-B iterations drive their primal-dual iterates towards a feasible but suboptimal point and thus cannot provide optimal feedback. In a tracking NMPC application this may lead, e.g., to a failure in tracking a desired setpoint if only level-B iterations are applied to the process, see the numerical investigations in Chapter 9.

5.2.4 Feedback iterations (level-A)

In level-A iterations, no QP data is updated at all, which also means that no condensing is necessary. If a parametric QP solver is used, the final matrix decomposition of the preceding QP can be reused for initialization. Level-A holds no own variables and thus, in contrast to the other levels, does no internal iterations. Level-A iterations aim solely at giving feedback as quickly as possible.

Level-A keeps fixed approximations B_k or J_k, C_k, and D_k of the Hessian and constraint Jacobians, and fixed approximations b_k, c_k, d_k of the objective gradient and the constraint residuals. Furthermore, level-A keeps a fixed set of reference variables $\bar{w}$ for feedback calculation. After solving QP (5.1), the control feedback $\bar{q}_0 + \Delta q_{0,k}^{\mathrm{A}}$ is given to the process, where Δw_k^{A} is the primal QP solution.

Level-A is essentially a *linear model predictive controller* (LMPC) that works on local linearizations provided by higher levels.

Explicit feedback law

Level-A iterations can even be used to generate a local feedback law which maps differences $\Delta x_0 = x_0^{\text{new}} - x_0$ to feedback updates and thus can be used as an explicit continuous feedback law between two following QP solutions.

To see this, consider the Karush-Kuhn-Tucker (KKT) system of the QP (5.1) after a successful solution

$$\underbrace{\begin{pmatrix} B & -\tilde{C}_k^\intercal & -D_{k,\mathbb{A}}^\intercal \\ \tilde{C}_k & & \\ D_{k,\mathbb{A}} & & \end{pmatrix}}_{:=K} \begin{pmatrix} \Delta w_k \\ \Delta \lambda_k \\ \Delta \mu_{k,\mathbb{A}} \end{pmatrix} = - \begin{pmatrix} b_k \\ \tilde{c}_k \\ d_{\mathbb{A}} \end{pmatrix}, \tag{5.8}$$

where $\mathbb{A}$ is the optimal active set, $\tilde{c}_k$ combines the IVE constraint (5.1b) and the equality constraints (5.1c), and $\tilde{C}_k$ is the corresponding combined Jacobian approximation. Let $\mathbb{I}_{\Delta q_0}$ be the part of an identity matrix with suitable dimensions that extracts Δq_0 from Δw. The part of the inverse of K which gives Δq_0 when applied to the right hand side is then obtained by solving

$$K^\intercal X = \mathbb{I}_{\Delta q_0}. \tag{5.9}$$

Since a decomposition of K is available from the QP solver, this amounts to only n_{u} backsolves. Assuming that $\mathbb{A}$ keeps constant for small changes in x_0, an update for Δq_0 can be determined by building

$$X^\intercal \begin{pmatrix} 0 \\ \begin{pmatrix} \Delta x_0 \\ 0 \end{pmatrix} \\ 0 \end{pmatrix}, \tag{5.10}$$

for which only a small part of the matrix X is actually needed. Furthermore, we can write

$$\begin{pmatrix} 0 \\ \begin{pmatrix} \Delta x_0 \\ 0 \end{pmatrix} \\ 0 \end{pmatrix} = \mathbb{I}_{\Delta x_0} \Delta x_0, \tag{5.11}$$

where $\mathbb{I}_{\Delta x_0}$ is the part of an identity matrix with suitable dimensions that maps Δx_0 to the position of the IVE constraint within a vector of the size of the right-hand side that is otherwise zero. It follows that we obtain an update for Δq_0 by building

$$(\mathbb{I}_{\Delta q_0})^\intercal K^{-1} \mathbb{I}_{\Delta x_0} \Delta x_0. \tag{5.12}$$

Thus, if $n_{\text{u}} >> n_{\text{x}_0}$, it is computationally more efficient to solve

$$KX = \mathbb{I}_{\Delta x_0} \tag{5.13}$$

instead of (5.9) and build

$$(\mathbb{I}_{\Delta q_0})^\intercal X \Delta x_0 \tag{5.14}$$

to obtain an update for Δq_0.

Euler steps

In some cases the limiting factor for feedback generation is the sampling rate of the system states x_0, e.g., if the current states are obtained from a measurement procedure with limited throughput.

If it is still desired to update the feedback control with a higher frequency, a possible remedy is to use the model to predict the next x_0^{k+1} by an Euler step

$$x_0^{k+1} = x_0^k + (t_{k+1} - t_k)\, f(t_k, x_0^k, q_{0,k}) \tag{5.15}$$

for t_{k+1} close to t_k and use x_0^{k+1} to obtain a new feedback $q_{0,k+1}$. In addition, as the explicit Euler scheme generates a linear affine homotopy path for $x_0^{k+1}(t)$ starting in t_k, it can be readily combined with the parametric QP strategy of section 7.1. This allows system state predictions to enter the QP solution even before the solution process has been completed.

5.3 Assembling MLI schemes

In the following, we describe how Multi-Level Iteration schemes may be assembled from the four levels presented above. Here, we will consider only schemes that are prescribed in advance before applying the scheme to the process. The adaptive choice of levels online is considered in the following chapter. We will discuss the choice of levels and the exchange of data between different levels.

5.3.1 Prescribed MLI schemes

We will discuss the assembly of Multi-Level Iteration schemes with prescribed level order for schemes that make use of all four levels presented. For a practically useful scheme it is reasonable to take the different computational times per level iteration into account. Let $\Delta_\mathrm{A}, \Delta_\mathrm{B}, \Delta_\mathrm{C}$, and Δ_D be upper bounds on the expected worst-case computational times for level-A, level-B, level-C, and level-D iterations, respectively. These upper bounds may be obtained from numerical experiments. As sampling interval we choose a suitable $\delta \geq \Delta_\mathrm{A}$ and define

$$n_\mathrm{B} = \left\lceil \frac{\Delta_\mathrm{B}}{\Delta_\mathrm{A}} \right\rceil, \quad n_\mathrm{C} = \left\lceil \frac{\Delta_\mathrm{C}}{\Delta_\mathrm{B}} \right\rceil, \quad n_\mathrm{D} = \left\lceil \frac{\Delta_\mathrm{D}}{\Delta_\mathrm{C}} \right\rceil. \tag{5.16}$$

Then, a natural choice for a Multi-Level Iteration scheme with prescribed level order is

1. basically apply level-A to generate feedback in each sample, except,
2. each n_B sample, perform a level-B iteration,
3. each $(n_\mathrm{B} \cdot n_\mathrm{C})$ sample, perform a level-C iteration,
4. each $(n_\mathrm{B} \cdot n_\mathrm{C} \cdot n_\mathrm{D})$ sample, perform a level-D iteration.

Alternatively, we can define

$$\bar{n}_{\mathrm{B}} = \left\lceil \frac{\Delta_{\mathrm{B}}}{\Delta_{\mathrm{A}}} \right\rceil, \quad \bar{n}_{\mathrm{C}} = \left\lceil \frac{\Delta_{\mathrm{C}}}{\Delta_{\mathrm{A}}} \right\rceil, \quad \bar{n}_{\mathrm{D}} = \left\lceil \frac{\Delta_{\mathrm{D}}}{\Delta_{\mathrm{A}}} \right\rceil, \tag{5.17}$$

and choose the level order as

1. basically perform level-A to generate feedback in each sample, except,
2. each $\bar{n}_{\mathrm{B}}$ sample, perform a level-B iteration,
3. each $\bar{n}_{\mathrm{C}}$ sample, perform a level-C iteration,
4. each $\bar{n}_{\mathrm{D}}$ sample, perform a level-D iteration.

In both cases, if a sample belongs to two or more levels according to the selection method, the highest applicable level is chosen.

Both level choices practically guarantee that between each two iterations of the same level enough time is available to finish the corresponding preparation phase in time before having to compute the feedback. As a compact notation for such a scheme we write $A^1 B^{n_{\mathrm{B}}} C^{(n_{\mathrm{B}} \cdot n_{\mathrm{C}})} D^{(n_{\mathrm{B}} \cdot n_{\mathrm{C}} \cdot n_{\mathrm{D}})}$ in the first case, and $A^1 B^{\bar{n}_{\mathrm{B}}} C^{\bar{n}_{\mathrm{C}}} D^{\bar{n}_{\mathrm{D}}}$ in the second case, e.g., $A^1 B^3 C^6 D^{30}$ for $n_{\mathrm{B}} = 3, n_{\mathrm{C}} = 2$, and $n_{\mathrm{D}} = 5$. A visualization of an incremental assembly of the MLI scheme $A^1 B^2 C^4 D^8$ beginning with a pure level-A scheme is given as example in Figure 5.1.

Of course, schemes can be assembled analogously using only a subset of the four levels, e.g., a combination of level-A and level-D for a scheme with fast feedback and occasional updates of the linearizations, or a pure level-C scheme as a fully nonlinear alternative to the standard RTI scheme. Using the notation introduced above, well known limit cases of the MLI schemes are the A^1 scheme, which is standard LMPC, and the D^1 scheme, which is the standard RTI scheme.

Example applications of such MLI schemes can be found, e.g., for mixed A and D level schemes for the disturbance rejection in a car crash scenario in [3], for pure C level schemes for the control of a chain of spring connected masses in [186, 187], and for a mixed B and D level scheme for the control of a chemical batch reactor in [129].

5.3.2 Data transfer between levels

The data communicated between the different levels can be divided into two parts: First, the matrix and vector data in the QP which generates the actual feedback given to the process and, second, the level-specific primal-dual variables, which determine the point in the space of the optimization variables where functions and derivatives are evaluated.

Regarding the matrix and vector data for the feedback-generating QP, we focus on the update algorithm (Algorithm 1).

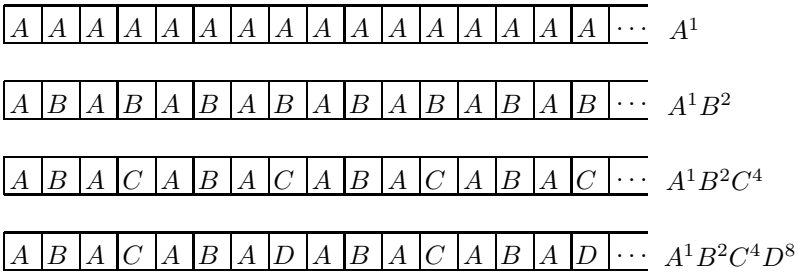


Figure 5.1: Incremental construction of the example MLI scheme $A^1B^2C^4D^8$. From top to bottom: Pure level-A scheme, A^1B^2 scheme, $A^1B^2C^4$ scheme, and $A^1B^2C^4D^8$ scheme. Note that higher levels take precedence in the case of a multi-occupancy of level slots.

Input: current level

switch *current level* **do**

case *level-D* **do**

In QP (5.1), set

$$\begin{aligned} b_k &= \nabla b(w_k^{\mathrm{D}}), \\ c_k &= c(w_k^{\mathrm{D}}), \\ d_k &= d(w_k^{\mathrm{D}}), \\ C_k &= \tfrac{\mathrm{d}c}{\mathrm{d}w}(w_k^{\mathrm{D}}), \\ D_k &= \tfrac{\mathrm{d}d}{\mathrm{d}w}(w_k^{\mathrm{D}}), \\ B_k &= B(w_k^{\mathrm{D}}, \lambda_k^{\mathrm{D}}, \mu_k^{\mathrm{D}}). \end{aligned}$$

Pass C_k and D_k to level-C for computation of modified gradient.

Pass B_k (or J_k), b_k, and w_k^{D} to level-B for computation of approximated gradient.

Set $\bar{q}_0 = q_{0,k}^{\mathrm{D}}$ as new reference in level-A.

case *level-C* **do**

In QP (5.1), set

$$\begin{aligned} b_k &= \nabla \mathcal{L}(w_k^{\mathrm{C}}, \lambda_k^{\mathrm{C}}, \mu_k^{\mathrm{C}}) + C_k^{\intercal}\lambda_k^{\mathrm{C}} + D_k^{\intercal}\mu_k^{\mathrm{C}}, \\ c_k &= c(w_k^{\mathrm{C}}), \\ d_k &= d(w_k^{\mathrm{C}}). \end{aligned}$$

Pass b_k and w_k^{C} to level-B for computation of approximated gradient.

Set $\bar{q}_0 = q_{0,k}^{\mathrm{C}}$ as new reference in level-A.

case *level-B* **do**

In QP (5.1), set

$$\begin{aligned} b_k &= \bar{b} + B_k\left(w_k^{\mathrm{B}} - \bar{w}\right) \text{ or } b_k = J_k^{\intercal} r(w_k^{\mathrm{B}}), \\ c_k &= c(w_k^{\mathrm{B}}), \\ d_k &= d(w_k^{\mathrm{B}}). \end{aligned}$$

Set $\bar{q}_0 = q_{0,k}^{\mathrm{B}}$ as new reference in level-A.

Algorithm 1: Data communication for feedback-generating QP

Algorithm 1 amounts to passing all available QP data from higher levels to lower levels.

However, other approaches are possible and interesting, such as using level-D iterations only to update the matrix data B_k, C_k, and D_k, and then using the most recent level-C or level-B right-hand-side data b_k, c_k, and d_k to calculate feedback. This can be done quite efficiently, because the matrix data enters only in the form of quickly computable matrix-vector products in the gradient approximations b_k of level C and level-B, so most parts of the computations can be done in parallel.

Since the levels B, C, and D each have their own sets of primal and possibly dual variables the question arises if and how they should exchange information in the case of level switching. There are four basic ways how such a communication can be organized.

Input: current level

switch *current level* **do**
  case *level-D* **do**
    Set $(w^C, \lambda^C, \mu^C) = (w^D, \lambda^D, \mu^D)$
    Set $(w^B, \lambda^B, \mu^B) = (w^D, \lambda^D, \mu^D)$
  case *level-C* **do**
    Set $(w^B, \lambda^B, \mu^B) = (w^C, \lambda^C, \mu^C)$
  case *level-B* **do**
    No variable transfer

Algorithm 2: Top-down data communication

In the top-down communication described in Algorithm 2, the internal primal-dual variables of lower levels are overwritten whenever a higher level iteration occurs. The interpretation is that the iteration scheme with the highest level is considered the main iteration scheme and lower level schemes are mainly used to give intermediate feedback until the next main iteration.

An advantage is that the highest level, which is either level-D or level-C, is a fully nonlinear feedback controller with the whole theoretical background of RTI and can be considered as a fallback controller which can reset a possibly suboptimal iteration behavior of level-B provided the higher level iterations occur sufficiently often. A potential drawback is that higher level iterations, in particular level-D iterations, take longer computational times and thus are set up at somewhat outdated iterates compared to the iterates of the lower levels. The iteration step then has to improve the iterate as well as compensate for the time delay, which becomes more difficult the less frequently the higher level iterations occur. It may thus be that the new iterate of the higher level is a less suitable solution approximation than more recently updated iterates of the lower levels. Usually this manifests in undesirable oscillations in the feedback control.

To avoid this, the top-down communication should preferably be used only if higher level iterations occur frequently, or the iterates of the higher level may be modified by a prediction taking the computational delay into account.

Input: current level

switch *current level* **do**
  case *level-D* **do**
    No variable transfer
  case *level-C* **do**
    Set $(w^D, \lambda^D, \mu^D) = (w^C, \lambda^C, \mu^C)$
  case *level-B* **do**
    Set $(w^C, \lambda^C, \mu^C) = (w^B, \lambda^B, \mu^B)$
    Set $(w^D, \lambda^D, \mu^D) = (w^B, \lambda^B, \mu^B)$

Algorithm 3: Bottom-up data communication

In the bottom-up communication described in Algorithm 3, the internal primal-dual variables of higher levels are overwritten whenever a lower level iteration occurs. The interpretation is that the iteration scheme with the lowest level (higher than level-A) is considered the main iteration scheme and higher level schemes are mainly used to update the feedback-generating QP and derivative information in the lower level schemes.

Advantages and drawbacks are similar but reversed compared to the top-down data communication: because lower level iterations occur more frequently, their iterates contain in general better predictions of the current and future system behavior. However, the feedback generated by level-B iterations might steer the system in undesirably suboptimal trajectories before the less frequent updates by higher levels improve the feedback controller towards optimality. Thus, MLI schemes should schedule higher level iterations sufficiently often if bottom-up data communication is used.

Input: current level

switch *current level* **do**
  case *level-D* **do**
    Set $(w^C, \lambda^C, \mu^C) = (w^D, \lambda^D, \mu^D)$
    Set $(w^B, \lambda^B, \mu^B) = (w^D, \lambda^D, \mu^D)$
  case *level-C* **do**
    Set $(w^D, \lambda^D, \mu^D) = (w^C, \lambda^C, \mu^C)$
    Set $(w^B, \lambda^B, \mu^B) = (w^C, \lambda^C, \mu^C)$
  case *level-B* **do**
    Set $(w^C, \lambda^C, \mu^C) = (w^B, \lambda^B, \mu^B)$
    Set $(w^D, \lambda^D, \mu^D) = (w^B, \lambda^B, \mu^B)$

Algorithm 4: Maximum data communication

In the maximum data communication described in Algorithm 4, the internal primal-dual variables of all levels are overwritten whenever an iteration (of level-B or higher) occurs. The interpretation is that all levels iterate on the same set of common primal-dual variables. This is the data communication approach we focus on for the adaptive level choice in the next chapter.

A potential drawback, which also applies in the case of bottom-up data communication, is the fact that level-B converges locally towards suboptimal points and thus produce dual

variable guesses that may be unfavorable for the calculation of the modified gradient in level-C or the Hessian (approximations) in level-D. A possible remedy is a preceding step for higher level iterations to obtain better dual variable guesses using the first order optimality conditions, which is subject to future investigation.

Input: current level

switch *current level* **do**
  case *level-D* **do**
    No variable transfer
  case *level-C* **do**
    No variable transfer
  case *level-B* **do**
    No variable transfer

Algorithm 5: Minimum data communication

In the minimum data communication described in Algorithm 5, no iterates are communicated across the different levels, i.e., every level iterates independently on its own set of primal-dual variables and only the feedback generating QP is updated by the different levels. The interpretation is that a standard RTI scheme based on level-D or level-C iterations with a longer sampling is applied to the process, and additionally the lower levels provide independently feedback with higher frequency in between two iterations of the higher levels. An advantage is that provided that the feedback of the lower levels does not drive the process to undesirably suboptimal trajectories, the higher level feedback will provide the theoretical and practical benefits of the well-established RTI scheme. Of course this is only true if the RTI with a sampling of the higher level iterations in itself is suitable for feedback control of the process.

It is also possible to combine some of the four presented data communications, e.g., to use maximum data communication for level-D and level-C, effectively iterating on a common set of primal-dual variables, and top-down or bottom-up data communication with respect to level-B. We do not study these variants in this thesis.

5.4 Convergence analysis

We give local convergence theory for pure level-B, level-C, and level-D iterations for the case $x_0^k = x_0$, i.e., that the iterations are always applied to the same problem. It can be regarded as local convergence analysis for inexact SQP methods where the steps are determined by level-B, level-C, and level-D iterations, respectively. This point of view motivates the alternative naming convention of level-C iterations as *optimality iterations* and of level-B iterations as *feasibility iterations*. We give the proof for level-C iterations, which has been carried out for the first time in L. Wirsching, *An SQP algorithm with inexact derivatives for a Direct Multiple Shooting method for optimal control problems* [184], and later in [60]. We then discuss the validity of the proof for level-B and level-D iterations, which was established in [29].

Stability analysis for the Multi-Level Iteration schemes is much more challenging since

the combined controller-system interaction has to be considered. We will briefly discuss the work that has been published on this topic.

The proof of local convergence for the level-C iterations for the case $x_0^k = x_0$ consists of two steps. First we establish stability of the QP active set near a solution of the NLP. This allows to regard the SQP iterations locally as a Newton-like method applied to the first order KKT conditions. Then local convergence theory for Newton-like methods yield the result.

5.4.1 Stability of the QP Active Set near an NLP Solution

We now deal with the question of the stability of the active set of the QP (2.30) near a solution of the NLP (2.24). The following theorem extends Theorem 2.4 and ensures this important property under rather mild conditions.

Theorem 5.1 (Stability of QP active set near NLP solution)**.**
Let (w^*, λ^*, μ^*) be a KKT-point of problem (2.24), $\|\cdot\|$ the Euclidean vector norm and $\|\cdot\|_{\mathrm{F}}$ the Frobenius matrix norm. Assume that w^* is a regular point and the strict complementary condition holds at (w^*, λ^*, μ^*).

Then for any constants $\alpha_1, \alpha_2 > 0$ there exists a neighborhood $\mathcal{N}$ of (w^*, λ^*, μ^*) and a constant $\gamma > 0$ such that for all $(w, \lambda, \mu) \in \mathcal{N}$ the following statement holds: For any three matrices $B \in \mathbb{R}^{n\times n}$, $C \in \mathbb{R}^{l\times n}$, $D \in \mathbb{R}^{m\times n}$ with B positive semi-definite and $\|C\|_{\mathrm{F}}$, $\|D\|_{\mathrm{F}} \leq \alpha_1$ and such that the matrix

$$J(B, C, D) := \begin{pmatrix} B & -C^\intercal & -\widetilde{D}^\intercal \\ C & 0 & 0 \\ \widetilde{D} & 0 & 0 \end{pmatrix} \tag{5.18}$$

with $\widetilde{D} := (D_j)_{j\in\mathcal{A}(w^*)}$, where D_j denotes the j-th row of D, is invertible and satisfies $\|J(B, C, D)^{-1}\|_{\mathrm{F}} \leq \alpha_2$, the QP

$$\begin{aligned} \mathrm{QP}(w, \lambda, \mu, B, C, D): \quad \min_{\Delta w} \quad & \frac{1}{2}\Delta w^\intercal B\, \Delta w + b(w, \lambda, \mu)^\intercal \Delta w && \text{(5.19a)} \\ \text{s. t.} \quad & c(w) + C\Delta w = 0 && \text{(5.19b)} \\ & d(w) + D\Delta w \geq 0 && \text{(5.19c)} \end{aligned}$$

with the modified gradient $b(w, \lambda, \mu) = \nabla_w \mathcal{L}(w, \lambda, \mu) + C^\intercal \lambda + D^\intercal \mu$ has a unique solution $(\Delta w, \lambda^{\mathrm{QP}}, \mu^{\mathrm{QP}})$ that satisfies

$$\|(\Delta w, \lambda^{\mathrm{QP}}, \mu^{\mathrm{QP}}) - (0, \lambda^*, \mu^*)\| \leq \gamma \|(w^*, \lambda^*, \mu^*) - (w, \lambda, \mu)\|. \tag{5.20}$$

This QP solution has the same active set $\mathcal{A}$ as the NLP solution w^*.

Proof. We start by noting that every solution Δw and multipliers λ^{QP}, μ^{QP} of QP (5.19)

have to satisfy the KKT conditions for the QP

$$\begin{aligned} B\,\Delta w + \nabla_w \mathcal{L}(w,\lambda,\mu) + C^\intercal\,(\lambda - \lambda^{\mathrm{QP}}) + D^\intercal\,(\mu - \mu^{\mathrm{QP}}) &= 0 && \text{(5.21a)}\\ c(w) + C\Delta w &= 0 && \text{(5.21b)}\\ d(w) + D\Delta w &\geq 0 && \text{(5.21c)}\\ \mu^{\mathrm{QP}} &\geq 0 && \text{(5.21d)}\\ \forall\, j = 1,\ldots,m: \quad \mu_j^{\mathrm{QP}}\,(d(w) + D\Delta w)_j &= 0. && \text{(5.21e)} \end{aligned}$$

For $(w,\lambda,\mu) = (w^*,\lambda^*,\mu^*)$ we obtain a KKT point of the QP by setting $(\Delta w, \lambda^{\mathrm{QP}}, \mu^{\mathrm{QP}}) = (0,\lambda^*,\mu^*)$. We now show that this KKT point satisfies the Jacobian uniqueness condition. The active set and the validity of the strict complementary condition are inherited from the NLP solution. The LICQ then holds if the columns of the matrix

$$\widetilde{C}^\intercal := \left(C^\intercal\,\widetilde{D}^\intercal\right)$$

are linearly independen, which follows from the invertibility of the matrix $J(B,C,D)$.

From this follows also that the matrix B is positive definite on the null space of the matrix $\widetilde{C}$. We see this by taking a $z \in \mathbb{R}^n$, $z \neq 0$ with $\widetilde{C}\,z = 0$ and assuming that $z^\intercal B\,z = 0$. Since B is positive semidefinite it can be written as $B = B^{1/2}\,B^{1/2}$. We obtain that z must be in the null space of $B^{1/2}$ and therefore in the null space of B. Then for $\tilde{z} := (z^\intercal, 0^\intercal)^\intercal \in \mathbb{R}^{n+l+m}$ we have $J(B,C,D)\,\tilde{z} = 0$ which contradicts to the invertibility assumption on $J(B,C,D)$. Thus it must be $z^\intercal B\,z \neq 0$. Since B is positive semidefinite, we have $z^\intercal B\,z > 0$.

By validity of the Jacobian uniqueness condition we conclude that the KKT point is a strict local minimizer of the QP and from the convexity of the QP it then follows that the KKT point is the unique global minimizer of the QP.

Let us now consider the case of the QP being formulated at a point (w,λ,μ). From the invertibility of $J(B,C,D)$ it follows that the solution of the linear system

$$J(B,C,D)\begin{pmatrix} s \\ u \\ v \end{pmatrix} = \underbrace{\begin{pmatrix} -b(w,\lambda,\mu) \\ -c(w) \\ -\widetilde{d}(w) \end{pmatrix}}_{=:\,\zeta(w,\lambda,\mu)} \tag{5.22}$$

with $\widetilde{d}(w) := (d_j(w))_{j\in\mathcal{A}(w^*)}$ is unique. Furthermore we have

$$\begin{pmatrix} s-0 \\ u-\lambda^* \\ v-\tilde{\mu}^* \end{pmatrix} = J(B,C,D)^{-1}\begin{pmatrix} b(w^*,\lambda^*,\mu^*) - b(w,\lambda,\mu) \\ c(w^*) - c(w) \\ \widetilde{d}(w^*) - \widetilde{d}(w) \end{pmatrix}$$

with $\tilde{\mu}^* := (\mu_j^*)_{j\in\mathcal{A}(w^*)}$. Since $b(w)$, $c(w)$ and $d(w)$ are assumed to be twice continuously differentiable in w, we can see that $\nabla_w\mathcal{L}(w,\lambda,\mu)$ is once continuously differentiable in (w,λ,μ) and therefore is locally Lipschitz-continuous in (w,λ,μ) with some Lipschitz

constant β_1. We obtain

$$\begin{aligned}\|b(w^*,\lambda^*,\mu^*) - b(w,\lambda,\mu)\| &\leq \|\nabla_w \mathcal{L}(w^*,\lambda^*,\mu^*) - \nabla_w \mathcal{L}(w,\lambda,\mu)\| \\ &\quad + \|C^\intercal(\lambda^* - \lambda)\| + \|D^\intercal(\mu^* - \mu)\| \\ &\leq \beta_1 \|(w^*,\lambda^*,\mu^*) - (w,\lambda,\mu)\| \\ &\quad + \alpha_1 \|\lambda^* - \lambda\| + \alpha_1 \|\mu^* - \mu\| \\ &\leq (\beta_1 + 2\alpha_1) \|(w^*,\lambda^*,\mu^*) - (w,\lambda,\mu)\|\end{aligned}$$

where we have used that $\|C\|_{\mathrm{F}} = \|C^\intercal\|_{\mathrm{F}}$. Let furthermore β_2 and β_3 be the Lipschitz constants of $c(w)$ and $d(w)$, respectively. Thus we obtain the estimate

$$\left\| \begin{pmatrix} s - 0 \\ u - \lambda^* \\ v - \tilde{\mu}^* \end{pmatrix} \right\| \leq \alpha_2(\beta_1 + \beta_2 + \beta_3 + 2\alpha_1) \left\| \begin{pmatrix} w^* - w \\ \lambda^* - \lambda \\ \mu^* - \mu \end{pmatrix} \right\|. \tag{5.23}$$

With the solution (s,u,v) of system (5.22) let us now recall the KKT conditions (5.21) for the QP:

- We first set $\Delta w := s$. Condition (5.21b) then holds by (5.22).
- Let us define $\check{d}(w) := (d_j(w))_{j \notin \mathcal{A}(w^*)}$ and $\check{D} := (D_j)_{j \notin \mathcal{A}(w^*)}$. With the estimate (5.23) we have

  $$\begin{aligned}\left\|\check{d}(w) + \check{D}\,\Delta w - \check{d}(w^*)\right\| &\leq \left\|\check{d}(w) - \check{d}(w^*)\right\| + \left\|\check{D}\,\Delta w\right\| \\ &\leq \beta_3 \|w^* - w\| + \alpha_1 \|\Delta w\| \\ &\leq (\beta_3 + \alpha_1\theta) \|(w^*,\lambda^*,\mu^*) - (w,\lambda,\mu)\|\end{aligned}$$

  with $\theta := \alpha_2(\beta_1 + \beta_2 + \beta_3 + 2\alpha_1)$. We set $c := \min_{j \notin \mathcal{A}(w^*)}\{d_j(w^*)\} > 0$ and choose $0 < c_1 < c$. By choosing a suitable neighborhood $\mathcal{N}^1 = \mathcal{N}^1_w \times \mathcal{N}^1_\lambda \times \mathcal{N}^1_\mu$ of (w^*,λ^*,μ^*) we can guarantee $(\beta_3 + \alpha_1\theta)\|(w^*,\lambda^*,\mu^*) - (w,\lambda,\mu)\| \leq c_1$ and therefore $\left\|\check{d}(w) + \check{D}\,\Delta w - \check{d}(w^*)\right\| \leq c_1$ for all $(w,\lambda,\mu) \in \mathcal{N}^1$. It follows that

  $$\begin{aligned}(d(w) + D\,\Delta w)_j &= d_j(w^*) + [(d(w) + D\,\Delta w)_j - d_j(w^*)] \\ &\geq c - c_1 > 0\end{aligned}$$

  for all $j \notin \mathcal{A}(w^*)$, i.e. $\check{d}(w) + \check{D}\,\Delta w > 0$. Furthermore, we have $\widetilde{d}(w) + \widetilde{D}\,\Delta w = 0$ by (5.22). Together, for $(w,\lambda,\mu) \in \mathcal{N}^1$ we obtain that condition (5.21c) holds, with the same inequalities being active as in the solution of the QP formulated in (w^*,λ^*,μ^*).
- Without loss of generality let us assume that $\mathcal{A}(w^*) = 1,\ldots,\tilde{m}$ with $\tilde{m} = |\mathcal{A}(w^*)|$. We then set $\mu^{\mathrm{QP}} := (v^\intercal, 0^\intercal)^\intercal$. Since $\tilde{\mu}^* > 0$ we can again find by inequalities (5.23) a neighborhood $\mathcal{N}^2 = \mathcal{N}^2_w \times \mathcal{N}^2_\lambda \times \mathcal{N}^2_\mu$ of (w^*,λ^*,μ^*) such that $v > 0$ for all $(w,\lambda,\mu) \in \mathcal{N}^2$. We therefore obtain for all $(w,\lambda,\mu) \in \mathcal{N} := \mathcal{N}^1 \cap \mathcal{N}^2$ that both condition (5.21d) and condition (5.21e) hold, with the strict complementary condition satisfied.

- Finally, with $\lambda^{\mathrm{QP}} := u$, the Lagrange gradient (5.21a) of the QP vanishes, as can also be seen by (5.22).

Thus, we obtain that for all $(w, \lambda, \mu) \in \mathcal{N}$ we have a KKT point of QP (5.19) at $(\Delta w, \lambda^{\mathrm{QP}}, \mu^{\mathrm{QP}}) = (s, u, v, 0)$ satisfying the Jacobian uniqueness condition and with the active set $\mathcal{A}(w^*)$. The same reasoning as above then tells us that this KKT point is the unique global solution of QP (5.19). Estimate (5.20) follows from (5.23), suitably augmented with zeros. □

Remark 5.1. *Since all vector norms in a finite-dimensional normed vector space are equivalent, one can also choose every other vector norm (and a corresponding compatible matrix norm) in the proof of Theorem 5.1. The choice of the Frobenius norm is convenient to be able to use the same bound for both a matrix and its transposed matrix.*

Remark 5.2. *Theorem 5.1 requires a positive semi-definite QP Hessian matrix B. This is always satisfied if we use SQP with BFGS Hessian approximations or SQP with Gauss-Newton Hessian approximations. However, for SQP with exact Hessians this condition is in general violated and Theorem 5.1 cannot be applied.*

Now we have established that QP (5.19) behaves in a neighborhood of its unique global minimizer like an equality-constrained QP. This allows to apply techniques for proving local convergence of inexact Newton methods for equality-constrained NLPs.

5.4.2 Local convergence

We now deal with the question of local convergence of the level-C iterations. An astounding aspect of Theorem 5.1 is the fact that the Jacobian approximations can be chosen rather arbitrarily, given that they satisfy the conditions required by the theorem. Particularly, no relation between the exact Jacobians and the approximations has to be assumed. However, we have to expect that such a relation is necessary to establish local convergence.

We make use of the following lemma which is a variant of standard results that have similarly been formulated, e.g. in [28, 50] and Theorem 2.5.

Lemma 5.2 (Local Convergence for Inexact Newton Methods)**.**
Assume $\Phi : \mathcal{D} \mapsto \mathbb{R}^{n_y}$, $\mathcal{D} \subset \mathbb{R}^{n_y}$ open, is continuously differentiable and consider the sequence

$$y_{k+1} = y_k + \Delta y_k, \quad \Delta y_k = -J_k^{-1}\, \Phi(y_k),$$

starting with some $y_0 \in D$. Let us make the following assumptions:

(i) The sequence of invertible matrices J_k is uniformly bounded and has uniformly bounded inverses.

(ii) There exists a $\kappa < 1$ such that for all $k \in \mathbb{N}$ it can be guaranteed that

$$\left\| J_{k+1}^{-1} \left(J_k - \frac{\partial \Phi}{\partial y} \left(y_k + t\, \Delta y_k \right) \right) \Delta y_k \right\| \leq \kappa \, \| \Delta y_k \|, \quad \forall\, t \in [0, 1].$$

(iii) The set

$$\mathcal{B}\left(y_0, \frac{\|\Delta y_0\|}{1-\kappa}\right) := \left\{ y \in \mathbb{R}^{n_y} \,\middle|\, \|y - y_0\| \leq \frac{\|\Delta y_0\|}{1-\kappa} \right\}$$

is contained in D.

Then the sequence $(y_k)_{k\in\mathbb{N}}$ remains in $\mathcal{D}$ and converges towards a point $y^* \in \mathcal{B}\left(y_0, \frac{\|\Delta y_0\|}{1-\kappa}\right)$ satisfying $\Phi(y^*) = 0$. Moreover, if

$$\lim_{k\to\infty} \frac{\left\| J_{k+1}^{-1} \left(J_k - \frac{\partial \Phi}{\partial y}(y^*) \right) \Delta y_k \right\|}{\|\Delta y_k\|} = 0,$$

the convergence rate is q-superlinear.

With this lemma we can prove the local convergence of level-C iterations.

Theorem 5.3 (Local Convergence of level-C iterations)**.**
Let (w^*, λ^*, μ^*) be a KKT-point of Problem (2.24), $\|\cdot\|$ be an arbitrary but fixed vector norm and let

$$\Phi(y) \triangleq \Phi(w, \lambda, \mu) \triangleq \begin{pmatrix} \nabla_w \mathcal{L}(w, \lambda, \mu) \\ c(w) \\ \widetilde{d}(w) \end{pmatrix} \tag{5.24}$$

be the function consisting of the Lagrange gradient, the equality constraints, and the inequality constraints that are active in w^*. Assume that the following conditions hold:

(i) w^* is a regular point and the strict complementary condition holds at (w^*, λ^*, μ^*).

(ii) Let (B_k), (C_k) and (D_k) be uniformly bounded sequences of matrices such that B_k is positive semidefinite for all $k \in \mathbb{N}$ and the sequence (J_k^{-1}) with $J_k := J(B_k, C_k, D_k)$ defined as in Theorem 5.1 is uniformly bounded.

(iii) There is a sequence $y_k := (w_k, \lambda_k, \mu_k)$ generated according to

$$y_{k+1} = y_k + \Delta y_k, \qquad \text{with} \qquad \Delta y_k := (\Delta w_k, \tilde{\lambda}_k - \lambda_k, \tilde{\mu}_k - \mu_k)$$

where Δw_k is the solution of the quadratic program

$$\min_{\Delta w} \quad \frac{1}{2} \Delta w^\intercal B_k \Delta w + b(w_k, \lambda_k, \mu_k)^\intercal \Delta w \tag{5.25a}$$

$$\text{s.t.} \quad c(w_k) + C_k \Delta w = 0, \tag{5.25b}$$

$$d(w_k) + D_k \Delta w \geq 0 \tag{5.25c}$$

and $\tilde{\lambda}_k$, $\tilde{\mu}_k$ are the multipliers of the equality and inequality constraints in the QP solution, respectively.

(iv) There exists a $\kappa < 1$ such that for all $k \in \mathbb{N}$ it can be guaranteed that

$$\left\| J_{k+1}^{-1} \left(J_k - \frac{\partial \Phi}{\partial y}(y_k + t\,\Delta y_k) \right) \Delta y_k \right\| \leq \kappa \, \|\Delta y_k\|, \quad \forall\, t \in [0, 1].$$

Then there exists a neighborhood $\bar{\mathcal{N}}$ of (w^*, λ^*, μ^*) such that for all initial guesses $(w_0, \lambda_0, \mu_0) \in \bar{\mathcal{N}}$ the sequence (w_k, λ_k, μ_k) converges q-linearly towards (w^*, λ^*, μ^*) with rate κ, and the solution of each QP (5.25) has the same active set as w^*.

Proof. Since the sequences (D_k) and (J_k^{-1}) are assumed to be uniformly bounded, we can find $\alpha_1, \alpha_2 > 0$ such that $\|D_k\|_{\mathrm{F}} \leq \alpha_1$ and $\|J_k^{-1}\|_{\mathrm{F}} \leq \alpha_2$ for all $k \in \mathbb{N}$.

Therefore we can apply Theorem 5.1 which yields the existence of a neighborhood $\mathcal{N}$ of (w^*, λ^*, μ^*) so that for all $(w_k, \lambda_k, \mu_k) \in \mathcal{N}$ and for all $k \in \mathbb{N}$ the QP

$$\begin{aligned} \min_{\Delta w} \quad & \frac{1}{2} \Delta w^\intercal B_k \, \Delta w + b(w_k, \lambda_k, \mu_k)^\intercal \Delta w && \text{(5.26a)} \\ \text{s. t.} \quad & c(w_k) + C_k \Delta w = 0, && \text{(5.26b)} \\ & d(w_k) + D_k \Delta w \geq 0, && \text{(5.26c)} \end{aligned}$$

has a unique global solution and the same active set $\mathcal{A}(w^*)$ as the NLP solution (w.l.o.g. we assume that $\mathcal{A}(w^*) = 1, \ldots, \tilde{m}$ with $\tilde{m} = |\mathcal{A}(w^*)|$). By the equivalence between SQP and Newton's method we can then calculate this solution not only by (5.22) but equivalently by solving

$$J(B_k, C_k, D_k) \begin{pmatrix} s_k \\ u_k \\ v_k \end{pmatrix} = \underbrace{\begin{pmatrix} -\nabla_w \mathcal{L}(w_k, \lambda_k, \mu_k) \\ -c(w_k) \\ -\widetilde{d}(w_k) \end{pmatrix}}_{\triangleq -\Phi(y_k),\ y_k \triangleq (w_k, \lambda_k, \mu_k)}. \tag{5.27}$$

The sequence generated by choosing a $y_0 \triangleq (w_0, \lambda_0, \tilde{\mu}_0, 0) \in \mathcal{N}$ and iterating $y_{k+1} = y_k + \Delta y_k$ with $\Delta y_k = (s_k, u_k, v_k, 0)$ is then well defined as long as we can guarantee that the iterates remain in $\mathcal{N}$. To show this as well as the convergence of the sequence to (w^*, λ^*, μ^*) we apply Lemma 5.2. In order to apply the lemma, the assumptions of the lemma have to be satisfied.

Because the sequences (B_k), (C_k) and (D_k) are uniformly bounded one can easily see that the sequence (J_k) is uniformly bounded, too. It remains to determine the neighborhood $\bar{\mathcal{N}}$ such that $\mathcal{B}\left(y_0, \frac{\|\Delta y_0\|}{1-\kappa}\right) \subset \mathcal{N}$ for all $y_0 \in \bar{\mathcal{N}}$. Because $\mathcal{N}$ is a neighborhood of $y^* = (w^*, \lambda^*, \mu^*)$, there is an $\epsilon > 0$ such that $\mathcal{B}(y^*, \epsilon) \subset \mathcal{N}$. Since $\Phi(y^*) = 0$ the first step $\Delta y_0 = -J_0^{-1} \, \Phi(y_0)$ can be made arbitrarily small if the distance $\|y^* - y_0\|$ is small enough, due to the differentiability of Φ and the boundedness of J_0^{-1}. For the given $\epsilon > 0$ we can therefore find a $\delta > 0$ such that $\|y^* - y_0\| + \frac{\|\Delta y_0\|}{1-\kappa} \leq \epsilon$ whenever $\|y^* - y_0\| \leq \delta$. Hence, for all $y_0 \in \bar{\mathcal{N}} := \mathcal{B}(y^*, \delta)$ we can guarantee that $\mathcal{B}(y_0, \frac{\|\Delta y_0\|}{1-\kappa}) \subset \mathcal{B}(y^*, \epsilon) \subset \mathcal{N}$. Now the convergence of the sequence (w_k, λ_k, μ_k) towards (w^*, λ^*, μ^*) follows from Lemma 5.2. □

This finishes the proof for the local convergence of level-C iterations if applied to the same problem in each iteration, i.e., for the case $x_0^k = x_0$. The application of level-D iterations to the same problem in each iteration amounts to standard SQP, and local convergence theory can be found in standard textbooks, cf. [140]. However, local convergence can also be obtained by Theorems 5.1 and 5.3, since level-D iterations are recovered

from level-C iterations by choosing the exact constraint Jacobians as constraint Jacobian approximations C_k and D_k. Note that for this choice the modified gradient b_k simply amounts to the objective gradient $\nabla b(w)$, as in standard SQP theory.

Concerning the local iteration of level-B iterations to the same problem in each iteration, it has been shown in [29] that if the iteration converges to a limit w^*, and (λ^*, μ^*) are the multipliers of the level-B QP in w^*, then (w^*, λ^*, μ^*) is a KKT point of the problem

$$\begin{aligned} \underset{w}{\text{minimize}} \quad & \frac{1}{2}\left(w-\bar{w}\right)^\intercal B\left(w-\bar{w}\right) + \left(\bar{b}+e\right)^\intercal w && \text{(5.28a)} \\ \text{subject to} \quad & 0 = c(w), && \text{(5.28b)} \\ & 0 \leq d(w), && \text{(5.28c)} \end{aligned}$$

with $e \triangleq \left(\frac{\mathrm{d}c}{\mathrm{d}w}(w^*) - C\right)^\intercal \lambda^* + \left(\frac{\mathrm{d}d}{\mathrm{d}w}(w^*) - D\right)^\intercal \mu^*$ and $B, C, D, \bar{w}, \bar{b}$ are the Hessian approximation, the equality and inequality constraint Jacobian approximations, the reference trajectory, and the reference gradient, respectively, see the description of level-B in Section 5.2. In the case of level-B iterations for a least-squares objective function, if the iteration converges to a limit w^*, and (λ^*, μ^*) are the multipliers of the level-B QP in w^*, then (w^*, λ^*, μ^*) is a KKT point of the problem

$$\begin{aligned} \underset{w}{\text{minimize}} \quad & \frac{1}{2}\left\|r(w)\right\|_2^2 + \tilde{e}^\intercal w && \text{(5.29a)} \\ \text{subject to} \quad & 0 = c(w), && \text{(5.29b)} \\ & 0 \leq d(w), && \text{(5.29c)} \end{aligned}$$

with $e \triangleq \left(\frac{\mathrm{d}c}{\mathrm{d}w}(w^*) - C\right)^\intercal \lambda^* + \left(\frac{\mathrm{d}d}{\mathrm{d}w}(w^*) - D\right)^\intercal \mu^* - \left(\frac{\mathrm{d}r}{\mathrm{d}w}(w^*) - J\right)^\intercal r(w^*)$ and J, C, D are the least-squares Jacobian approximation and the equality and inequality constraint Jacobian approximations.

With these results, one can prove stability of the active set in a neighborhood of the KKT points analogously to and under the conditions of Theorem 5.1. With Φ defined as

$$\Phi(y) \triangleq \Phi(w, \lambda, \mu) \triangleq \begin{pmatrix} \bar{b} + B\left(w-\bar{w}\right) - C^\intercal \lambda - \widetilde{D}^\intercal \mu \\ c(w) \\ \widetilde{d}(w) \end{pmatrix} \tag{5.30}$$

in the case of an economical objective and

$$\Phi(y) \triangleq \Phi(w, \lambda, \mu) \triangleq \begin{pmatrix} J^\intercal r(w) - C^\intercal \lambda - \widetilde{D}^\intercal \mu \\ c(w) \\ \widetilde{d}(w) \end{pmatrix} \tag{5.31}$$

in the case of a least-squares objective, one can prove local convergence of the level-B iterations analogously to Theorem 5.3. It should be noted, that the limit points of level-B iterations are feasible with respect to the original problem (2.24), however, in general they do not satisfy the necessary KKT conditions of (2.24) and thus are suboptimal.

Convergence results for the general case with varying initial states x_0^k are technically more difficult because the combined system-controller dynamic has to be taken into account. Existing work addresses the RTI scheme and presents stability proofs for various

problem formulations: RTI with shifting and a zero end point constraint in [58], RTI with shrinking horizon for batch processes in [54], RTI with receding horizon without shifting in [57], and RTI with inequality constraints in [174]. However, the proofs are formulated in terms of inexact Newton methods similar to Theorems 5.1 and 5.3 and do not require exact constraint Jacobians. Thus, the proofs are also valid for MLI schemes with pure level-C iterations and with mixed level-C and level-D iterations for the maximum data communication, cf. Algorithm 4. MLI schemes using level-B iterations are not covered by the existing theory because the inexact Newton schemes are required to converge to solutions of the NLPs (2.24).

5.5 Mixed-level and fractional-level iterations

The approach presented in this section has first been published in J.V. Frasch, L. Wirsching, S. Sager, and H.G. Bock, *Mixed-level iteration schemes for nonlinear model predictive control* [85]. The text at hand follows closely the presentation in the cited work.

The idea of hierarchically updating the data in the feedback-generating QPs in the various MLI levels can be carried further by not only choosing different update levels for the full horizon at different sampling times but also by choosing different update levels over the prediction horizon *within* one sample. This approach is motivated by the observation that in each sample the approximation of an *open-loop solution* of OCP (3.1) is calculated over the whole NMPC horizon. It therefore can be expected that this solution approximation becomes more and more outdated as the process evolves, since disturbances or measurement errors add up and require a recalculation at later sampling times. Hence, when performing an MLI iteration for a single instance of OCP (3.1), it is more important to model the process well in earlier parts of the prediction horizon than in later parts. Therefore, two modifications to the MLI approach are proposed:

Fractional-level iterations apply one of the update levels described in Chapter 5 only to the first $N_{\text{frac}} < N$ Multiple Shooting intervals. For example, a fractional-D (or D$'$) iteration only evaluates new constraint residuals and Jacobians, objective gradients, and Hessian approximations on the first N_{frac} Multiple Shooting intervals and reuses the old data for the other intervals.

Mixed-level iterations apply one of the update levels D or C as described in Chapter 5 to the first N_{frac} multiple shooting intervals and another update level lower in the hierarchy for the other intervals. For example, a D/B iteration evaluates new constraint residuals and Jacobians, objective gradients, and Hessian approximations on the first N_{frac} Multiple Shooting intervals and evaluates new constraint residuals and gradient approximations on the other intervals.

The most obvious benefit of using fractional-level or mixed-level iterations are the savings in computational effort for function and derivative evaluation. However, further savings arise in D/$\cdot$ and D' iterations by using a tailored condensing for the QP solution, see Chapter 7. For further details and a numerical test case the reader is referred to [85].

6 Adaptive level choice for MLI

In the preceding chapter, we presented the various MLI levels and discussed how to assemble MLI schemes with fixed level choices. In this chapter, we deal with the naturally arising question how to choose the levels adaptively. The goal of an adaptive level choice is to be computationally as efficient as possible, i.e., use lower levels as often as possible, while ensuring the quality of the feedback generating scheme by applying higher levels if needed.

We start with preliminary considerations, then we present two approaches for the estimation of the contraction rate, and formulate and discuss an adaptive level choice algorithm which makes use of these estimations. For the case of an MHE estimator we present a criterion that may also be used in the decision algorithm for the adaptive level choice.

Finally, we outline a feedback approach, which, while not fitting exactly in the MLI approach of this thesis, makes use of the contraction rate estimates and is motivated by the stability results for suboptimal NMPC given in [168, 156].

6.1 Adaptive level selection: preliminary considerations

The adaptive level selection for the MLI schemes aims at two objectives which, in general, contradict each other, namely numerical efficiency and scheme quality. Here, scheme quality is determined by the difference of an MLI iteration to a single exact SQP iteration initialized from the current set of primal-dual variables. MLI scheme quality is in general improved by using higher level iterations. In particular, level-D iterations are the best possible choice with respect to scheme quality. However, level-D iterations are also the computationally most expensive choice, in general exceeding the computational cost for level-C and level B iterations by far. Between level-C and level-B, the assumption that function evaluation is the dominant part of the computational effort per iteration, together with a well-known result from automatic differentiation which bounds the cost of a gradient computation to at most five times the cost for the respective function evaluation [90], yields that level-C iterations are expected to be at most five times more computationally expensive than level-B iterations.

Thus, an adaptive level selection algorithm which makes use of all levels has to address two main questions: first, when to schedule a level-D iteration, and second, how to choose between level-B and level C iterations. In this work, we propose to address the first question by looking at the *contraction* of the currently used level iteration. This is motivated by the stability results for the RTI scheme from [54] and related work, where it was pointed out that contraction of the current feedback scheme is essential for the contraction of the combined controller-state system, as well as the results for a fixed current state x_0 given

in the last chapter, where contraction implies convergence to a local solution, which is optimal in the case of level-C iterations and feasible in the case of level-B iterations.

Using the contraction rate also as a sole criterion for the second question is not suitable, as even in the case of good contraction, the fact that level-B iterations are converging to suboptimal points may lead to suboptimal feedback. We propose to use the QP right-hand side of the iterations, i.e., the norm of the constraint vector and of the Lagrange gradient or its level-B equivalent. The main idea is to force level-C iterations if the right-hand side in level-B iterations becomes too small, thus improving the right-hand side and guiding the feedback towards optimality.

6.2 Contraction rate estimates by postiterations

The first approach to estimate the contraction rate δ of the current feedback scheme is by *postiterations*, i.e., by making additional iterations after the feedback phase while keeping x_0 fixed. If the active set remains the same, making additional iterations amounts to the application of a simplified Newton to the level-B or level-C right hand side, respectively. Thus, motivated by the a priori estimate in Theorem 2.5 (Local Contraction Theorem) the contraction rate can be estimated as described in Algorithm 6. In practice, the postiterations will usually contract even if the correct active set is not determined by the first iteration as long as the constraints are approximated well enough in the QP [29].

Input: x_0, step $(\Delta w, \Delta\lambda, \Delta\mu)$, current estimate (w, λ, μ), current active set $\mathbb{A}$

switch *current level* **do**

  case *level-B* **do**

    evaluate constraint vectors c_k, d_k

    calculate gradient approximation b_k as in (5.5) or (5.6)

    solve QP (5.1) to obtain step $(\Delta w^+, \Delta\lambda^+, \Delta\mu^+)$ and active set $\mathbb{A}^+$

  case *level-C* **do**

    evaluate constraint vectors c_k, d_k

    calculate modified gradient b_k as in (5.3)

    solve QP (5.1) to obtain step $(\Delta w^+, \Delta\lambda^+, \Delta\mu^+)$ and active set $\mathbb{A}^+$

if $\mathbb{A}^+ = \mathbb{A}$ **then**

  estimate $\hat{\delta} = \frac{\|(\Delta w^+, \Delta\lambda^+, \Delta\mu^+)\|}{\|(\Delta w, \Delta\lambda, \Delta\mu)\|}$

iterate $(w^+, \lambda^+, \mu^+) = (w, \lambda, \mu) + (\Delta w^+, \Delta\lambda^+, \Delta\mu^+)$

Algorithm 6: Estimation of δ by postiterations.

If the estimate $\hat{\delta}$ satisfies $\hat{\delta} \leq \Delta < 1$, with a prescribed contraction bound Δ, then the contraction of the scheme is assumed to be satisfactory. As a side benefit, in this case we can use the improved primal-dual guess (w^+, λ^+, μ^+) in the preparation phase for the next sampling period. Even more, the constraints and the gradient (approximation), which have been updated during the postiteration process, can be used in an improved controller if the subsequent MLI iteration is a level-A iteration. If the estimate $\hat{\delta}$ is too

large or even exceeds 1 then the linearization information is considered outdated and we have to schedule a level-D iteration for the next sampling period.

In the case of changing active sets we may repeat the postiteration process until we obtain subsequent iterations with the same active set. Provided that the NMPC subproblems are well initialized by the solution of the previous subproblems, which is the basic assumption for the validity of the Real-Time Iteration idea, cf. Chapter 4, we can assume that not too many iterations are necessary to obtain an estimate $\hat{\delta}$. An interesting and computationally less expensive but more heuristic alternative is to just *assume* that the active set keeps constant and continue the iteration with the matrix M which is given implicitly by the current factorization of the KKT matrix of the QP, i.e., to backsolve the QP matrix factorization belonging to the current active set with the new right-hand-side to obtain the step $(\Delta w^+, \Delta\lambda^+, \Delta\mu^+)$.

Algorithm 6 may calculate meaningless $\hat{\delta}$ estimates if the size of the step $(\Delta w, \Delta\lambda, \Delta\mu)$ is in the scale of the prescribed function evaluation accuracy, i.e., if the current estimate (w, λ, μ) is already close to the numerical solution. In this case we cannot see the theoretically predicted linear decrease in the norm of the step increments. However, being already in or close to the solution is a situation which is highly desirable in the first place and which makes contraction estimation unnecessary. In the adaptive level choice algorithm, this case will be considered by tracking feasibility and (approximated) optimality in addition to contractivity.

Since the estimate $\hat{\delta}$ by Algorithm 6 is actually an underestimate of the real contraction rate δ, it could be possible that a small $\hat{\delta}$ doeas not necessarily imply a small δ. This is hard to detect since by definition of $\hat{\delta}$ the increments then still behave as if the iteration scheme is contracting. In that case it is to be expected that later iterations eventually detect possibly insufficient contraction behavior. Furthermore, the estimate $\hat{\delta}$ depends on the norm chosen to measure the increment size. It can happen that for an overall contractive matrix iteration the first iterates do not get smaller if measured in the wrong norm, see [147].

A more robust indicator of asymptotic contraction behavior than $\hat{\delta}$ is given by the spectral radius of a matrix. We will present a contraction rate estimator based on the spectral radius in the following section.

6.3 Contraction rate estimates by spectral radius

After solving QP (5.1) in a level-B or level-C iteration we obtain an active set $\mathbb{A}$ so that we can rewrite the QP as purely equality-constrained QP

$$\begin{aligned} \underset{\Delta w}{\text{minimize}} \quad & \frac{1}{2}\Delta w^\intercal B_k \Delta w + b_k^\intercal \Delta w && \text{(6.1a)} \\ \text{subject to} \quad & 0 = \widetilde{C}_k \Delta w + \widetilde{c}_k, && \text{(6.1b)} \\ & 0 = \widetilde{D}_k \Delta w + \widetilde{d}_k, && \text{(6.1c)} \end{aligned}$$

where the constraints (6.1b) combine the IVE constraint (5.1b) and the equality constraints (5.1c), and the constraints (6.1c) are the subset of active inequality constraints

(5.1d) selected by the active set $\mathbb{A}$. The QPs differ for level-B and level-C in the choice of b_k, which is either one of the gradient approximations (5.5) and (5.6) or the modified gradient (5.3). We can transform (6.1) to a linear system by writing down the KKT conditions

$$\begin{pmatrix} B_k & -\widetilde{C}_k^\intercal & -\widetilde{D}_k^\intercal \\ \widetilde{C}_k & & \\ \widetilde{D}_k & & \end{pmatrix} \begin{pmatrix} \Delta w \\ \lambda^{\mathrm{QP}} \\ \mu_{\mathbb{A}}^{\mathrm{QP}} \end{pmatrix} = \begin{pmatrix} -b_k \\ -\widetilde{c}_k \\ -\widetilde{d}_k \end{pmatrix}. \tag{6.2}$$

Adding the term $\widetilde{C}_k^\intercal \lambda_k + \widetilde{D}_k^\intercal \mu_{\mathbb{A},k}$ to the first block equation in (6.2), yields a linear system which determines the step in both primal and dual variables

$$\begin{pmatrix} B_k & -\widetilde{C}_k^\intercal & -\widetilde{D}_k^\intercal \\ \widetilde{C}_k & & \\ \widetilde{D}_k & & \end{pmatrix} \begin{pmatrix} \Delta w \\ \Delta\lambda_k \\ \Delta\mu_{\mathbb{A},k} \end{pmatrix} = \begin{pmatrix} -b_k + \widetilde{C}_k^\intercal \lambda_k + \widetilde{D}_k^\intercal \mu_{\mathbb{A},k} \\ -\widetilde{c}_k \\ -\widetilde{d}_k \end{pmatrix}. \tag{6.3}$$

Using these reformulations, we can consider the QP solution as one step of an inexact Newton method with

$$x_{k+1} = x_k - MF(x_k), \tag{6.4}$$

where $x_k = (w_k, \lambda_k, \mu_{\mathbb{A},k})$, and

$$M \triangleq \begin{pmatrix} B_k & -\widetilde{C}_k^\intercal & -\widetilde{D}_k^\intercal \\ \widetilde{C}_k & & \\ \widetilde{D}_k & & \end{pmatrix}^{-1} \tag{6.5}$$

approximates the inverse of the KKT matrix

$$K_{\mathbb{A}} \triangleq \begin{pmatrix} \nabla^2 \mathcal{L}(w_k, \lambda_k, \mu_{\mathbb{A},k}) & -J_{c,k}^\intercal & -J_{d,k}^\intercal \\ J_{c,k} & & \\ J_{d,k} & & \end{pmatrix}, \tag{6.6}$$

where $J_{c,k}$ is the Jacobian of the combined IVE and equality constraints and $J_{d,k}$ is the Jacobian of the active inequality constraints selected by the active set $\mathbb{A}$. The right hand side $f(x_k)$ in (6.4) comprises the gradient (approximation) with the step correction, the combined equality constraints and the active inequality constraints. For level B, $F(x_k)$ is given by

$$F^{\mathrm{B}}(x_k) = \begin{pmatrix} b_k - \widetilde{C}_k^\intercal \lambda_k - \widetilde{D}_k^\intercal \mu_{\mathbb{A},k} \\ \widetilde{c}_k \\ \widetilde{d}_k \end{pmatrix}, \tag{6.7}$$

with b_k either given by (5.5) or (5.6). For level-C, $F(x_k)$ is given by

$$F^{\mathrm{C}}(x_k) = \begin{pmatrix} \nabla_w \mathcal{L}(w_k, \lambda_k, \mu_{\mathbb{A},k}) \\ \widetilde{c}_k \\ \widetilde{d}_k \end{pmatrix}. \tag{6.8}$$

The Jacobian of the right-hand-sides (6.7) and (6.8) are

$$J^{\mathrm{B}}(x_k) = \begin{pmatrix} B_k & -\widetilde{C}_k^{\mathsf{T}} & -\widetilde{D}_k^{\mathsf{T}} \\ J_{c,k} & 0 & 0 \\ J_{d,k} & 0 & 0 \end{pmatrix} \tag{6.9}$$

and

$$J^{\mathrm{C}}(x_k) = \begin{pmatrix} \nabla^2 \mathcal{L}(w_k, \lambda_k, \mu_{\mathbb{A},k}) & -J_{c,k}^{\mathsf{T}} & -J_{d,k}^{\mathsf{T}} \\ J_{c,k} & 0 & 0 \\ J_{d,k} & 0 & 0 \end{pmatrix}, \tag{6.10}$$

respectively.

For the adaptive level choice algorithm, the question is to determine whether the matrix M in (6.4) is suitable for further use or if M should be updated by a level-D iteration. In contrast to the approach in the last section, which, under the assumption of a constant active set, amounts to continuing the iteration (6.4) and estimating the contraction by comparing the magnitude of the iteration steps, we will investigate directly properties of the matrix M. To this end, we consider the matrix (with F one of $F^{\mathrm{B}}, F^{\mathrm{C}}$)

$$V = \mathbb{I} - M \frac{\mathrm{d}F}{\mathrm{d}x}(x_{k+1}) \tag{6.11}$$

which should be small in some sense if M is a good approximation of $\frac{\mathrm{d}F}{\mathrm{d}x}^{-1}(x_{k+1})$. In particular, we propose to estimate the *spectral radius* $\sigma(V)$ and check for $\sigma(V) \leq \kappa_0 < 1$. In this case, the following well-known theorem, see, e.g., Saad [160, Theorem 4.1], holds, see also [147].

Theorem 6.1. *Let*

$$\kappa \triangleq \sigma(V). \tag{6.12}$$

If $\sigma(V) < 1$ then M and $\frac{\mathrm{d}F}{\mathrm{d}x}(x_k)$ are invertible and the iteration

$$\chi_{j+1} = \chi_j - M\left(\frac{\mathrm{d}F}{\mathrm{d}x}(x_{k+1})\chi_j + F(x_{k+1})\right) = V\chi_j - MF(x_{k+1}) \tag{6.13}$$

converges for every $F(x_{k+1}), \chi_0$ to the solution of

$$\frac{\mathrm{d}F}{\mathrm{d}x}(x_{k+1})\chi = F(x_{k+1}). \tag{6.14}$$

The asymptotic R-linear convergence factor is κ.

Theorem 6.1 states that in principle the matrix M is suitable to recover the exact Newton step for the right-hand-side $F(x)$ through the linear iteration (6.13), provided that the condition (6.12) is satisfied. We consider this to be a good and promising criterion for the usefulness of M as an approximation of $\frac{\mathrm{d}F}{\mathrm{d}x}^{-1}(x_{k+1})$.

For the level-C, iteration (6.13) would actually provide the corresponding level-D iteration step with exact Hessian, which is the optimal step choice within our algorithmic framework from a contraction-based point of view. For level-B, iteration (6.13) would

provide the level-D step for the modified problem (5.28) or (5.29), which is simply an implication from the suboptimality of the level-B iterations.

However, we are not interested in performing too many steps of iteration (6.13), since this would be computationally too expensive. Our main focus is in deciding whether $\sigma(V) \leq \kappa_0 < 1$ holds true or not. This also means that we need to determine only a rather coarse approximation of $\sigma(V)$ up to, e.g., two digits.

Although convergence of the linear iteration is guaranteed as long as $\kappa < 1$ holds, smaller values of κ are preferable since then M is a better approximation of $\frac{\mathrm{d}F}{\mathrm{d}x}^{-1}(x_{k+1})$ and the asymptotic linear convergence factor is smaller, i.e., fewer iterations are necessary to obtain a specified accuracy. We impose a maximum threshold value κ_0 which is a tuning parameter for the adaptive level choice algorithm and which indicates the upper bound for acceptable contraction.

For reasons of computational effort we want to avoid building either M or $\frac{dF}{dx}(x_{k+1})$ explicitly but rather use iterative methods for the estimation of the largest eigenvalue which only demand matrix-vector products with the investigated matrix. We consider efficient numerical methods for the spectral radius estimation in Chapter 7.

To sum up the comparison between the postiteration approach and the spectral radius approach: the former checks if continuing the nonlinear iteration with constant matrices leads to contraction, the latter checks if the next step can be bend towards the Newton direction by a linear iteration using the current QP inverse M. In principle, one could also consider the matrix $V = I - MJ^{\mathrm{B|C}}(x_k)$ of the current step, however, we have to calculate $F(x_{k+1})$ for the adaptive level choice algorithm anyway, and thus looking at the next step for a decision about the upcoming level choice seems more logical.

6.4 Adaptive level choice algorithm

The methods described above in this chapter aim to quantify the quality of a current feedback generating scheme using either level-B or level-C purely from a contractivity point of view. Ideally, we would like to calculate feedback using only these levels due to their favorable computational complexity. Level-D iterations are considered as an instrument to improve contraction properties only if necessary. Since level-D iterations are the best we can do regarding contractivity improvement, we do not need to estimate the contraction rate for level-D iterations.

However, an adaptive level choice algorithm which is solely based on contraction estimates for the levels B and C, selecting in each sample the level with the smallest contraction estimate, is likely to produce suboptimal feedback which steers the system to undesired operating points. The reason for this is that the level-B iterations may have excellent contraction properties, and thus being chosen in each sample, yet still converge to suboptimal points which result in suboptimal feedback.

Thus, an effective adaptive level choice algorithm has to take into account additional information. We propose to use the norm of the QP right-hand side $F^{\mathrm{B}}(x_{k+1})$ in level-B iterations. This is motivated by the interpretation of MLI as continuously iterating inexact

SQP method with changing parameters. If the level-B iteration shows good contraction and the controller steers the system to some fixed operating point we can expect the norm of the right-hand-side to shrink, indicating convergence to a critical point of the modified problem (5.28) or (5.29). As discussed above, this may well be a suboptimal operating point, however, it is the best point currently achievable by level-B iterations. To improve the controller performance, we therefore switch to level-C iterations if the norm of the QP right-hand side gets smaller than some prescribed value, even if the level-B contraction estimates are good. This allows to improve the performance of the controller towards optimality.

While making level-C iterations we do not mind a small right-hand side norm since this indicates optimality of the controller. We propose, however, to check the norm of the Lagrange gradient $\nabla_w \mathcal{L}(x_{k+1})$ and continue with a level-D iteration if the norm is larger than a prescribed value. This avoids using level-C iterations for iterates that are too far away from a critical point.

For both level-B and level-C iterations, we continue with level-D iterations if the estimated contraction κ exceeds prescribed contraction bounds κ_{B} or κ_{C}, respectively. For reasons of computational efficiency, after each level-D iteration we schedule a lower level iteration, preferably a level-B iteration. The described adaptive level selection is summed up in Algorithm 7. A vizualization of the adaptive level selection as finite state machine is given in Figure 6.1.

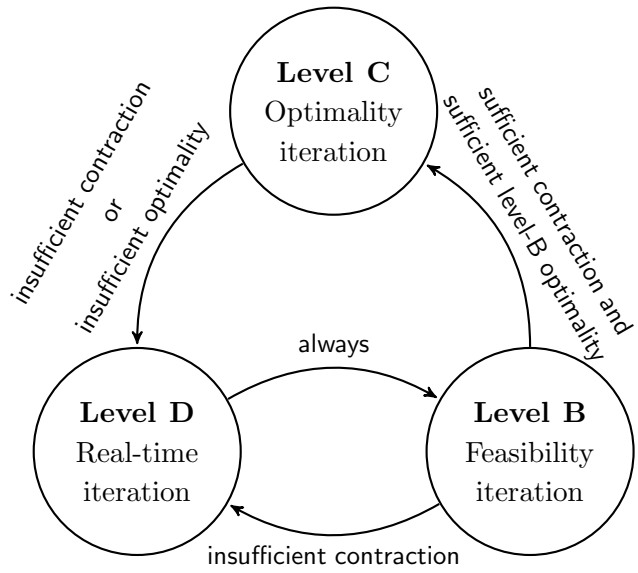


Figure 6.1: Adaptive level choice as finite state machine. Visualization of Algorithm 7.

Of course, Algorithm 7 can also be used with the postiteration estimation approach, using thresholds $\delta_{\mathrm{B}}, \delta_{\mathrm{C}}$ for the nonlinear contraction estimate instead of $\kappa_{\mathrm{B}}, \kappa_{\mathrm{C}}$ for the linear contraction estimate.

Input : $\kappa_{\mathrm{B}}, \kappa_{\mathrm{C}}$ contraction bounds for level-B and C,
γ_{B} right-hand-side bound for level-B,
λ_{C} Lagrange gradient bound for level-C,
l old level

Output: Level choice l^+ for next iteration

Calculate feedback with the current level selection and obtain x_{k+1}.
switch l **do**
  case B **do**
    Estimate spectral radius κ for level-B.
    Evaluate right-hand-side norm $\gamma = \left\| F^{\mathrm{B}}(x_{k+1}) \right\|$.
    if $\kappa \leq \kappa_{\mathrm{B}}$ **then**
      if $\gamma \leq \gamma_{\mathrm{B}}$ **then**
        Set $l^+ = \mathrm{C}$.
      else
        Set $l^+ = \mathrm{B}$.
    else
      Set $l^+ = \mathrm{D}$.
  case C **do**
    Estimate spectral radius κ for level-C.
    Evaluate Lagrange gradient norm $\lambda = \|\nabla_w \mathcal{L}(x_{k+1})\|$.
    if $\lambda \geq \lambda_{\mathrm{C}}$ **then**
      Set $l^+ = \mathrm{D}$.
    else
      if $\kappa \leq \kappa_{\mathrm{C}}$ **then**
        Set $l^+ = \mathrm{C}$.
      else
        Set $l^+ = \mathrm{D}$.
  case D **do**
    Set $l^+ = \mathrm{B}$.

Algorithm 7: Decision algorithm for the adaptive level choice.

6.5 χ^2 criterion for MHE

Another reason to choose level-D iterations is the case of changes in the model, e.g., jumps in parameters or states. These jumps can be detected by using online state and parameter estimation such as the MHE, see also Chapter 3. The state and parameter values given to the controller will gradually change to the new values with the corresponding data moving through the estimation horizon. However, we propose to additionally use the following idea to determine *whether* such a jump event probably has occurred so that the controller can take appropriate action, e.g., by recalculating system linearizations with level-D iterations.

The idea is to check the result of the state and parameter estimation by employing the χ^2-test. This test checks the hypothesis that the difference of the measurements and the model response is $\mathcal{N}(0, \sigma^2)$ distributed. We make use of the following theorem.

Theorem 6.2 (χ^2-test)**.** *Let $\eta_1, \ldots, \eta_n$ be independently $\mathcal{N}(\mu, \sigma^2)$-distributed measurements with unknown mean μ and unknown variance σ^2. Define*

$$T(\eta_1, \ldots, \eta_n) \triangleq \frac{n-1}{\sigma_0^2} \sum_{i=1}^{n} (\eta_i - \bar{\eta})^2,$$

with $\bar{\eta} = \frac{1}{n} \sum_{i=1}^{n} \eta_i$ the sample mean and some σ_0. The null hypothesis $\sigma = \sigma_0$ is to be rejected if $T(\eta_1, \ldots, \eta_n) > \chi^2_{n-1,1-\alpha}$ for some given confidence level $\alpha \in (0, 1)$, typically $\alpha = 0.05$ or $\alpha = 0.01$.

The measurement residuals in the MHE problem (3.18) are assumed to have zero mean. However, they can in principle have different variances, e.g., due to different measurement methods. In order to be able to apply Theorem 6.2, the measurement residuals of the MHE problem are normalized with their respective variances to obtain $\mathcal{N}(0, 1)$-distributed values for all measurement residuals. Then we can test the χ^2-distribution of the sum of squares of the normalized measurement residuals and, assuming that the state and parameter estimates are close to the true values, conclude if the data in the estimation process can be consistently explained by the estimates.

If the model is correct, i.e., if the true parameters and states have been estimated, and the process data is explained by these estimates, then the least-squares residual will be χ^2-distributed with $n - 1$ degrees of freedom.

Failure of the test indicates significant changes in the process from which the measurements are taken. This can be due to jumps in process parameters or state disturbances. Then one part of the measurement data in the estimation horizon is based on the states and parameters before the disturbance and the other part is based on the states and parameters after the disturbance, which prohibits the explanation of the overall data with one set of state and parameter estimates. In this case, the estimates given by the estimator should be taken with caution. Several reactions may be considered:

- In case of Multi-Level Iterations for the estimator, we should perform level-D iterations for the estimator.

- In case of Multi-Level Iterations for the controller, we should perform level-D iterations for the controller.
- A failed test may trigger robust or experimental design iterations to reduce the impact of wrong state and parameter values or to improve the information gain for estimating the new values.
- One can try to identify the measurements that have been taken after the disturbance, e.g., by using a t-test, re-initialize the MHE with a shorter horizon containing only the measurements taken after the disturbance, and apply the growing horizon initialization until the horizon is filled again completely.

6.6 Suboptimal NMPC with level-B iterations

The final idea presented in this chapter does not exactly fit into the MLI framework because the division in preparation phase, feedback phase, and transition phase does not exist. It rather works like an offline algorithm applied to NMPC. However, it makes use of the computational efficiency of level-B iterations as well as their property to converge to feasible points.

The idea is based on a classical result first published by Scokaert and co-workers [168], and described in detail in [156], which states that for NMPC problems in the setting of Chapter 3, stability essentially follows from finding in each sample feasible states and controls which yield a decrease in the objective value compared to the previous sample.

Level-B iterations provide a computationally efficient way to obtain feasible points. The algorithm works like follows:

1. If x_0^k is available, perform level-B iterations until feasibility up to a prescribed accuracy ϵ is achieved.
2. While iterating, monitor contraction with the postiteration approach described in Section 6.2. If contraction is worse than a prescribed $\bar{\delta}$, perform intermediate level-D iteration (or at least update Hessian approximation and constraint Jacobians).
3. Compare objective value Φ of current sample with objective value Φ^- from previous sample. If $\Phi < \Phi^-$, return first control move to process, otherwise try level-C iterations until $\Phi < \Phi^-$.

An idea to turn the algorithm above into an online algorithm in the spirit of RTI and MLI, the following variant may be considered:

1. In the preparation phase, perform level-B iterations until feasibility up to a prescribed accuracy ϵ is achieved.
2. While iterating, monitor contraction with the postiteration approach described in Section 6.2. If contraction is worse than a prescribed $\bar{\delta}$, perform intermediate level-D iteration (or at least update Hessian approximation and constraint Jacobians).

3. Compare objective value Φ of current sample with objective value Φ^- from previous sample. If $\Phi \geq \Phi^-$, try level-C iterations until $\Phi < \Phi^-$.

(optional) If a feasible solution with lower objective value is obtained, make a single level-D preparation to set up the best possible tangential predictor.

4. If x_0^k is available, evaluate the tangential predictor, i.e., solve the QP, and immediately return the first control to the process.

One can interpret this algorithm as an extension to RTI, where the next sample iteration is initialized in a point that is at least iterated to feasibility instead of simply using the tangential predictor of the previous sample for initialization.

The algorithm assumes that the sampling interval is long enough to obtain the feasible point with lower objective. Furthermore, one actually finds a feasible point of the NLP for the preceding state x_0^{k-1} and compares objective values with a feasible point of the NLP for the state x_0^{k-2}. The new solution for x_0^k is then actually given only approximately by the tangential predictor, which is why setting up a level-D tangential predictor seems reasonable. Furthermore, this approach could be combined with the idea of the *advanced step controller*, cf. [192], where we seek a feasible point of the NLP for a prediction $\hat{x}_0^k$ of the (not yet available) state x_0^k, and then apply the tangential predictor in the feedback phase to account for the difference between the prediction and the actual current state.

7 Numerical methods for Multi-Level Iteration schemes

In this chapter, we consider details for the numerical realization of several tasks that are part of the algorithmic approach of MLI iterations as presented in the last chapters.

First, we consider efficient numerics for solving a sequence of parametric QPs, a task which is central to the MLI approach. We then give an extension of the presented method of *online active set strategy* (OASES) to the solution of parametric linear least-squares problems, which also occur frequently in MLI, in particular if an MHE estimator is used for online state and parameter estimation.

Next, we consider a tailored condensing for mixed- and fractional-level MLI. Aside from function and derivative evaluation, matrix condensing is one of the computationally most expensive parts per MLI iteration. Matrix condensing is needed, whenever the matrix data in the QP is updated, i.e., during level-D iterations for standard MLI or during D' or D/$\cdot$ iterations for fractional- or mixed-level MLI. In the latter case, only part of the matrix data is updated and part of the matrix data remains constant, and this allows a significant reduction of computational effort for condensing.

Finally, we consider efficient iterative methods for the estimation of the spectral radius of the MLI iteration matrix $V = I - MJ$ which is needed as part of the adaptive level choice algorithm to decide whether the current matrices are good enough to obtain contraction of the MLI scheme or whether a level-D iteration has to be scheduled to update the Hessian approximation and the constraint linearizations.

7.1 Parametric quadratic programming

When applying MLI, we have to repeatedly solve the parametric QPs (5.1) for changing values of x_0^k. For the approach considered in this work, the QP matrices are fixed in iterations with the levels C, B, and A. An efficient approach to solve such sequences of parametric QPs is the *Online Active Set Strategy* (OASES). This approach builds on the expectation that the active set does not change much from one QP to the next, but is different from conventional warm starting techniques. It has first been published in [74] and is described and extended extensively in the follow-up publications [75, 148]. The method is implemented in the software package `qpOASES` [73, 76]. OASES was first applied to level-C MLI in L. Wirsching, H.J. Ferreau, H.G. Bock, and M. Diehl, *An online active set strategy for fast adjoint based nonlinear model predictive control* [187], and the following presentation follows widely the presentation in this work.

After condensing of QP (5.1), i.e., elimination of the state variable step Δs_k, we obtain the condensed QP

$$\mathrm{QP}(x_0^k): \min_{\Delta q_k} \quad \tfrac{1}{2}\Delta q_k^\mathsf{T} \hat{B}_k \Delta q_k + \left(\hat{b}_k + F_k x_0^k\right)^\mathsf{T} \Delta q_k \tag{7.1a}$$

$$\text{s. t.} \qquad \hat{D}_k \Delta q_k \geq \hat{d}_k + E_k x_0^k, \tag{7.1b}$$

with suitable matrices $\hat{B}_k$, $\hat{D}_k$, F_k, E_k and vectors $\hat{b}_k$, $\hat{d}_k$. Note that condensing of the QP matrices only occurs in level-D iterations. In iterations of level A, B, and C, the matrices $\hat{B}_k$, $\hat{D}_k$ and therefore also the matrices F_k, E_k are constant and we denote them in the following by $\hat{B}, \hat{D}, F, E$. We only have to build the condensed gradient vector $b_k(x_0^k) \triangleq \hat{b}_k + F x_0^k$ and the condensed constraint vector $d_k(x_0^k) \triangleq \hat{d}_k + E x_0^k$.

For transition from the solved kth quadratic program $\mathrm{QP}(x_0^k)$ to the next one $\mathrm{QP}(x_0^{k+1})$, the Online Active Set Strategy moves on a straight line in the parameter space, i.e., in the set

$$\mathbb{P} \triangleq \{x_0 \in \mathbb{R}^{n_{x_0}} \mid \mathrm{QP}(x_0) \text{ is feasible}\}. \tag{7.2}$$

Using the definitions

$$\begin{aligned}
\Delta x_0 &\triangleq x_0^{k+1} - x_0^k, && \text{(7.3a)}\\
\Delta b &\triangleq b_k(x_0^{k+1}) - b_k(x_0^k) && \text{(7.3b)}\\
&= \left(\hat{b}_{k+1} - \hat{b}_k\right) + F\Delta x_0,\\
\Delta d &\triangleq d_k(x_0^{k+1}) - d_k(x_0^k) && \text{(7.3c)}\\
&= \left(\hat{d}_{k+1} - \hat{d}_k\right) + E\Delta x_0,
\end{aligned}$$

gradient and constraint vector are reparametrized as follows:

$$\tilde{x}_0 \colon [0,1] \to \mathbb{R}^{n_x}, \quad \tilde{x}_0(\tau) \triangleq x_0^k + \tau \Delta x_0\,, \tag{7.4a}$$

$$\tilde{b} \colon [0,1] \to \mathbb{R}^{n}, \quad \tilde{b}_k(\tau) \triangleq b_k(x_0^k) + \tau \Delta b\,, \tag{7.4b}$$

$$\tilde{d} \colon [0,1] \to \mathbb{R}^{m}, \quad \tilde{d}_k(\tau) \triangleq d_k(x_0^k) + \tau \Delta d\,. \tag{7.4c}$$

We start from the known optimal solution x_k^* and μ_k^* (and a corresponding working set $\mathbb{A}$) of the kth $\mathrm{QP}(x_0^k)$ and want to solve $\mathrm{QP}(x_0^{k+1})$. The basic idea of the online active set strategy is to move from x_0^k towards x_0^{k+1}, and thus from (x_k^*, μ_k^*) towards (x_{k+1}^*, μ_{k+1}^*), while keeping primal and dual feasibility (i.e. optimality) for all intermediate points. This means that we are looking for homotopies ($\mathbb{M} \triangleq \{1, \ldots, m\}$)

$$\tilde{x}^* \colon [0,1] \to \mathbb{R}^{n}, \quad \tilde{x}^*(0) = x_k^*,\ \tilde{x}^*(1) = x_{k+1}^*, \tag{7.5a}$$

$$\tilde{\mu}^* \colon [0,1] \to \mathbb{R}^{m}, \quad \tilde{\mu}^*(0) = \mu_k^*,\ \tilde{\mu}^*(1) = \mu_{k+1}^*, \tag{7.5b}$$

$$\tilde{\mathbb{A}} \colon [0,1] \to 2^{\mathbb{M}}, \quad \tilde{\mathbb{A}}(0) = \mathbb{A},\ \tilde{\mathbb{A}}(\tau) \subseteq \mathbb{M}, \tag{7.5c}$$

$$\tilde{\mathbb{I}} \colon [0,1] \to 2^{\mathbb{M}}, \quad \tilde{\mathbb{I}}(\tau) \triangleq \mathbb{M} \setminus \tilde{\mathbb{A}}(\tau), \tag{7.5d}$$

that satisfy the well-known KKT conditions (see e.g. [36]) at every point $\tau \in [0,1]$:

$$\begin{pmatrix} \hat{B} & \hat{D}_{\tilde{\mathbb{A}}(\tau)}^{\intercal} \\ \hat{D}_{\tilde{\mathbb{A}}(\tau)} & 0 \end{pmatrix} \begin{pmatrix} \tilde{x}^*(\tau) \\ -\tilde{\mu}^*_{\tilde{\mathbb{A}}(\tau)}(\tau) \end{pmatrix} = \begin{pmatrix} -\tilde{b}(\tau) \\ \tilde{d}_{\tilde{\mathbb{A}}(\tau)}(\tau) \end{pmatrix}, \tag{7.6a}$$

$$\tilde{\mu}^*_{\tilde{\mathbb{I}}(\tau)}(\tau) = 0\,, \tag{7.6b}$$

$$\hat{D}_{\tilde{\mathbb{I}}(\tau)}\tilde{x}^*(\tau) \geq d_{\tilde{\mathbb{I}}(\tau)}(\tau)\,, \tag{7.6c}$$

$$\tilde{\mu}^*_{\tilde{\mathbb{A}}(\tau)}(\tau) \geq 0\,. \tag{7.6d}$$

Since $\tilde{x}^*(\tau)$ and $\tilde{\mu}^*(\tau)$ are piecewise affine linear functions ($\tilde{x}^*(\tau)$ is even continuous), as already shown in [77], locally we must have a relation of the form

$$\tilde{x}^*(\tau) \triangleq x_k^* + \tau \Delta x^*\,, \tag{7.7a}$$

$$\tilde{\mu}^*_{\mathbb{A}}(\tau) \triangleq (\mu_k^*)_{\mathbb{A}} + \tau \Delta \mu^*_{\mathbb{A}}\,, \tag{7.7b}$$

which holds for sufficiently small $\tau \in [0, \tau_{\max}]$, $\tau_{\max} \geq 0$.

We know that conditions (7.6) are met at $\tau = 0$, as we start from an optimal solution. Therefore equality (7.6a) is satisfied for all $\tau \in [0, \tau_{\max}]$ if and only if

$$\begin{pmatrix} H & \hat{D}_{\mathbb{A}}^{\intercal} \\ \hat{D}_{\mathbb{A}} & 0 \end{pmatrix} \begin{pmatrix} \Delta x^* \\ -\Delta \mu^*_{\mathbb{A}} \end{pmatrix} = \begin{pmatrix} -\Delta b \\ \Delta d_{\mathbb{A}} \end{pmatrix} \tag{7.8}$$

holds. Because linear independence of the rows of $\hat{D}_{\mathbb{A}}$ can easily be ensured [22], Eq. (7.8) has a unique solution.

As long as we stay in one critical region, the QP solution depends affinely on x_0, but it might happen that we have to cross the boundaries of critical regions during our way along the straight line. The active set stays constant as long as no previously inactive constraint becomes active, i.e.

$$\hat{D}_i^{\intercal}(x_k^* + \tau \Delta x^*) = d_i(x_0^k) + \tau \Delta d_i$$

for some $i \in \mathbb{I}$, and no previously active constraint becomes inactive, i.e.

$$(\mu_k^*)_i + \tau \Delta \mu_i = 0$$

for some $i \in \mathbb{A}$. Thus, we determine the maximum possible homotopy step length $\tau_{\max}$ as follows:

$$\tau_{\max}^{\text{prim}} \triangleq \min_{\substack{i \in \mathbb{I} \\ \hat{D}_i^{\intercal} \Delta x^* < \Delta d_i}} \frac{d_i(x_0^k) - \hat{D}_i^{\intercal} x_k^*}{\hat{D}_i^{\intercal} \Delta x^* - \Delta d_i}\,, \tag{7.9a}$$

$$\tau_{\max}^{\text{dual}} \triangleq \min_{\substack{i \in \mathbb{A} \\ \Delta \mu_i < 0}} -\frac{(\mu_k^*)_i}{\Delta \mu_i}\,, \tag{7.9b}$$

$$\tau_{\max} \triangleq \min\left\{1, \tau_{\max}^{\text{prim}}, \tau_{\max}^{\text{dual}}\right\}\,. \tag{7.9c}$$

This choice of $\tau_{\max}$ ensures that conditions (7.6c), (7.6d) remain fulfilled. Moreover, if we define $\Delta \mu_{\mathbb{I}}^* \triangleq 0$ then also equality (7.6b) holds for all $\tau \in [0, \tau_{\max}]$.

If $\tau_{\max}$ equals one, a full step along the homotopy path can be taken and the solution of the new quadratic program $\mathrm{QP}(x_0^{k+1})$ is found. Otherwise, a primal or dual *blocking constraint* – which causes the restriction of $\tau_{\max}$ to below one – indicates a constraint to be added or removed from the working set $\mathbb{A}$. After updating the working set and the KKT matrix in Eq. (7.8), a new step direction is calculated and the whole procedure repeats until the solution of $\mathrm{QP}(x_0^{k+1})$ is found. This idea is illustrated in Fig. 7.1.

Two issues have to be addressed: First, adding a new row d^+ to the constraint matrix $\hat{D}_{\mathbb{A}}$ may lead to rank deficiencies, if d^+ is linearly dependent on the rows of $\hat{D}_{\mathbb{A}}$. This can be checked by solving

$$\begin{pmatrix} H & \hat{D}_{\mathbb{A}}^\intercal \\ \hat{D}_{\mathbb{A}} & 0 \end{pmatrix} \begin{pmatrix} s \\ \xi \end{pmatrix} = \begin{pmatrix} (d^+)^\intercal \\ 0 \end{pmatrix}, \tag{7.10}$$

and checking that $s \neq 0$. Otherwise, another primal blocking constraint has to be found to be added to the active set, or the problem is infeasible.

Second, removing a constraint may expose a direction of zero curvature on the (larger) null space of the new constraint matrix. This can be checked by solving

$$\begin{pmatrix} H & \hat{D}_{\mathbb{A}}^\intercal \\ \hat{D}_{\mathbb{A}} & 0 \end{pmatrix} \begin{pmatrix} s \\ \xi \end{pmatrix} = \begin{pmatrix} 0 \\ -\mathbb{I}_{|\mathbb{A}|,k} \end{pmatrix}, \tag{7.11}$$

where $\mathbb{I}_{|\mathbb{A}|,k}$ is the k-th column of the identity matrix of the size $|\mathbb{A}| \times |\mathbb{A}|$. If $\xi = 0$ then the Hessian H is singular on the null space of the new constraint matrix, and we have to find another dual blocking constraint to be removed from the active set, or the problem is unbounded. Note that in both issues we can reuse the matrix factorization used for the step computation. For a more detailed description of the algorithm, including the handling of ties and degeneracy, the reader is referred to [148].

It should be noted that the idea of OASES to follow the homotopy of a primal-dual optimal solution of $\mathrm{QP}(x_0^k)$ to a primal-dual optimal solution of $\mathrm{QP}(x_0^{k+1})$ can also be extended to the case of a new Hessian and a new constraint matrix, i.e., for level-D iterations, by constructing a primal-dual optimal solution for a help problem $\mathrm{QP}_{\mathrm{new}}(x_0^k)$ with the new matrices and then following the homotopy to the primal-dual solution of $\mathrm{QP}_{\mathrm{new}}(x_0^{k+1})$, cf. [72].

7.1.1 Extension of OASES to linear least-squares problems

Quite regularly, we have to solve a parametric *linear least-squares* (LLS) problem instead of a parametric QP problem during the iterations, e.g., for quadratic tracking-type objectives for the controller or when using RTI- or MLI-based MHE for online state and parameter estimation. Analogously to (7.1), after condensing we obtain the condensed LLS problem

$$\mathrm{LLS}(p^k): \min_{\Delta q_k} \quad \left\| \hat{J}_k \Delta q_k + \left(\hat{r}_k + \tilde{F}_k p^k \right) \right\|_2^2 \tag{7.12a}$$

$$\text{s.t.} \quad \hat{D}_k \Delta q_k \geq \hat{d}_k + \tilde{E}_k p^k, \tag{7.12b}$$

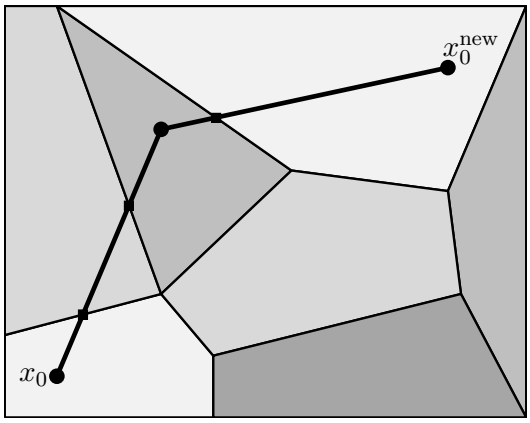


Figure 7.1: Homotopy paths from one QP to the next across multiple critical regions.

again, with suitable matrices $\hat{J}_k, \hat{D}_k, \tilde{F}_k, \tilde{E}_k$ and vectors $\hat{r}_k, \hat{d}_k$. The parameter p^k is either the current state x_0^k for controller problems or the current measurement η^k for estimator problems. As described above, we can assume that the LLS problem matrices are constant between LLS(p^k) and LLS(p^{k+1}), and we will denote them again as $\hat{J}, \hat{D}, \tilde{F}, \tilde{E}$. To see that we can apply OASES also to problem (7.12), we start by noting that the problem can be equivalently written as parametric QP (7.1) with

$$\hat{B} \triangleq 2\hat{J}^\intercal \hat{J}, \tag{7.13a}$$
$$\hat{b}_k \triangleq 2\hat{J}^\intercal \hat{r}_k, \tag{7.13b}$$
$$F \triangleq 2\hat{J}^\intercal \tilde{F}. \tag{7.13c}$$

It follows immediately that OASES can in principle be used to solve problem (7.12) by using the reformulation as QP. However, we want to avoid building the Hessian (7.13a) explicitly, because of possible ill-conditioning due to the well-known relation $cond_2(\hat{J}^\intercal \hat{J}) = cond_2(\hat{J}^\intercal)^2$.

Using the definitions

$$\Delta p \triangleq p^{k+1} - p^k, \tag{7.14a}$$
$$\Delta r \triangleq (\hat{r}_{k+1} - \hat{r}_k) + \tilde{F}\Delta p, \tag{7.14b}$$
$$\Delta d \triangleq \left(\hat{d}_{k+1} - \hat{d}_k\right) + \tilde{E}\Delta p, \tag{7.14c}$$

we can do the re-parametrization of the LS residual and constraint vector analogously to (7.4), and seek homotopies (7.5) that satisfy the KKT conditions (7.6). The key difference to OASES for QPs is that we do not determine the steps $\Delta x^*, -\Delta\mu_{\mathbb{A}}^*$ by using the KKT conditions and solving (7.8), but rather by solving an equivalent LLS problem. Using the definitions for $\Delta r, \Delta d$ from (7.14) and the definition of the Hessian (7.13a), we can derive

from (7.8) the following LS problem for the step calculation.

$$\min_{\Delta x^*} \quad \left\| \hat{J}\Delta x^* + \Delta r \right\|_2^2 \tag{7.15a}$$

$$\text{s.t.} \quad \hat{D}_{\mathbb{A}}\Delta x^* = \Delta d_{\mathbb{A}} \tag{7.15b}$$

There are various approaches to solve problem (7.15), we will give an algorithm based on QR decompositions, derived and presented in [37]. The following conditions for $\hat{J}, \hat{D}_{\mathbb{A}}$ have to hold:

$$\hat{D}_{\mathbb{A}} \text{ has linear independent rows, and } \begin{bmatrix} \hat{J} \\ \hat{D}_{\mathbb{A}} \end{bmatrix} \text{ has linear independent columns.} \tag{7.16}$$

Then we obtain the steps $\Delta x^*, -\Delta\mu_{\mathbb{A}}^*$ by

1. computing the QR factorizations
$$\begin{bmatrix} \hat{J} \\ \hat{D}_{\mathbb{A}} \end{bmatrix} = \begin{bmatrix} Q_1 \\ Q_2 \end{bmatrix} R, \quad Q_2^\intercal = \tilde{Q}\tilde{R},$$
2. computing $u = \tilde{R}^{-\intercal}\Delta d_{\mathbb{A}}$ by forward substitution,
3. solving $\tilde{R}w = -2\tilde{Q}^\intercal Q_1^\intercal \Delta r - 2u$ by back substitution,
4. solving $R\Delta x^* = -Q_1^\intercal \Delta r - \frac{1}{2}Q_2^\intercal w$ by back substitution,
5. computing $-\Delta\mu_{\mathbb{A}}^* = w + 2\Delta d_{\mathbb{A}}$.

With the resulting steps $\Delta x^*, -\Delta\mu_{\mathbb{A}}^*$ we can then determine the maximum possible homotopy step length $\tau_{\max}$ as in (7.9), and continue the homotopy updating $\mathbb{A}$ and $\hat{D}_{\mathbb{A}}$ until problem (7.12) is solved.

Within the presentation of the extension of OASES to LLS we have to address how to perform the linear independence test (7.10) and the zero curvature test (7.11).

For the linear independence test, we have to check for the potential new constraint row d^+ whether there is a $\xi \neq 0$ with $\hat{D}_{\mathbb{A}}^\intercal \xi = (d^+)^\intercal$. With the matrix factorizations from the step calculation we obtain

$$\begin{aligned} \hat{D}_{\mathbb{A}}^\intercal \xi &= (d^+)^\intercal, \\ \Rightarrow R^\intercal \tilde{Q}\tilde{R}\xi &= (d^+)^\intercal, \\ \Rightarrow \tilde{R}\xi &= \tilde{Q}^\intercal R^{-\intercal}(d^+)^\intercal, \end{aligned}$$

which yields that $\xi \neq 0$ if and only if $\tilde{Q}^\intercal R^{-\intercal}(d^+)^\intercal \neq 0$.

For the zero curvature test, we can see by comparison that solving (7.11) is equivalent to solving the LLS

$$\begin{aligned} \min_{s} \quad & \left\| \hat{J}s + 0 \right\|_2^2 \\ \text{s.t.} \quad & \hat{D}_{\mathbb{A}}s = \mathbb{I}_{|_{\mathbb{A}}|,k}, \end{aligned}$$

where again $\mathbb{I}_{|_{\mathbb{A}}|,k}$ is the k-th column of the identity matrix of the size $|_{\mathbb{A}}| \times |_{\mathbb{A}}|$, and checking whether $\Delta\mu_{\mathbb{A}}^* = 0$. For both the linear independence test and the zero curvature test we thus can reuse the matrix factorizations performed for the step computation.

7.2 Condensing for mixed- and fractional-level MLI

In Chapter 5 we have outlined the idea of mixed- and fractional-level MLI, where higher level updates are used in the leading N_{frac} shooting intervals and lower level updates are used in the remaining shooting intervals. In particular, when performing mixed- or fractional-level iterations involving level-D updates, we evaluate new Jacobians and Hessians or approximations of Hessians for the leading shooting intervals, and keep the old ones in the remaining shooting intervals. Apart from the savings of computational effort, this can be also exploited in the condensing algorithm to avoid building parts of the condensed matrices and vectors that do not change because they do not involve the new data. More details and numerical testing can be found in [85], and the presentation follows widely the presentation in this work.

We start by noting that the condensing vector c' does not have to be recalculated in a fractional-D iteration for $N_{\text{frac}} = 1$, which can be seen by the recursion given in (2.36). For $N_{\text{frac}} > 1$ or for mixed-level iterations involving lower levels B or C, however, c' has to be calculated completely with the described recursion.

Consider the condensing matrix in the relation (2.35), which connects the eliminated variables and the variables that remain in the QP. With the definition $\Xi_{i_1}^{i_2} \triangleq \prod_{i=i_1}^{i_2} G_i^s \triangleq G_{i_2}^s \cdots G_{i_1}^s$ we can write this matrix as

$$\tilde{A} \triangleq \begin{pmatrix} G_0^s & G_0^q & & & \\ G_1^s G_0^s & G_1^s G_0^q & G_1^q & & \\ \vdots & \vdots & \vdots & \ddots & \\ \Xi_0^{N-1} & \Xi_1^{N-1} G_0^q & \Xi_2^{N-1} G_1^q & \dots & G_{N-1}^q \end{pmatrix} \tag{7.17}$$

From (7.17), we can see that during fractional-D and D/$\cdot$ iterations the last $N - N_{\text{frac}}$ block columns of $\tilde{A}$ do not change, i.e., we only have to recalculate the first $N_{\text{frac}}+1$ block columns. To efficiently build these first columns, we can recursively build and save the constant blocks $\Xi_{N_{\text{frac}}}^{\overline{i}}, \overline{i} = N_{\text{frac}}+1, \dots, N-1$ during a full level-D update and then only right-multiply the new matrices $G_i^{s|q}, i = 0, \dots, N_{\text{frac}} - 1$ in the appropriate sequence in every following fractional-D or D/$\cdot$ update.

To show the computational savings for the update of the condensed Hessian by a fractional-D or D/$\cdot$ update, we write the condensed Hessian in (2.37) as

$$B_{\text{cond}} = \tilde{A}^\intercal B_{11} \tilde{A} + B_{21} \tilde{A} + \tilde{A}^\intercal B_{12} + B_{22}, \tag{7.18}$$

following the notation in [125], and specify the parts of B_{cond},

$$\tilde{A}^\intercal B_{11} \tilde{A} = \begin{pmatrix} \hat{B}^{s_0,s_0} & \hat{B}_0^{s_0,q} & \hat{B}_1^{s_0,q} & \cdots & \hat{B}_{N-1}^{s_0,q} \\ * & \hat{B}_{0,0}^{q,q} & \hat{B}_{0,1}^{q,q} & \cdots & \hat{B}_{0,N-1}^{q,q} \\ * & * & \hat{B}_{1,1}^{q,q} & \cdots & \hat{B}_{1,N-1}^{q,q} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ * & * & * & \cdots & \hat{B}_{N-1,N-1}^{q,q} \end{pmatrix}, \tag{7.19}$$

where

$$\hat{B}^{s_0,s_0} \triangleq B_0^{ss} + \Xi_0^{0\intercal} B_1^{ss}\ \Xi_0^0 + \Xi_0^{1\intercal} B_{ss}^2\ \Xi_0^1 + \ldots + \Xi_0^{N-1\intercal} B_{ss}^N\ \Xi_0^{N-1}, \tag{7.20a}$$

$$\hat{B}_l^{s_0,q} \triangleq \Xi_0^{l\,\intercal} B_{l+1}^{ss}\ G_l^q + \Xi_0^{l+1\intercal} B_{l+2}^{ss}\ \Xi_{l+1}^{l+1}\ G_l^q + \ldots + \Xi_0^{N-1\intercal} B_N^{ss}\ \Xi_{l+1}^{N-1}\ G_l^q, \tag{7.20b}$$

$l = 0, 1, \ldots, N-1$, and

$$\hat{B}_{k,l}^{q,q} \triangleq G_k^{q\intercal}\ \Xi_{k+1}^{l}{}^{\intercal} B_{l+1}^{ss}\ G_l^q + \ldots + G_k^{q\intercal}\ \Xi_{k+1}^{N-1\intercal} B_N^{ss}\ \Xi_{l+1}^{N-1}\ G_l^q, \tag{7.20c}$$

$k = 0, 1, \ldots, N-1$ and $l = k, k+1, \ldots, N-1$. From (7.19) and (7.20c), we can see that for $k, l \geq N_{\text{frac}}$ all involved Hessian blocks B^{ss} and constraint Jacobian blocks $G^{s|q}$ are constant, thus the lower right $(N - N_{\text{frac}}) \times (N - N_{\text{frac}})$ blocks in (7.19) do not have to be recomputed. Furthermore, for computation of (7.20c), the intermediate terms

$$\left(\Xi_{N_{\text{frac}}}^l\right)^\intercal B_{l+1}^{ss} G_l^q + \ldots + \left(\Xi_{N_{\text{frac}}}^{N-1}\right)^\intercal B_N^{ss}\ \Xi_{l+1}^{N-1}\ G_l^q \tag{7.21}$$

can be saved for $l = N_{\text{frac}}, \ldots, N-1$, thus only requiring left-multiplication with the matrix $G_k^{q\intercal}\ \left(\Xi_{k+1}^{(N_{\text{frac}}-1)}\right)^\intercal$ during a fractional-D or D/$\cdot$ re-condensing. This works analogously for (7.20b). For the second composed part $\tilde{A}^\intercal B_{12}$ of B_{cond} we obtain

$$\tilde{A}^\intercal B_{12} = \begin{pmatrix} 0 & 0 & G_0^s B_1^{sq} & \Xi_0^{1\intercal} B_2^{sq} & \cdots & \Xi_0^{N-2\intercal} B_{N-1}^{sq} \\ & 0 & G_0^q B_1^{sq} & G_0^{q\intercal}\ \Xi_1^{1\intercal} B_2^{sq} & \cdots & G_0^{q\intercal}\ \Xi_1^{N-2\intercal} B_{N-1}^{sq} \\ & & 0 & G_1^{q\intercal}\ B_2^{sq} & \cdots & G_1^{q\intercal}\ \Xi_2^{N-2\intercal} B_{N-1}^{sq} \\ & & & 0 & \ddots & \vdots \\ & & & & \ddots & G_{N-2}^q{}^\intercal B_{N-1}^{sq} \\ & & & & & 0 \end{pmatrix} \tag{7.22}$$

From (7.22) we can see that in the last $N - N_{\text{frac}}$ block rows all involved Hessian blocks B^{sq} and constraint Jacobian blocks $G^{s|q}$ are constant, thus the last $(N - N_{\text{frac}})$ block rows do not have to be recomputed. Since the second addend $B_{21}\tilde{A}$ in (7.18) is the transpose of (7.22), we can conclude that also the lower right $(N - N_{\text{frac}}) \times (N - N_{\text{frac}})$ blocks of the sum of the second and third addend in (7.22) are constant and thus do not have to be recomputed. Finally, the same fact is trivially true for the fourth addend B_{22}, thus the lower right $(N - N_{\text{frac}}) \times (N - N_{\text{frac}})$ blocks of the whole condensed Hessian B_{cond} are constant.

The complexity of a full condensing step can easily be verified to be $O(N^3)$ submatrix-submatrix multiplications, each of which is at most of runtime complexity $O(n^3)$, for $n = n_x + n_u$. Reusing intermediate products as outlined above leads to a computational complexity of $O(N \cdot N_{\text{frac}} + N_{\text{frac}}^3)$ submatrix-submatrix multiplications. Hence, for small N_{frac} we can expect the computational effort for condensing in fractional-D or D/$\cdot$ updates to be only a small part of the effort in full level D updates. It should still be noted that in practice, due to almost triangular shape of $\tilde{A}$, the observed runtime complexity of a full condensing step is rather quadratic in the horizon length (cf. [104]).

7.3 Spectral radius estimation by iterative methods

In Chapter 6 we have discussed an adaptive level choice algorithm for MLI which uses an estimate of the spectral radius of matrix (6.11) to decide if a level-D iteration is necessary to improve contraction. We now discuss how such an estimate can be computed and how to do this efficiently. In the following we use the notation of Chapter 6.

From (6.13) in Theorem 6.1 we can immediately see that

$$(\chi_{j+1} - \chi_j) = V\,(\chi_j - \chi_{j-1}) \tag{7.23}$$

This means, that the iteration (6.13) behaves like a Power iteration, cf. [181, 88]. We can use it to estimate the largest absolute eigenvalue of V, which is by definition the spectral radius of V. Efficient estimators for the spectral radius can be found described and derived in detail in [147], and we briefly outline two of these estimators.

Rayleigh estimator If $V \in \mathbb{R}^{N\times N}$ is diagonalizable, and its eigenvalues $\lambda_i, i = 1,\ldots,N$ are ordered by

$$|\lambda_1| > |\lambda_2| \geq \ldots \geq |\lambda_N|,$$

and $(\chi_1 - \chi_0)$ has a component in the direction of the eigenvector corresponding to λ_1, then

$$\hat{\kappa}_j \triangleq \frac{(\chi_j - \chi_{j-1})^\mathsf{T}\,(\chi_{j+1} - \chi_j)}{(\chi_j - \chi_{j-1})^\mathsf{T}\,(\chi_j - \chi_{j-1})} \to \sigma(V), \quad (j \to \infty)$$

This estimator is computationally cheap and robust, but may converge slowly, if $\frac{|\lambda_2|}{|\lambda_1|}$ is close to 1. Extensions and improvements of this method are, e.g., the Subspace Iteration and the Arnoldi Iteration approach, which work on subspaces with dimension greater than one and have faster convergence. However, they are also computationally significantly more expensive.

Root estimator If $\sigma(V) > 0$ and $(\chi_1 - \chi_0)$ has a component in the dominant invariant subspace corresponding to the eigenvalues of V with the largest modulus, then

$$\hat{\kappa}_{j+1} \triangleq \frac{\|\chi_{j+1} - \chi_j\|^{(1/j)}}{\|\chi_1 - \chi_0\|^{(1/j)}}, \quad j \geq 1,$$

yields an asymptotically correct estimate of $\sigma(V)$ for $j \to \infty$.

For both estimators, we have to perform several steps of iteration (6.13). The computationally most expensive part of one such iteration is the calculation of the product

$$V\chi = M\left(\frac{\mathrm{d}F}{\mathrm{d}x}(x_{k+1})\right)\chi.$$

Building the matrix V explicitly is computationally undesirable, since this would be much more expensive than the actual MLI iteration due to the computation of a full Jacobian. However, we can first compute the directional derivative $\left(\frac{\mathrm{d}F}{\mathrm{d}x}(x_{k+1})\right)\chi$ and then apply M to the resulting vector to obtain the desired product much more efficiently.

In order to calculate a matrix-vector product $J^{\mathrm{B}}(x_{k+1})\chi$ or $J^{\mathrm{C}}(x_{k+1})\chi$, we partition the direction $\chi = (\chi^1, \chi^2, \chi^3)$ and then obtain

$$J^{\mathrm{B}}(x_{k+1})\chi = \begin{pmatrix} B_k\chi^1 - \widetilde{C}_k^\intercal\chi^2 - \widetilde{D}_k^\intercal\chi^3 \\ J_{c,k+1}\chi^1 \\ J_{d,k+1}\chi^1 \end{pmatrix} \tag{7.24}$$

and

$$J^{\mathrm{C}}(x_{k+1})\chi = \begin{pmatrix} \nabla_w^2\mathcal{L}(x_{k+1})\chi^1 - J_{c,k+1}^\intercal\chi^2 - J_{d,k+1}^\intercal\chi^3 \\ J_{c,k+1}\chi^1 \\ J_{d,k+1}\chi^1 \end{pmatrix}. \tag{7.25}$$

Note that $B_{k+1} = B_k, \widetilde{C}_{k+1} = \widetilde{C}_k, \widetilde{D}_{k+1} = \widetilde{D}_k$ in (7.24), since in this case we test level-B for suitability for the next MLI iteration, and thus the matrices are kept constant.

In both level-B and level-C, we have to compute $F^{\mathrm{B|C}}(x_{k+1})$ anyway for the adaptive level choice algorithm. We can thus see that the computational effort for level-B is mainly the forward sensitivity computation $J_{c,k+1}\chi^1$, which costs only a small multiple of the function evaluation $\widetilde{c}(w_{k+1})$. For level-C the dominant part is the computation of the directional second derivative $\nabla_w^2\mathcal{L}(x_{k+1})\chi^1$ which costs only a small multiple of the Lagrange gradient evaluation $\nabla_w\mathcal{L}(x_{k+1})$. This means that evaluation of $J^{\mathrm{B}}(x_{k+1})\chi$ or $J^{\mathrm{C}}(x_{k+1})\chi$ is not significantly more expensive than evaluation of the respective right-hand sides. In particular, when evaluating the right-hand-sides, during the solution of the dynamic model equations the quite expensive adaptive choices are determined and can be reused for the computation of the directional derivatives.

To efficiently compute the full matrix-vector product $V\chi$ we then exploit the fact, that M is given implicitly by the factorization of the QP matrices for the current active set. Multiplication of M with the vector $v = J^{\mathrm{B|C}}(x_{k+1})\chi$ is actually just a backsolve of the current factorized KKT matrix in the QP with v.

Note that if we start the iteration (6.13) with $\chi_0 = 0$, then we obtain $\chi_1 = -MF(x_{k+1})$ which is the inexact Newton increment in (6.4). This means, that in general the estimation of the spectral radius is computationally more expensive than the contraction estimation by postiterations. On the other hand, the spectral radius estimate is a more reliable quantity to determine the quality of M as an approximation of $J^{-1}(x_{k+1})$ and thus its suitability for reuse in upcoming MLI iterations.

Part III

Applications

8 Methodological applications

In this chapter, we consider several methodological fields of application for the MLI approach. First, we briefly discuss and give some results for MLI for NMPC on long horizons. Then, we outline an approach for the calculation of robust feedback, and introduce a formulation for a Dual NMPC problem to avoid the conservatism of the robust worst-case approach. Both the robust and the Dual NMPC problem are significantly larger and computationally more expensive than their nominal counterpart, and thus are natural candidates for the application of MLI.

8.1 NMPC on long horizons

An interesting field for application of the MLI approach is NMPC on long horizons. In NMPC theory, in general instant feedback is assumed as soon as the new current state is available. Moreover, ideally the optimal control problem is to be solved on an infinite time horizon, or otherwise one has to approximate the infinite horizon by end constraints and/or end weightings in the objective function, cf. [137, 156].

On the other hand, numerical NMPC schemes most often try to work on short control horizons to reduce the computational effort and thus calculate feedback quickly to reduce the feedback delay. In [92, 91] and related works it has been shown that by choosing a long enough control horizon, stability can be achieved by numerical schemes without any end point constraints or weightings.

Furthermore, for the approach of Multiple Shooting discretized NMPC with condensing for problem size reduction considered in this work, NMPC on long horizons can be considered as a particular instance of the more general problem of NMPC with many control variables. These problems may also occur from, e.g., the application of outer convexification to optimal control and NMPC with mixed-integer variables, cf. [161, 162, 103]. Thus, the efficient solution of NMPC problems on long horizons is an interesting and important problem.

In C. Kirches, L. Wirsching, H.G. Bock, and J.P. Schlöder, *Efficient direct multiple shooting for nonlinear model predictive control on long horizons* [106], MLI with level-C iterations is tested and compared to RTI with a sparse QP solver and RTI with a tailored block-structured QP solver (cf. [107]) on long horizons for several test problems. The problem sizes vary from 18 variables and 10 constraints to 57,690 variables and 55,680 constraints, with control horizon lengths up to $N = 640$ Multiple Shooting intervals. The study finds the MLI with level-C iterations combined with vector condensing "to perform best by a wide margin for systems with larger state space dimensions"([106]).

8.2 Robust Model Predictive Control

Robust control becomes essential whenever parameter values appearing in the problem formulation are only known within a given set of uncertainty. In this case, optimizing only for the nominal parameter values may lead to control profiles which perform poorly in the case of different parameter realizations or even lead to violation of process constraints.

The approach used in this work for robustification of control problems applies the work of Bock et al. [27, 52] for the min-max-robustification of nonlinear programs by Ben-Tal and Nemirovskii [17]. The min-max-robustification aims to minimize the maximum of the objective value over the uncertainty set, subject to feasibility of the maxima of the constraints over the uncertainty set. Belonging to the class of semi-infinite optimization these problems are computationally highly demanding. Thus, in [52] it is proposed to linearize the objective and constraint functions around the nominal parameter values. This gives rise to analytically solvable inner maximization problems and thus allows to simplify the problem to a nonlinear program.

For robust NMPC, we apply the methods presented and proposed in this work to the sequence of nonlinear programs obtained by the robustification approach described above, i.e., we apply RTI or MLI to the robust nonlinear programs with changing initial state in each iteration. The arising nonlinear programs are significantly larger and computationally more expensive than their nominal counterparts due to the fact that we have to compute first-order sensitivity information already for the formulation of the problem and thus higher-order sensitivity information to set up and solve the subproblems. Since MLI aims particularly at reducing the cost of function and derivative evaluation, it lends itself as efficient numerical approach for robust NMPC.

8.2.1 Robust nonlinear programming

Let us briefly describe the approach for a nonlinear program with equality constraints which contain uncertain parameters p. We will use the notation and closely follow the presentation of [52] and refer to this paper for further details. We will denote the objective function with f_0, the inequality constraints with $f_i, i = 1, \dots, n_f$, and the equality constraints with $g_j, j = 1, \dots, n_x$.

$$\min_{x \in \mathbb{R}^{n_x}, u \in \mathbb{R}^{n_u}} \quad f_0(x,u) \quad \text{s.t.} \quad \begin{cases} f_i(x,u) & \leq & 0 & \text{for} & i = 1, \dots, n_f, \\ g_j(x,u,p) & = & 0 & \text{for} & j = 1, \dots, n_x. \end{cases} \tag{8.1}$$

Assume that the parameters p are contained in the ellipsoidal uncertainty set

$$\mathbb{P} := \left\{ p \in \mathbb{R}^{n_p} \;\middle|\; \left\| \Sigma_p^{-\frac{1}{2}} (p - \bar{p}) \right\|_2 \leq \gamma \right\}$$

centered around the nominal values $\bar{p}$ and using the covariance matrix Σ_p as well as a scalar confidence level parameter γ for determining the size of the ellipsoid.

By defining the worst-case functions

$$\phi_i(u) \quad := \quad \max_{x \in \mathbb{R}^{n_x}, p \in \mathbb{P}} \quad f_i(x,u) \quad \text{s.t.} \quad g(x,u,p) = 0. \tag{8.2}$$

we can formulate the "robust counterpart" of the uncertain optimization problem as the following worst-case problem.

$$\text{(RC)} \qquad \min_{u \in \mathbb{R}^{n_u}} \quad \phi_0(u) \quad \text{s.t.} \quad \phi_i(u) \leq 0 \quad \text{for} \quad i = 1, \ldots, n_f. \tag{8.3}$$

As already mentioned above, this is a semi-infinite optimization problem (in the sense that (8.2) amounts to infinitely many constraints) and thus computationally demanding and usually intractable for discretizations of large-scale optimal control problems. However, it is possible to simplify the worst-case functions (8.2) by linearization around the nominal parameter values to come up with much simpler functions $\tilde{\phi}_i(u)$, which can be represented as

$$\tilde{\phi}_i(u) = f_i(\bar{x}, u) + \gamma \left\| \Sigma_p^{\frac{1}{2}} \left(\frac{\partial g}{\partial p}(\bar{x}, u, \bar{p}) \right)^T \left(\frac{\partial g}{\partial x}(\bar{x}, u, \bar{p}) \right)^{-T} \left(\frac{\partial f_i}{\partial x}(\bar{x}, u) \right)^T \right\|_2 . \tag{8.4}$$

Here, $\bar{x}$ are the discrete states satisfying the equality constraints for given control parameters u and nominal parameter values $\bar{p}$. Note that the approximated worst-case functions $\phi_i(u)$ contain penalty terms which increase with both the desired confidence level parameter γ and the covariance Σ_p. Thus, we come up with the approximated nonlinear problem

$$\text{(ARC)} \qquad \min_{u \in \mathbb{R}^{n_u}} \quad \tilde{\phi}_0(u) \quad \text{s.t.} \quad \tilde{\phi}_i(u) \leq 0 \quad \text{for} \quad i = 1, \ldots, n_f \tag{8.5}$$

which can then be treated by NLP solvers.

8.2.2 Application to feedback control

To illustrate the application of the robustification to NMPC consider the following NMPC optimal control problem with Mayer objective for the current state x_0^k

$$\begin{aligned} \underset{x,u}{\text{minimize}} \quad & E(x(t_{\text{end}})) \\ \text{subject to} \quad & 0 = x(t_0) - x_0^k, \\ & 0 = \dot{x}(t) - f(t, x(t), u(t), p), \\ & 0 \leq c^{\text{path}}(x(t), u(t)), \\ & 0 \leq r(x(t_{\text{end}})). \end{aligned} \tag{8.6}$$

Application of Direct Multiple Shooting as well as the linearized min-max robustification described in Section 8.2.1 yields a nonlinear program where the equality constraints represent the Multiple Shooting discretization of the dynamical equations containing uncertain parameters. In this case, in (8.1) x denotes the discrete state trajectory, u are the control parameters, and p are uncertain parameters (and possibly uncertain initial

values). The robustified nonlinear program reads

$$
\begin{aligned}
\underset{s,q,D}{\text{minimize}} \quad & E(s_N) + \gamma\sqrt{\nabla E(s_N)^\intercal D_N \Sigma D_N^\intercal \nabla E(s_N)} \qquad (8.7)\\
\text{subject to} \quad & 0 = s_0 - x_0^k,\\
& 0 = D_0,\\
& 0 = x_j(t_{j+1}; t_j, s_j, q_j) - s_{j+1}, \quad j = 0, \dots, N-1,\\
& 0 = D(t_{j+1}; D_j, s_j, q_j) - D_{j+1}, \quad j = 0, \dots, N-1,\\
& 0 \le c^{\text{path}}(s_j, q_j) - \gamma\sqrt{(\nabla c^{\text{path}}(s_j, q_j))^\intercal D_j \Sigma D_j^\intercal \nabla c^{\text{path}}(s_j, q_j)}, \quad j = 0, \dots, N-1,\\
& 0 \le r(s_N) - \gamma\sqrt{(\nabla r(s_N))^\intercal D_N \Sigma D_N^\intercal \nabla r(s_N)},
\end{aligned}
$$

where Σ is the covariance of the parameter uncertainty and $D(t_{j+1}; D_j, s_j, q_j)$ is the solution of the variational differential equation $\dot{D} = f_x D + f_p, D(t_j) = D_j$, which gives the sensitivity of the states with respect to the uncertain parameters. Since D has the size $n_x \times n_p$, NLP (8.7) has $(N+1) \cdot n_x \cdot n_p$ more equality constraints than its nominal counterpart. This NLP can now be handled with the numerical NMPC methods presented in this work. For the application of RTI to robust NMPC for fed-batch reactors see [117, 116].

8.3 Dual Control and NMPC

Using the approach described in Section 8.2.2 we obtain a robust feedback control. However, the expression $D_j \Sigma D_j^\intercal$ only propagates the initial uncertainty of the parameters through the system dynamics. The robust feedback which is computed in this way will be conservative in the sense of ignoring any possible additional information gain on the parameter values obtained within the prediction horizon. This may lead to computation of controls that accept unnecessary performance degradation.

The idea of *Dual Control*, which originates in the work of Feldbaum [68, 69, 70, 71], is to combine the optimal control of a process with uncertainties with the information gain about the uncertainties, in particular the uncertainties which have the most influence on the optimization goal. This means that we try to find controls which not only steer the process well with respect to the optimization goal, but also lead to "information-rich" measurement data from which the uncertainties with the most influence towards the optimization goal can be reduced quickly. A popular analogy is the time-optimal steering of a car on unknown slippery ground. One wants to drive fast, but in order to learn about the driving behavior of the car on the ground, one would also do some breaking and steering maneuvers which would be counterproductive with respect to the original goal on a well-known ground.

Therefore, one idea to apply this approach to the NMPC problem (8.7) and improve on the conservatism of the computed controls is to directly account for measurement information obtained over the prediction horizon within the problem formulation. This means to replace the expression $D_j \Sigma D_j^\intercal$ by some function $C(s, u)$, which on the one hand accounts

for measurement information, and on the other hand quantifies the information gain depending on state and control parameter values. By doing so, we enable the controller to choose control profiles which make best possible use of future measurements.

8.3.1 Propagation of covariance for linearized min-max robustification

A first approach to account for measurement information is to include the estimator equations in the controller problem formulation of the covariance. For the continuous extended Kalman filter equations, the new problem formulation is:

$$\begin{aligned}
\underset{s,q,C}{\text{minimize}} \quad & E(s_N) + \gamma\sqrt{\nabla E(s_N)^\intercal C_N \nabla E(s_N)} && (8.8)\\
\text{subject to} \quad & 0 = x_0^{(i)} - s_0,\\
& 0 = \bar{C} - C_0,\\
& 0 = x_j(t_{j+1}; t_j, s_j, q_j) - s_{j+1}, \quad j = 0, \dots, N-1,\\
& 0 = C(C_j, s_j, q_j) - C_{j+1}, \quad j = 0, \dots, N-1,\\
& 0 \le c^{\text{path}}(s_j, q_j) - \gamma\sqrt{(\nabla c^{\text{path}}(s_j, q_j))^\intercal\, C_j \nabla c^{\text{path}}(s_j, q_j)}, \quad j = 0, \dots, N-1,\\
& 0 \le r(s_N) - \gamma\sqrt{(\nabla r(s_N))^\intercal\, C_N \nabla r(s_N)},
\end{aligned}$$

where $C(C_j, s_j, q_j)$ is the solution of the initial value problem

$$\dot{C} = f_x C + C f_x^\intercal - C h_x^\intercal R^{-1} h_x C, \quad C(t_j) = C_j$$

and h is the measurement function.

This problem formulation adds $(N+1)\cdot(n_x + n_p)^2$ equality constraints to the nominal problem formulation and is thus also well suited for the application of Multi-Level Iteration schemes due to the computationally expensive dynamics. Note that the nonlinearity of the robustification terms both for robust NMPC and the Dual NMPC formulation is a strong argument for using exact Lagrange Hessians in level-D iterations.

8.3.2 Other approaches and further reading

More details on Dual Control for NMPC can be found, e.g., in [121, 120]. An interesting alternative approach to address the gain of measurement information in order to reduce conservatism in the robust feedback control is the scenario tree NMPC approach, see [167, 133]. The main idea is to divide up the prediction horizon in stages and branch on each stage after a common first stage for different realizations of the uncertainties. This allows to compute different control moves for each branch and thus satisfy bounds and minimize the objective more flexible than having to rely on a single control sequence over the whole horizon. The resulting NLPs are large-scale but structured and can be efficiently treated by structure-exploiting methods, see, e.g., [123, 122, 109].

9 Applications: MLI with prescribed level choice

In this chapter, we present the application of MLI with prescribed level choice to two numerical test cases, the continuous stirred-tank reactor (CSTR), and a single-track car model with nonlinear tire model (TESTDRIVE). In both cases, NMPC is used to reject a disturbance and track a desired state. We compare the performance of MLI with the established approach of RTI, investigate MLI for various data communication strategies, and show the advantage of higher sampling rates for the considered test cases.

If not cited from another publication, all computations presented in the following were performed on an Intel® Xeon® W3565@3.20GHz with 12 GB RAM. The software used is MLI, a MATLAB® based implementation of the Multi-Level Iteration approach written by the author of this work, which makes use of the parametric QP solver qpOASES [73], and the integrator/evaluator package SolvIND [5, 6] which provides implementations of BDF and Runge-Kutta methods with forward and backward sensitivity calculation of first and higher order.

9.1 CSTR: MLI with prescribed level choice

As a numerical test case for the Multi-Level Iteration schemes we use the already introduced continuous stirred-tank reactor (CSTR). For the description of the states, controls, and parameters of the CSTR, as well as the dynamic model equations we refer to the presentation in Chapter 2.

In general, the dynamics of the CSTR are comparably slow and the system size is small, thus computational times are not an issue on modern day desktop computers and the use of particularly efficient numerical schemes to generate feedback quickly is not critical. Still, the system is significantly nonlinear and illustrates nicely rather typical workload distribution with respect to the tasks that are part of a single Multi-Level Iteration, and thus we use the test case as a proof-of-concept for the usability of the presented approach of MLI.

9.1.1 Control scenario

For the numerical testing, we consider the following scenario. Units are left out for the sake of briefness of the presentation. The CSTR reactor starts in the steady-state and

controls

$$x_\mathrm{s} = \begin{pmatrix} 2.1402 \\ 1.0903 \\ 114.19 \\ 112.91 \end{pmatrix}, \quad \text{and} \quad u_\mathrm{s} = \begin{pmatrix} 14.19 \\ -1113.5 \end{pmatrix}. \tag{9.1}$$

At time $t = 2$, the feed temperature jumps from $\theta_0 = 104.9$ to $\theta_0 = 109.0$. The state evolution resulting from this disturbance for constant controls u_s is depicted in Figure 9.1. We can see a significant increase in the reactor temperature θ and thus an undesired deviation of the product concentration c_B from the target value $x_{\mathrm{s},2}$.

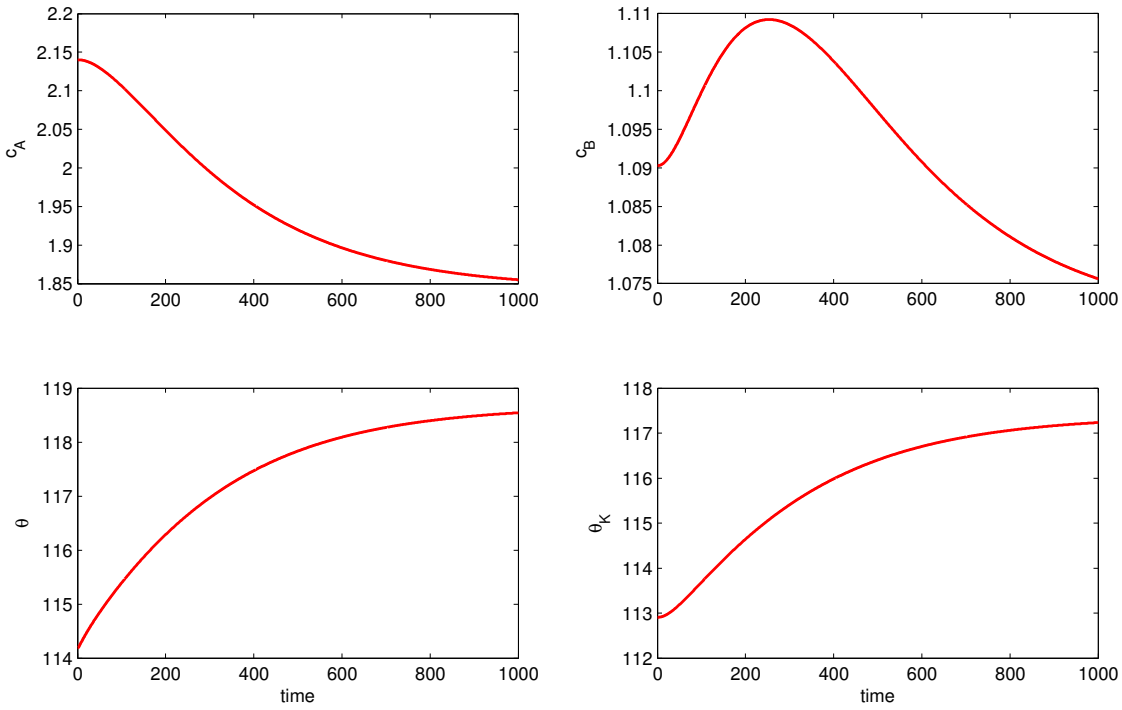


Figure 9.1: Evolution of CSTR states until $t = 1000$ after parameter jump under steady-state controls.

The aim of the controller is to reject the disturbance by tracking the product concentration $x_{\mathrm{s},2}$ and the reactor temperature $x_{\mathrm{s},3}$ while staying close to the steady-state controls u_s. To do this, we apply NMPC with the objective

$$\min_{x,u} \int_0^T L(x,u)\mathrm{d}t, \tag{9.2}$$

with

$$L(x,u) = \begin{pmatrix} x_2 - x_{\mathrm{s},2} \\ x_3 - x_{\mathrm{s},3} \end{pmatrix}^\intercal Q \begin{pmatrix} x_2 - x_{\mathrm{s},2} \\ x_3 - x_{\mathrm{s},3} \end{pmatrix} + (u - u_\mathrm{s})^\intercal R\,(u - u_\mathrm{s})\,, \tag{9.3}$$

with weighting matrices

$$\begin{pmatrix} 100 & 0 \\ 0 & 50 \end{pmatrix}, \quad \text{and} \begin{pmatrix} 5 \cdot 10^{-2} & 0 \\ 0 & 5 \cdot 10^{-8} \end{pmatrix}. \tag{9.4}$$

As time horizon for Multiple Shooting, we use $[0, T] = [0, 100]$, divided into 10 Multiple Shooting intervals. The scenario runs for 300 time units.

9.1.2 Failure of LMPC and RTI reference

In order to see that a nonlinear feedback technique such as NMPC is required, we set up an LMPC controller by linearizing the constraints and evaluating the exact Lagrange Hessian and the objective gradient in the steady-state, using the Lagrange multipliers obtained by solving the steady-state optimal control problem. As sampling period we choose $\delta = 0.1$ time units.

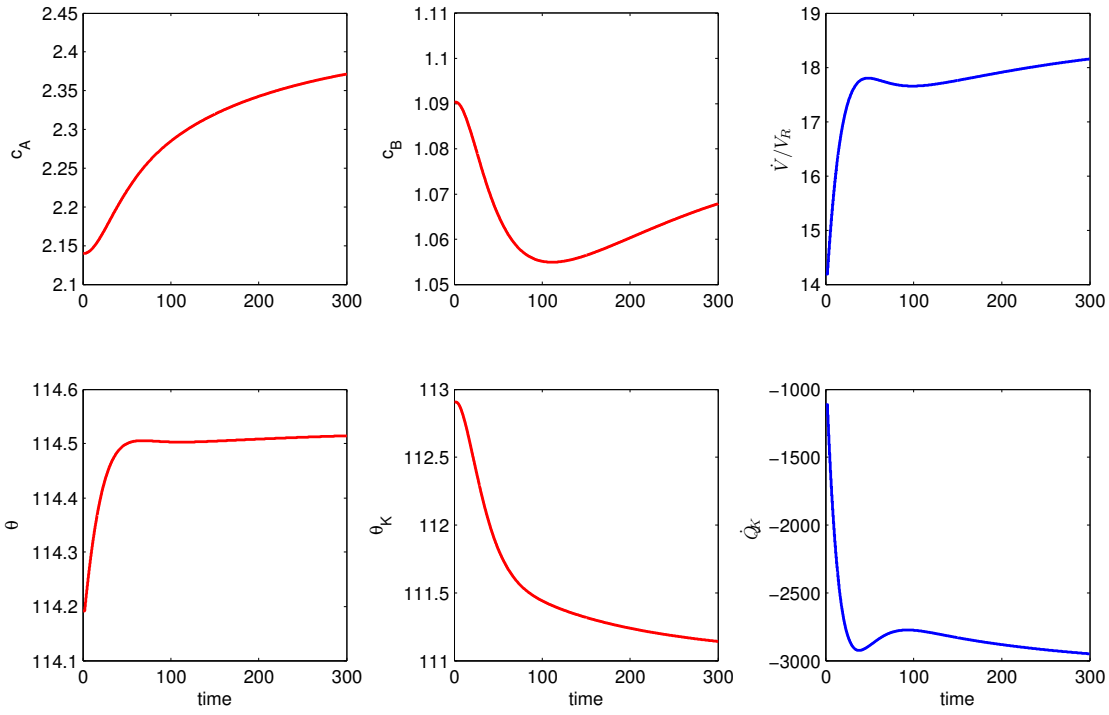


Figure 9.2: CSTR state (*left, red*) and control (*right, blue*) trajectories: LMPC fails to track the product concentration and reactor temperature.

The resulting control and state trajectories are given in Figure 9.2. It is clear that the controller is not able to track the reactor temperature and the product concentration. The chosen control trajectory deviates significantly from the steady-state controls. This is partly due to the fact that the linearization of the dynamical system, i.e., the linearized matching conditions, do not contain any information of the parameter change. A possible improvement would be to add the term $G_i^p \Delta p$ to the ith matching condition, where G_i^p is the sensitivity of the steady-state solution on the ith Multiple Shooting interval with respect to the parameters, and Δp is the difference of the current parameters to the steady-state parameters, which could be obtained, e.g., by a concurrently running online estimator.

As a reference solution for comparison of the MLI controller performances, we give the

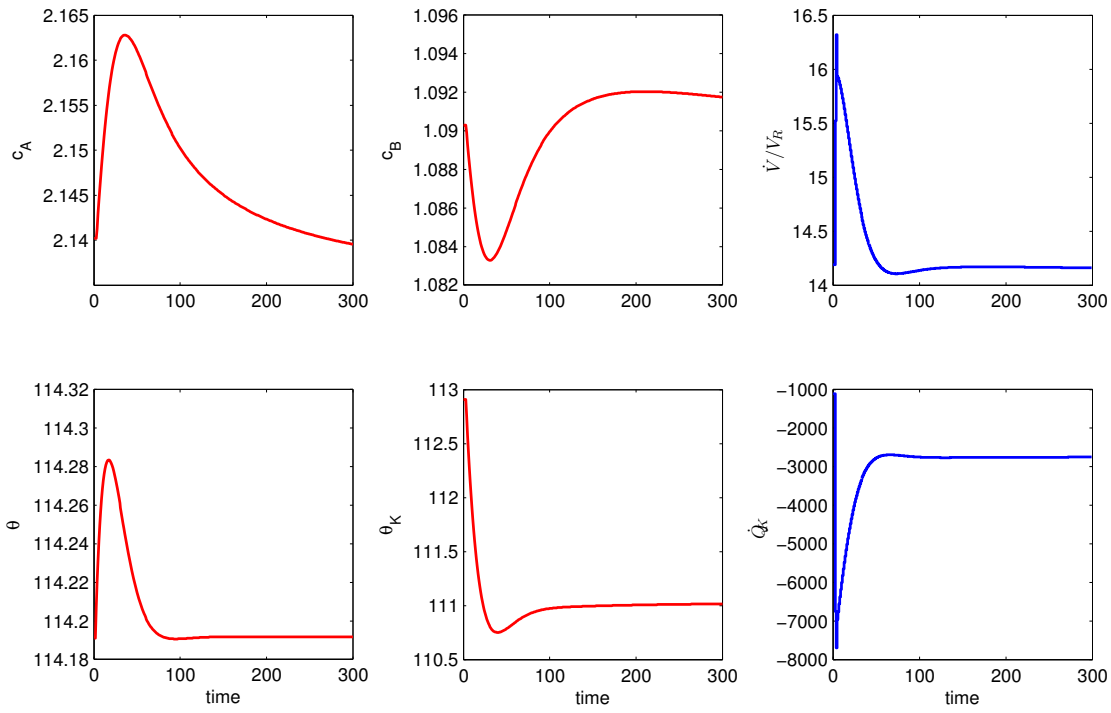


Figure 9.3: CSTR state (*left, red*) and control (*right, blue*) trajectories: RTI reference solution tracks product concentration and reactor temperature.

RTI control and state trajectories for a sampling period of $\delta = 1$ time units using exact Hessians in Figure 9.3. RTI is able to reject the disturbance and track both the product concentration and the reactor temperature. In Table 9.1 we give average computational times per sample for the six main tasks which are part of one real-time iteration. These tasks, with shorthands in parentheses, are function and derivative evaluation (EVAL), assembling the Hessian (HESS), condensing (COND), solving the QP (QP), recovering the step variables and multipliers eliminated by condensing (BLOWUP), and updating the primal and dual variables (STEP).

The accumulated CPU times over the whole scenario runtime for the six tasks are given in Figure 9.4. The figure shows clearly that the main part of the computational effort goes into the preparation phase, which mostly consists of the function and derivative evaluation and condensing. Since MLI aims particularly at reducing the computational effort for these two tasks, it can be expected to decrease the computational effort considerably. Furthermore, the figure shows a slight decrease of the computational effort for function and derivative evaluation over runtime. This is due to the fact that with approaching the new operating point, the error control algorithm within the integrator allows to choose longer step sizes, thus reducing the computational cost of function and derivative evaluation. Note that the tasks which do not depend on the evaluation point but only on the size of the matrices and vectors involved, such as condensing and recovery of step variables and multipliers, require a constant computational effort and thus gain on the function and derivative evaluation towards the end of the scenario runtime.

Table 9.1: Average computational times of tasks in milliseconds per sample for RTI reference solution. It is clearly visible that the preparation phase is the computationally dominant phase and that function and derivative evaluation contributes a major part. Note also the small amount of CPU time spent in the feedback phase, making feedback effectively immediate.

RTI phase	**Task**	**⌀ CPU time/sample [ms]**
Preparation phase	EVAL	15.91
	HESS	0.38
	COND	10.17
	$\sum$	26.46
Feedback phase	QP	0.31
Transition phase	BLOWUP	0.80
	STEP	0.06
	$\sum$	0.86
	Total $\sum$	27.63

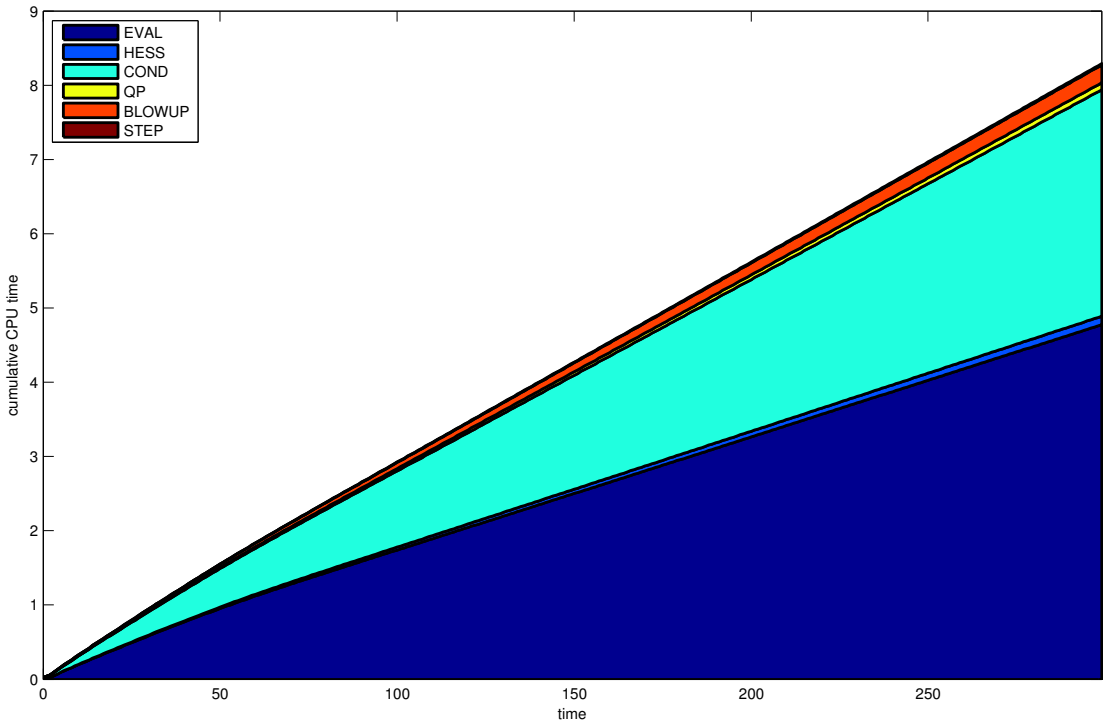


Figure 9.4: RTI for CSTR: accumulated CPU times for six tasks function and derivative evaluation (EVAL), Hessian assembly (HESS), condensing (COND), QP solution (QP), recovery of step variables and multipliers (BLOWUP), and step computation (STEP). The figure shows the proportions of the CPU time spent for the various tasks over time. Dominant parts are function and derivative evaluation and condensing.

9.1.3 Numerical results for pure level-C iterations

As a first example for an MLI scheme let us consider the pure level-C scheme. In the controller initialization, the exact Hessian and constraint Jacobians are evaluated in the original steady state. In each sample, then only the Lagrange gradient and the constraint residuals are evaluated. In particular, no new Jacobians or Hessian approximations are computed and thus no matrix condensing is necessary. As discussed in Chapter 5, the pure level-C iteration scheme has the same theoretical properties as RTI, provided that the Jacobian and Hessian approximations are in a defined way close enough to their exact counterparts.

Note that we choose the same sampling rate $\delta = 1$ here and for the MLI schemes used later in Section 9.1 if not stated otherwise. This is in slight contradiction to the overall philosophy of using MLI schemes to to be able to choose faster sampling periods to obtain better controller performances. However, the point of the investigations at hand is to show that MLI schemes generate suitable feedback control even for the same sampling rate as the RTI scheme. We also give results for a smaller sampling rate to show how the controller performance benefits from the faster feedback provided by the MLI schemes, see Table 9.4.

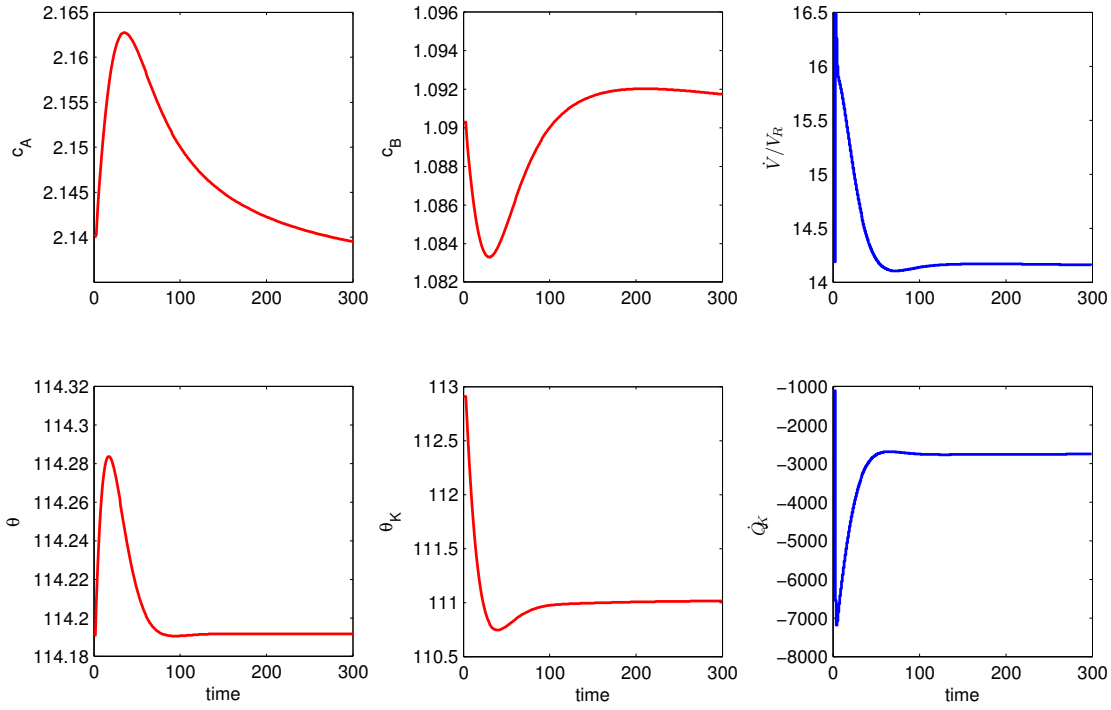


Figure 9.5: CSTR state (*left, red*) and control (*right, blue*) trajectories: pure level-C MLI tracks product concentration and reactor temperature.

The resulting state and control trajectories for the pure level-C scheme are given in Figure 9.5. Clearly, the controller is able to reject the disturbance and track the desired states. There are only marginal differences in state and control trajectories compared

Table 9.2: Average computational times of tasks in milliseconds per sample for pure level-C MLI. Note the significant decrease in the CPU time spent for function and derivative evaluation as well as condensing.

MLI phase	**Task**	**∅ CPU time/sample [ms]**
Preparation phase	EVAL	4.59
	HESS	0.0
	COND	2.74
	$\sum$	7.33
Feedback phase	QP	0.28
Transition phase	BLOWUP	0.93
	STEP	0.10
	$\sum$	1.03
	Total $\sum$	8.64

to the RTI reference solution, most notably in the size of the controls closely after the instant of time when the disturbance occurs.

The computational times for the six main tasks introduced above are given in Table 9.2. As expected, we observe a significant reduction in the function and derivative evaluation, which amounts for the test case at hand to a factor of almost 3.5, and in the condensing, which amounts to a factor of a bit more than 3.7, both in comparison with the computational times for the RTI controller, cf. Table 9.1. There is also a reduction in the QP solution times, which is due to the saving of the initial matrix decompositions, which are necessary whenever new Jacobians or Hessian approximations are computed, as it is the case in the RTI scheme. This effect seems small in this example, however, for larger QPs it amounts to a significant part of the computations needed for the QP solution. Furthermore, the QP solution time constitutes the actual delay for giving feedback as soon as the new current state is available, thus MLI allows not only faster sampling but also reduces the actual feedback delay. The accumulated CPU times over the whole scenario runtime for the six tasks are given in Figure 9.6. It is clear that the total time consumption as well as the share of the preparation phase in the overall computations per sample have decreased significantly.

9.1.4 Suboptimality of level-B iterations

Next we consider the pure level-B MLI scheme and investigate its suitability for the control scenario at hand. In the controller initialization, the exact Hessian and constraint Jacobians are evaluated in the original steady state. In each sample, then only the gradient approximation and the constraint residuals are evaluated. In particular, no new derivative information is computed, which makes level-B iterations computationally even less expensive than level-C iterations. Also, no matrix condensing is necessary. As discussed in Chapter 5, due to the use of a gradient approximation, the level-B iterations

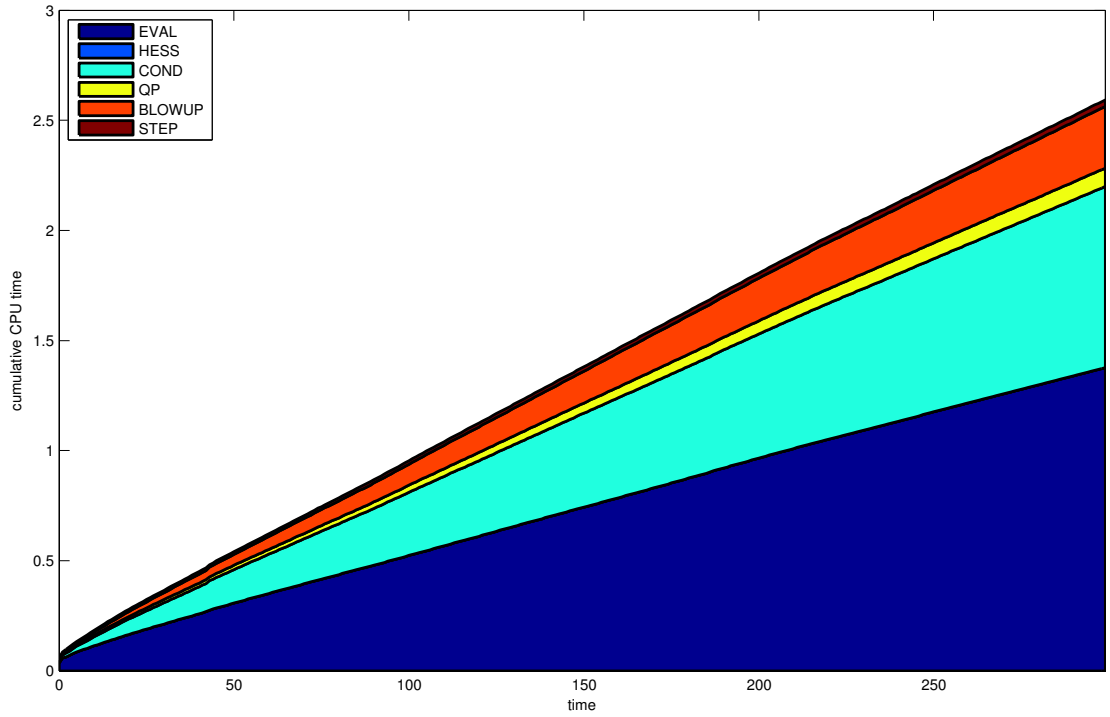


Figure 9.6: Pure level-C MLI for CSTR: accumulated CPU times for six tasks function and derivative evaluation (EVAL), Hessian assembly (HESS), condensing (COND), QP solution (QP), recovery of variables and multipliers (BLOWUP), and step computation (STEP). Note the difference in the y-axis range compared to Figure 9.4. The amount of CPU time reduction for EVAL and COND is clearly visible. As expected, there is no reduction in the CPU time for the calculations that are independent of the level choice.

converge in general to suboptimal points, and this can be observed for the test case.

The resulting state and control trajectories for the pure level-B scheme with exact Hessian are given in Figure 9.7. For comparison, the dashed lines show the RTI reference solution. While tracking the product concentration c_B satisfactorily, the controller is unable to track the reactor temperature. Also, the control profile of the first control $\dot{V}/V_R$ shows an offset to the RTI control profile. It is thus clear that the controller drives the process into a suboptimal operating point. The controller performance is still much better than the LMPC controller performance, cf. Figure 9.2.

As an alternative to using the exact Hessian in the steady state as Hessian approximation for the level-B iterations, one can formulate the NMPC objective approximatively by the following least-squares objective, which samples and sums up the original objective in the Multiple Shooting nodes

$$\min_{s,q} \sum_{i=0}^{N-1} \left(\left\| \begin{pmatrix} s_{i,2} - x_{\mathrm{s},2} \\ s_{i,3} - x_{\mathrm{s},3} \end{pmatrix} \right\|_Q^2 + \|q_i - u_\mathrm{s}\|_R^2 \right) + \left\| \begin{pmatrix} s_{N,2} - x_{\mathrm{s},2} \\ s_{N,3} - x_{\mathrm{s},3} \end{pmatrix} \right\|_Q^2, \tag{9.5}$$

and then build the Gauss-Newton Hessian approximation in the steady state and use the

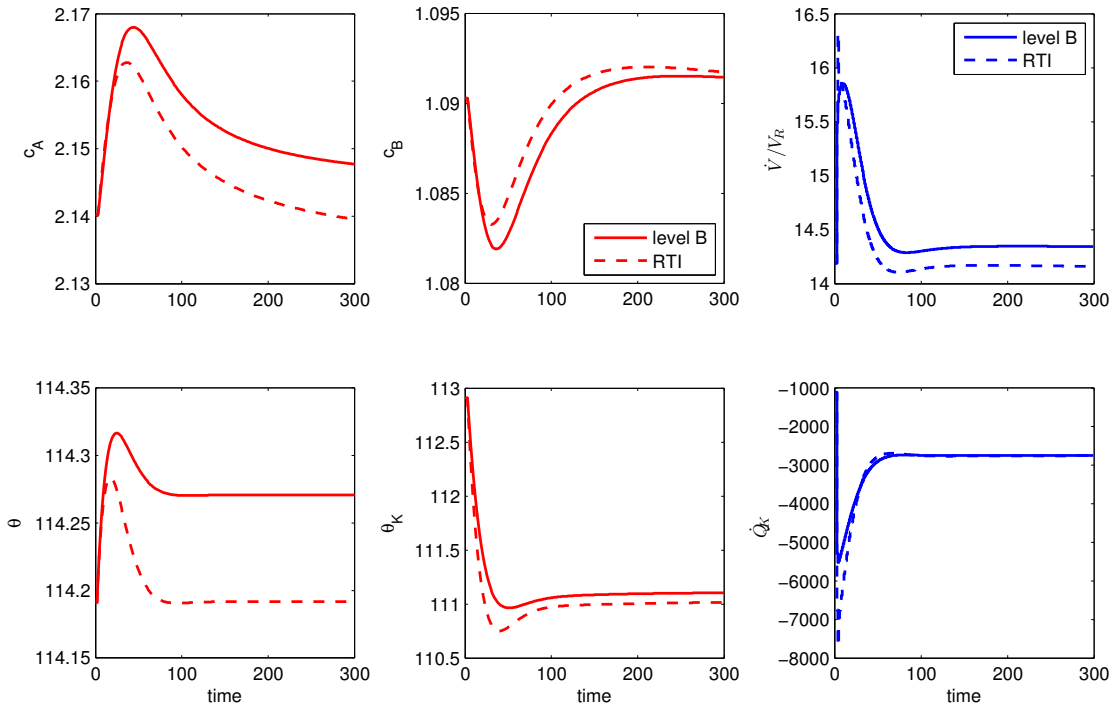


Figure 9.7: CSTR state (*left, red*) and control (*right, blue*) trajectories: pure level-B MLI with exact Hessian (solid line) steers process to suboptimal operating point; reference RTI solution as dashed line.

Gauss-Newton gradient approximation during the MLI iterations. In this case, the Gauss-Newton Hessian approximation is constant and thus the exact Gauss-Newton Hessian approximation in each sample, therefore also the Gauss-Newton gradient approximation is the exact objective gradient in each sample.

The resulting state and control trajectories for the pure level-B scheme with Gauss-Newton Hessian approximation are given in Figure 9.8. For comparison, the dashed lines show the RTI reference solution. The controller is able to track both the product concentration and the reactor temperature. Still, the control profile of the first control $\dot{V}/V_R$ shows an offset to the RTI control profile, which means that the controller still drives the process into a slightly suboptimal operating point. However, the level-B controller with the Gauss-Newton Hessian approximation is much better than the level-B controller with exact Hessian and almost as good as the level-C controller.

The computational times for the six main tasks introduced above are given for both the controller with the exact Hessian and the controller with the Gauss-Newton Hessian approximation in Table 9.3. As expected, we observe a further reduction in the function and derivative evaluation, which amounts for the test case at hand to a factor of roughly 2 compared to the computational times for the level-C controller, cf. Table 9.2. The reason for the comparably small reduction of the cost compared to the theoretical factor of up to 5 is the cost for the adaptive choices made for the constraint evaluation, see also the discussion in Chapter 5.

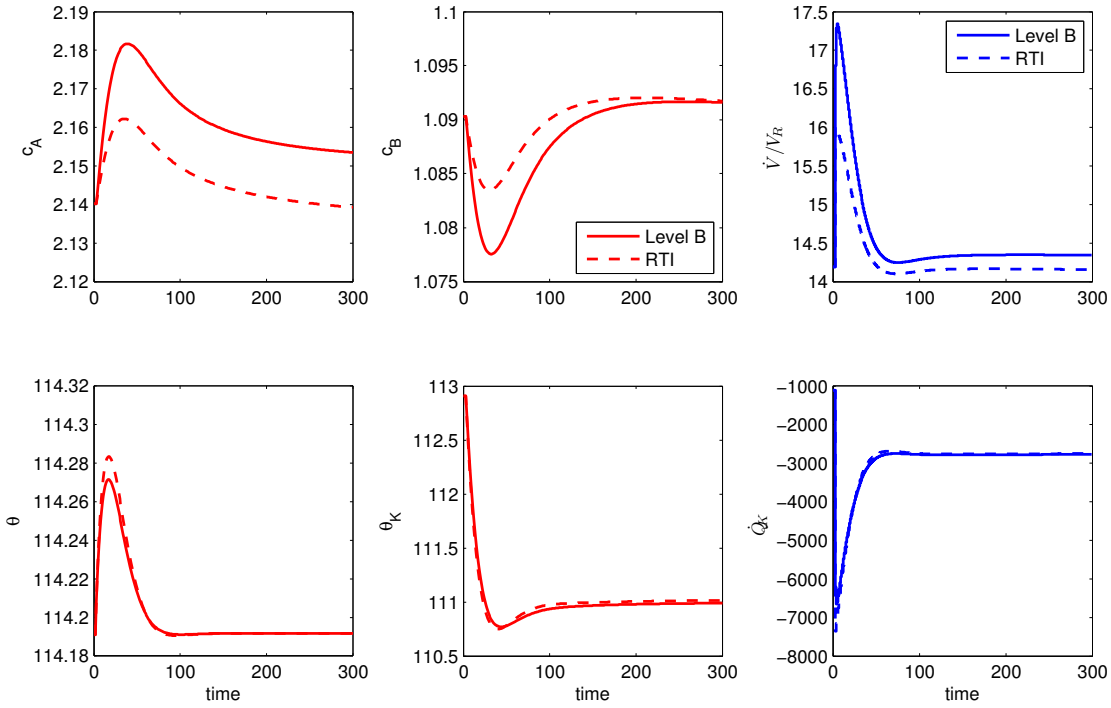


Figure 9.8: CSTR state (*left, red*) and control (*right, blue*) trajectories: pure level-B MLI with Gauss-Newton Hessian approximation tracks product concentration and reactor temperature, but slight offset in control $\dot{V}/V_R$; reference RTI solution as dashed line.

The other tasks show similar computational times than their level-C counterparts. The slight difference in the evaluation time between the level-B controllers with exact Hessian and Gauss-Newton Hessian approximation comes from the more expensive evaluation of the Gauss-Newton gradient approximation, which includes the evaluation of the least-squares residual function.

We finish the investigation of MLI schemes using a single level by comparing the performances of the presented controllers. To this end, we consider as performance index the objective function, evaluated over the whole scenario runtime. The results are given in Table 9.4. As discussed, the level-C controller achieves almost the same performance as the RTI controller, the level-B controller with the Gauss-Newton Hessian approximation is about 3.6% worse than the RTI controller. Only the level-B controller with exact Hessian shows significant suboptimal performance. In summary, it can be stated that for the considered test case, the level C and level-B controllers provide a computationally attractive and well-performing alternative to RTI. Furthermore, as a first indication that giving faster feedback can be beneficial for the controller performance, we also give the performance for a level-C controller for a sampling period of 0.1 time units, which is well within real-time feasibility, cf. Table 9.2 (the physical time unit of the sampling period δ that we have dropped for notational reasons are seconds). Compared to the RTI controller, the level-C controller achieves a performance gain of roughly 3.5%.

Table 9.3: Average computational times of tasks in milliseconds per sample for pure level-B MLI with exact Hessian (left) and Gauss-Newton Hessian approximation (right). The further reduction of CPU time for function evaluation compared to level-C is clearly visible.

MLI phase	**Task**	**∅ CPU exHess**	**∅ CPU GNHess**
Preparation phase	EVAL	2.23	2.93
	HESS	0.0	0.0
	COND	2.50	2.72
	$\sum$	4.74	5.65
Feedback phase	QP	0.22	0.27
Transition phase	BLOWUP	0.81	0.92
	STEP	0.05	0.10
	$\sum$	0.86	1.02
	Total $\sum$	5.82	6.94

Table 9.4: Comparison of performance index $I_{\text{perf}} = \int_0^{300} L(x,u)\mathrm{d}t$ for RTI, level-C MLI, and level-B MLI with exact Hessian and Gauss-Newton Hessian approximation. Additionally, performance of level-C MLI with sampling period of $\delta = 0.1$ time units is given. A smaller value of I_{perf} means a better controller performance. Level-C MLI is competitive with RTI even for the same sampling rate. The suboptimality of level-B iterations is clearly visible, with GN level-B still losing only about 3.6% of performance compared to GN RTI. The best performance is achieved by level-C with a sampling rate of $\delta = 0.1$, which shows the benefit of smaller sampling rates.

	exact Hessian				**GN Hessian**		
	RTI	C	B	C	RTI	C	B
Sampling δ	1.0			0.1	1.0		
I_{perf}	74.9066	74.9477	166.9260	72.3007	74.8453	75.0560	77.5548

9.1.5 MLI schemes with more than one level

In the following, we investigate MLI schemes with more than one level. More specifically, we investigate the combination of level-B iterations with level-D iterations as a representative for such MLI schemes. This seems like a reasonable choice, because the level-B iterations are suboptimal, but computationally cheap and thus allow fast feedback, while level-D iterations are computationally expensive, but provide the controller with updated system linearizations, Hessian approximations, and gradient information. We investigate the controller behavior and compare the scheme performances for the four data communication strategies presented in Chapter 5, and check if the interaction of level-B and level-D iterations leads to an improvement of the suboptimal performance of the level-B controller as observed above.

Specifically, we consider the B^1D^2 scheme as an example for an MLI scheme with frequent level-D updates, and the B^1D^{20} scheme as an example for an MLI scheme with less frequent level-D updates. Since we have observed significant differences in the controller

behavior in the test cases above, we again use level-B iterations either with exact Hessians or with Gauss-Newton Hessian approximations, each provided by the respective level-D iterations.

We give the performance index, defined as in the preceding subsection, for the four data communication strategies described in Chapter 5, namely bottom-up, maximum, minimum, and top-down communication in Table 9.5. The results for exact Hessians are given in the left part, the results for Gauss-Newton Hessian approximations are given in the right part.

Table 9.5: Comparison of performance index $I_{\text{perf}} = \int_0^{300} L(x,u)\mathrm{d}t$ for B^1D^2 and B^1D^{20} MLI schemes using data communication strategies from Chapter 5, with exact Hessian and Gauss-Newton Hessian approximation. In this example, the different data communication strategies result in almost identical controller performances. For the MLI schemes with sparser level-D iterations, the Gauss-Newton Hessian approximation yields significantly better controller performances, since it does not rely on Lagrange multiplicator information provided by the suboptimal level-B iterations.

	exact Hessian		**GN Hessian**	
I_{perf}	B^1D^2	B^1D^{20}	B^1D^2	B^1D^{20}
Bottom-up com.	74.9848	82.8482	74.8454	76.7225
Maximum com.	74.9815	82.8482	74.8451	76.7225
Minimum com.	74.9557	82.8574	74.8459	76.7210
Top-down com.	74.8281	82.8573	74.8455	76.7212

From the performance indices in Table 9.5, we can conclude that the B^1D^2 schemes for both exact Hessian and Gauss-Newton Hessian approximation achieve virtually the same performance as the respective RTI schemes, at a significantly lower computational cost. The B^1D^{20} schemes perform, in line with expectations, slightly worse. However, for both exact Hessian and Gauss-Newton Hessian approximation they are better than the respective pure level-B scheme. In particular, the exact Hessian B^1D^{20} is able to track both the product concentration and the reactor temperature. The B^1D^{20} scheme with Gauss-Newton Hessian approximation achieves a better performance index compared to the scheme with exact Hessian because it is able to track the reactor temperature with a smaller deviation from the set point value.

For this test case, only small differences can be observed between the four data communication strategies. The state trajectories and feedback control profiles generated by the schemes using the Gauss-Newton Hessian approximation are virtually identical. For the schemes using the exact Hessian, mostly small changes in the feedback control trajectory closely after the parameter jump event can be observed. As an instance for the B^1D^2 and the B^1D^{20} schemes with exact Hessian we give the state trajectories and the feedback control profiles for the bottom-up communication in Figure 9.9 and Figure 9.10, respectively. For the B^1D^2 and the B^1D^{20} schemes with Gauss-Newton Hessian approximation, state trajectories and feedback control profiles are given in Figure 9.11 and Figure 9.12, respectively.

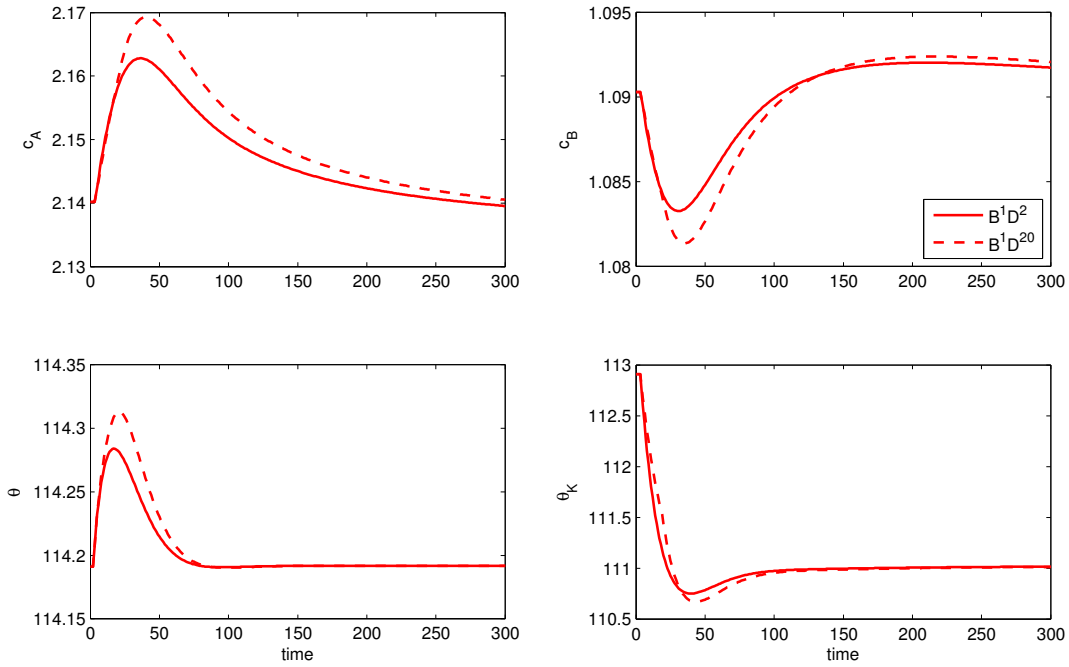


Figure 9.9: CSTR state trajectories: comparison of B^1D^2 and B^1D^{20} MLI scheme with exact Hessian and bottom-up data communication. The B^1D^2 scheme rejects the disturbance slightly faster and with less deviations from the set point.

The benefit of the combination of level-B and level-D iterations in MLI can be observed particularly well in the control profiles given in Figure 9.10 for the B^1D^{20} scheme. In the first part, the controller follows the control profiles generated by the pure level-B scheme, cf. Figure 9.7. At time $t = 20$, the linearizations, the gradient, and the Hessian approximation are updated. As a result, the controller jumps to a new feedback control arc which then allows the following level-B iterations to track the reactor temperature successfully.

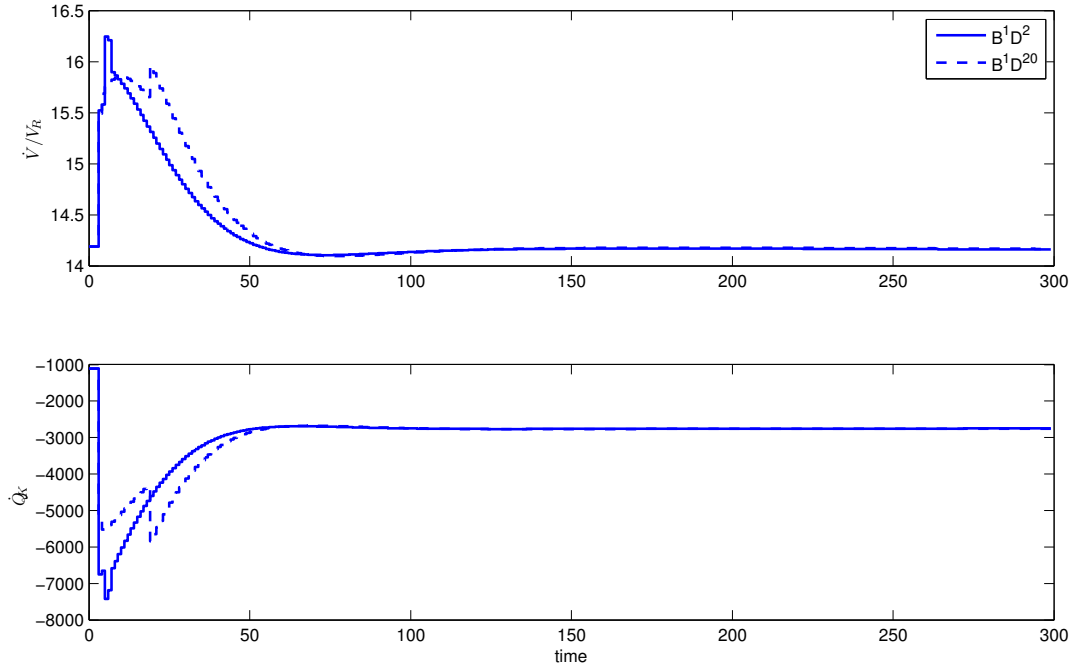


Figure 9.10: CSTR control trajectories: comparison of B^1D^2 and B^1D^{20} MLI scheme with exact Hessian and bottom-up data communication. The B^1D^2 scheme takes more control actions early on and thus the disturbance is rejected faster.

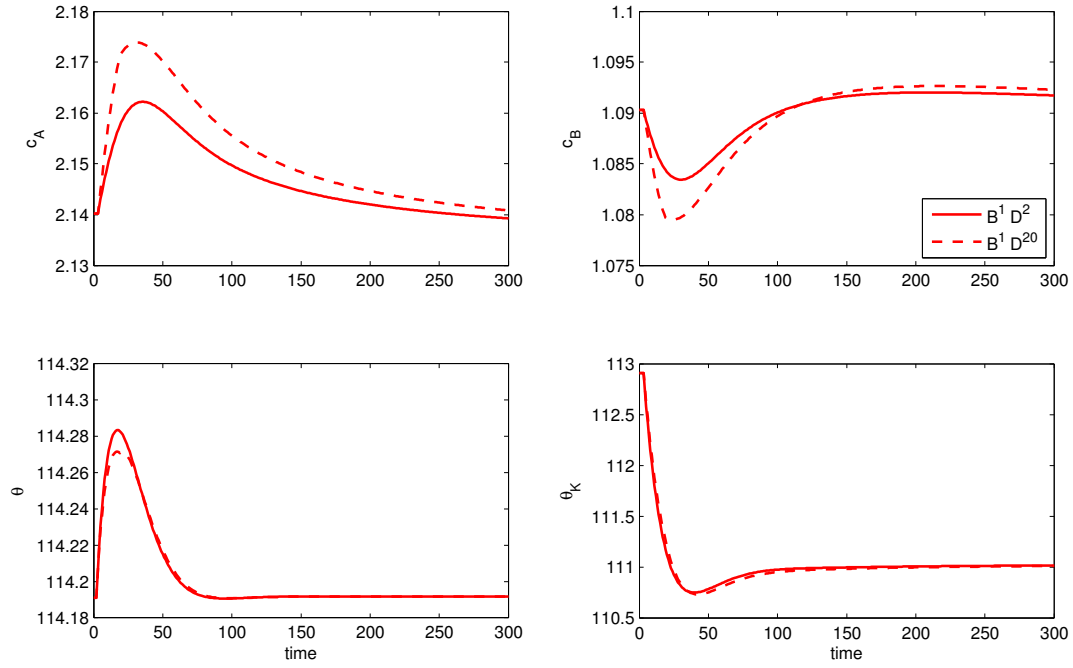


Figure 9.11: CSTR state trajectories: comparison of B^1D^2 and B^1D^{20} MLI scheme with Gauss-Newton Hessian approximation and bottom-up data communication.

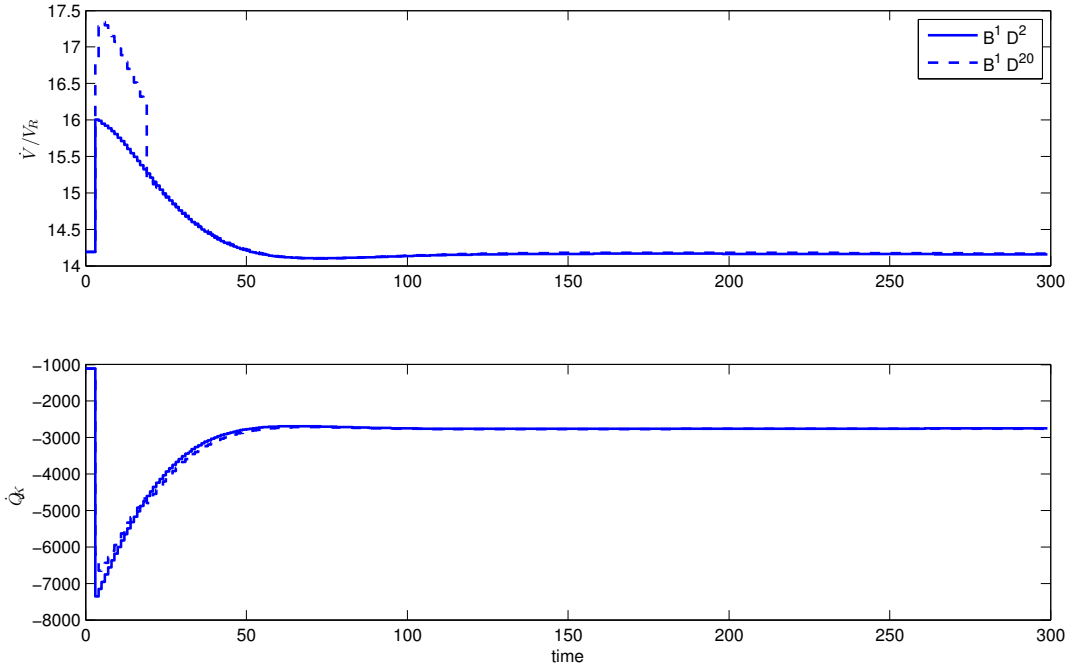


Figure 9.12: CSTR control trajectories: comparison of B^1D^2 and B^1D^{20} MLI scheme with Gauss-Newton Hessian approximation and bottom-up data communication.

9.2 TESTDRIVE: Performance improvement by faster feedback

In this section, we consider as test case the application of feedback control to a moving car that gets disturbed by a lateral impact. The goal of the feedback control is to recover speed and direction of the car. The dynamics of the system are fast and thus quick feedback is required to reject the disturbance. We provide the car model, specify the control scenario, and compare numerical results of RTI with various schemes consisting of level-A and level-D iterations. The results show that giving feedback faster is more important than using exact derivative information in each sample. The section is mostly a citation from the paper J. Albersmeyer, D. Beigel, C. Kirches, L. Wirsching, H.G. Bock, and J.P. Schlöder, *Fast nonlinear model predictive control with an application in automotive engineering* [3], with adaption of notation and some additional commentary.

9.2.1 Vehicle model

A nonlinear single-track car model featuring a Pacejka type tire model is used for the vehicle dynamics. It is described by an ODE system in the states $x = (c_{\mathrm{x}}, c_{\mathrm{y}}, v, \delta, \beta, \psi, w_{\mathrm{z}})$ and the controls $u = (w_\delta, F_{\mathrm{B}}, \phi)$. The coordinates, angles, and forces are visualized in Figure 9.13. For a detailed description of the model equations and the system parameters we refer to [105] and the references therein. The model is nonlinear in the states as well as in the control ϕ.

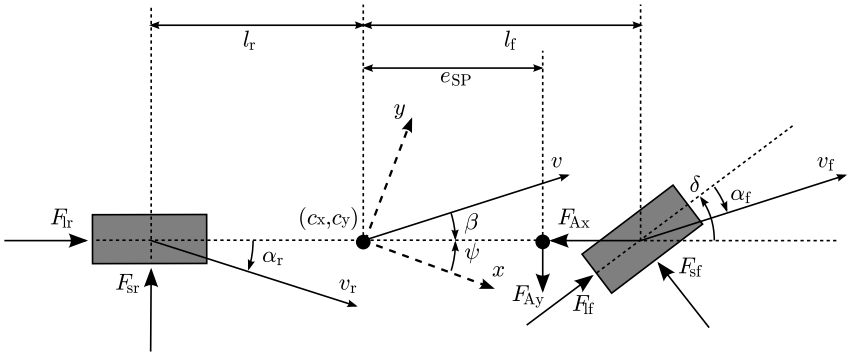


Figure 9.13: Coordinates, forces, and angles of the single-track vehicle model. The figure is aligned with the vehicle's local coordinate system, while the dashed pair of vectors (x, y) depicts the global coordinate system used for computations.

9.2.2 Control scenario

A vehicle of $1,300$ kg mass is driving on a straight lane at a speed of 30 m/s. After 2 seconds, an impulse of magnitude $2 \cdot 10^4$ N is acting on the rear axle perpendicular to the driving direction for 0.1 seconds. Applying the optimal offline control for driving on the straight lane, the impact is strong enough to make the car spin multiple times and force it off the lane. The model thus is not open-loop stable for this scenario.

Aim of the controller is to keep the vehicle on the lane while retaining a speed of 30 m/s. The full system state information is available at a resolution of 0.05 seconds.

The NMPC formulation of the scenario contains a least-square objective to minimize the deviation from the straight lane as well as the prescribed velocity. Further, the controls w_δ, F_B, ϕ are regularized over the prediction horizon with weight vector $(1.0, 10^{-12}, 10^{-4})$, which puts most weight on tracking the setpoint for the steering angle velocity w_δ with about equal weighting of the braking force and the pedal position two orders of magnitude less than the steering angle velocity.

Two-sided simple bounds on all states and controls are formulated, while no nonlinear constraints are present. The upper and lower bounds on the states are

$$\begin{aligned} \overline{x} &= (2000, 1000, 60, 30, 30, 30, 30, 100), \\ \underline{x} &= -2000, -1000, 0, -30, -30, -30, -30, -100), \end{aligned}$$

and the upper and lower bounds on the controls are

$$\begin{aligned} \overline{u} &= (0.5, 15000, 1), \\ \underline{u} &= (-0.5, 0, 0). \end{aligned}$$

9.2.3 Numerical results

The delay due to computation time spent in the feedback phase is properly taken into account in the vehicle system simulation. The computations have been performed with the optimal control software `MUSCOD-II` [126, 127]. All computation times are given for a

single-core *Intel Pentium 4* machine with *2.8 GHz* and *3 GB RAM*, running *SuSE Linux 10.1*.

9.2.4 Classical Real-Time Iterations

From Figure 9.14 we can see that a sampling rate of 0.1 s is insufficient. The vehicle spins uncontrollably, the final deviation from the straight lane exceeds 5 meters, and the velocity of 30 m/s is not maintained. Figure 9.15 shows the controller performance for a sampling rate of 0.05 s. Obviously, the shorter sampling intervals allow the RTI scheme to reject the disturbance.

9.2.5 Multi-Level Iteration Schemes

The MLI controller schemes A^1D^2, A^1D^4, and A^1D^6 with a sampling rate of 0.05 s were applied to the control scenario. All controller schemes A^1D^2 to A^1D^6 are able to reject the disturbance, while those which apply mode D more often are able to catch nonlinear effects better and generate "cleaner" control profiles, although at higher computational cost. The results depicted in Figure 9.16 show the vehicle behavior under control profiles generated by an A^1D^4 Multi-Level Iteration scheme. Again, choosing the sampling rate for MLI equal to the sampling rate for the RTI is not fully in line with the philosophy of MLI to give feedback as quickly as possible. The point of the study at hand is to illustrate that for this example, it is more important to generate feedback quickly enough than to calculate new linearizations. In fact, for sufficiently fast feedback, the linearization can be re-used over up to six sampling intervals.

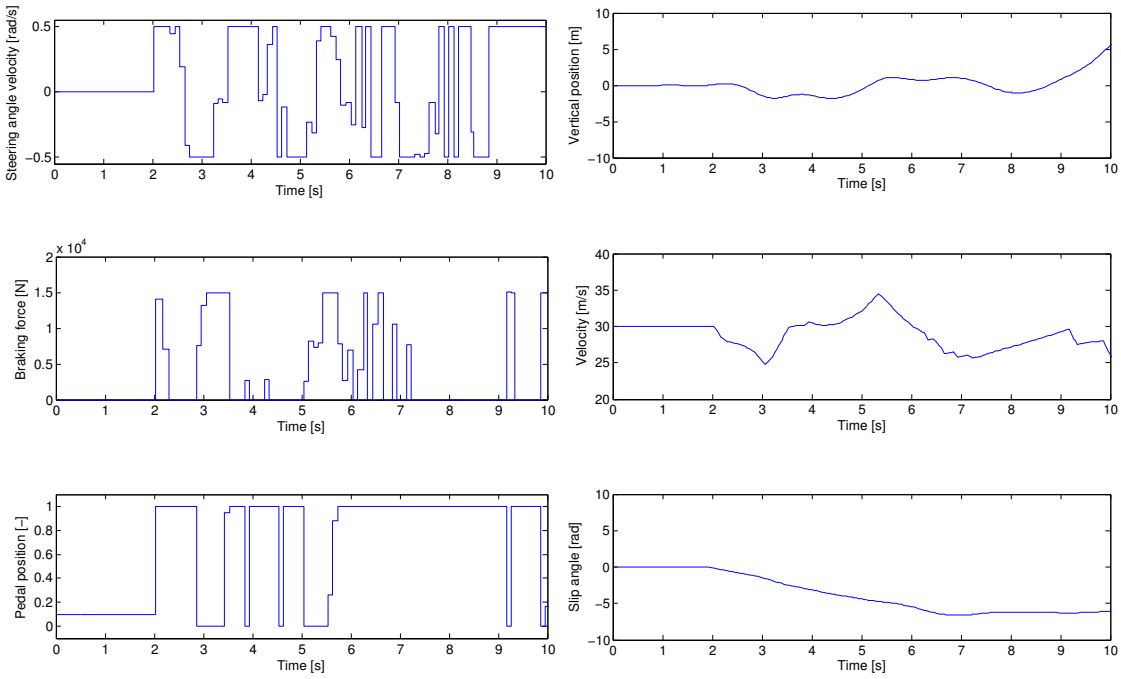
9.2.6 Computation Times

In Table 9.6 computation times for the multi-level NMPC implementation within the optimal control software package `MUSCOD-II` [126, 127] are shown. All controllers are faster than the classical Real-Time Iterations, while the most performant controller is A^1D^4.

Notably, when the nonlinear updates performed by mode D are applied scarcely to save on computation time (A^1D^6 scheme), the mode A workload starts to increase again due to more active set changes occurring during the dense QP solution.

Controller	Mode A		Mode D		Total
	Calls	Avg. [s/call]	Calls	Avg. [s/call]	[s]
RTI	200	–	200	0.019	3.76
A^1D^2	200	0.005	101	0.015	2.43
A^1D^4	200	0.005	51	0.015	1.77
A^1D^6	200	0.006	34	0.016	1.86

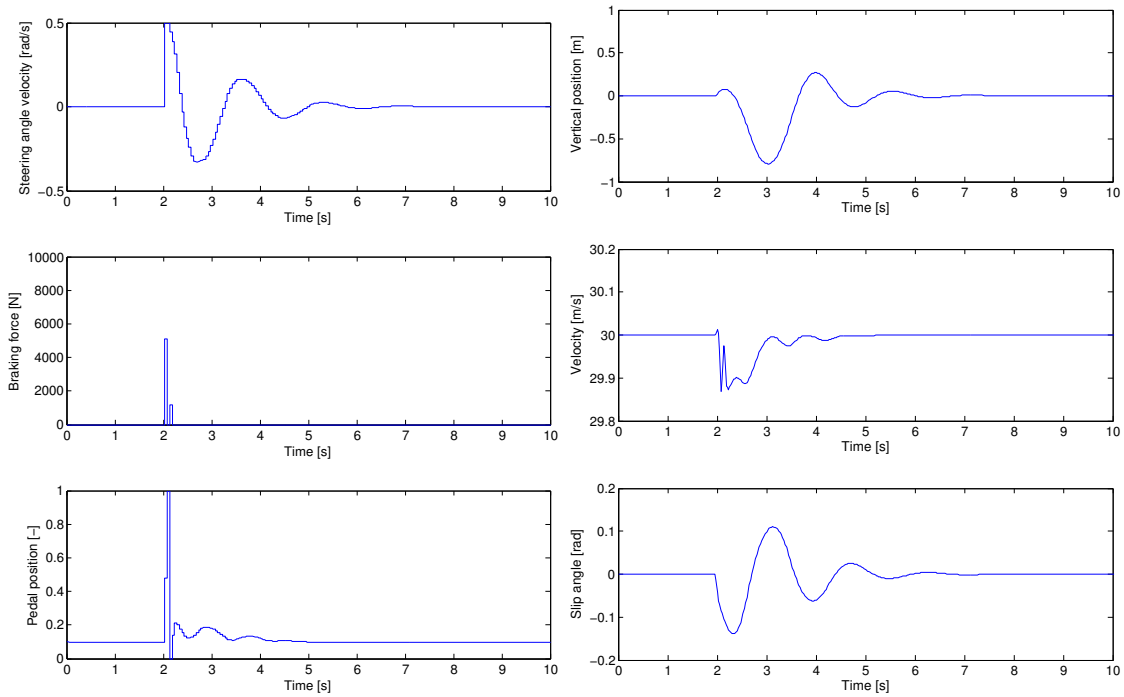
Table 9.6: Computation times for the results presented in Figures 9.15, and 9.16.



(a) Control profile.

(b) Differential state profile.

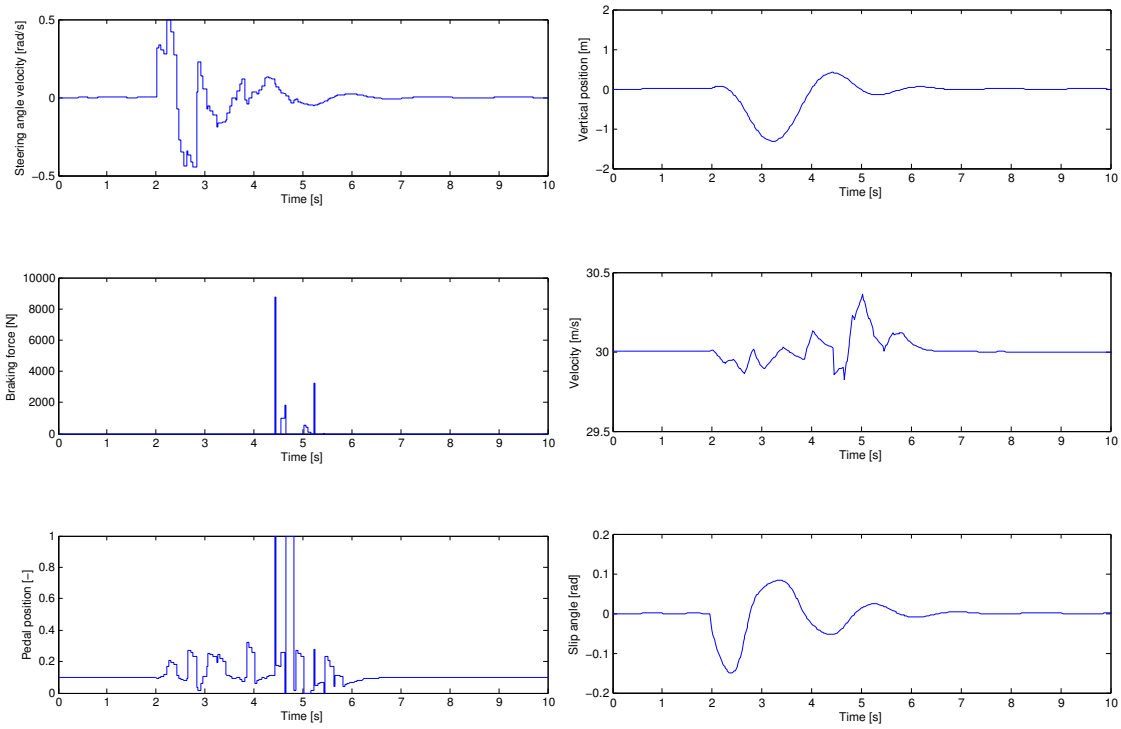
Figure 9.14: Unsatisfactory performance of the RTI controller scheme for a sampling time of 0.1 seconds. The controller is unable to reject the impact. From the right column it can be seen that the vehicle leaves the straight lane (top), the velocity v drops below 25 m/s (center), while the vehicle spins.



(a) Control profile.

(b) Differential state profile.

Figure 9.15: Performance of the RTI controller scheme for a sampling time of 0.05 seconds. Within less than 5 seconds, the controller recovers from the impact suffered at $t = 2$ s. The left column depicts the optimal controls (from top to bottom: steering angle velocity w_δ, braking force F_B, pedal position ϕ). The right column shows a selection of resulting differential states (deviation from the straight lane c_y, velocity v, slip angle β).



(a) Control profile.

(b) Differential state profile.

Figure 9.16: Performance of the A^1D^4 controller scheme for a sampling time of 0.05 seconds. Even doing a mode D step only every 0.2 seconds is still sufficient. Note the impact of the nonlinear updates of mode D on the control trajectories generated by the mode A controller. Every 4 control interval (0.2 seconds time), a new piecewise constant approximation to a control arc starts.

10 Applications with adaptive level choice

In this chapter, we investigate MLI with adaptive level choice by applying the algorithms described in Chapter 6 to MLI for two test cases: a chain of masses connected by springs, which is to be brought to rest, and the TESTDRIVE problem described in the previous chapter. For the chain application, we give the model equations, describe the control scenario, and discuss numerical results for both the postiteration approach and the spectral radius estimation approach. For the TESTDRIVE model, we concentrate on the postiteration approach, which meets the performance requirements best due to the rather small problem size and the fast dynamics involved.

10.1 CHAIN: Control of a chain of masses connected by springs

As a first test case for the adaptive level choice for MLI we consider NMPC for a chain of masses connected by springs without friction. The model as well as the application of MLI with pure level-C iterations have been described in [186, 184], with more extensive analysis of the system dynamics in the latter work. We consider the control task of bringing the chain to rest and compare the performance of various schemes with adaptive level choice both by postiterations and spectral radius estimation with the performance of the corresponding RTI scheme. The adaptive schemes show comparable performance and are, in contrast to the RTI scheme, real-time feasible by construction. We give the dynamic model of the chain, describe the control scenario, and present and discuss numerical results for the adaptive level choice by postiterations and by spectral radius estimation.

10.1.1 ODE model

Consider a system of N point masses, each having the mass m, that are connected by springs to form a chain. Each spring has the spring constant D and the rest length L. The mass at the one end of the chain is connected to a fixed point and the mass at the other end of the chain is connected to a freely movable point, both via another spring having the same spring constant and rest length. The whole chain of masses is situated in a homogeneous gravitational field.

Let the state variables $x_i(t) = (x_i^1(t), x_i^2(t), x_i^3(t))^\intercal \in \mathbb{R}^3$ for $i = 1, \ldots, N$, describe the position of the center of the mass i at time t. Let $x_0 \in \mathbb{R}^3$ be the point in space where the chain is fixed and $x_{N+1} \in \mathbb{R}^3$ the point in space where the chain can be moved. Without restriction of generality we choose x_0 to be the zero vector.

For the N balls, Newton's axioms for classical mechanics (see e.g. [173]) yield

$$\ddot{x}_i(t) = \frac{1}{m}\left(F_{i,i+1}(t) - F_{i-1,i}(t)\right) + g, \quad i = 1, \ldots, N \tag{10.1}$$

with $g \in \mathbb{R}^3$ being the gravitational acceleration and $F_{i,i+1}(t) \in \mathbb{R}^3$ being the force acting on the i-th mass due to the spring between ball i and ball $i+1$ at time t (see also Figure 10.1).

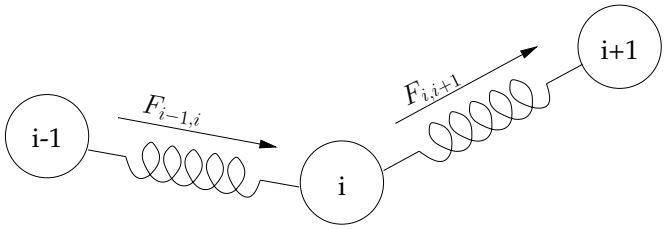


Figure 10.1: Spring forces between masses.

We assume that the radii of the balls are small enough to approximate the length of the (possibly elongated or shortened) spring between ball i and ball $i+1$ by $\|x_{i+1}(t) - x_i(t)\|$. From Hooke's law it then follows that the absolute value of $F_{i,i+1}(t)$ is given by

$$\|F_{i,i+1}(t)\| = |D(\|x_{i+1}(t) - x_i(t)\| - L)|, \tag{10.2}$$

and the force should point from $x_i(t)$ to $x_{i+1}(t)$ if $(\|x_{i+1}(t) - x_i(t)\| - L) > 0$, so we obtain

$$\begin{aligned} F_{i,i+1}(t) &= D(\|x_{i+1}(t) - x_i(t)\| - L)\frac{x_{i+1}(t) - x_i(t)}{\|x_{i+1}(t) - x_i(t)\|} \\ &= D\left(1 - \frac{L}{\|x_{i+1}(t) - x_i(t)\|}\right)(x_{i+1}(t) - x_i(t)), \end{aligned} \tag{10.3}$$

which is a nonlinear force.

We formulate the equations of motion for $x_{N+1}(t)$ also as differential equations by setting

$$\dot{x}_{N+1}(t) = u(t), \tag{10.4}$$

which means we prescribe the velocity of $x_{N+1}(t)$. The complete second-order model is then reformulated as first-order model, cf. Remark 2.1. We introduce the variables $v_i(t) = (v_i^1(t), v_i^2(t), v_i^3(t))^\intercal \in \mathbb{R}^3$ for $i = 1, \ldots, N$ and set

$$\dot{x}_i(t) = v_i(t), \quad i = 1, \ldots, N, \tag{10.5a}$$

$$\dot{v}_i(t) = \frac{1}{m}\left(F_{i,i+1}(t) - F_{i-1,i}(t)\right) + g, \quad i = 1, \ldots, N, \tag{10.5b}$$

$$\dot{x}_{N+1}(t) = u(t). \tag{10.5c}$$

In the following we drop units. However, all quantities and parameter values are formulated in the standard SI system and should be understood to have the correct corresponding unit.

Table 10.1: Parameter values for the chain model used in the test case.

Parameter	Value
D	1.0
L	0.055
m	0.03
g	$(0, 0, -9.81)^\intercal$
x_{end}	$(5, 0, 0)^\intercal$

10.1.2 Control scenario

We consider feedback control for a chain with $N = 9$ masses. In the initial state of the chain, the chain is at rest. The masses are lined up equally spaced along the x-axis between $x = 0.5$ and $x = 4.5$, and the point of the chain where the control is applied is at $x = 5.0$, i.e., the initial states for the model (10.5) are $x_i = (i \cdot 0.5, 0, 0)^\intercal, i = 1, \ldots, 10$ and $v_i = (0, 0, 0)^\intercal, i = 1, \ldots, 9$, see also Figure 10.2. The scenario runs for 30 time units. Because no friction is involved, the system with no controls applied is energy conservative and thus would keep oscillating forever.

The goal of the feedback control is to bring the chain to rest in its natural steady state with the control point in a prescribed x_{end}. To this end, we use the objective function

$$L(x, v, u) = \alpha \left\| x_{N+1}(t) - x_{\text{end}}(t) \right\|_2^2 + \beta \sum_{j=1}^{N} \left\| v_j(t) \right\|_2^2 + \gamma \left\| u(t) \right\|_2^2, \tag{10.6}$$

with the weighting factors $\alpha = 25$, $\beta = 1$ and $\gamma = 0.01$ for the formulation of the NMPC optimal control problem. The units of the weighting factors are chosen to obtain a dimensionless objective function value. For the control scenario, we consider the controls to be bounded by

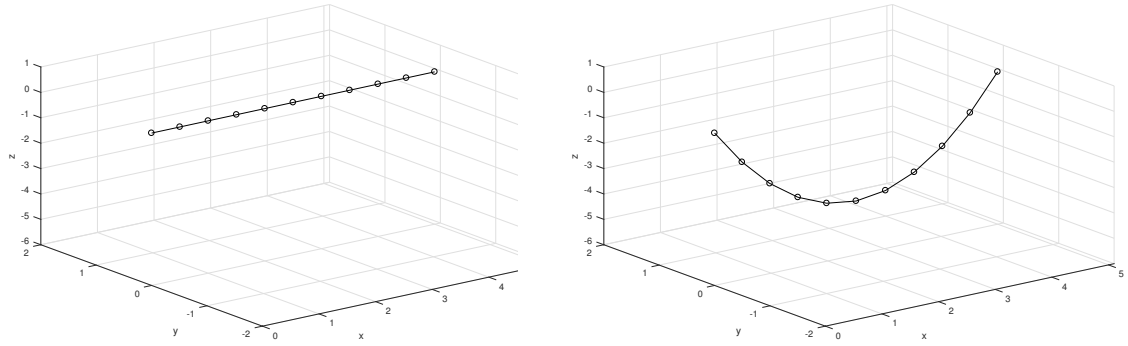
$$\left\| u(t) \right\|_\infty \leq 1 \quad \forall t \in [0, T]. \tag{10.7}$$

The parameter values used in the chain model and objective function are given in Table 10.1.

As time horizon for Multiple Shooting, we use $[0, T] = [0, 4]$, divided into 20 Multiple Shooting intervals. We use the Gauss-Newton approach and initialize the controller in the steady state, which can be obtained by solving the right-hand-side of equations (10.5) for $u = 0$ and the parameter values from Table 10.1, or by pseudo-timestepping. The steady state of the chain is depicted in Figure 10.2.

10.1.3 Numerical results

We aim at a sampling period of $\delta = 0.04$ to address the fast dynamics of the chain. RTI with this sampling yields the feedback control profiles depicted in Figure 10.3 and the state trajectories partly depicted in Figures 10.4 and 10.5. The overall quality of the RTI scheme is good, with a performance index of $I_{\text{perf}} = 550.3029$, where the performance



(a) Initial state of the chain.

(b) Steady state of the chain.

Figure 10.2: Initial state (left) and steady state (right) of the chain with $N = 9$ balls and parameter values from Table 10.1.

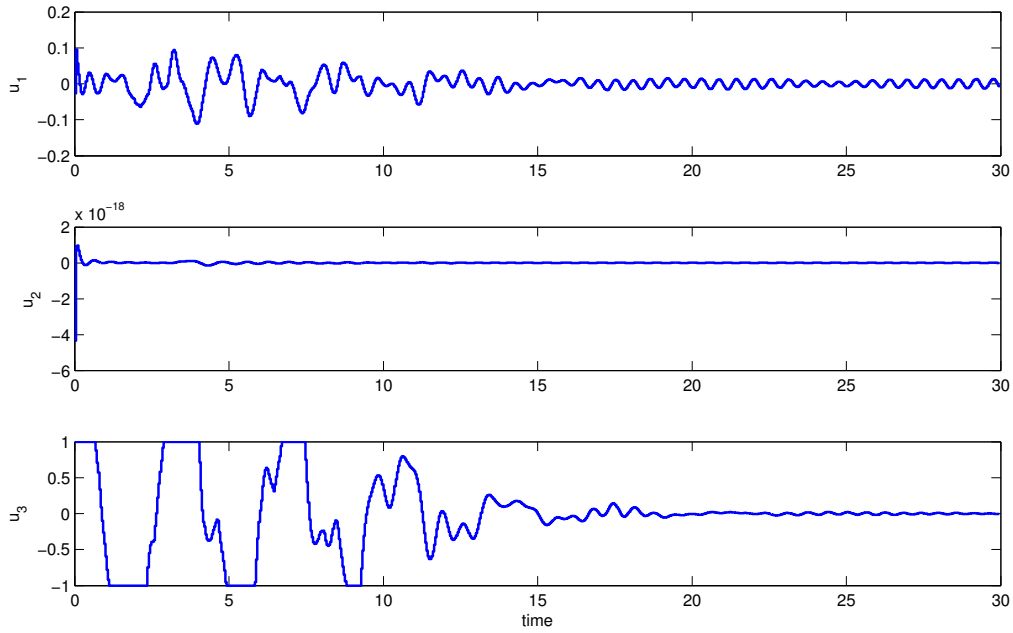


Figure 10.3: RTI solution: control profiles. The controls are the prescribed velocities at chain end mass. The scenario dynamics take place solely in the x_1-x_3-plane.

index is defined as

$$I_{\text{perf}} = \int_0^{30} L(x, v, u)\mathrm{d}t. \tag{10.8}$$

On first sight, the x_1-velocities depicted in Figure 10.4 may seem unsatisfactorily large, however, one has to take into account that the scale of the x_1-velocities differs quite significantly from the scale of the x_3-velocities. The magnitude of the remaining residual velocities is actually of equal order for both x_1- and x_3-velocities and for the controls u_1 and u_3, which are just the controlled velocities of the end point of the chain. The reason for the remaining residual velocities is the fact that the Gauss-Newton approach samples the objective function only in the Multiple Shooting nodes. Thus, small high-frequent

residual oscillations remain which cannot be eliminated with the resolution determined by the Multiple Shooting discretization. The oscillations can be reduced by refining the Multiple Shooting grid, however, the increase of the computational effort soon outweighs the benefits.

While the control performance of the RTI scheme is good, the scheme fails to be real-time feasible by a significant margin. With a peak of about 232 ms CPU time per sample and an average of about 158 ms CPU time per sample, the RTI scheme takes four to six times the sampling period CPU time per sample. We thus apply MLI to the control scenario and aim to obtain an MLI scheme which is assembled adaptively, is real-time feasible, and shows a comparable control performance.

For the adaptive level choice, we essentially employ Algorithm 7, with the small modification that we separately calculate the norms of the Lagrange gradient approximation and the constraint vector in level-B, and check each for a prescribed lower bound. We use both the postiteration approach and the spectral radius estimation approach for various tuning parameters, and compare the resulting MLI schemes. As data communication strategy, we employ the maximum data communication, cf. Section 5.3.2, i.e., the MLI levels iterate on a common set of primal-dual variables.

To account for the different computational times required for each level, we specify for the levels B, C, and D the numbers $n_{\mathrm{B}}, n_{\mathrm{C}}, n_{\mathrm{D}}$ of sampling periods needed to perform the respective calculations. Then, whenever an iteration of level $(\cdot)$ is scheduled, we perform $n_{(\cdot)} - 1$ level-A iterations in advance. This is a sequential model for a parallel computation of the level updates while continually giving feedback with the best available level-A controller. In the following, we use for both the postiterations approach and the spectral radius estimation approach the values $n_{\mathrm{B}} = 1, n_{\mathrm{C}} = 2, n_{\mathrm{D}} = 10$, which have been obtained by numerical experiments with an added safety margin.

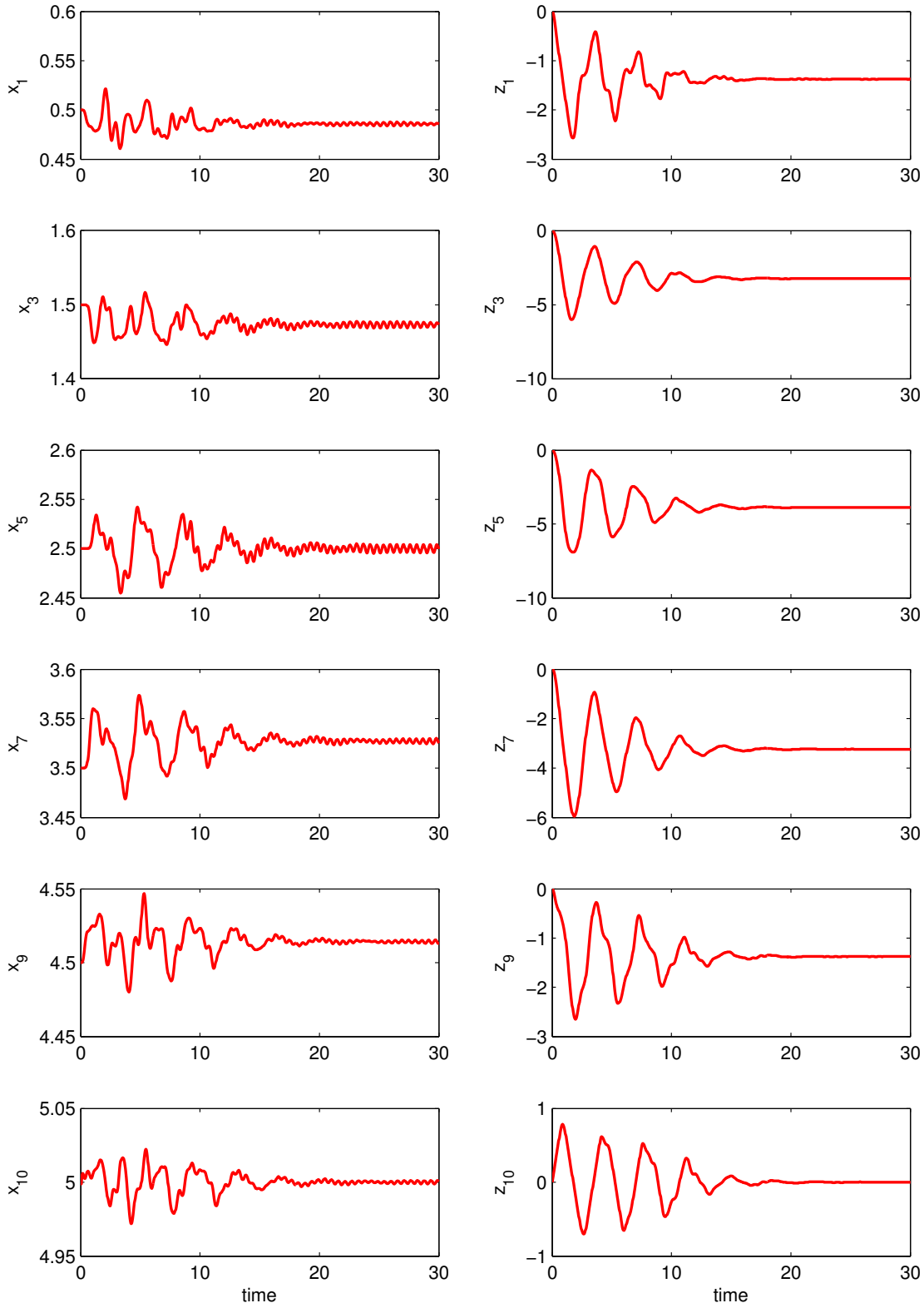


Figure 10.4: RTI solution: positions of masses with odd index and of chain end mass. The RTI controller brings the chain to rest.

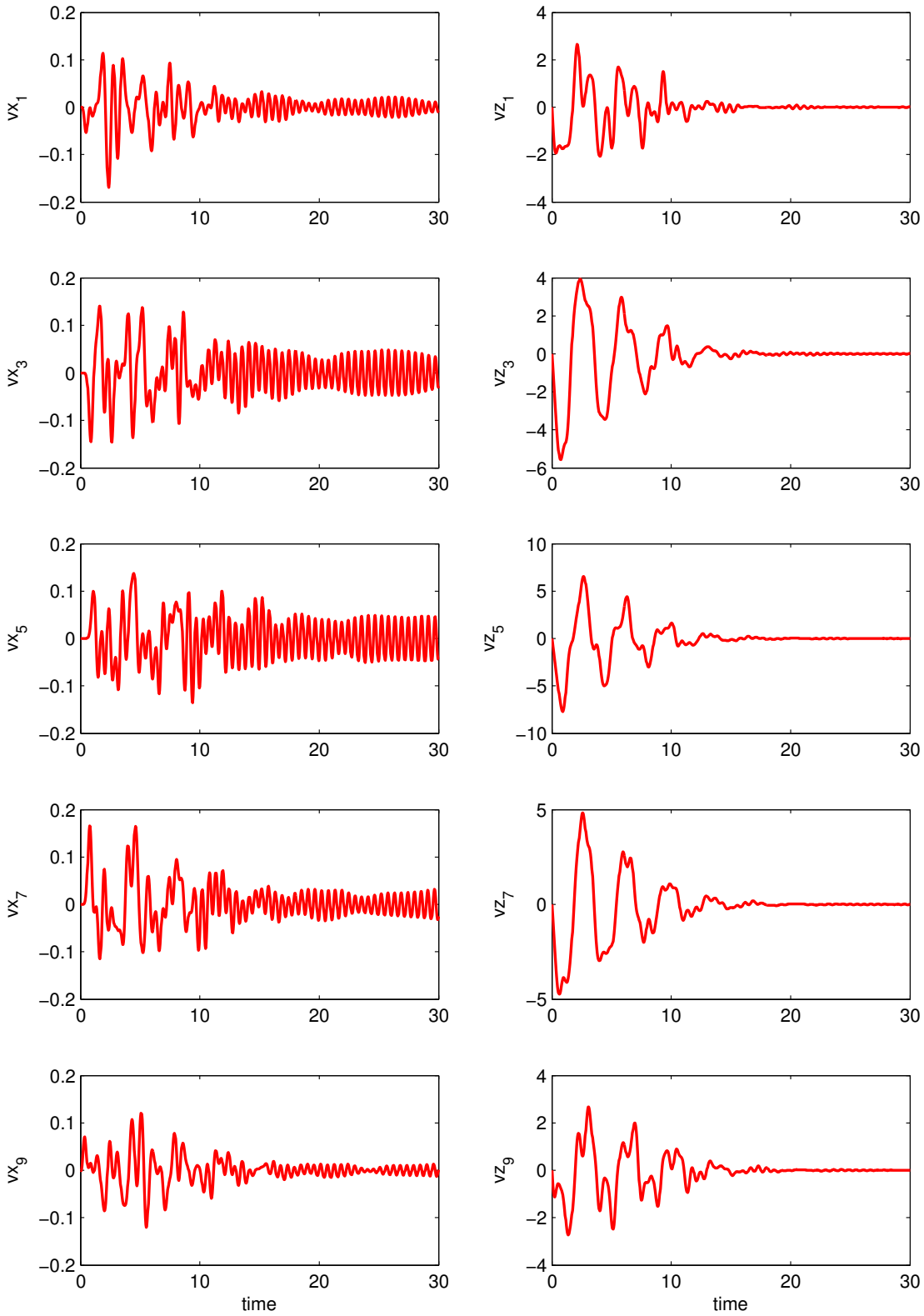


Figure 10.5: RTI solution: velocities of masses with odd index. Note the different scales of x_1- and x_3-velocities. Small residual oscillations remain and cannot be removed completely.

Adaptive level choice by postiterations

For the adaptive level choice by postiterations, we choose $\delta_{\mathrm{B}} = \delta_{\mathrm{C}} = 0.5$ as the upper limits for acceptable contraction of the level-B and level-C iterations. We vary the lower limits for the constraint vector norm $\gamma_{\mathrm{B,c}}$ and for the Lagrange gradient approximation norm $\gamma_{\mathrm{B,l}}$. For choosing these values, it is reasonable to orient oneself at the function evaluation accuracy, which is in this case essentially the integrator accuracy chosen at 10^{-5}. Thus, we vary $\gamma_{\mathrm{B,c}}$ and $\gamma_{\mathrm{B,l}}$ between 10^{-1} and 10^{-4}. For these values, we choose the upper limit for the acceptable Lagrange gradient norm in level-C as $\lambda_{\mathrm{C}} = 1$, which is a rather conservative value but is suitable to illustrate the influence of this parameter on the scheme generation.

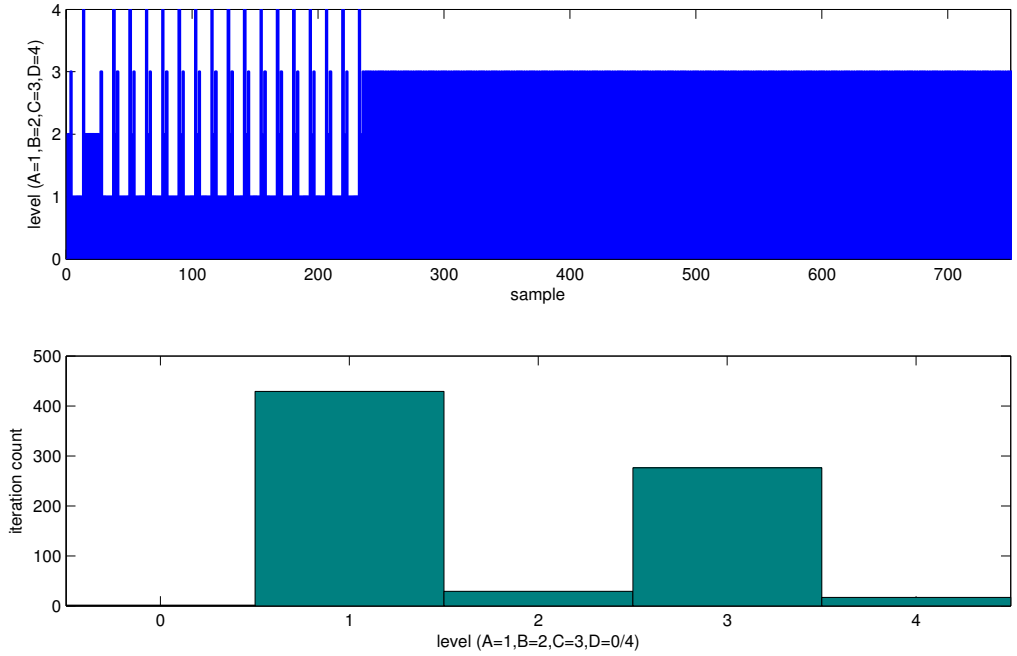


Figure 10.6: Adaptive level choice by postiterations. *Top:* MLI scheme for $\gamma_{\mathrm{B,c}} = \gamma_{\mathrm{B,l}} = 10^{-1}$. The levels A to D are denoted by the numbers 1 to 4 on the y-axis. The first part of the scheme is dominated by cycles of level-B/level-C/level-D sequences, the second part consists of level C iterations. Note that each level-C iteration has a preceding level A iteration, which is not visible due to resolution limits. *Bottom:* Histogram of level iterations used in the scheme. The levels A to D are denoted by the numbers 1 to 4 on the x-axis (0 denotes the initial iteration and is of type level D). The number of iterations of the respective level occurring in the scheme are given on the y-axis, summing up to a total of 750 iterations/sampling times. The most frequently scheduled level is level C, the most frequently occurring level is level A. Every level-C iteration has one, every level-D iteration except the initial iteration has nine preceding level-A iterations.

The results for $\gamma_{\mathrm{B,c}} = \gamma_{\mathrm{B,l}} = 10^{-1}$ are depicted in Figures 10.6, 10.7, and 10.8. In Figure 10.6, the resulting MLI scheme and the number of iterations of each of the levels occurring in the scheme are shown. The first part of the scheme, up to around sample 230, shows many level changes in the form of cycles of level-B/level-C/level-D sequences.

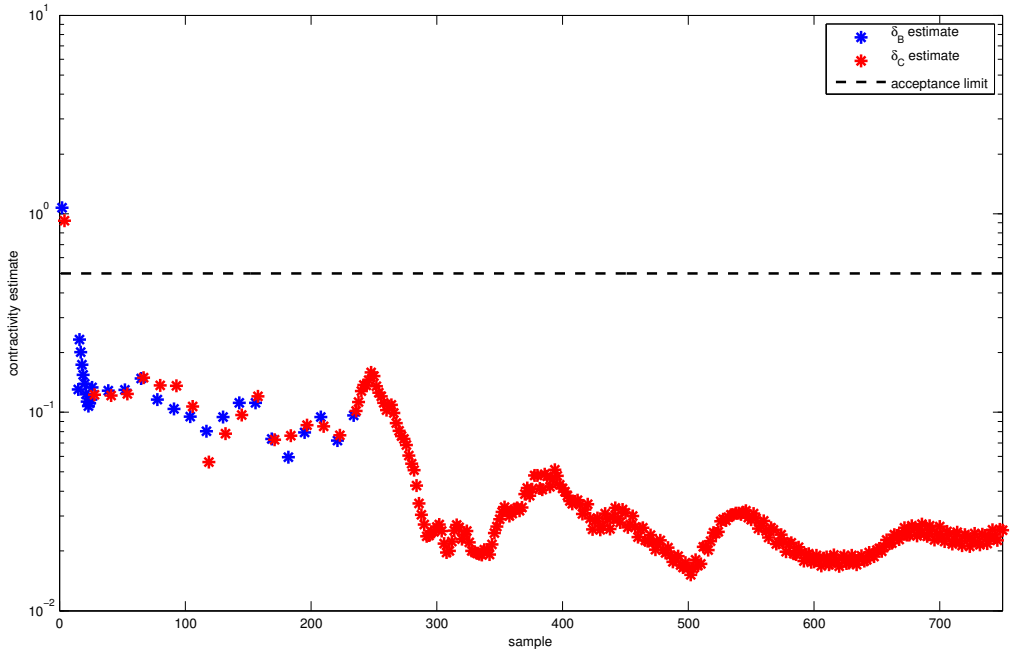


Figure 10.7: Adaptive level choice by postiterations for $\gamma_{\mathrm{B,c}} = \gamma_{\mathrm{B,l}} = 10^{-1}$. Contractivity estimates. For most samples, the contractivity estimates are well below the acceptable threshold of $\delta_{\mathrm{B/C}} = 0.5$.

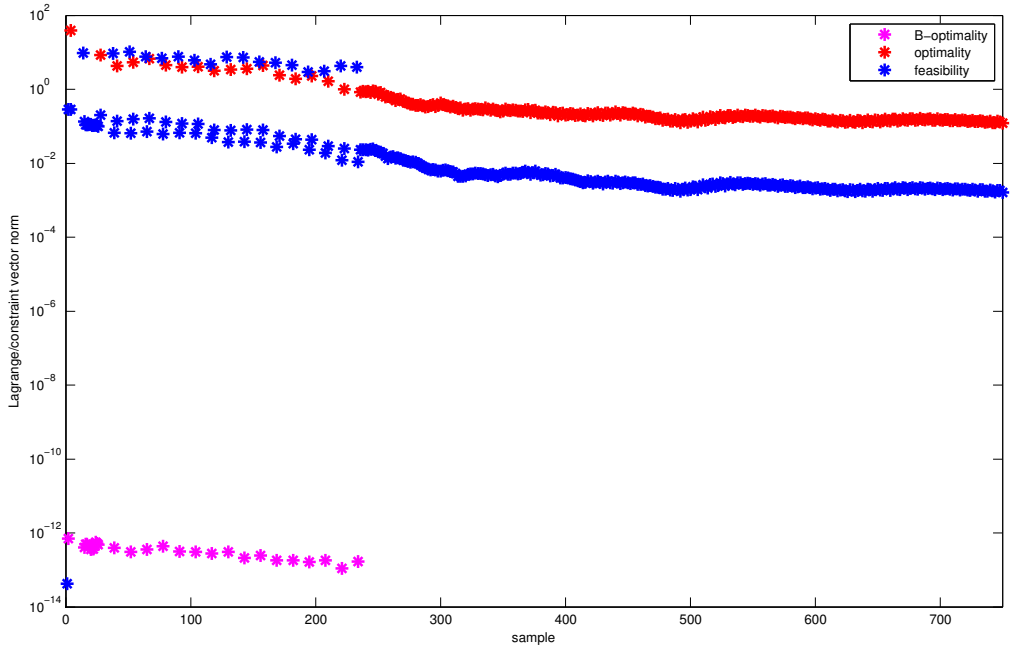


Figure 10.8: Adaptive level choice by postiterations for $\gamma_{\mathrm{B,c}} = \gamma_{\mathrm{B,l}} = 10^{-1}$. Constraint and gradient vector norms. In the first part, constraint norms in level-B iterations and Lagrange gradient norms in level-C iterations trigger level changes.

The second part consists of level-C iterations. In the discussion, we mainly refer to the

iteration types scheduled by the adaptive level choice. It should be kept in mind that each level-C iteration is preceded by one, each level-D iteration by nine level-A iterations. Thus, level-A iterations are the most often occurring iteration type in the resulting scheme, while level-C iterations are the most often scheduled iteration type, as can be seen in the level histogram, which renders the overall computational effort for this scheme significantly lower than for the corresponding RTI scheme.

From Figure 10.7 we can see that, with the exception of the very first samples, the contraction estimates are well below the acceptance threshold of $\delta_{\mathrm{B/C}} = 0.5$, which means that only the first level-B/level-C/level-D sequence is scheduled because of lack of contractivity. The reason for the other level-B/level-C/level-D sequences are shown in Figure 10.8, which shows the norms of the approximated Lagrange gradient and the constraints for level-B iterations and the norms of the Lagrange gradient and the constraints for level-C iterations. The level-C iterations are scheduled because the norm of the constraint vector is below the threshold $\gamma_{\mathrm{B,c}} = 10^{-1}$, and in the following the level-D iterations are scheduled because the norm of the Lagrange gradient is above $\lambda_C = 1$. Since the level-D iterations continue with a level-B iteration next, the cycle is then completed. In the second part of the scheme, the norm of the Lagrange gradient is below the threshold λ_C in the level-C iterations, and, since the contraction estimate is also below the threshold δ_{C}, no further level changes have to be scheduled.

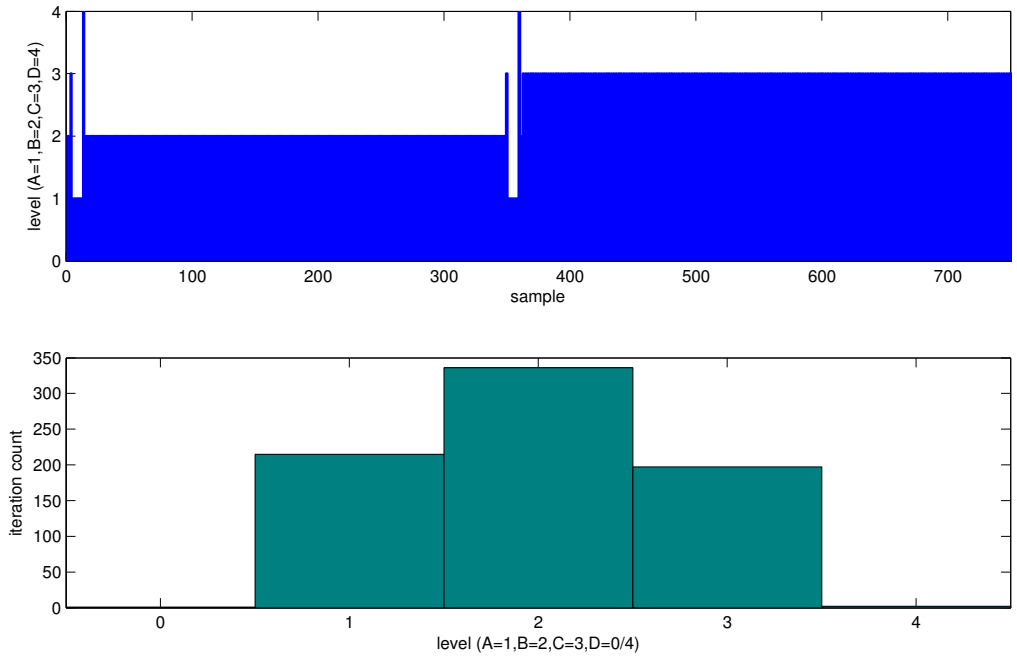


Figure 10.9: Adaptive level choice by postiterations. *Top:* MLI scheme for $\gamma_{\mathrm{B,c}} = \gamma_{\mathrm{B,l}} = 10^{-2}$. *Bottom:* Histogram of level iterations used in the scheme. The first part of the scheme is dominated by level-B iterations, the second part is dominated by level-C iterations, with two level-C/level-D-combinations preceding the parts.

The results for $\gamma_{\mathrm{B,c}} = \gamma_{\mathrm{B,l}} = 10^{-2}$ are depicted in Figures 10.9, 10.10, and 10.11. From

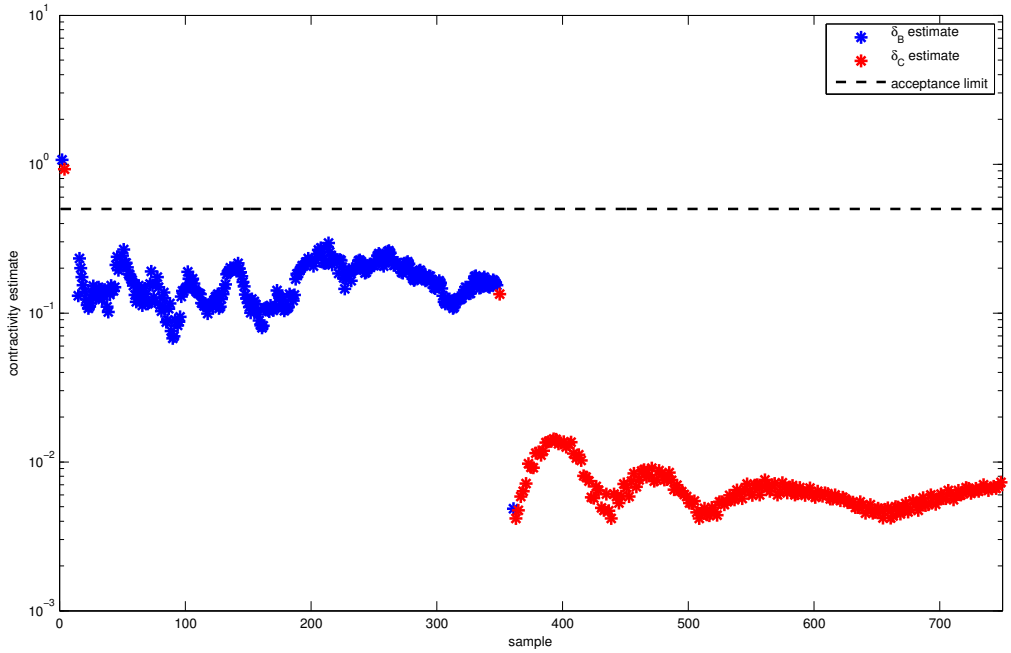


Figure 10.10: Adaptive level choice by postiterations for $\gamma_{\mathrm{B,c}} = \gamma_{\mathrm{B,l}} = 10^{-2}$. Contractivity estimates. For most samples, the contractivity estimates are well below the acceptable threshold of $\delta_{\mathrm{B/C}} = 0.5$.

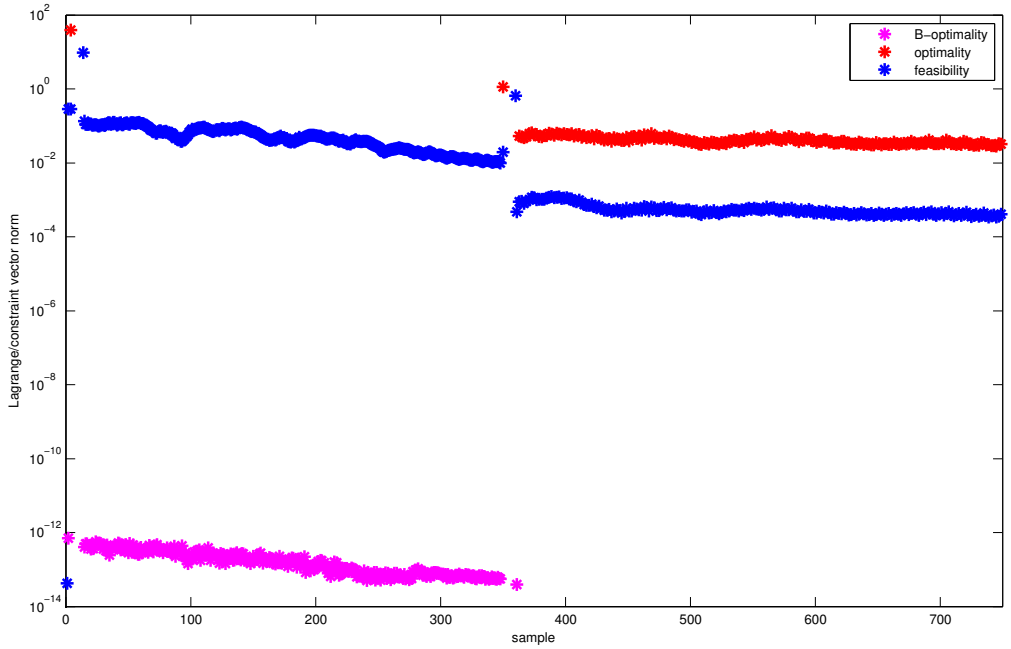


Figure 10.11: Adaptive level choice by postiterations for $\gamma_{\mathrm{B,c}} = \gamma_{\mathrm{B,l}} = 10^{-2}$. Constraint and gradient vector norms. Level-B constraint vector norms decrease until the threshold 10^{-2} is transgressed.

Figure 10.9 we can see that the first part of the scheme consists of level-B iterations, the second part consists of level-C iterations, and both parts are preceded by a single level-C/level-D-combination. Overall, the most often scheduled iteration type are level-B iterations, and the overall computational effort for this scheme is again significantly less than for the corresponding RTI scheme.

From Figure 10.10, which depicts the estimated contraction for level-B and level-C iterations, we can see that the first of the level-C/level-D-combinations is scheduled because in the beginning the contraction condition is violated for the level-B iterations and then also for the subsequent scheduled level-C iteration. The new linearizations provided by the level D iteration then yield contraction estimates well below the threshold of $\delta_{\mathrm{B}} = 0.5$, thus level-B iterations are performed until the norm of the constraint vector is smaller than the threshold 10^{-2}, which happens around sample 350, cf. Figure 10.11. The second level-C/level-D-combination is then scheduled because the norm of the Lagrange gradient in the following level-C iteration is above the threshold of $\lambda_{\mathrm{C}} = 1$. The following level-B iteration again schedules level-C next due to the constraint norm condition, and for the rest of the scheme, the level-C iterations satisfy both the conditions for contractivity and Lagrange gradient norm.

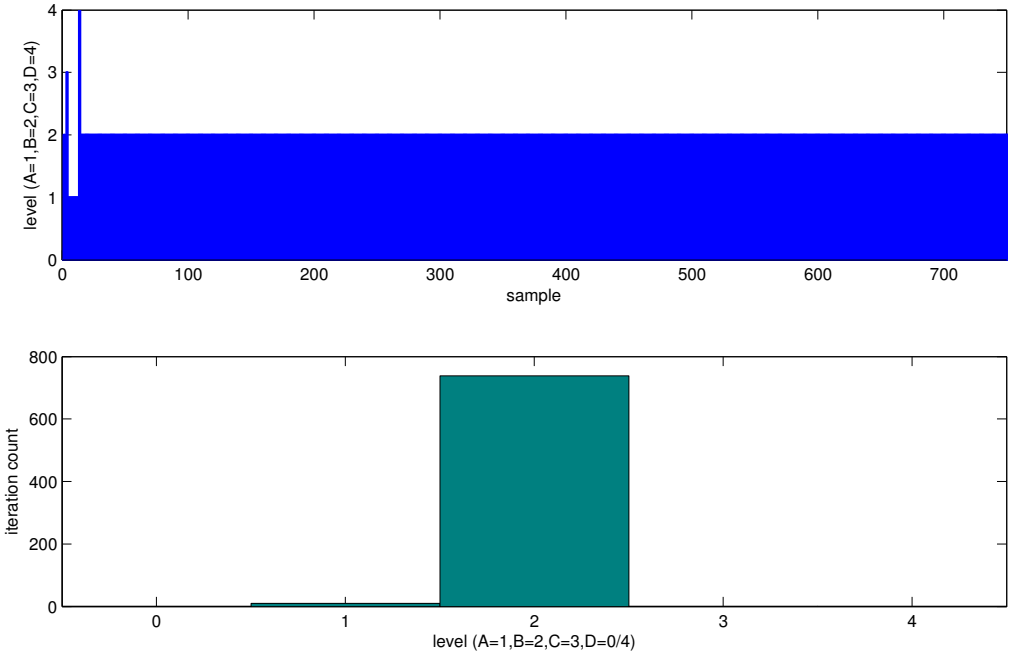


Figure 10.12: Adaptive level choice by postiterations. *Top:* MLI scheme for $\gamma_{\mathrm{B,c}} = \gamma_{\mathrm{B,l}} = 10^{-3}$. *Bottom:* Histogram of level iterations used in the scheme. This scheme is mostly a level-B scheme and is also obtained for smaller values of $\gamma_{\mathrm{B,c}}$ and $\gamma_{\mathrm{B,l}}$.

The results for $\gamma_{\mathrm{B,c}} = \gamma_{\mathrm{B,l}} = 10^{-3}$ are depicted in Figures 10.12, 10.13, and 10.14. From Figure 10.12 we can see that the scheme is essentially a level-B scheme preceded by a single level-C/level-D-combination. From Figure 10.13, we can see that the level-C/level-D-combination is scheduled because in the beginning the contraction condition

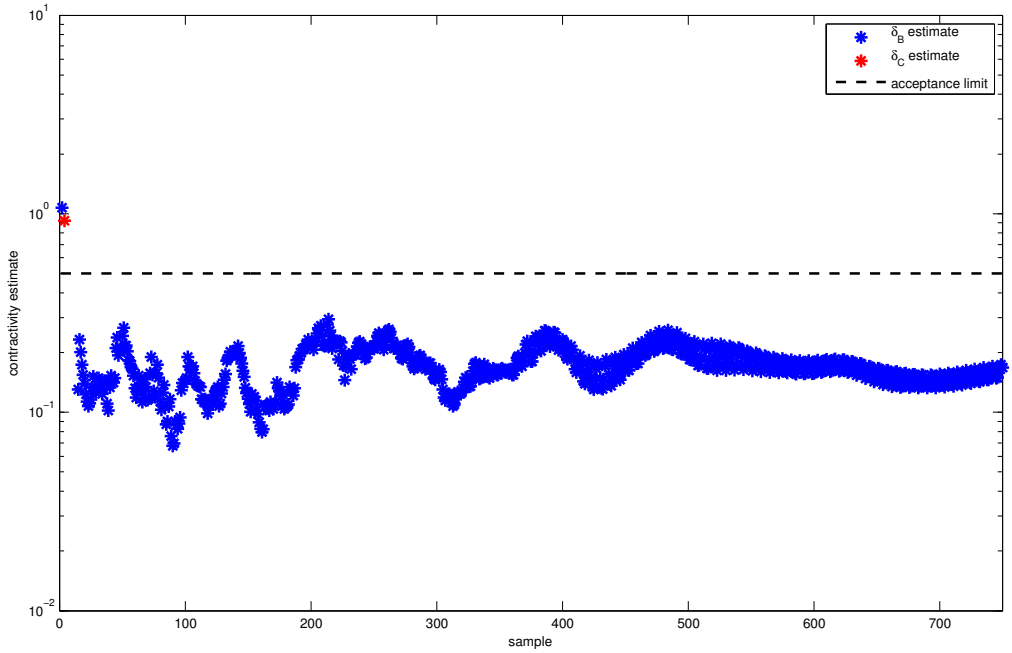


Figure 10.13: Adaptive level choice by postiterations for $\gamma_{\mathrm{B,c}} = \gamma_{\mathrm{B,l}} = 10^{-3}$. Contractivity estimates. For most samples, the contractivity estimates are well below the acceptable threshold of $\delta_{\mathrm{B/C}} = 0.5$.

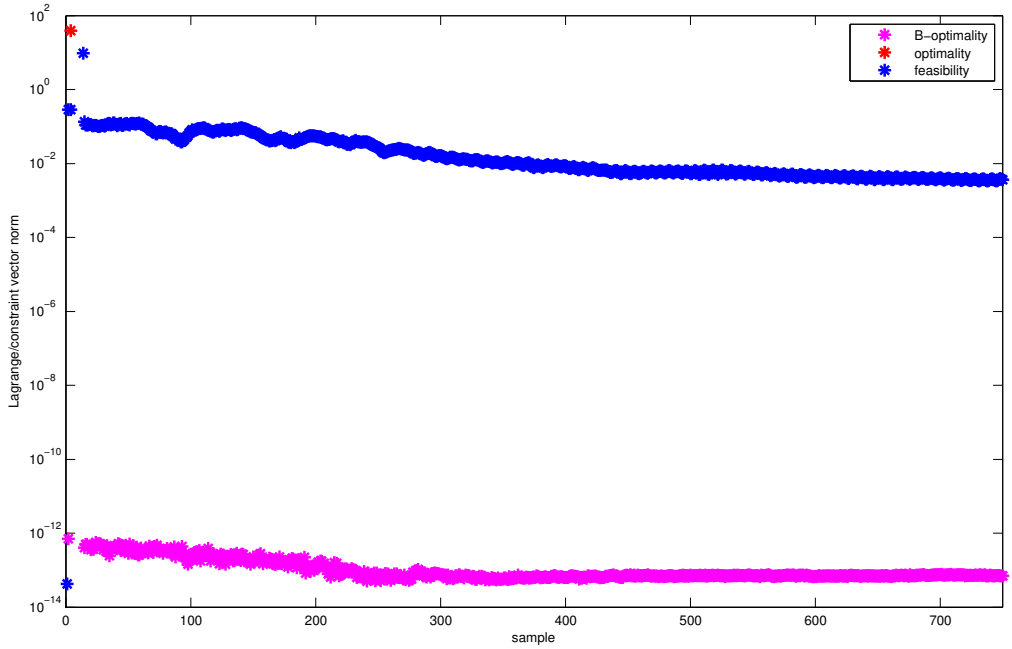


Figure 10.14: Adaptive level choice by postiterations for $\gamma_{\mathrm{B,c}} = \gamma_{\mathrm{B,l}} = 10^{-3}$. Constraint and gradient vector norms. The feasibility threshold of 10^{-3} is not transgressed at any sample.

is violated for the level-B iterations and then also for the subsequent scheduled level-C

iteration. The new linearizations provided by the level D iteration then yield contraction estimates well below the threshold of $\delta_{\mathrm{B}} = 0.5$, thus level-B iterations are performed in the following. The threshold $\gamma_{\mathrm{B,c}} = 10^{-3}$ is at no time transgressed, as depicted in Figure 10.14, thus no further level change has to be scheduled. The exact same scheme, contractivity estimates, and constraint and gradient vector norms are also obtained for the choice $\gamma_{\mathrm{B,c}} = \gamma_{\mathrm{B,l}} = 10^{-4}$.

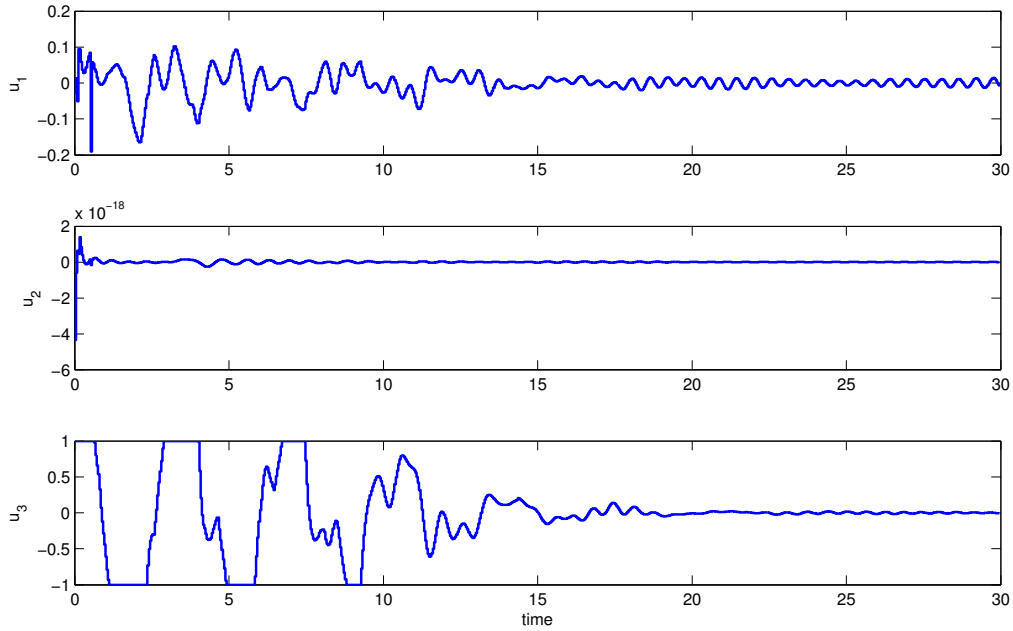


Figure 10.15: Adaptive level choice by postiterations for $\gamma_{\mathrm{B,c}} = \gamma_{\mathrm{B,l}} = 10^{-2}$. Feedback control profiles.

The feedback control profiles generated by the adaptive schemes are almost identical and only small changes to the RTI feedback control profile in Figure 10.3 are visible. As an example, we show the control profiles of the adaptive MLI scheme for $\gamma_{\mathrm{B,c}} = \gamma_{\mathrm{B,l}} = 10^{-2}$ in Figure 10.15. The performance indices (10.8) for the discussed MLI schemes with adaptive level choice by postiterations are given in Table 10.2. For comparison, the performance indices of the RTI scheme and the fixed MLI schemes A^1C^2 and A^1D^{10} are also given (note that the A^1C^2 scheme even fails to bring the chain to rest). The MLI schemes with adaptive level choice by postiterations all succeed to bring the chain to rest, have performance indices comparable to the RTI scheme, and are computationally significantly less expensive than the RTI scheme. Even more, the adaptive schemes are, by construction, real-time feasible in the sense that in a practical realization using parallelization, the updated matrix and vector data, i.e., the computations performed in level-B, level-C, and level-D iterations, could be provided in real-time to the controller running level-A iterations. By the choice of the tuning parameters $\gamma_{\mathrm{B,c}}$, $\gamma_{\mathrm{B,l}}$, and λ_{C}, the user can balance between preferring cheap level-B iterations or more frequent updates of the derivative information.

Adaptive level choice by spectral radius estimation

For the adaptive level choice by spectral radius estimation, we choose $\sigma_B = \sigma_C = 0.25$ as the upper limits for an acceptable spectral radius estimate of the level-B and level-C iterations. The spectral radius estimation is realized via the MATLAB® function `eigs` [172] with a convergence tolerance of tol $= 10^{-3}$. For this proof-of-concept test case, we have calculated the estimates of the spectral radius for both level-B and level-C iteration in each level-B or level-C iteration. The integrator accuracy is again chosen at 10^{-5}. We vary the lower limits for the constraint vector norm $\gamma_{B,c}$ and for the Lagrange gradient approximation norm $\gamma_{B,l}$ between 10^{-1} and 10^{-4}. For these values, we choose the upper limit for the acceptable Lagrange gradient norm in level-C as $\lambda_C = 1$.

The results for $\gamma_{B,c} = \gamma_{B,l} = 10^{-1}$ are depicted in Figures 10.16 to 10.18, for $\gamma_{B,c} = \gamma_{B,l} = 10^{-2}$ in Figures 10.19 to 10.21, and for $\gamma_{B,c} = \gamma_{B,l} = 10^{-3}$ in Figures 10.22 to 10.24. The results for $\gamma_{B,c} = \gamma_{B,l} = 10^{-4}$ are again identical to the results for $\gamma_{B,c} = \gamma_{B,l} = 10^{-3}$. Since the resulting schemes and the explanations are very similar to their respective counterparts for the adaptive level choice with postiterations, we do not repeat the full discussion but concentrate on the differences.

For $\gamma_{B,c} = \gamma_{B,l} = 10^{-1}$, the main differences between the schemes from postiterations and spectral radius estimation are that in the beginning it takes for the latter a few more iterations until the estimates are consistently below the threshold of $\sigma_{B/C} = 0.25$, and that an additional level-D iteration is scheduled in the part dominated by the level-C iterations, which is due to the Lagrange gradient norm growing larger than the threshold of $\lambda_C = 1$. Overall, slightly less level-B iterations and slightly more level-C iterations are performed with the spectral radius estimation approach, with the number of level-D iterations remaining the same as for the postiterations approach.

For $\gamma_{B,c} = \gamma_{B,l} = 10^{-2}$, the schemes by postiterations and spectral radius estimation differ most, while still being structurally quite similar. For the spectral radius estimation approach, two instead of one level-D iterations are scheduled in the beginning of the scheme, and none in the later part. Both level-D iterations are scheduled due to the Lagrange gradient norm being larger than the threshold of $\lambda_C = 1$. Also, a few more iterations are needed until the estimates are consistently below the threshold of $\sigma_{B/C} = 0.25$. Furthermore, the part of the scheme consisting of level-B iterations is shorter, and the part consisting of level-C iterations is longer, because the feasibility threshold $\gamma_{B,c} = 10^{-2}$ is transgressed earlier, namely around sample 300. Overall, more level-C (and thus more level-A iterations) iterations and less level-B iterations are performed than for the postiterations approach.

For $\gamma_{B,c} = \gamma_{B,l} = 10^{-2}$, the main difference between the schemes from postiterations and spectral radius estimation is, that for the latter two instead of one level-D iterations are scheduled in the beginning of the scheme. This is again due to the Lagrange gradient norm being larger than the threshold of $\lambda_C = 1$ and the spectral radius estimates being larger than the threshold of $\sigma_{B/C} = 0.25$ for the first samples. For the rest of the scheme, level-B iterations are scheduled as for the postiteration approach. Overall, due to the additional level-D iteration, the scheme for spectral radius estimation is computationally

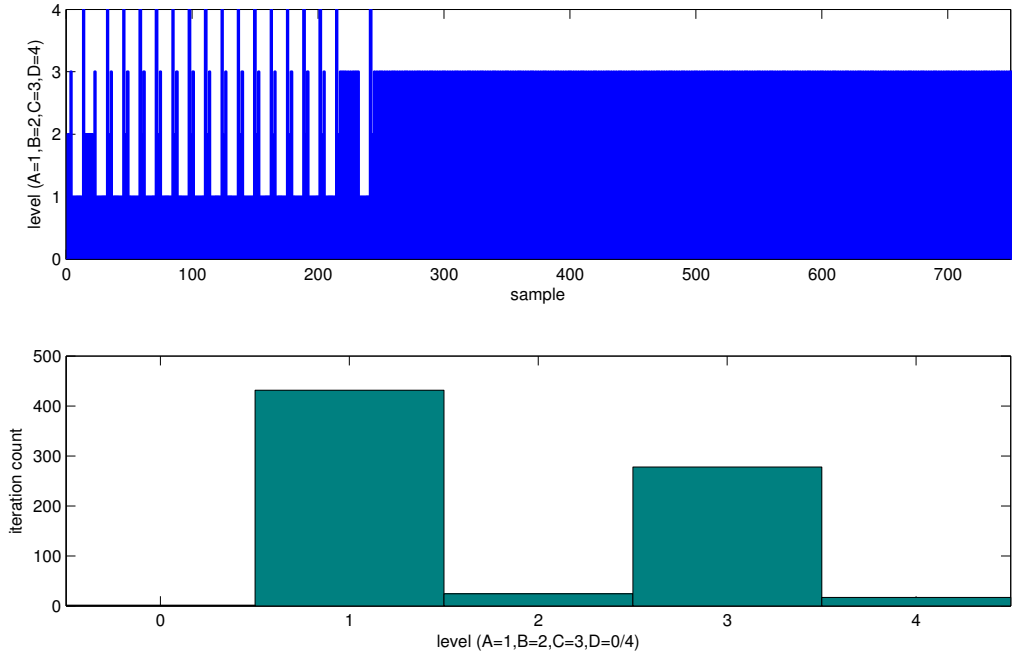


Figure 10.16: Adaptive level choice by spectral radius estimation. *Top:* MLI scheme for $\gamma_{\mathrm{B,c}} = \gamma_{\mathrm{B,l}} = 10^{-1}$. The levels A to D are denoted by the numbers 1 to 4 on the y-axis. The first part of the scheme is dominated by cycles of level-B/level-C/level-D sequences, the second part consists of level C iterations. Note that each level-C iteration has a preceding level A iteration, which is not visible due to resolution limits. *Bottom:* Histogram of level iterations used in the scheme. The levels A to D are denoted by the numbers 1 to 4 on the x-axis (0 denotes the initial iteration and is of type level D). The number of iterations of the respective level occurring in the scheme are given on the y-axis, summing up to a total of 750 iterations/sampling times. The most frequently adaptively scheduled level is level C, the most frequently occurring level is level A. Every level-C iteration has one, every level-D iteration except the initial iteration has nine preceding level-A iterations.

slightly more expensive than the corresponding scheme for the postiteration approach.

An interesting observation from Figures 10.17, 10.20, and 10.23 is the fact, that the spectral radius estimates for level-C are consistently larger than the spectral radius estimates for level-B. A possible explanation could be that the inverse of the iteration matrix M is closer to the exact Jacobian of the level-B iterations than to the exact Jacobian of the level-C iterations in the sense that

$$M^{-1} - J_B(x_k) = \begin{pmatrix} 0 & 0 & 0 \\ \widetilde{C}_k - J_{c,k} & 0 & 0 \\ \widetilde{D}_k - J_{d,k} & 0 & 0 \end{pmatrix}, \quad \text{and}$$

$$M^{-1} - J_C(x_k) = \begin{pmatrix} B_k - \nabla_w^2 \mathcal{L}(x_k) & -(\widetilde{C}_k - J_{c,k})^\intercal & -(\widetilde{D}_k - J_{d,k})^\intercal \\ \widetilde{C}_k - J_{c,k} & 0 & 0 \\ \widetilde{D}_k - J_{d,k} & 0 & 0 \end{pmatrix},$$

with the notation and definitions from Chapter 6. However, this explanation is pending future investigations and clarification.

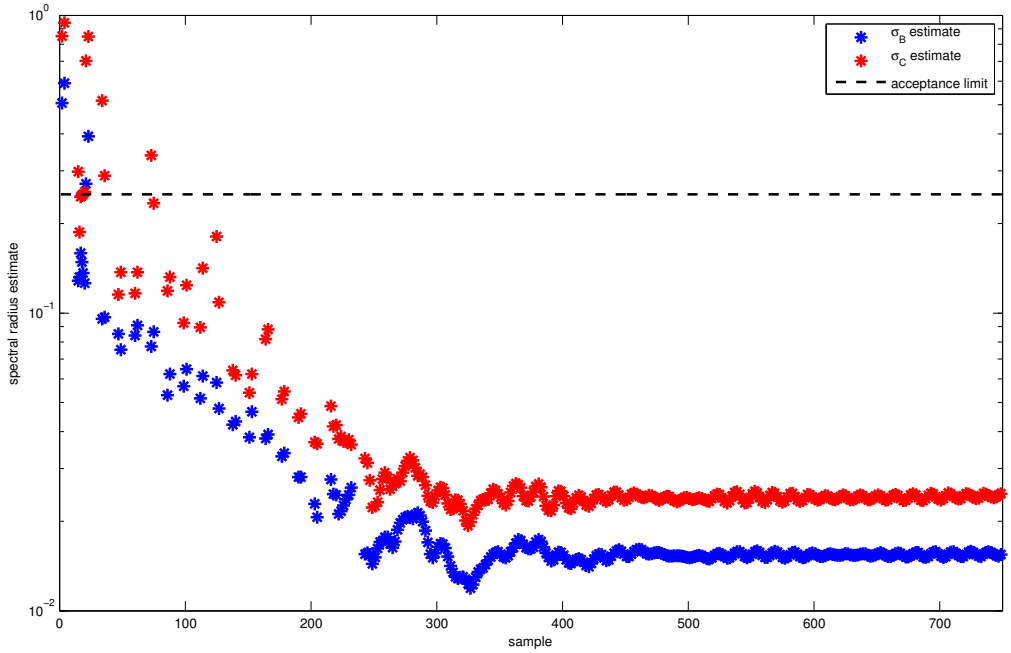


Figure 10.17: Adaptive level choice by spectral radius estimation for $\gamma_{\mathrm{B,c}} = \gamma_{\mathrm{B,l}} = 10^{-1}$. Contractivity estimates. For most samples, the contractivity estimates are well below the acceptable threshold of $\sigma_{\mathrm{B/C}} = 0.25$. Note that the estimates for level-C are consistently larger than the estimates for level-B.

The feedback control profiles generated by the adaptive schemes are again almost identical and only small changes to the RTI feedback control profile in Figure 10.3 are visible. As an example, we show the control profiles of the adaptive MLI scheme for $\gamma_{\mathrm{B,c}} = \gamma_{\mathrm{B,l}} = 10^{-2}$ in Figure 10.25. The performance indices (10.8) for the discussed MLI schemes with adaptive level choice by spectral radius estimation are given in Table 10.2. For comparison, the performance indices of the RTI scheme and the fixed MLI schemes A^1C^2 and A^1D^{10} are also given. The MLI schemes with adaptive level choice by spectral radius estimation all succeed to bring the chain to rest, have performance indices almost identical to the RTI scheme and slightly better than for the postiteration approach, and are computationally significantly less expensive than the RTI scheme. By the choice of the tuning parameters $\gamma_{\mathrm{B,c}}$, $\gamma_{\mathrm{B,l}}$, and λ_{C}, the user can balance between preferring cheap level-B iterations or more frequent updates of the derivative information. For the test case at hand, a slight preference towards level-C iterations at the expense of level-B iterations, as well as more frequent level-D iterations in the beginning can be observed for the spectral radius estimation approach compared to the schemes generated by the postiterations approach.

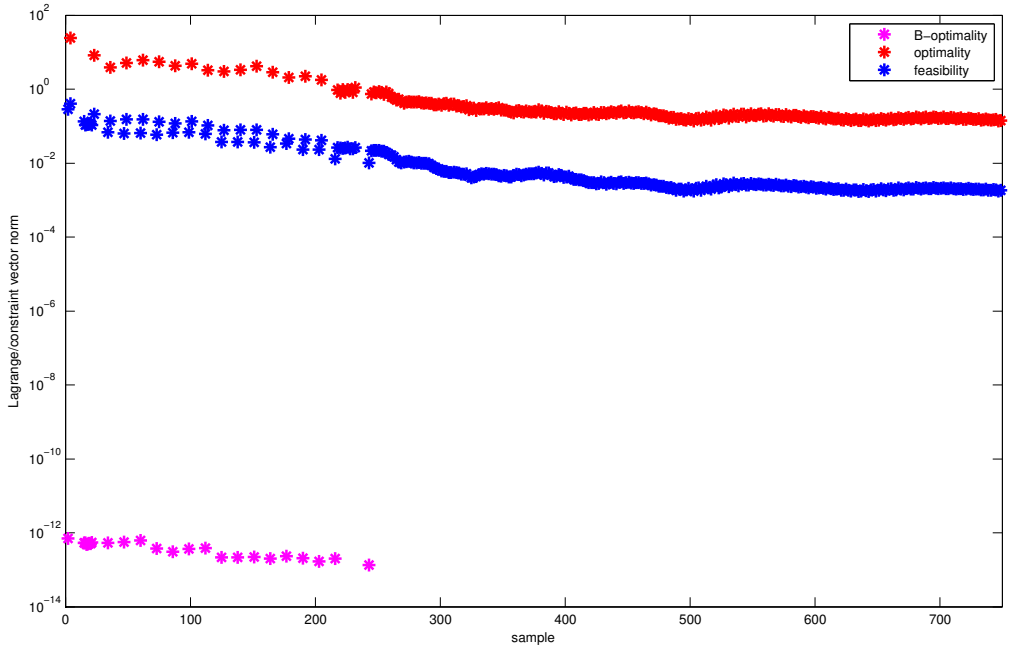


Figure 10.18: Adaptive level choice by spectral radius estimation for $\gamma_{\mathrm{B,c}} = \gamma_{\mathrm{B,l}} = 10^{-1}$. Constraint and gradient vector norms. In the first part, constraint norms in level-B iterations and Lagrange gradient norms in level-C iterations trigger level changes.

Table 10.2: Performance indices $I_{\mathrm{perf}} = \int_0^{30} L(x, v, u)\mathrm{d}t$ for adaptive level choice by postiterations and spectral radius estimation. Performance indices for RTI, fixed MLI scheme A^1C^2, and fixed MLI scheme A^1D^{10} are given for comparison.

Scheme	$\gamma_{\mathrm{B,c}}$	$\gamma_{\mathrm{B,l}}$	λ_{C}	I_{perf}
RTI	n/a	n/a	n/a	550.3029
A^1C^2	n/a	n/a	n/a	580.0345
A^1D^{10}	n/a	n/a	n/a	551.3768
Postiterations 1	10^{-1}	10^{-1}	1	550.6639
Postiterations 2	10^{-2}	10^{-2}	10	551.4191
Postiterations 3	10^{-3}	10^{-3}	10	551.4168
Postiterations 4	10^{-4}	10^{-4}	10	551.4168
Spectral radius est. 1	10^{-1}	10^{-1}	10	550.3925
Spectral radius est. 2	10^{-2}	10^{-2}	10	550.3755
Spectral radius est. 3	10^{-3}	10^{-3}	10	550.3772
Spectral radius est. 4	10^{-4}	10^{-4}	10	550.3772

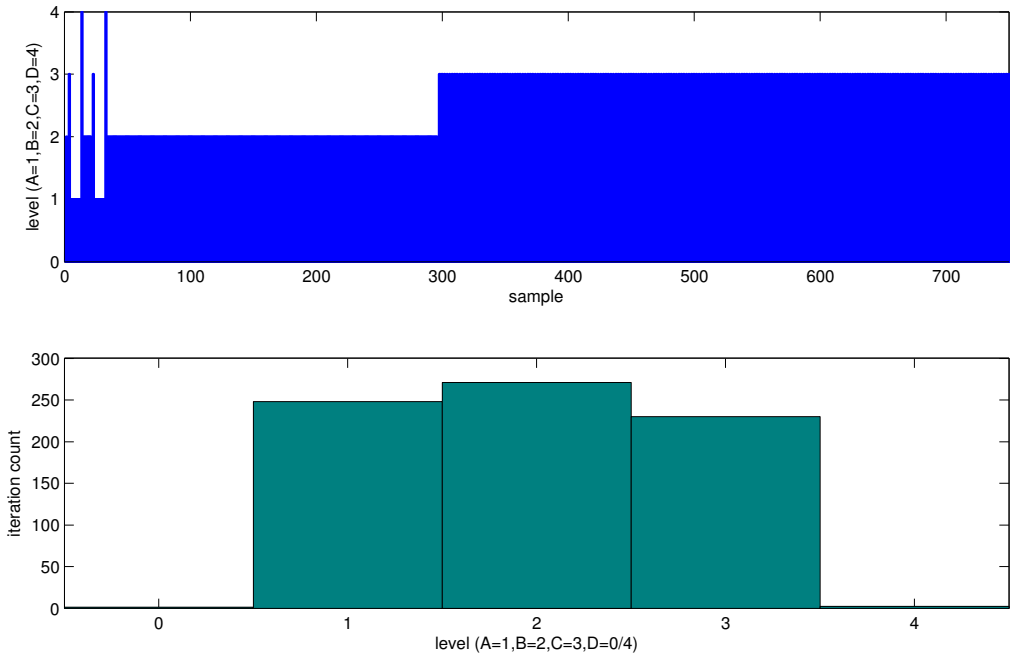


Figure 10.19: Adaptive level choice by spectral radius estimation. *Top:* MLI scheme for $\gamma_{\mathrm{B,c}} = \gamma_{\mathrm{B,l}} = 10^{-2}$. *Bottom:* Histogram of level iterations used in the scheme. The first part of the scheme is dominated by level-B iterations, the second part is dominated by level-C iterations, with two level-C/level-D-combinations preceding the first part.

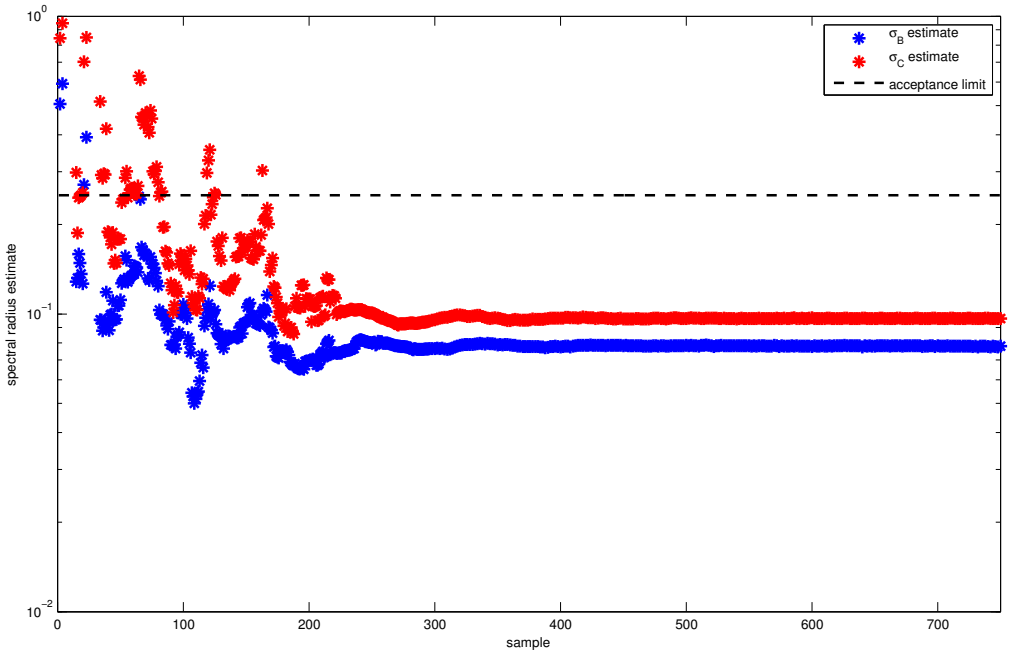


Figure 10.20: Adaptive level choice by spectral radius estimation for $\gamma_{\mathrm{B,c}} = \gamma_{\mathrm{B,l}} = 10^{-2}$. Contractivity estimates. For most samples, the contractivity estimates are well below the acceptable threshold of $\sigma_{\mathrm{B/C}} = 0.25$.

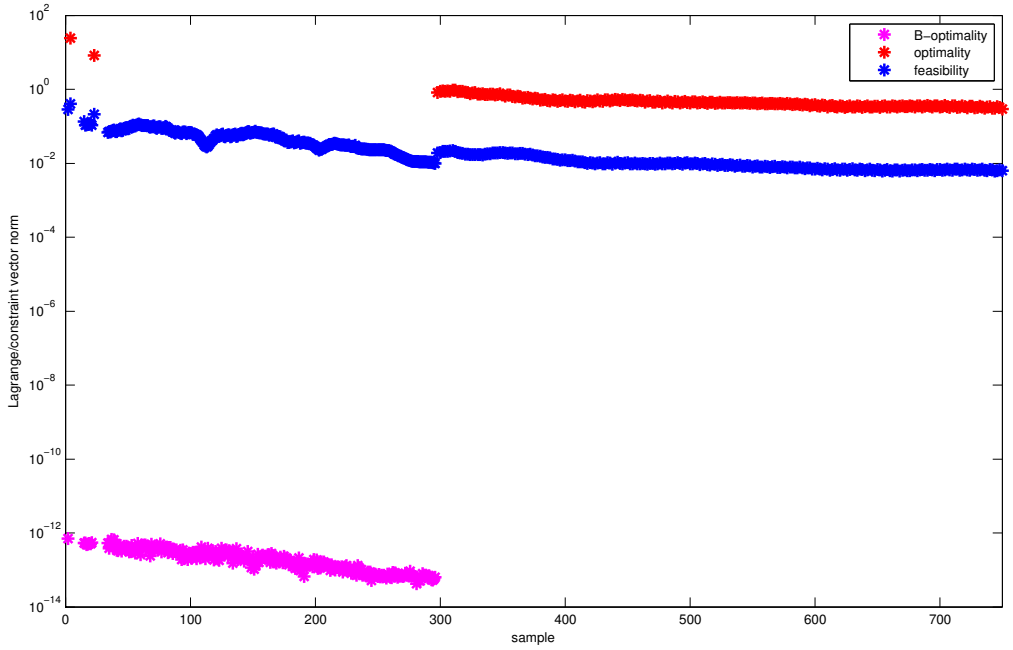


Figure 10.21: Adaptive level choice by spectral radius estimation for $\gamma_{\mathrm{B,c}} = \gamma_{\mathrm{B,l}} = 10^{-2}$. Constraint and gradient vector norms. Level-B constraint vector norms decrease until the threshold 10^{-2} is transgressed.

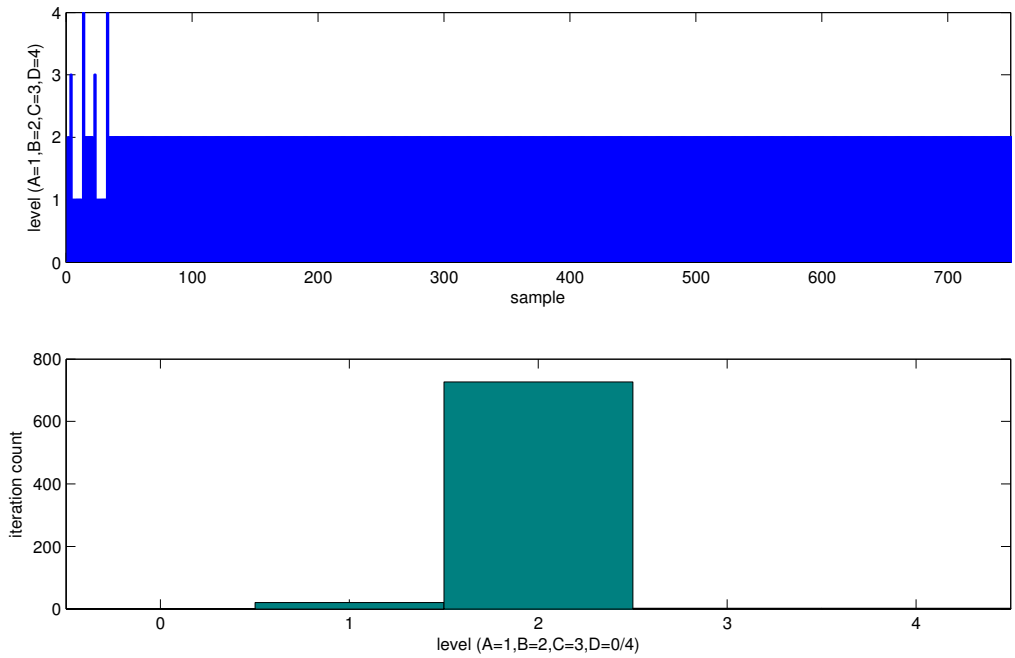


Figure 10.22: Adaptive level choice by spectral radius estimation. *Top:* MLI scheme for $\gamma_{\mathrm{B,c}} = \gamma_{\mathrm{B,l}} = 10^{-3}$. *Bottom:* Histogram of level iterations used in the scheme. This scheme is mostly a level-B scheme and is also obtained for smaller values of $\gamma_{\mathrm{B,c}}$ and $\gamma_{\mathrm{B,l}}$.

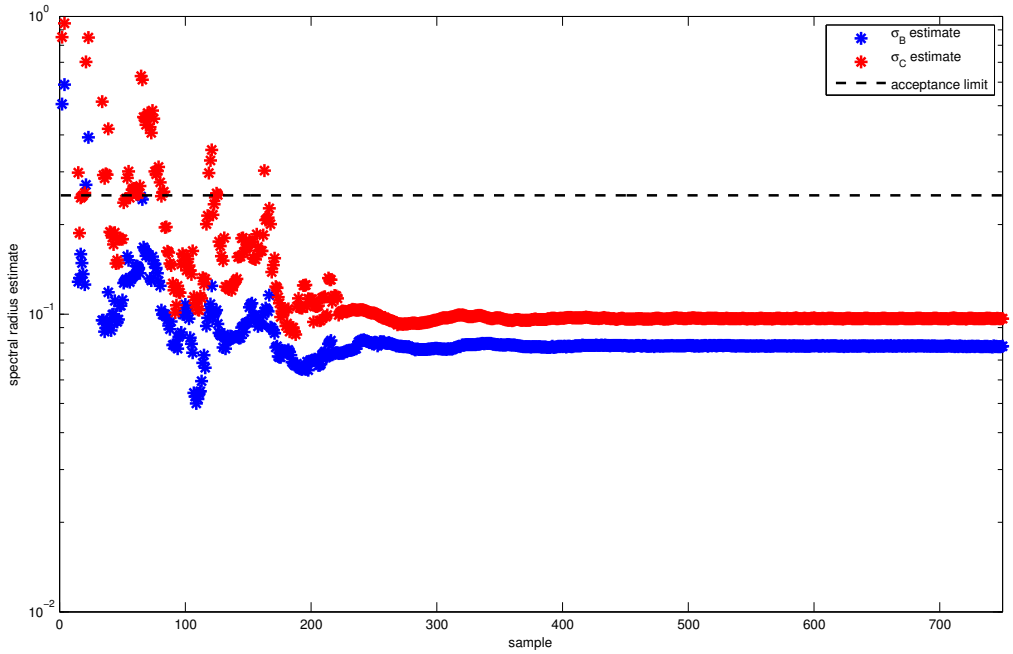


Figure 10.23: Adaptive level choice by spectral radius estimation for $\gamma_{\mathrm{B,c}} = \gamma_{\mathrm{B,l}} = 10^{-3}$. Contractivity estimates. For most samples, the contractivity estimates are well below the acceptable threshold of $\sigma_{\mathrm{B/C}} = 0.25$.

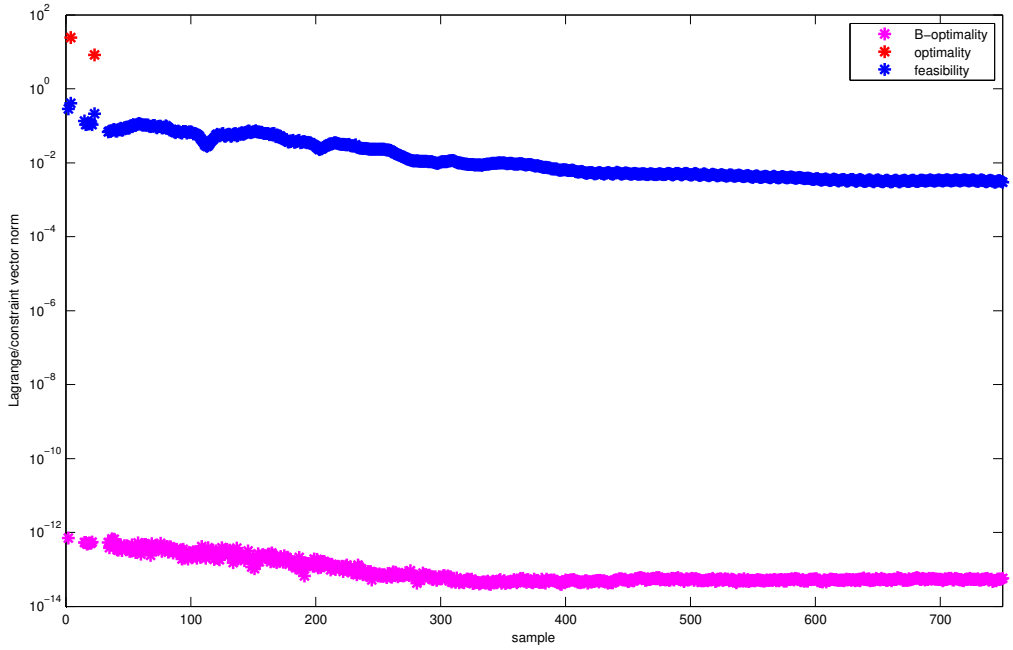


Figure 10.24: Adaptive level choice by spectral radius estimation for $\gamma_{\mathrm{B,c}} = \gamma_{\mathrm{B,l}} = 10^{-3}$. Constraint and gradient vector norms. The feasibility threshold of 10^{-3} is not transgressed at any sample.

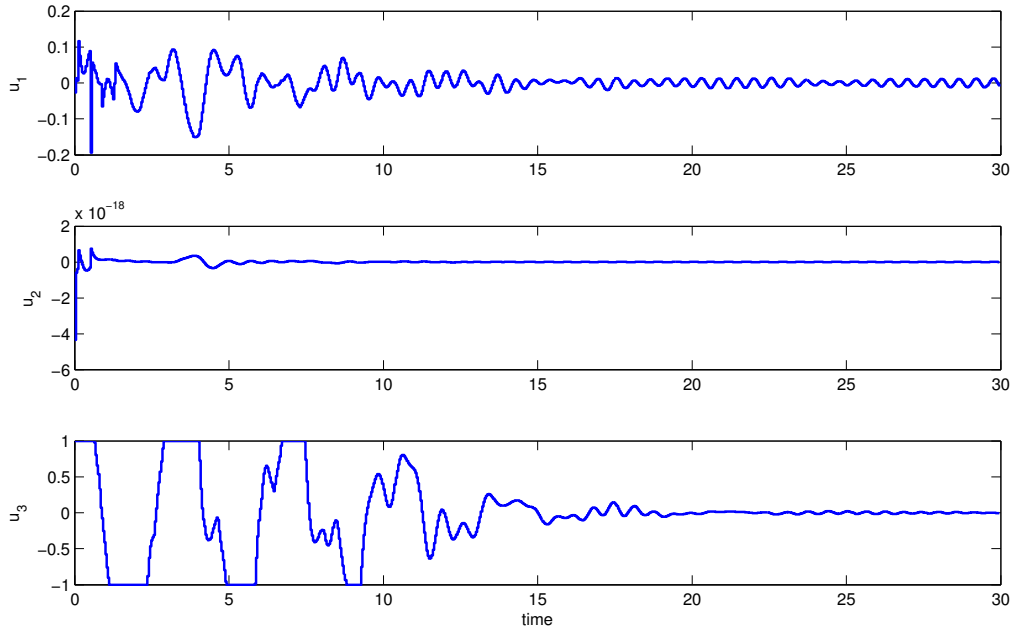


Figure 10.25: Adaptive level choice by spectral radius estimation for $\gamma_{\mathrm{B,c}} = \gamma_{\mathrm{B,l}} = 10^{-2}$. Feedback control profiles.

10.2 TESTDRIVE revisited: Adaptive level choice MLI

We have considered the TESTDRIVE scenario in the last chapter as a test case for MLI with fixed level choices. The results indicated that the process benefits from faster feedback. In this section, we apply MLI with adaptive level choice by postiterations to the test case. We give the control scenario and controller setup, and present and discuss the numerical results. The MLI schemes with adaptive level choice are able to reject the disturbance, and due to their ability to generate feedback faster than RTI they show clearly superior performance compared to the latter.

10.2.1 Control scenario

For the adaptive level choice test case, we use the dynamic model described in Section 9.2.1. The mass of the car is $1,239$ kg, and initially it is driving on a straight lane at a speed of 30 m/s. After 0.5 seconds, an impulse of magnitude $2.5 \cdot 10^4$ N is acting on the rear axle perpendicular to the driving direction for 0.05 seconds.

Aim of the controller is to keep the vehicle on the lane while retaining a speed of 30 m/s. The full system state information is available at a resolution of 0.01 seconds. The scenario runtime is 8 seconds.

The NMPC formulation of the scenario contains a least-square objective $L(x, u)$ to minimize the deviation from the straight lane as well as the prescribed velocity. Further, the controls $w_\delta, F_\mathrm{B}, \phi$ are regularized over the prediction horizon. The objective $L(x, u)$ reads

$$L(x, u) = c_\mathrm{y}^2 + (v - 30)^2 + w_\delta^2 + 10^{-12} F_\mathrm{B}^2 + 10^{-4} \phi^2. \tag{10.9}$$

Two-sided simple bounds on all states and controls are formulated, while no nonlinear constraints are present. As before, we leave out units in the following, however, all quantities and parameter values are formulated in the standard SI system and should be understood to have the correct corresponding unit.

10.2.2 Numerical results

For all numerical schemes, we use a prediction horizon of length 5, divided into 20 shooting intervals. The integrator tolerance is chosen as 10^{-5}. We use a Gauss-Newton approach, and the controller variables are initialized in the offline solution for the undisturbed moving car. All computations were performed with the software package MLI.

As in the previous chapter, we present the solution of the RTI scheme for a sampling period of $\delta = 0.05$. The RTI scheme is able to reject the disturbance and bringthe car back to the desired speed and driving direction. However, the scheme is not real-time feasible, since the CPU time per sample exceeds the sampling period of 0.05 seconds. The RTI solution (control profiles and most interesting states) is given in Figure 10.26.

We apply MLI with adaptive level choice by postiterations to the test case, because the postiterations approach is numerically less expensive and thus more suitable than the spectral radius estimation approach for small to medium size problems with fast

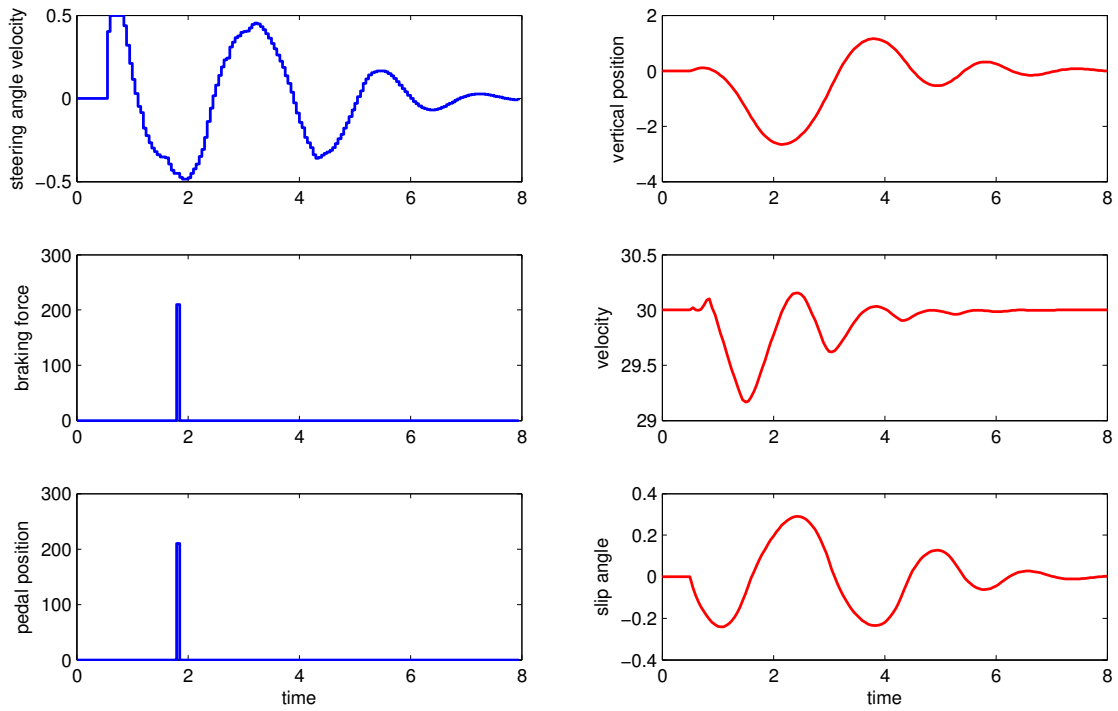


Figure 10.26: RTI solution for sampling $\delta = 0.05$. State and control profiles. The RTI controller is able to reject the disturbance, but the controller is not real-time feasible.

dynamics. We again use the modification of Algorithm 7 as described above, and employ maximum data communication. The upper limits for acceptable contraction of the level-B and level-C iterations are again set to $\delta_{\mathrm{B}} = \delta_{\mathrm{C}} = 0.5$. .

For the sampling rate we choose $\delta = 0.01$, exploiting the fact that we can reduce the sampling time significantly by selecting suitable values for $n_{\mathrm{B}}, n_{\mathrm{C}}, n_{\mathrm{D}}$. Here, numerical experiments suggest $n_{\mathrm{B}} = 2$, $n_{\mathrm{C}} = 4$, and $n_{\mathrm{D}} = 8$. We vary $\gamma_{\mathrm{B,c}}$ and $\gamma_{\mathrm{B,l}}$ from 10^{-2} to 10^{-4} and choose $\lambda_{\mathrm{C}} = 10$.

All three adaptive MLI schemes are able to reject the disturbance and bring back the car to the desired speed and driving direction. We give the schemes for $\gamma_{\mathrm{B,c}} = \gamma_{\mathrm{B,l}} = 10^{-1}$ in Figure 10.27 and for $\gamma_{\mathrm{B,c}} = \gamma_{\mathrm{B,l}} = 10^{-3}$ in Figure 10.28. In the first part, up to about sample 480, the schemes are quite similar, in the second part level-C iterations are dominantly scheduled in the former, level-B iterations are dominantly scheduled in the latter scheme. In absolute numbers, level-A iterations are by far the most often occurring iterations, which shows that computationally, this scheme is, even considering the higher feedback frequency, much cheaper than the RTI scheme.

We show the contraction estimates and the constraint and (approximated) Lagrange vector norms for $\gamma_{\mathrm{B,c}} = \gamma_{\mathrm{B,l}} = 10^{-1}$ in Figures 10.29 and 10.30, and for $\gamma_{\mathrm{B,c}} = \gamma_{\mathrm{B,l}} = 10^{-3}$ in Figures 10.31 and 10.32. The level-C and level-D iterations scheduled in both schemes in the first part are mostly due to insufficient contraction of the preceding level-B and level-C iterations. From around sample 600 on, the schemes show sufficient contraction, and in the following the scheme for $\gamma_{\mathrm{B,c}} = \gamma_{\mathrm{B,l}} = 10^{-1}$ schedules level-C iterations, since

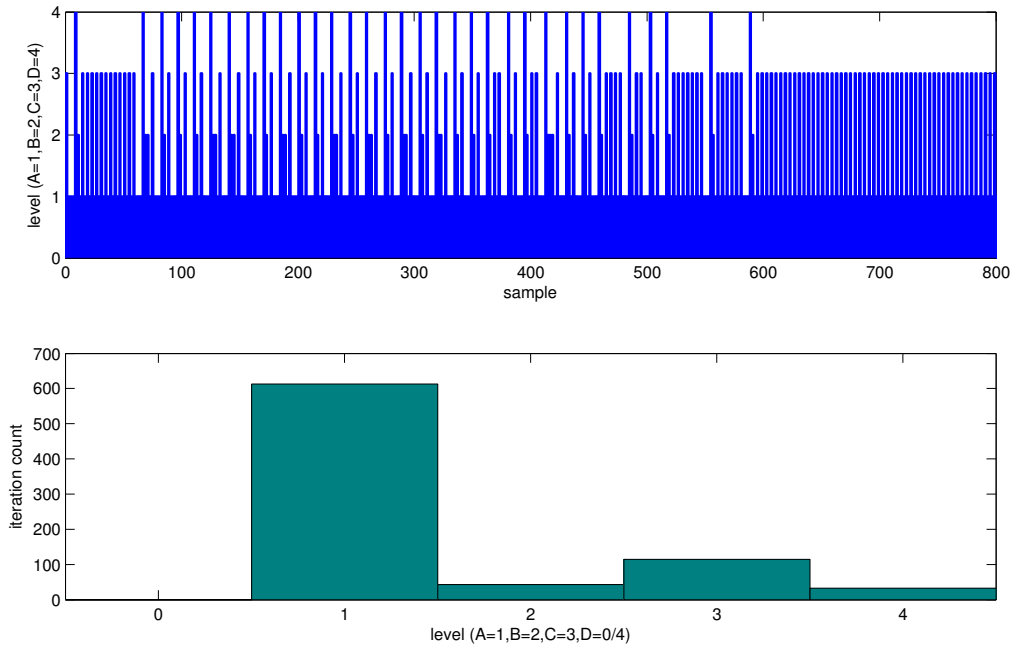


Figure 10.27: Adaptive level choice by postiterations. *Top:* MLI scheme for $\gamma_{\mathrm{B,c}} = \gamma_{\mathrm{B,l}} = 10^{-1}$. *Bottom:* Histogram of level iterations used in the scheme.

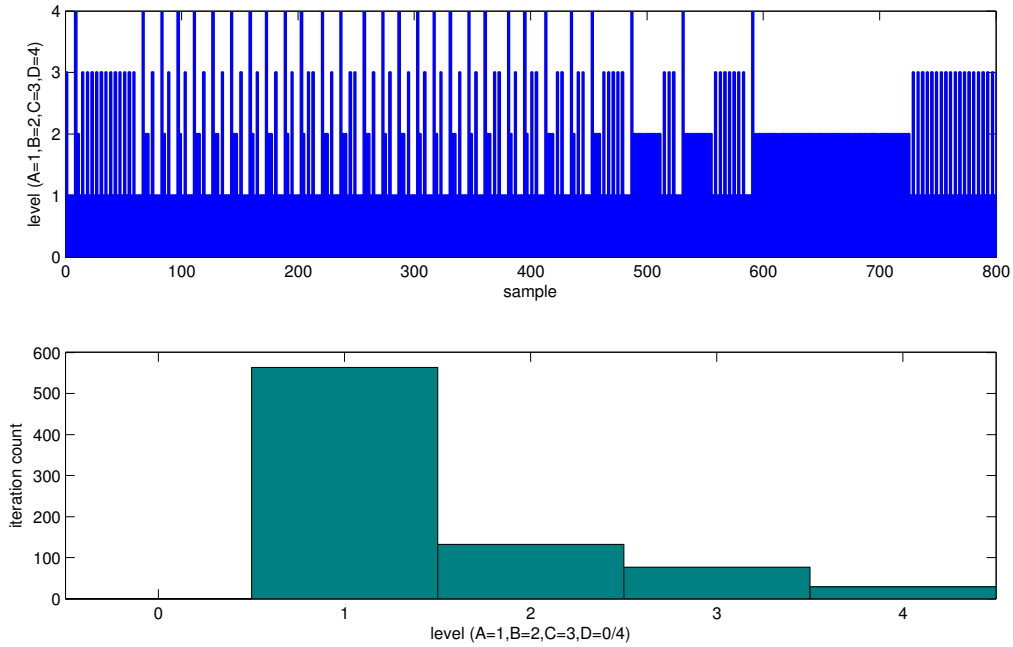


Figure 10.28: Adaptive level choice by postiterations. *Top:* MLI scheme for $\gamma_{\mathrm{B,c}} = \gamma_{\mathrm{B,l}} = 10^{-3}$. *Bottom:* Histogram of level iterations used in the scheme.

the constraint vector norm is smaller than $\gamma_{\mathrm{B,c}}$, while the scheme for $\gamma_{\mathrm{B,c}} = \gamma_{\mathrm{B,l}} = 10^{-3}$ first schedules level-B iterations, and level-C iterations after the constraint vector norm becomes smaller than 10^{-3}. In summary, both schemes schedule level-D iterations in

the critical part of the dynamics, and computationally cheaper iterations as soon as they become feasible.

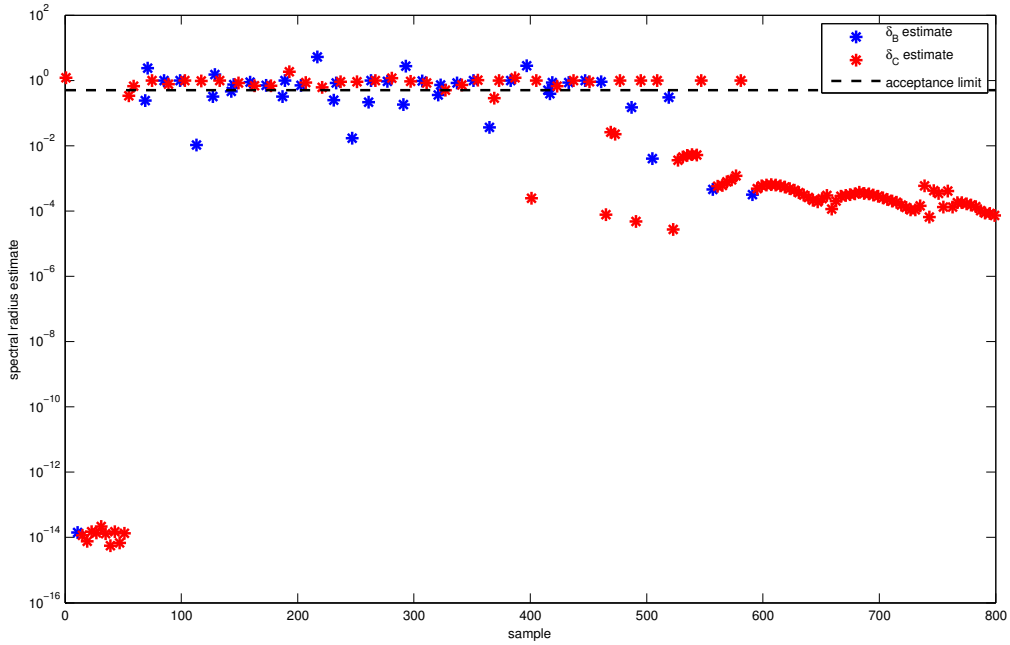


Figure 10.29: Adaptive level choice by postiterations for $\gamma_{\mathrm{B,c}} = \gamma_{\mathrm{B,l}} = 10^{-1}$. Contractivity estimates.

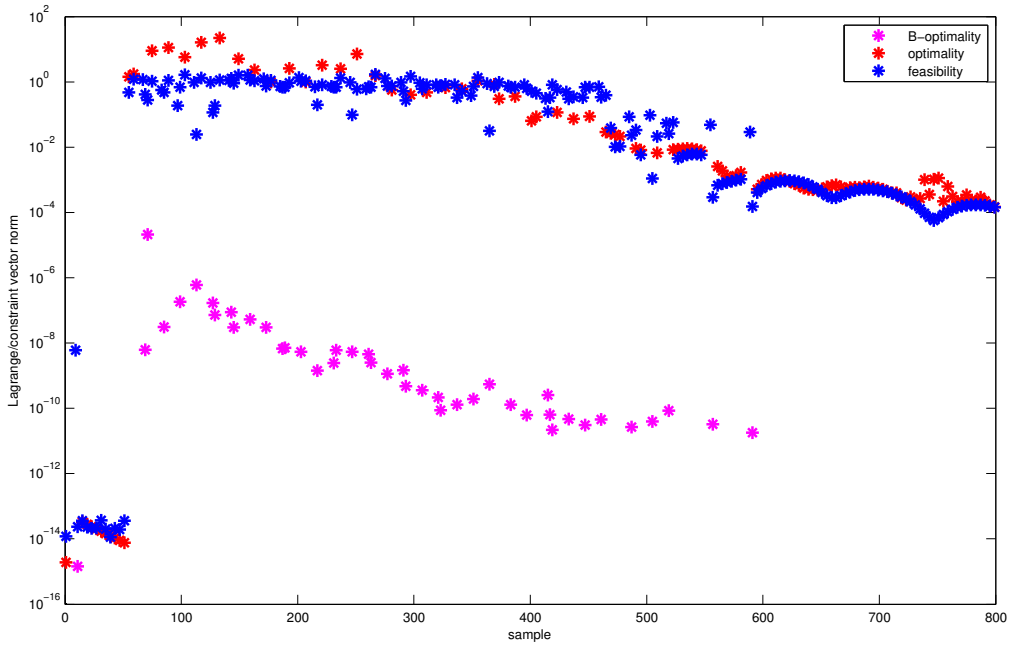


Figure 10.30: Adaptive level choice by postiterations for $\gamma_{\mathrm{B,c}} = \gamma_{\mathrm{B,l}} = 10^{-1}$. Constraint and gradient vector norms.

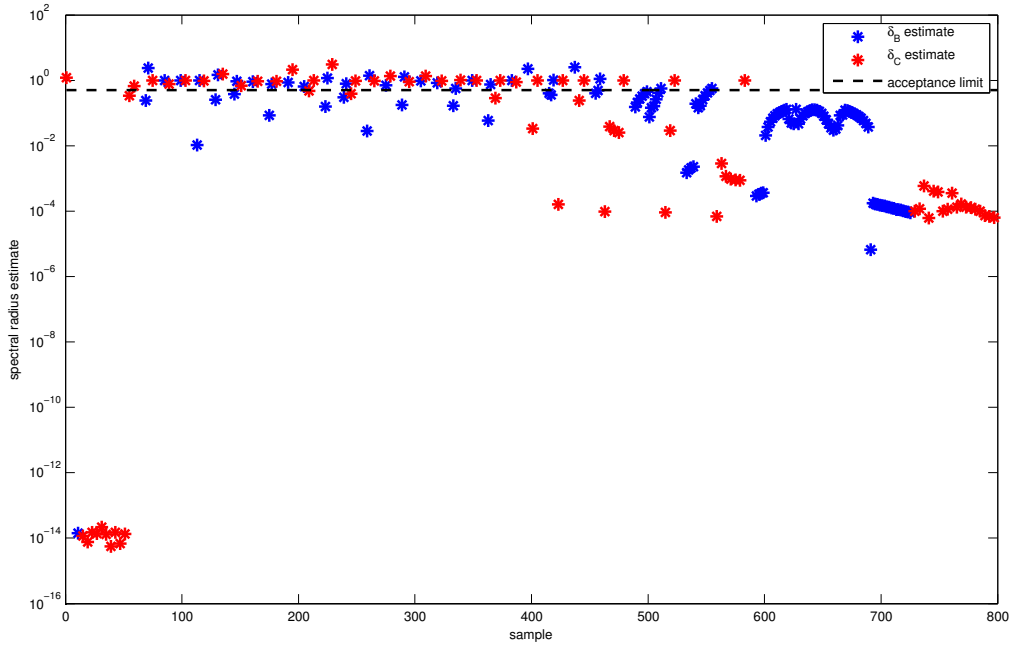


Figure 10.31: Adaptive level choice by postiterations for $\gamma_{\mathrm{B,c}} = \gamma_{\mathrm{B,l}} = 10^{-3}$. Contractivity estimates.

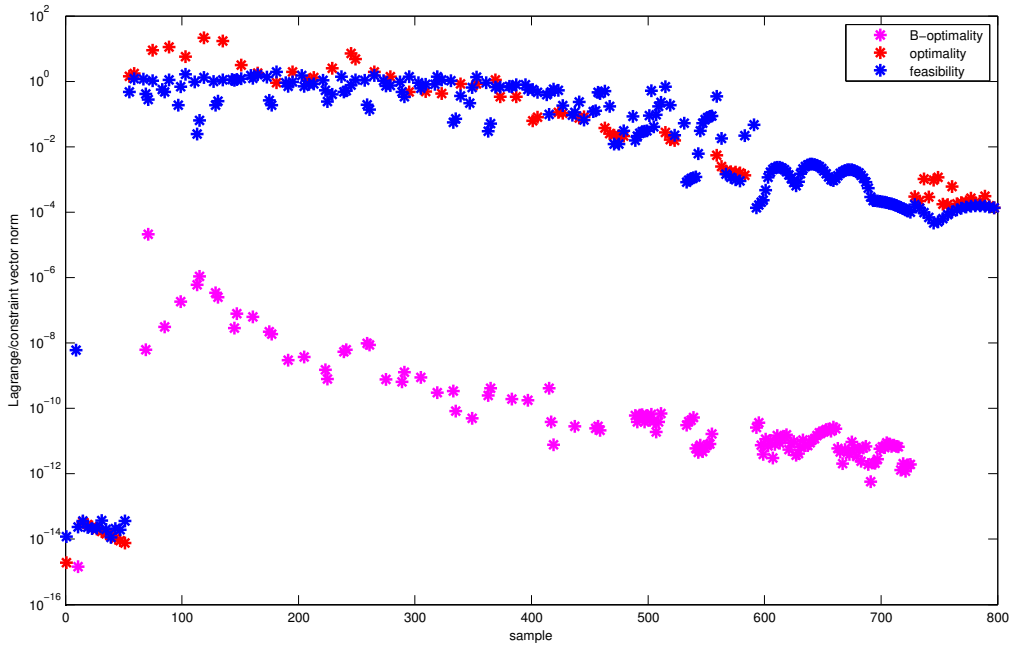


Figure 10.32: Adaptive level choice by postiterations for $\gamma_{\mathrm{B,c}} = \gamma_{\mathrm{B,l}} = 10^{-3}$. Constraint and gradient vector norms.

In Figures 10.33 and 10.34 we give the control profiles and the most important state trajectories for the discussed schemes. Apart from small changes in the structure and magnitude of the breaking force F_{B} and the pedal position ϕ, the control profiles and

state trajectories are quite similar. Compared to the RTI solution in Figure 10.26 the state trajectories generated by the MLI schemes show smaller deviations from the target values than those generated by RTI. This can also be seen from the performance indices $I_{\text{perf}} = \int_0^8 L(x,u)\mathrm{d}t$ for RTI and the MLI schemes given in Table 10.3, which shows clearly the benefit of giving feedback faster for the test case at hand.

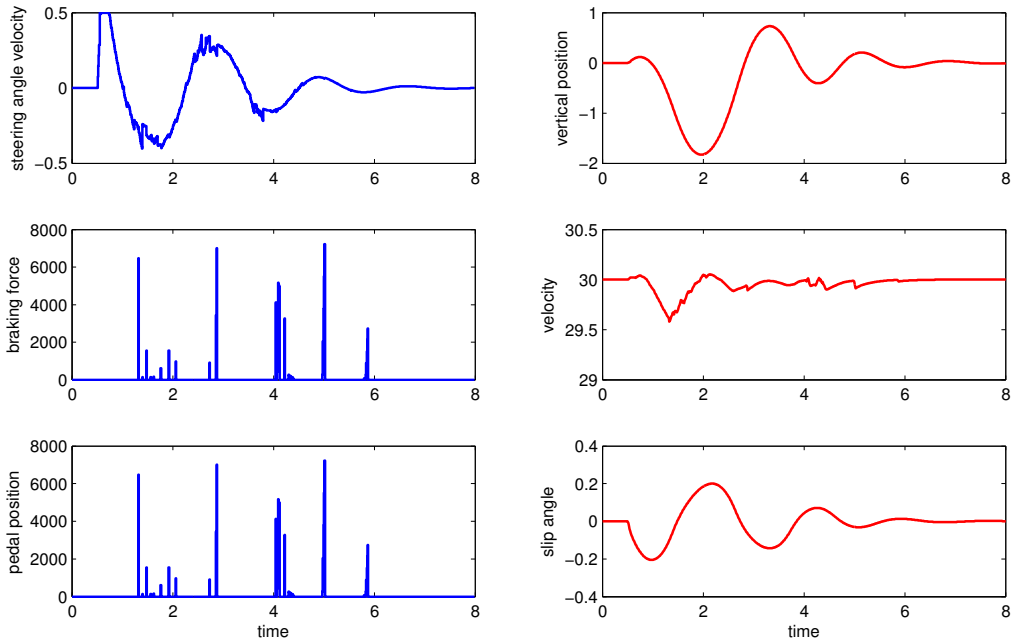


Figure 10.33: MLI solution by postiterations for $\gamma_{\text{B,c}} = \gamma_{\text{B,l}} = 10^{-1}$ and for sampling $\delta = 0.01$. State and control profiles.

Table 10.3: Performance indices $I_{\text{perf}} = \int_0^8 L(x,u)\mathrm{d}t$ for adaptive level choice by postiterations and for sampling $\delta = 0.01$. Performance index for RTI is given for comparison. The faster sampling of the adaptive MLI schemes yield a considerably improved controller performance.

Scheme	$\gamma_{\text{B,c}}$	$\gamma_{\text{B,l}}$	λ_{C}	I_{perf}
RTI ($\delta = 0.05$)	n/a	n/a	n/a	9.2057
Postiterations 1	10^{-2}	10^{-2}	10	3.5544
Postiterations 2	10^{-3}	10^{-3}	10	3.5305
Postiterations 3	10^{-4}	10^{-4}	10	3.5306

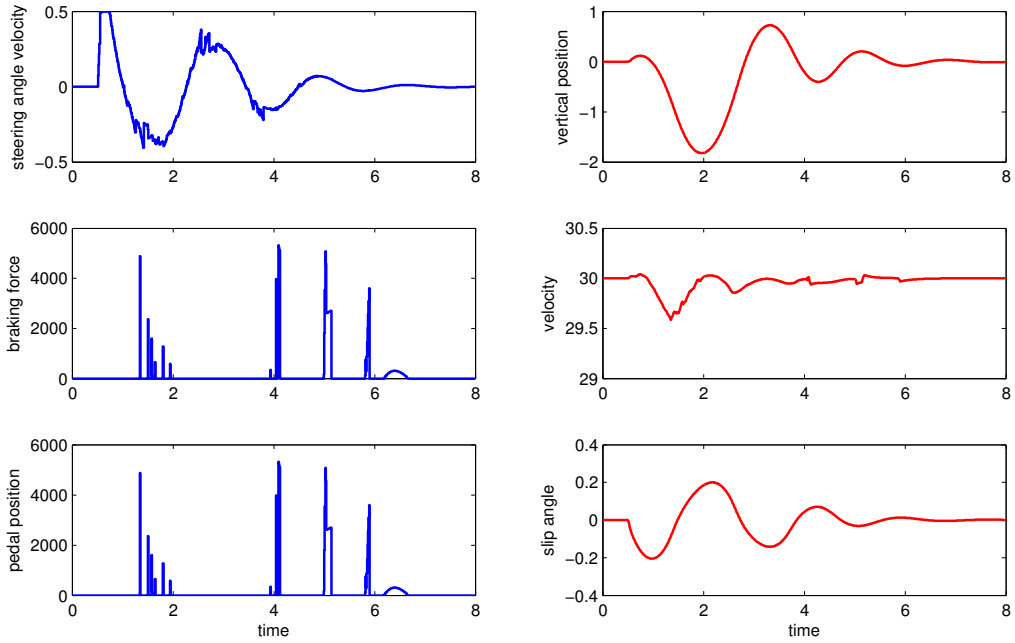


Figure 10.34: MLI solution by postiterations for $\gamma_{\mathrm{B,c}} = \gamma_{\mathrm{B,l}} = 10^{-3}$ and for sampling $\delta = 0.01$. State and control profiles.

11 Applications: MLI with χ^2-test and MLI for Dual NMPC

In this chapter, we give some numerical results for using the χ^2-test as described in Chapter 6 as additional tool for MLI with adaptive level choice by postiterations, and for the application of MLI to solve the problems arising in the Dual NMPC approach outlined in Chapter 8. As test case, we use in both cases a Lotka-Volterra model for the dynamics of a predator-prey system [132, 178].

11.1 LOTKA: MLI with χ^2-test

We state the Lotka-Volterra dynamic model, describe the control scenario and discuss numerical results. It turns out that using the χ^2-criterion can improve the schemes and controller performance of MLI schemes with adaptive level choice in the case of a combined estimator-controller system.

11.1.1 ODE model

For the test case we consider the following Lotka-Volterra model for the dynamics of a predator-prey system

$$\dot{x}(t) = k_1\, x(t) - k_2\, x(t)\, y(t), \tag{11.1a}$$
$$\dot{y}(t) = -k_3\, y(t) + k_4\, x(t)\, y(t) - u(t)\, y(t), \tag{11.1b}$$

with parameter values $k_1 = k_2 = k_3 = k_4 = 1.0$. In this model, the state x describes the size of the prey population, and the state y describes the size of the predator population. We have augmented the classical Lotka-Volterra model [132, 178] by a control dependent term which models removal of predators at a controlled time-dependent rate u proportional to the current population size, i.e., u is essentially a hunting quota.

11.1.2 Control scenario

For a given reference control u_s, we can calculate steady-state values x_s and y_s. Goal of the optimal control is to retain the steady-state values for the both populations by choosing a suitable control $u(t)$. To this end, we define the objective function

$$L(x,u) = (x(t) - x_\mathrm{s})^2 + (y(t) - y_\mathrm{s})^2 + 0.1\,(u(t) - u_\mathrm{s})^2\,. \tag{11.2}$$

In the test case, we choose $u_\mathrm{s} = 0.1$, yielding with the parameter values given above the steady-state values $x_\mathrm{s} = 1.1$ and $y_\mathrm{s} = 1.0$.

The real-time scenario runs for 100 time units and consists of a jump of k_1 to the value 1.3 at time $t = 10$. The sampling period is 1 time units. Only x can be measured, with a measurement covariance of 10^{-5} following the definitions in Chapter 3, i.e., the values of x, y, and k_1 have to be obtained by online estimation.

11.1.3 Numerical results

For the online estimation, we use MHE as described in Chapter 3, with an estimation horizon of length 4, divided into 4 shooting intervals. The estimator uses the growing horizon initialization, starting from the values x_s, y_s and $k_1 = 1.0$. The initial covariance of states and parameter are chosen as $10^{-4} \cdot \mathbb{I}_3$, and the tuning matrix Q_{T-N+1} for the arrival cost term is chosen as $10^{-6} \cdot \mathbb{I}_3$.

For the controller, we choose a prediction horizon of length 10, divided into 10 shooting intervals. We use an exact Hessian approach, and choose for the adaptive level choice by postiterations the parameters $\delta_{B/C} = 0.5$ for the acceptable contraction, $\gamma_{B,c} = \gamma_{B,l} = 10^{-3}$ and $\lambda_C = 10$. Although the sampling is not time-critical, we choose for comparison two configurations for the numbers (n_B, n_C, n_D), namely $(1, 1, 1)$ and $(1, 2, 4)$.

We compare MLI with adaptive level choice by postiterations for the configurations described above, once without using the χ^2-criterion, and once using the χ^2-criterion by forcing a level-D iteration whenever the test fails.

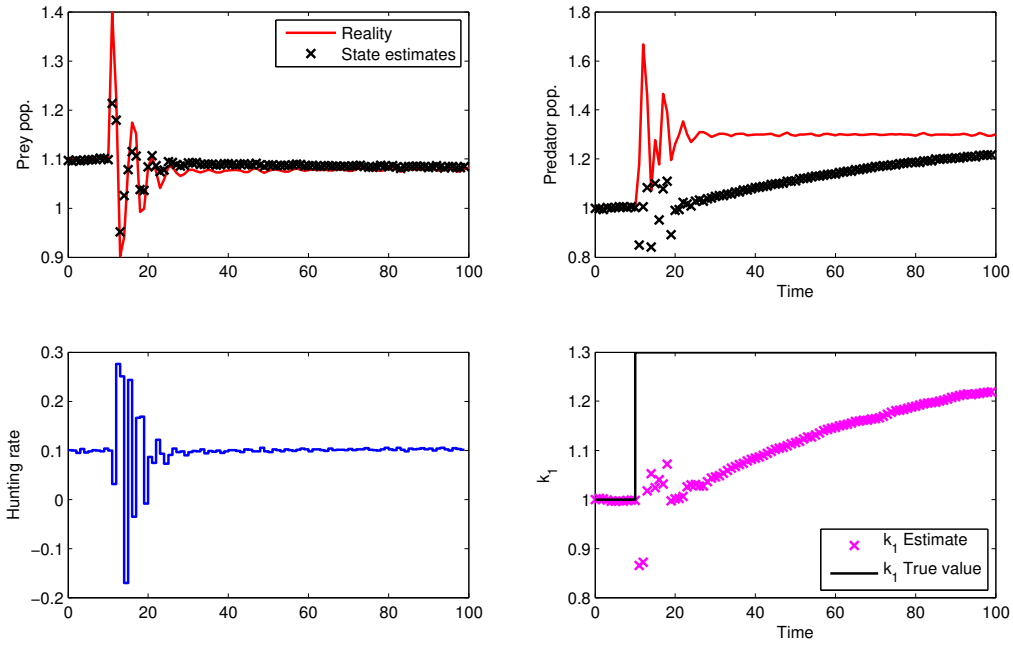
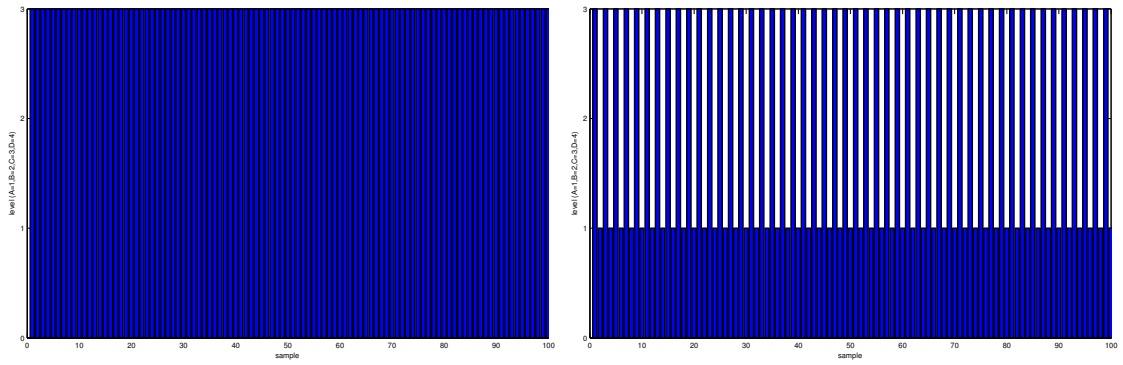


Figure 11.1: RTI reference solution. The controller tries to keep the states close to the steady-state values. Since x is fitted quite well, the estimator lacks information for a fast identification of k_1.

In Figure 11.1, the solution of the RTI scheme is given. While x is estimated quite well, the estimator-controller system only slowly approaches the true values of y and k_1. This is due to the fact that the controller tries to keep the states close to the steady-state values,

while the good match of the steady-state value of x also provides insufficient information for a quicker identification of k_1.



(a) Scheme for $(n_B, n_C, n_D) = (1, 1, 1)$ (b) Scheme for $(n_B, n_C, n_D) = (1, 2, 4)$

Figure 11.2: Schemes for MLI with adaptive choice by postiterations without χ^2-criterion. Both schemes are essentially pure level-C schemes.

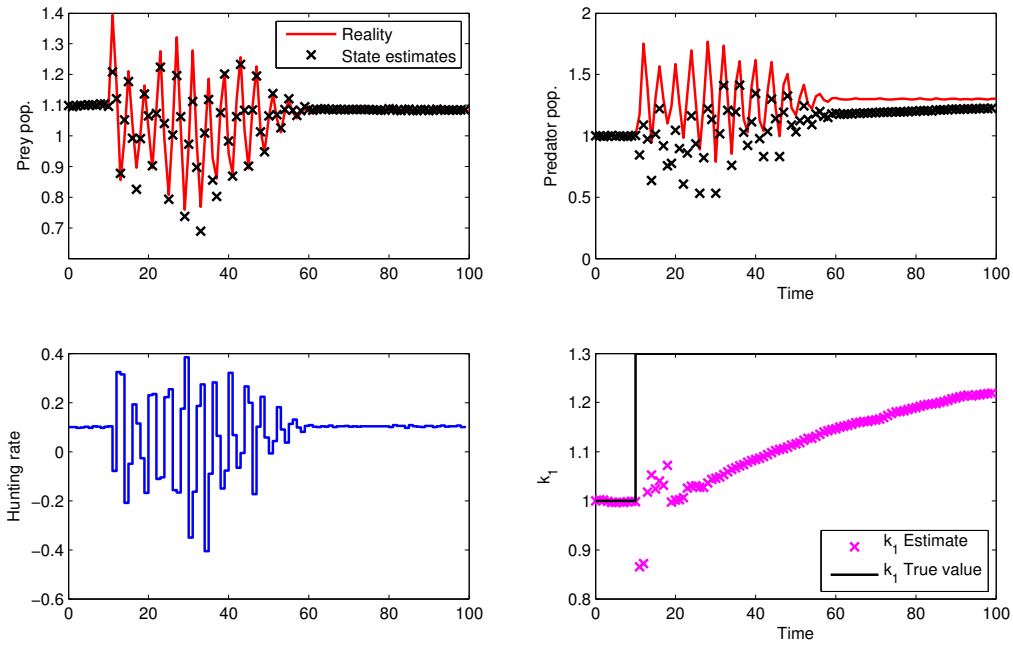
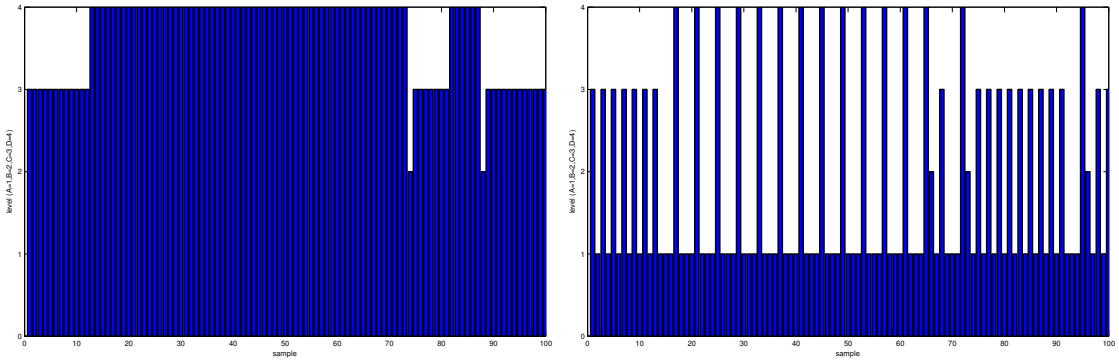


Figure 11.3: MLI solution for $(n_B, n_C, n_D) = (1, 2, 4)$ without χ^2-criterion. The estimator-controller system needs more than half the samples to approach RTI trajectories.

In Figure 11.2, the MLI schemes generated by postiterations without using the χ^2-criterion are shown. Both schemes are (essentially) pure level-C schemes, which means that contraction and optimality was satisfactorily in each sample. While the scheme for $(n_B, n_C, n_D) = (1, 1, 1)$ shows similar performance than the RTI scheme, the performance for $(n_B, n_C, n_D) = (1, 2, 4)$ is rather poor, as can be also seen from the state trajectories, the control profile, and the estimated parameter depicted in Figure 11.3. For more than

half of the samples, the state trajectories and estimates show an erratic and oscillating behavior caused by the feedback control profiles. This hints at insufficiently accurate linearization approximations due to the parameter jump, and it is the aim of the χ^2-criterion to fix this behavior.



(a) Scheme for $(n_\mathrm{B}, n_\mathrm{C}, n_\mathrm{D}) = (1, 1, 1)$ (b) Scheme for $(n_\mathrm{B}, n_\mathrm{C}, n_\mathrm{D}) = (1, 2, 4)$

Figure 11.4: Schemes for MLI with adaptive choice by postiterations with χ^2-criterion. All level-D iterations are triggered by χ^2-criterion. Still both schemes are computationally significantly less expensive than full RTI.

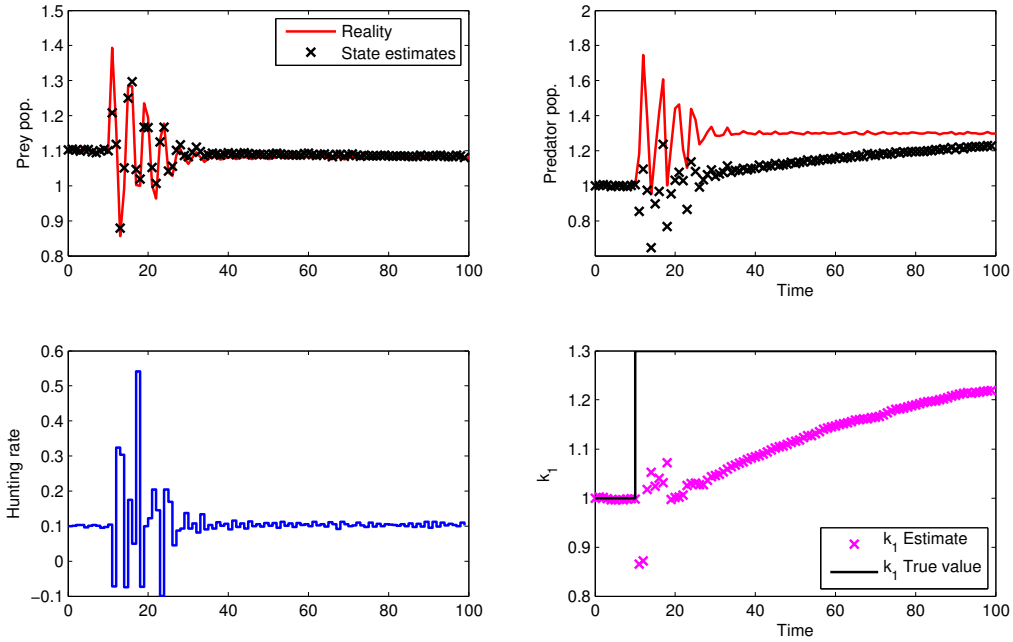


Figure 11.5: MLI solution for $(n_\mathrm{B}, n_\mathrm{C}, n_\mathrm{D}) = (1, 2, 4)$ with χ^2-criterion. The estimator-controller system performs almost as well as RTI.

In Figure 11.4, the MLI schemes generated by postiterations with using the χ^2-criterion are shown. All level-D iterations were triggered by the χ^2-criterion. Changes from level-B to level-C iterations were scheduled due to transgression of the thresholds $\gamma_{\mathrm{B,c}}$ and

$\gamma_{\mathrm{B,l}}$, contractivity was sufficient for all samples. Although the χ^2-criterion trigger many level-D iterations, the resulting schemes, in particular for $(n_{\mathrm{B}}, n_{\mathrm{C}}, n_{\mathrm{D}}) = (1, 2, 4)$, are computationally significantly cheaper than a full RTI scheme. For comparison to Figure 11.3, we show the state trajectories, the control profile, and the estimated parameter for $(n_{\mathrm{B}}, n_{\mathrm{C}}, n_{\mathrm{D}}) = (1, 2, 4)$ with using the χ^2-criterion in Figure 11.5. The scheme shows similarly good performance as the RTI scheme. The performance indices of the discussed schemes are also given in Table 11.1, which shows that both schemes using the χ^2-criterion are better than their counterparts, with a drastic improvement of the scheme for $(n_{\mathrm{B}}, n_{\mathrm{C}}, n_{\mathrm{D}}) = (1, 2, 4)$. This underlines the usefulness of the χ^2-criterion as a supporting tool for MLI with adaptive level choice in the case of a combined estimator-controller system.

Table 11.1: Performance indices $I_{\mathrm{perf}} = \int_0^{100} L(x, u)\mathrm{d}t$ for adaptive level choice by postiterations. Performance index for RTI is given for comparison.

Scheme	**with χ^2-crit.**	n_{B}	n_{c}	n_{D}	I_{perf}
RTI	n/a	n/a	n/a	n/a	8.6490
Postit. 1	no	1	1	1	8.8126
Postit. 2	no	1	2	4	11.3388
Postit. 3	yes	1	1	1	8.6931
Postit. 4	yes	1	2	4	9.0643

11.2 LOTKA: MLI for Dual NMPC

In this section, we consider the application of MLI with adaptive level choice for the problems arising from Dual NMPC as described in Chapter 8. We give the extended ODE model which includes covariance data to assess the contribution of the controls to the information gain specific for the computation of the objective function. We describe the control scenario and present and discuss numerical results. The focus here is on the computational issues rather than the interpretation of the results in the Dual NMPC context.

11.2.1 Extended ODE model

As outlined in Chapter 8, we can quantify and incorporate the influence of the controls to the information gain by suitably augmenting the dynamic system. We start by including the objective function into the dynamical system

$$\dot{x}(t) = k_1\, x(t) - k_2\, x(t)\, y(t), \tag{11.3a}$$

$$\dot{y}(t) = -k_3\, y(t) + k_4\, x(t)\, y(t) - u(t)\, y(t), \tag{11.3b}$$

$$\dot{m}(t) = q_1\, (x(t) - x_{\mathrm{s}})^2 + r_1\, (u(t) - u_{\mathrm{s}})^2\,, \tag{11.3c}$$

with $q_1 = 1.0$ and $r_1 = 10^{-2}$, where we consider here a quadratic objective function only in x and u, in contrast to the example considered in the last section. In this scenario, all parameters $k_i, i = 1, \ldots, 4$ are fixed to the value 1.0. With the shorthand $z \triangleq (x, y, m)$ for the states, and denoting the right-hand-side of (11.3) as $f(z)$, we can set up the dynamical equations for the covariance matrix as

$$\dot{C}(t) = \frac{\mathrm{d}f}{\mathrm{d}z}(t)\, C(t) + C(t)\, \frac{\mathrm{d}f}{\mathrm{d}z}(t)^\intercal - C(t)\, \frac{\mathrm{d}h}{\mathrm{d}z}(t)^\intercal\, R^{-1}\, \frac{\mathrm{d}h}{\mathrm{d}z}(t)\, C(t). \tag{11.3d}$$

Since C is symmetric, we only have to propagate the elements on and above the main diagonal.

11.2.2 Control scenario

We choose again $u_\mathrm{s} = 0.1$, yielding with the parameter values given above the steady-state values $x_\mathrm{s} = 1.1$ and $y_\mathrm{s} = 1.0$. As mentioned above, the control goal for this test case is to track the prey population, while regularizing the control. The real-time scenario runs for 100 time units, and the sampling period is 1 time unit.

Following (8.8), the Dual NMPC objective function is

$$\begin{aligned} E(z(T), C(T)) &= m(T) + \gamma \sqrt{(\nabla_z m(T))^\intercal\, C(T) \nabla_z m(T)} \\ &= m(T) + \gamma \sqrt{C_{3,3}(T)} \end{aligned} \tag{11.4}$$

The disturbance in this scenario is an immigration of prey at each sampling time between 10 and 19 (including) into the area of consideration, at an absolute value of 0.3 per sample.

Only x can be measured, with a measurement covariance of $R = 10^{-5}$ following the definitions in Chapter 3, i.e., the values of x and y have to be obtained by online estimation. That means the measurement function is $h(z) = x$, which yields $\frac{\mathrm{d}h}{\mathrm{d}z} = (1, 0, 0)$ in (11.3d).

11.2.3 Numerical results

For the online estimation, we use MHE as described in Chapter 3, with an estimation horizon of length 4, divided into 4 shooting intervals. The estimator uses the growing horizon initialization, starting from the values x_s and y_s. The initial covariance of states and parameter are chosen as $10^{-4} \cdot \mathbb{I}_2$, and the tuning matrix Q_{T-N+1} for the arrival cost term is chosen as $10^{-6} \cdot \mathbb{I}_2$.

For the controller, we choose a prediction horizon of length 10, divided into 10 shooting intervals. We use an exact Hessian approach, and choose for the adaptive level choice by postiterations the parameters $\delta_{\mathrm{B/C}} = 0.5$ for the acceptable contraction, $\gamma_{\mathrm{B,c}} = \gamma_{\mathrm{B,l}} = 10^{-3}$ and $\lambda_\mathrm{C} = 10$. Although the sampling is not time-critical, we choose for comparison again two configurations for the numbers $(n_\mathrm{B}, n_\mathrm{C}, n_\mathrm{D})$, namely $(1, 1, 1)$ and $(1, 2, 4)$.

In Figure 11.6 we give the RTI solution. The controller manages the changes caused by the immigration of the prey well and recovers the desired steady-state.

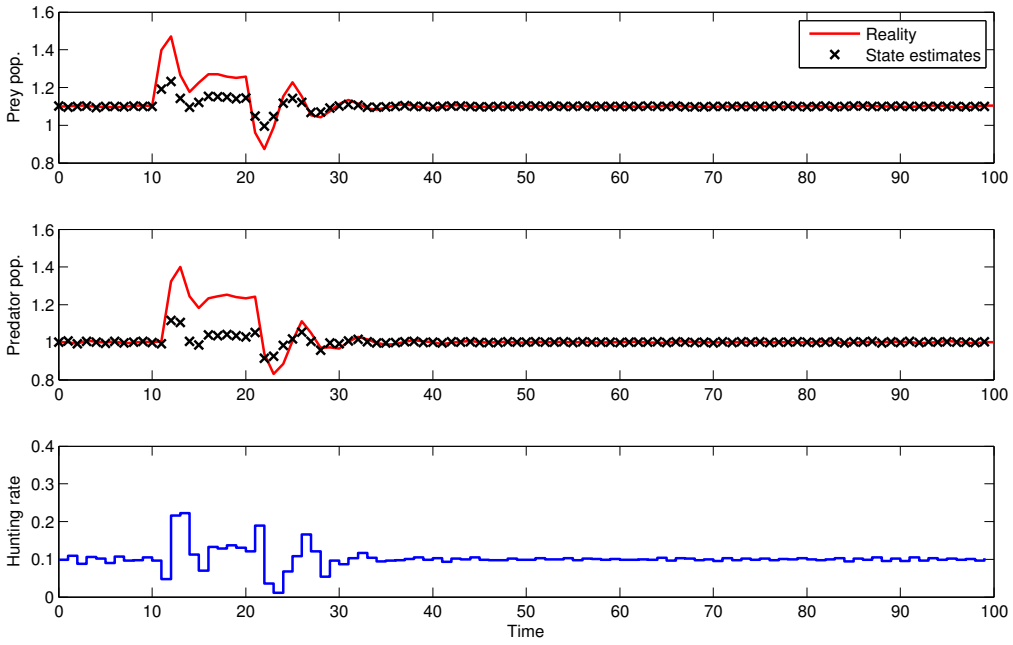
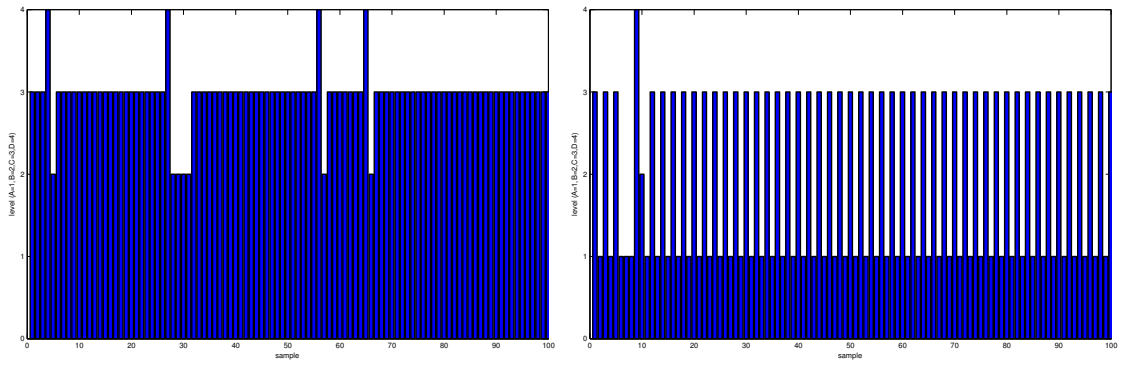


Figure 11.6: RTI reference solution. State trajectories and feedback control profiles.



(a) Scheme for $(n_{\mathrm{B}}, n_{\mathrm{C}}, n_{\mathrm{D}}) = (1, 1, 1)$

(b) Scheme for $(n_{\mathrm{B}}, n_{\mathrm{C}}, n_{\mathrm{D}}) = (1, 2, 4)$

Figure 11.7: Schemes for MLI with adaptive choice by postiterations. Level-D iterations are triggered by insufficient contraction. Transitions from level-B to level-C are due to transgression of thresholds $\gamma_{\mathrm{B,c}}$ and $\gamma_{\mathrm{B,l}}$. Overall, both schemes consist dominantly of level-C iterations.

The application of MLI with a fixed scheme consisting of level-C iterations fails due to the controller iterates deteriorating to a point where function and derivative evaluation cannot be performed anymore. The application of MLI with adaptive level choice by postiterations, however, generates well performing schemes which consist dominantly of level-C iterations, cf. Figure 11.7. The few level-D iterations are triggered by the detection of insufficient contraction according to the given thresholds $\delta_{\mathrm{B/C}} = 0.5$. The short part of level-B iterations in the scheme for $(n_{\mathrm{B}}, n_{\mathrm{C}}, n_{\mathrm{D}}) = (1, 1, 1)$ is ended by the constraint norm and approximated Lagrange vector norm falling below the thresholds $\gamma_{\mathrm{B,c}}$ and $\gamma_{\mathrm{B,l}}$.

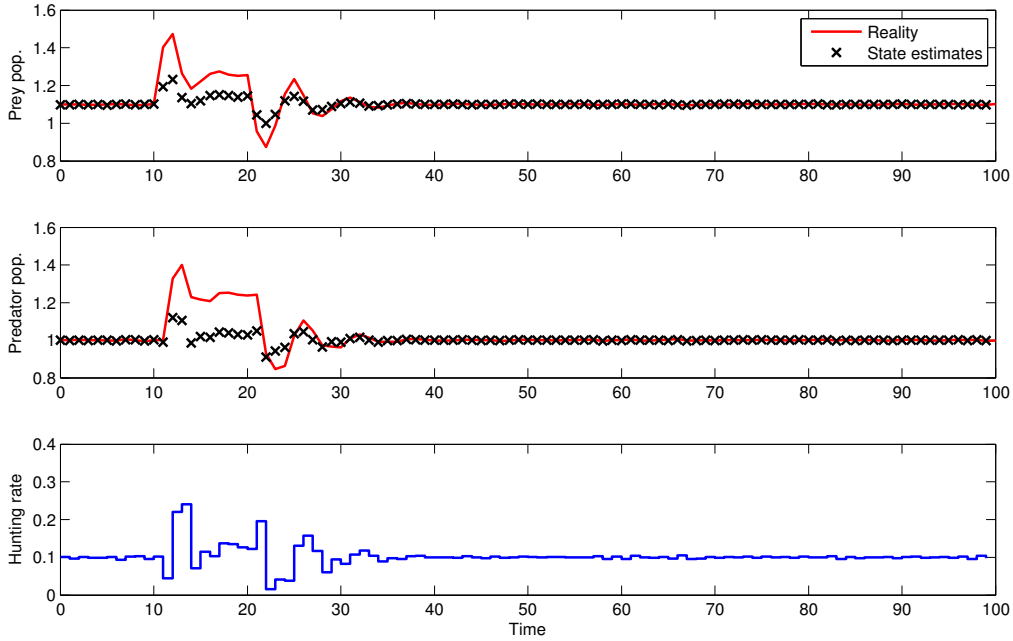


Figure 11.8: MLI solution for $(n_\mathrm{B}, n_\mathrm{C}, n_\mathrm{D}) = (1, 2, 4)$. The estimator-controller system performs almost as well as RTI.

The performances of the MLI schemes with adaptive level choice and the average CPU time per sample are also given in Table 11.2, with the numbers for the RTI scheme given for comparison. While the performance loss of the MLI schemes is less than 1%, the MLI scheme for $(n_\mathrm{B}, n_\mathrm{C}, n_\mathrm{D}) = (1, 1, 1)$ is in average computationally less expensive by a factor of about five, for $(n_\mathrm{B}, n_\mathrm{C}, n_\mathrm{D}) = (1, 2, 4)$ even by a factor of 10. This shows that the application of MLI with adaptive level choice is a promising approach to reduce the computational burden of the problems arising in Dual NMPC.

Table 11.2: Performance indices $I_\mathrm{perf} = m(100)$ for adaptive level choice by postiterations. Performance index for RTI is given for comparison.

Scheme	n_B	n_c	n_D	I_perf	**∅ CPU/sample**
RTI	n/a	n/a	n/a	0.2688	146.0 ms
Postit. 1	1	1	1	0.2712	29.7 ms
Postit. 2	1	2	4	0.2724	14.6 ms

12 Conclusions and future work

In this thesis, we developed Multi-Level Iteration schemes, an algorithmic approach for fast and efficient numerics for Nonlinear Model Predictive Control. The Multi-Level Iteration schemes are based on parametric optimization and generalized tangential predictors, and have been developed and tested in a Direct Multiple Shooting framework for the discretization of optimal control problems. Four levels form a hierarchy for updating the vector and matrix data in the feedback controller which requires the evaluation of the model functions and their derivatives. Multi-Level Iteration schemes then schedule the various update levels in a fixed or adaptive fashion to successively generate feedback laws in the form of generalized tangential predictors. These feedback laws are then evaluated efficiently to obtain feedback for the process.

In this thesis, we described the four levels, discussed the data communication between the levels within a feedback scheme, presented convergence theory for the levels with fixed initial state, and developed an algorithm for the efficient adaptive assembly of the Multi-Level Iteration schemes during runtime. We discussed details for an efficient numerical implementation. We successfully applied Multi-Level Iteration schemes with fixed level choice and with adaptive level choice to various nonlinear test cases and compared the results to the state-of-the-art approach of Real-Time Iterations.

In the course of the research done for this thesis, many interesting questions pointing to further research emerged. We will briefly comment on the most pressing topics.

Further investigation of level-A As discussed, level-A iterations are highly suited to provide quick feedback and even give rise to a local feedback law. In this work, they are used to provide the feedback while the higher levels compute the controller data updates. The methods for level selection developed in this thesis can, however, not be applied to level-A iterations since they work on the primal-dual iterates owned by the higher levels, and level-A iterations do not use such iterates. Still, it would be highly interesting to have a criterion which indicates if the current level-A controller is at least locally suitable as a feedback controller for the nonlinear process. Essentially, one would like to answer the question of the quality of the linearized model and constraints as approximation of their nonlinear counterparts. This is a question which would be, e.g., also important in the field of LMPC switching between several linearized models. Furthermore, the criterion would have to be evaluated in a time comparable to the time used by the level-A iteration itself, or at least clearly within one sampling period.

Convergence theory for level-B It is not strictly necessary for a good practical performance of a numerical feedback scheme to have convergence or stability theory, e.g.,

for the rejection of a disturbance in regulatory NMPC. However, it is desirable, reassuring, and an interesting research topic in itself. As discussed in this thesis, the convergence theory of RTI applies for schemes consisting of level-C and/or level-D iterations and suitable data communication schemes. There is so far no stability or convergence theory for the general case of schemes that also make use of level-B iterations. However, it seems that in the case of schemes consisting of level-B and higher level iterations, theory similar to the theory for RTI is likely to hold, provided the level-B iterations do not drive the primal-dual iterates out of the region of local convergence. Even in the case of pure level-B iterations, it seems possible to formulate and prove a similar theory considering the well known result that stability essentially requires only feasibility and a sufficient decrease of the objective function in each sample.

Parallelization and distributed computing The multi-level iterations approach is highly suited for parallelization in several ways. The computations for each level can and should be performed in parallel, with a thread for the computation of the feedback control and additional threads for the computation of the data update in the higher levels. In the numerical results for the adaptive level choice in this thesis, this parallelization is simulated, but this is obviously only a surrogate for a real parallel implementation, which would have to properly and efficiently distribute the computations and ensure real-time communication to bring the computed data in due time to the feedback controller. Furthermore, the calculations within each level are particularly suited for parallelization, since the Multiple Shooting discretization introduces a natural decoupling in the function and derivative evaluation. Beside load balancing, there is the interesting question of the possibility to intertwine the different levels of parallelization in the sense of the mixed and fractional level iterations described in this thesis, i.e., to update the controller asynchronous with parts of the data computed in parallel.

Embedded control In embedded control, optimization algorithms have to be adapted to or developed for resource-constrained platforms such as micro-controllers. The Multi-Level Iteration approach opens the possibility to efficiently implement nonlinear feedback control even for computing platforms that are significantly less powerful than a desktop computer. Therefore, tailoring the Multi-Level Iteration approach to specific hardware, implementing it in a suitably fast and resource-efficient way, and testing it in real-world applications is a highly interesting task. Probably, this research direction would also have overlappings with the research on parallelization.

Application to challenging non-standard numerical problems In this thesis, we have demonstrated numerically that the Multi-Level Iteration approach is in principle useful and efficient in the numerical treatment of the nonstandard and possibly large-scale problems arising from Dual NMPC. It would be interesting to perform an extensive testing of the Multi-Level Iterations to the problems arising from various approaches in Robust and Dual NMPC, also with regard to the statistical interpretation of the

results, and investigate and possibly quantify the effect of using approximations in particular of the derivatives, which often have interpretations as statistical quantities such as covariance matrices, on the quality of the feedback control under uncertainties and the online estimation. Furthermore, it would be interesting to test and apply the Multi-Level Iterations for the feedback control of processes with discrete decisions, in particular in combination with the outer convexification approach, which yields continuous problems with significantly larger numbers of constraints and decision variables compared to the original mixed-integer problems.

List of Figures

Bibliography

[1] J. Albersmeyer. Effiziente Ableitungserzeugung in einem adaptiven BDF-Verfahren. Diploma thesis, Universität Heidelberg, 2005.

[2] J. Albersmeyer. *Adjoint based algorithms and numerical methods for sensitivity generation and optimization of large scale dynamic systems*. PhD thesis, Ruprecht–Karls–Universität Heidelberg, 2010.

[3] J. Albersmeyer, D. Beigel, C. Kirches, L. Wirsching, H.G. Bock, and J.P. Schlöder. Fast nonlinear model predictive control with an application in automotive engineering. In L. Magni, D.M. Raimondo, and F. Allgöwer, editors, *Lecture Notes in Control and Information Sciences*, volume 384, pages 471–480. Springer Verlag Berlin Heidelberg, 2009.

[4] J. Albersmeyer and H.G. Bock. Sensitivity Generation in an Adaptive BDF-Method. In Hans Georg Bock, E. Kostina, X.H. Phu, and R. Rannacher, editors, *Modeling, Simulation and Optimization of Complex Processes: Proceedings of the International Conference on High Performance Scientific Computing, March 6–10, 2006, Hanoi, Vietnam*, pages 15–24. Springer Verlag Berlin Heidelberg New York, 2008.

[5] J. Albersmeyer and H.G. Bock. Efficient sensitivity generation for large scale dynamic systems. Technical report, SPP 1253 Preprints, University of Erlangen, 2009.

[6] J. Albersmeyer and C. Kirches. The `SolvIND` webpage. http://www.solvind.org, 2007.

[7] J. Andersson. *A general-purpose software framework for dynamic optimization*. PhD thesis, KU Leuven, Belgium, 2013.

[8] D.F. Andrews. Sequentially designed experiments for screening out bad models with F-Tests. *Biometrika*, 58:427–432, 1971.

[9] K.H. Ang, G.C.Y. Chong, and Y. Li. Pid control system analysis, design, and technology. *IEEE Transactions on Control Systems Technology*, 13(4):559–576, 2005.

[10] A.C. Atkinson. A method for discriminating between models. *Journal of the Royal Statistical Society, Series B (Methodological)*, 32(3):323–353, 1970.

[11] A.C. Atkinson and D. Cox. Planning experiments for discriminating between models. *Journal of the Royal Statistical Society*, B36:321–348, 1974.

[12] Y. Bard. *Nonlinear Parameter Estimation.* Academic Press, 1974.

[13] J.V. Beck and K.J. Arnold. *Parameter estimation in Engineering and Science.* Wiley, New York, 1977.

[14] R.E. Bellman. *Dynamic Programming.* University Press, Princeton, N.J., 6th edition, 1957.

[15] R.E. Bellman. Dynamic programming and a new formalism in the calculus of variations. *Proceedings of the national academy of sciences*, 40(4):231–235, 1954.

[16] R.E. Bellman and S.E. Dreyfus. *Applied dynamic programming.* Princeton university press, 2015.

[17] A. Ben-Tal and A. Nemirovski. *Lectures on Modern Convex Optimization: Analysis, Algorithms, and Engineering Applications.* MPS-SIAM Series on Optimization. MPS-SIAM, Philadelphia, 2001.

[18] D.P. Bertsekas. *Dynamic Programming and Optimal Control*, volume 1 and 2. Athena Scientific, Belmont, MA, 1995.

[19] D.P. Bertsekas. *Nonlinear Programming.* Athena Scientific, 2003.

[20] D.P. Bertsekas. *Dynamic programming and optimal control, Volume 1.* Athena Scientific, Belmont, Mass., 3rd edition, 2005.

[21] D.P. Bertsekas. *Dynamic programming and optimal control, Volume 2.* Athena Scientific, Belmont, Mass., 3rd edition, 2007.

[22] M.J. Best. An algorithm for the solution of the parametric quadratic programming problem. In H. Fischer, B. Riedmüller, and S. Schäffler, editors, *Applied Mathematics and Parallel Computing – Festschrift for Klaus Ritter*, chapter 3, pages 57–76. Physica-Verlag, Heidelberg, 1996.

[23] L.T. Biegler. Solution of dynamic optimization problems by successive quadratic programming and orthogonal collocation. *Computers & Chemical Engineering*, 8:243–248, 1984.

[24] L.T. Biegler. *Nonlinear Programming: Concepts, Algorithms, and Applications to Chemical Processes.* Series on Optimization. SIAM, 2010.

[25] L.T. Biegler, A.M. Cervantes, and A. Waechter. Advances in simultaneous strategies for dynamic process optimization. *Chemical Engineering Science*, 4(57):575–593, 2002.

[26] N. Boccara. *Modeling complex systems.* Springer Science & Business Media, 2010.

[27] H.G. Bock and E. Kostina. Robust experimental design. In H. G. Bock, editor, *Projekt A4, Optimization methods for reactive flows of Sonderforschungsbereich 359 Reactive Flows, Diffusion, and Transport, Report 1999-2001*, pages 133–135. University of Heidelberg, 2001.

[28] H.G. Bock. *Randwertproblemmethoden zur Parameteridentifizierung in Systemen nichtlinearer Differentialgleichungen*, volume 183 of *Bonner Mathematische Schriften*. Universität Bonn, Bonn, 1987.

[29] H.G. Bock, M. Diehl, E. Kostina, and J.P. Schlöder. Constrained Optimal Feedback Control for DAE. In L. Biegler, O. Ghattas, M. Heinkenschloss, D. Keyes, and B. van Bloemen Waanders, editors, *Real-Time PDE-Constrained Optimization*, chapter 1, pages 3–24. SIAM, 2007.

[30] H.G. Bock, M. Diehl, P. Kühl, E. Kostina, J.P. Schlöder, and L. Wirsching. Numerical methods for efficient and fast nonlinear model predictive control. In R. Findeisen, F. Allgöwer, and L. T. Biegler, editors, *Assessment and future directions of Nonlinear Model Predictive Control*, volume 358 of *Lecture Notes in Control and Information Sciences*, pages 163–179. Springer, 2005.

[31] H.G. Bock and R.W. Longman. Optimal control of velocity profiles for minimization of energy consumption in the new york subway system. In *Proceedings of the Second IFAC Workshop on Control Applications of Nonlinear Programming and Optimization*, pages 34–43. International Federation of Automatic Control, 1980.

[32] H.G. Bock and K.J. Plitt. A Multiple Shooting algorithm for direct solution of optimal control problems. In *Proceedings of the 9th IFAC World Congress*, pages 242–247, Budapest, 1984. Pergamon Press. Available at http://www.iwr.uni-heidelberg.de/groups/agbock/FILES/Bock1984.pdf.

[33] H.G. Bock and J.P. Schlöder. Fit, fitter, the fittest: Methods for modelling and validation of dynamical systems. In D.P.F. Möller, editor, *Advances in System Analysis*, volume 2. Vieweg, 1986.

[34] G.E.P. Box and H.L. Lucas. Design of experiments in non-linear situations. *Biometrika*, 46:77–90, 1959.

[35] M.J. Box. Planning experiments to test the adequacy of non-linear models. *Journal of the Royal Statistical Society*, 18(3):241–248, 1969.

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

[37] S. Boyd and L. Vandenberghe. Introduction to applied linear algebra. Vectors, matrices, and least squares. Available at https://web.stanford.edu/ boyd/vmls/vmls.pdf, 2017.

[38] A.E. Bryson and Y.-C. Ho. *Applied Optimal Control*. Wiley, New York, 1975.

[39] J.-B. Caillau, J. Gergaud, T. Haberkorn, P. Martinon, and J. Noailles. Numerical optimal control and orbital transfers. In *Proceedings of the Workshop Optimal Control, Sonderforschungsbreich 255: Transatmosphärische Flugsysteme, Heronymus Munchen, ISBN 3-8979-316-X*, pages 39–49, Greifswald, Germany, 2002.

[40] A.M. Cervantes and L.T. Biegler. Large-scale DAE optimization using a simultaneous NLP formulation. *AIChE Journal*, 44(5):1038–1050, 1998.

[41] A.M. Cervantes, A. Wächter, R.H. Tütüncü, and L.T. Biegler. A reduced space interior point strategy for optimization of differential algebraic systems. *Computers & Chemical Engineering*, 24(1):39–51, 2000.

[42] H. Chen. *Stability and Robustness Considerations in Nonlinear Model Predictive Control*. Fortschritt-Berichte VDI Reihe 8 Nr. 674. VDI Verlag, Düsseldorf, 1997.

[43] 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.

[44] D.W. Clarke, C. Mohtadi, and P.S. Tuffs. Generalized predictive control – Part i. The basic algorithm. *Automatica*, 23(2):137–148, 1987.

[45] D.W. Clarke, C. Mohtadi, and P.S. Tuffs. Generalized predictive control – Part ii. Extension and interpretations. *Automatica*, 23(2):149–160, 1987.

[46] United States. President's Information Technology Advisory Committee. *Computational Science: Ensuring America's Competitiveness*. National Coordination Office for Information Technology Research & Development, 2005.

[47] H. Cox. On the estimation of state variables and parameters for noisy dynamic systems. *IEEE Transactions on Automatic Control*, 9:5–12, 1964.

[48] R.K. Cox, J.F. Smith, and Y. Dimitratos. Can simulation technology enable a paradigm shift in process control? Modeling for the rest of us. *Computers & Chemical Engineering*, 30:1542–1552, 2006.

[49] C.R. Cutler and B.L. Ramakar. Dynamic matrix control – a computer control algorithm. In *Proceedings of Joint Automatic Control Conference*, San Francisco, 1980.

[50] J.E. Dennis and J.J. Moré. Quasi-Newton methods, motivation and theory. *SIAM Review*, 19(1):46–89, January 1977.

[51] M. Diehl. *Real-Time Optimization for Large Scale Nonlinear Processes*. PhD thesis, Universität Heidelberg, 2001.

[52] M. Diehl, H.G. Bock, and E. Kostina. An approximation technique for robust nonlinear optimization. *Mathematical Programming*, 107:213–230, 2006.

[53] 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.

[54] M. Diehl, H.G. Bock, and J.P. Schlöder. A real-time iteration scheme for nonlinear optimization in optimal feedback control. *SIAM Journal on Control and Optimization*, 43(5):1714–1736, 2005.

[55] M. Diehl, H.G. Bock, J.P. Schlöder, R. Findeisen, Z. Nagy, and F. Allgöwer. Real-time optimization and nonlinear model predictive control of processes governed by differential-algebraic equations. *Journal of Process Control*, 12(4):577–585, 2002.

[56] M. Diehl, H.J. Ferreau, and N. Haverbeke. Efficient numerical methods for nonlinear MPC and moving horizon estimation. In L. Magni, D.M. Raimondo, and F. Allgöwer, editors, *Nonlinear Model Predictive Control*, volume 384 of *Springer Lecture Notes in Control and Information Sciences*, pages 391–417. Springer-Verlag, Berlin, Heidelberg, New York, 2009.

[57] M. Diehl, R. Findeisen, and F. Allgöwer. A stabilizing real-time implementation of nonlinear model predictive control. In L. Biegler, O. Ghattas, M. Heinkenschloss, D. Keyes, and B. van Bloemen Waanders, editors, *Real-Time and Online PDE-Constrained Optimization*. SIAM, 2006.

[58] M. Diehl, R. Findeisen, F. Allgöwer, H.G. Bock, and J.P. Schlöder. Nominal stability of the real-time iteration scheme for nonlinear model predictive control. *IEE Proceedings - Control Theory and Applications*, 152(3):296–308, 2005.

[59] M. Diehl, P. Kuehl, H.G. Bock, and J.P. Schlöder. Schnelle Algorithmen für die Zustands- und Parameterschätzung auf bewegten Horizonten. *Automatisierungstechnik*, 54(12):602–613, 2006.

[60] M. Diehl, A. Walther, H.G. Bock, and E. Kostina. An adjoint-based SQP algorithm with quasi-Newton Jacobian updates for inequality constrained optimization. *Optimization Methods and Software*, 2009.

[61] C. Dym. *Principles of mathematical modeling*. Academic press, 2004.

[62] D. Edwards and M. Hamson. *Guide to mathematical modelling*. Industrial Press, 2007.

[63] H. Elmqvist, S.E. Mattsson, and M. Otter. Modelica: The new object-oriented modeling language. In *Proceedings of the 12th European Simulation Multiconference*, Manchester, UK, 1998.

[64] S. Engell. *Nichtlineare Regelung – Methoden, Werkzeuge, Anwendungen*. Number 1026 in Fortschritt-Berichte. VDI–Verlag, Düsseldorf, 1993.

[65] W.H. Enright, D.J. Higham, B. Owren, and W. Sharp. A survey of the explicit Runge–Kutta method. Technical Report 291/94, Department of Computer Science, University of Toronto, Canada, 1995.

[66] V.V. Fedorov. Optimal experimental design. *Wiley Interdisciplinary Reviews: Computational Statistics*, 2:581–589, 2010.

[67] E. Fehlberg. Klassische Runge-Kutta-Formeln fünfter und siebenter Ordnung mit Schrittweiten-Kontrolle. *Computing*, 4:93–106, 1969.

[68] A.A. Feldbaum. Dual control theory. i. *Avtomatika i Telemekhanika*, 21(9):1240–1249, 1960.

[69] A.A. Feldbaum. Dual control theory. ii. *Avtomatika i Telemekhanika*, 21(11):1453–1464, 1960.

[70] A.A. Feldbaum. Dual control theory. iv. *Avtomatika i Telemekhanika*, 22(2):129–142, 1961.

[71] A.A. Feldbaum. Theory of dual control. *Avtomatika i Telemekhanika*, 22(1):3–16, 1961.

[72] H.J. Ferreau. *Model Predictive Control Algorithms for Applications with Millisecond Timescales*. PhD thesis, K.U. Leuven, 2011.

[73] H.J. Ferreau, C. Kirches, A. Potschka, H.G. Bock, and M. Diehl. qpOASES: A parametric active-set algorithm for quadratic programming. *Mathematical Programming Computation*, 6(4):327–363, 2014.

[74] H.J. Ferreau. An online active set strategy for fast solution of parametric quadratic programs with applications to predictive engine control. Diploma thesis, Ruprecht–Karls–Universität Heidelberg, 2006.

[75] H.J. Ferreau, H.G. Bock, and M. Diehl. An online active set strategy to overcome the limitations of explicit MPC. *International Journal of Robust and Nonlinear Control*, 18(8):816–830, 2008.

[76] H.J. Ferreau, A. Potschka, and C. Kirches. qpOASES webpage. http://www.qpOASES.org/, 2007–2018.

[77] A.V. Fiacco. *Introduction to sensitivity and stability analysis in nonlinear programming*. Academic Press, New York, 1983.

[78] A.V. Fiacco and G.P. McCormick. Nonlinear programming: Sequential unconstrained minimization techniques. *SIAM publications*, 1990.

[79] R. Findeisen, F. Allgöwer, and L. Biegler, editors. *Assessment and Future Directions of Nonlinear Model Predictive Control*. Lecture Notes in Control and Information Sciences. Springer, 2006.

[80] O. Forster. *Analysis II*. Vieweg, Wiesbaden, 1996.

[81] A.R. Forsyth. *Theory of Differential Equations*. Dover, 1959.

[82] R. Fourer, D.M. Gay, and B.W. Kernighan. *AMPL: A Modeling Language for Mathematical Programming*. Duxbury Press, 2002.

[83] G. Franceschini and S. Macchietto. Model-based design of experiments for parameter precision: State of the art. *Chemical Engineering Science*, 63:4846–4872, 2008.

[84] G. François, B. Srinivasan, and D. Bonvin. Equivalence between neighboring-extremal control and self-optimizing control for the steady-state optimization of dynamical systems. *Industrial & Engineering Chemistry Research*, 53(18):7470–7478, 2014.

[85] J.V. Frasch, L. Wirsching, S. Sager, and H.G. Bock. Mixed-level iteration schemes for nonlinear model predictive control. In *Proceedings of the IFAC Conference on Nonlinear Model Predictive Control*, 2012.

[86] M. Gerdts. *Optimal Control of ODEs and DAEs*. De Gruyter, 2012.

[87] P.E. Gill, W. Murray, and M.H. Wright. *Practical optimization*. Academic Press, London, 1999.

[88] G.H. Golub and C.F. van Loan. *Matrix Computations*. Johns Hopkins University Press, Baltimore, 3rd edition, 1996.

[89] E. Griepentrog and R. März. *Differential–Algebraic Equations and Their Numerical Treatment*. BSB B.G. Teubner Verlagsgesellschaft, 1986.

[90] A. Griewank and A. Walther. *Evaluating Derivatives, Principles and Techniques of Algorithmic Differentiation*. SIAM, Philadelphia, 2nd edition, 2008.

[91] L. Grüne and J. Pannek. *Nonlinear Model Predictive Control: Theory and Algorithms*. Communications and Control Engineering. Springer International Publishing, 2016.

[92] L. Grüne. NMPC without terminal constraints. *IFAC Proceedings Volumes*, 45(17):1 – 13, 2012. 4th IFAC Conference on Nonlinear Model Predictive Control.

[93] S.P. Han. Superlinearly convergent variable-metric algorithms for general nonlinear programming problems. *Mathematical Programming*, 11:263–282, 1976.

[94] S.P. Han. A globally convergent method for nonlinear programming. *Journal of Optimization Theory and Applications*, 22:297–310, 1977.

[95] C.R. Hargraves and S.W. Paris. Direct trajectory optimization using nonlinear programming and collocation. *AIAA Journal of Guidance, Control, and Dynamics*, 10(4):338–342, 1987.

[96] C. Hoffmann. *Numerical aspects of uncertainty in the design of optimal experiments for model discrimination*. PhD thesis, Heidelberg, 2017.

[97] S.J. Julier and J.K. Uhlmann. Unscented filtering and nonlinear estimation. *Proceedings of the IEEE*, 92(3):401– 422, 2004.

[98] S.J. Julier, J.K. Uhlmann, and H.F. Durrant-Whyte. A new method for the nonlinear transformation of means and covariances in filters and estimators. *IEEE Transactions on Automatic Control*, 45:477–482, 2000.

[99] J.V. Kadam and W. Marquardt. Sensitivity-based solution updates in closed-loop dynamic optimization. *IFAC Proceedings Volumes*, 37(9):947–952, 2004.

[100] J. Kallrath. *Modeling Languages in Mathematical Optimization*. Springer Publishing Company, Incorporated, 2012.

[101] R.E. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82:35–45, 1960.

[102] W.G. Kelley and A.C. Peterson. *The Theory of Differentail Equations*. Springer, 2nd edition, 2010.

[103] C. Kirches. *Fast numerical methods for mixed-integer nonlinear model-predictive control*. Advances in Numerical Mathematics. Springer Vieweg, Wiesbaden, 2011.

[104] C. Kirches, H.G. Bock, J.P. Schlöder, and S. Sager. Block structured quadratic programming for the direct multiple shooting method for optimal control. *Optimization Methods and Software*, 26(2):239–257, April 2011.

[105] C. Kirches, S. Sager, H.G. Bock, and J.P. Schlöder. Time-optimal control of automobile test drives with gear shifts. *Optimal Control Applications and Methods*, 31(2):137–153, 2010.

[106] C. Kirches, L. Wirsching, H.G. Bock, and J.P. Schlöder. Efficient direct multiple shooting for nonlinear model predictive control on long horizons. *Journal of Process Control*, 22(3):540–550, 2012.

[107] C. Kirches, L. Wirsching, S. Sager, and H.G. Bock. Efficient numerics for nonlinear model predictive control. In M. Diehl, F. Glineur, E. Jarlebring, and W. Michiels, editors, *Recent Advances in Optimization and its Applications in Engineering*, pages 339–359. Springer, 2010.

[108] D. Klatte. First Order Constraint Qualifications. In C. Floudas, P. Pardalos, editors, *Encyclopedia of Optimization*, pages 1055–1060. Springer, Boston, MA, 2008.

[109] E. Klintberg, J. Dahl, J. Fredriksson, and S. Gros. An improved dual Newton strategy for scenario-tree mpc. In *2016 IEEE 55th Conference on Decision and Control (CDC)*, pages 3675–3681, 2016.

[110] S. Körkel, E. Kostina, H.G. Bock, and J.P. Schlöder. Numerical methods for optimal control problems in design of robust optimal experiments for nonlinear dynamic processes. *Optimization Methods and Software*, 19:327–338, 2004.

[111] P. Krämer-Eis. *Ein Mehrzielverfahren zur numerischen Berechnung optimaler Feedback–Steuerungen bei beschränkten nichtlinearen Steuerungsproblemen*, volume 166 of *Bonner Mathematische Schriften*. Universität Bonn, Bonn, 1985.

[112] 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 and B. Engquist, editors, *Large Scale Scientific Computing*, pages 287–306. Birkhäuser, Basel Boston Berlin, 1987.

[113] T. Kraus, P. Kühl, L. Wirsching, H.G. Bock, and M. Diehl. A Moving Horizon State Estimation algorithm applied to the Tennessee Eastman benchmark process. In *Proceedings of IEEE Robotics and Automation Society conference on Multisensor Fusion and Integration for Intelligent Systems*, 2006.

[114] T. Kraus. Real-time state and parameter estimation for NMPC-based feedback control with application to the Tennessee Eastman benchmark process. Diploma thesis, Universität Heidelberg, 2007.

[115] P. Kühl, M. Diehl, T. Kraus, J.P. Schlöder, and H.G. Bock. A real-time algorithm for moving horizon state and parameter estimation. *Computers & Chemical Engineering*, 35:71–83, 2011.

[116] P. Kühl, M. Diehl, A. Milewska, E. Molga, and H.G. Bock. Robust NMPC for a benchmark fed-batch reactor with runaway conditions. In R. Findeisen, F. Allgoewer, and L.T. Biegler, editors, *Assessment and Future Directions of Nonlinear Model Predictive Control*, volume 358 of *Lecture Notes in Control and Information Sciences*, pages 455–464. Springer Berlin/Heidelberg, 2007.

[117] P. Kühl, A. Milewska, M. Diehl, E. Molga, and H.G. Bock. NMPC for runaway-safe fed-batch reactors. In *Proceedings of the International Workshop on Assessment and Future Directions of NMPC*, pages 467–474, 2005.

[118] H.W. Kuhn and A.W. Tucker. Nonlinear programming. In J. Neyman, editor, *Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability*, pages 481–492, University of California Press, Berkeley, 1951.

[119] P. Kunkel and V. Mehrmann. *Differential–Algebraic Equations: Analysis and Numerical Solution*. European Mathematical Society Publishing House, 2006.

[120] H.C. La, A. Potschka, J.P. Schlöder, and H.G. Bock. Dual control and online optimal experimental design. *SIAM Journal on Scientific Computing*, 39(4):B640–B657, 2017.

[121] H.C. La. *Dual control for nonlinear model predictive control*. PhD thesis, Universität Heidelberg, 2016.

[122] C. Leidereiter, A. Potschka, and H. G. Bock. Dual decomposition for QPs in scenario tree NMPC. In *2015 European Control Conference (ECC 2015)*, pages 1608–1613, 2015.

[123] C. Leidereiter, A. Potschka, and H.G. Bock. Quadrature-based scenario tree generation for nonlinear model predictive control. *IFAC Proceedings Volumes*, 47(3):11087 – 11092, 2014.

[124] D.B. Leineweber. Analyse und Restrukturierung eines Verfahrens zur direkten Lösung von Optimal-Steuerungsproblemen. Diploma thesis, Ruprecht–Karls–Universität Heidelberg, 1995.

[125] D.B. Leineweber. *Efficient reduced SQP methods for the optimization of chemical processes described by large sparse DAE models*, volume 613 of *Fortschritt-Berichte VDI Reihe 3, Verfahrenstechnik*. VDI Verlag, Düsseldorf, 1999.

[126] D.B. Leineweber, I. Bauer, A.A.S. Schäfer, H.G. Bock, and J.P. Schlöder. An efficient multiple shooting based reduced SQP strategy for large-scale dynamic process optimization. Part I: Theoretical aspects. *Computers & Chemical Engineering*, 27:157–166, 2003.

[127] D.B. Leineweber, I. Bauer, A.A.S. Schäfer, H.G. Bock, and J.P. Schlöder. An efficient multiple shooting based reduced SQP strategy for large-scale dynamic process optimization. Part II: Software Aspects and Applications. *Computers & Chemical Engineering*, 27:167–174, 2003.

[128] Y. Li, K.H. Ang, and G.C.Y. Chong. PID control system analysis and design. *IEEE Control Systems Magazine*, 26(1):32–41, 2006.

[129] C. Lindscheid, D. Hakerl, A. Meyer, A. Potschka, H.G. Bock, and S. Engell. Parallelization of modes of the multi-level iteration scheme for nonlinear model-predictive control of an industrial process. In *2016 IEEE Conference on Control Applications (CCA)*, pages 1506–1512, 2016.

[130] A. Locatelli. *Optimal control – an introduction*. Birkhäuser, Basel Boston Berlin, 2001.

[131] F.A. Lootsma. A survey of methods for solving constrained minimization problems via unconstrained minimization. In F.A. Lootsma, editor, *Numerical Methods for Nonlinear Optimization*, pages 313–347. Academic Press, London, 1972.

[132] A.J. Lotka. Contribution to the theory of periodic reactions. *The Journal of Physical Chemistry*, 14(3):271–274, 1909.

[133] S. Lucia, T. Finkler, D. Basak, and S. Engell. A new robust NMPC scheme and its application to a semi-batch reactor example. *IFAC Proceedings Volumes*, 45(15):69 – 74, 2012.

[134] W.L. Luyben. *Process modeling, simulation, and control for chemical engineers*. McGraw-Hill, New York, 1990.

[135] L. Magni, D.M. Raimondo, and F. Allgöwer, editors. *Nonlinear Model Predictive Control: Towards New Challenging Applications*, volume 384 of *Lecture Notes in Control and Information Sciences*. Springer, 2009.

[136] A.G. Marchetti and D. Zumoffen. On the links between real-time optimization, neighboring-extremal control, and self-optimizing control. In *2013 European Control Conference (ECC 2013)*, pages 4466–4471.

[137] D.Q. Mayne, J.B. Rawlings, C.V. Rao, and P.O.M. Scokaert. Constrained model predictive control: stability and optimality. *Automatica*, 26(6):789–814, 2000.

[138] D.Q. Mayne. Model predictive control: Recent developments and future promise. *Automatica*, 50(12):2967 – 2986, 2014.

[139] A. Neumaier. *Mathematical Model Building*, pages 37–43. Springer US, Boston, MA, 2004.

[140] J. Nocedal and S.J. Wright. *Numerical Optimization*. Springer Verlag, Berlin Heidelberg New York, 2nd edition, 2006.

[141] H.J. Pesch. *Numerische Berechnung optimaler Flugbahnkorrekturen in Echtzeitrechnung*. PhD thesis, TU München, 1978.

[142] H.J. Pesch. A practical guide to the solution of real-life optimal control problems. *Control and Cybernetics*, 23:7–60, 1994.

[143] K.J. Plitt. Ein superlinear konvergentes Mehrzielverfahren zur direkten Berechnung beschränkter optimaler Steuerungen. Diploma thesis, Rheinische Friedrich–Wilhelms–Universität Bonn, 1981.

[144] L.S. Pontryagin, V.G. Boltyanski, R.V. Gamkrelidze, and E.F. Miscenko. *The Mathematical Theory of Optimal Processes*. Wiley, Chichester, 1962.

[145] A. Potschka. Handling path constraints in a direct multiple shooting method for optimal control problems. Diploma thesis, Universität Heidelberg, 2006.

[146] A. Potschka, H.G. Bock, and J.P. Schlöder. A minima tracking variant of semi-infinite programming for the treatment of path constraints within direct solution of optimal control problems. *Optimization Methods and Software*, 24:237–252, 2009

[147] A. Potschka. *A direct method for the numerical solution of optimization problems with time-periodic PDE constraints*. PhD thesis, Universität Heidelberg, 2012.

[148] A. Potschka, C. Kirches, H.G. Bock, and J.P. Schlöder. Reliable solution of convex quadratic programs with parametric active set methods. Technical Report 2010–11–2828, Heidelberg University, Interdisciplinary Center for Scientific Computing, Heidelberg University, Im Neuenheimer Feld 368, 69120 Heidelberg, Germany, 2010.

[149] M.J.D. Powell. The convergence of variable metric methods for nonlinearly constrained optimization calculations. In O.L. Mangasarian, R.R. Meyer, and S.M. Robinson, editors, *Nonlinear Programming 3*, Academic Press, 1978.

[150] M.J.D. Powell. Algorithms for nonlinear constraints that use Lagrangian functions. *Mathematical Programming*, 14(3):224–248, 1978.

[151] 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*, Springer, Berlin, 1978.

[152] F. Pukelsheim. *Optimal Design of Experiments*. Classics in Applied Mathematics 50. SIAM, 2006.

[153] 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.

[154] S.J. Qin and T.A. Badgwell. Review of nonlinear model predictive control applications. In B. Kouvaritakis and M. Cannon, editors, *Nonlinear model predictive control: theory and application*, pages 3–32, London, 2001. The Institute of Electrical Engineers.

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

[156] J.B. Rawlings, D.Q. Mayne, and M. Diehl. *Model Predictive Control: Theory, Computation, and Design*. Nob Hill Publishing, LLC, 2nd edition, 2017.

[157] J.J. Renes. *On the use of splines and collocation in a trajectory optimization algorithm based on mathematical programming*. NLR, 1978.

[158] S.M. Robinson. Perturbed Kuhn-Tucker points and rates of convergence for a class of nonlinear programming algorithms. *Mathematical Programming*, 7:1–16, 1974.

[159] W. Rudin. *Real and Complex Analysis*. McGraw-Hill, 1966.

[160] Y. Saad. *Iterative Methods for Sparse Linear Systems*. SIAM, Philadelpha, PA, 2nd edition, 2003.

[161] S. Sager. *Numerical methods for mixed–integer optimal control problems*. PhD thesis, Universität Heidelberg, 2006.

[162] S. Sager. On the integration of optimization approaches for mixed-integer nonlinear optimal control. Habilitation, University of Heidelberg, August 2011.

[163] 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.

[164] H. Sayama. *Introduction to the modeling and analysis of complex systems*. Open SUNY Textbooks, 2015.

[165] H. Schichl. *Models and the History of Modeling*, pages 25–36. Springer US, Boston, MA, 2004.

[166] R. Scholz. Stabiles Condensing für Optimale Steuerung. Master thesis, Universität Heidelberg, 2016.

[167] P.O.M. Scokaert and D.Q. Mayne. Min-max feedback model predictive control for constrained linear systems. *IEEE Transactions on Automatic Control*, 43:1136–1142, 1998.

[168] P.O.M. Scokaert, D.Q. Mayne, and J.B. Rawlings. Suboptimal model predictive control (feasibility implies stability). *IEEE Transactions on Automatic Control*, 44(3):648–654, 1999.

[169] E.D. Sontag. *Mathematical Control Theory: Deterministic Finite Dimensional Systems*. Number 6 in Textbooks in Applied Mathematics. Springer-Verlag, New York, 2nd edition, 1998.

[170] M.C. Steinbach, H.G. Bock, and R.W. Longman. Time optimal extension or retraction in polar coordinate robots: A numerical analysis of the switching structure. In *Proceedings of the 1989 AIAA Guidance, Navigation and Control Conference*, Boston, 1989.

[171] R.F. Stengel. *Optimal Control and Estimation*. Dover Publications, 1994.

[172] G.W. Stewart. A Krylov-Schur algorithm for large eigenproblems. *SIAM Journal on Matrix Analysis and Applications*, 23(3):601–614, 2002.

[173] P.A. Tipler and G. Mosca. *Physics for scientists and engineers*. Freeman, New York, 6th edition, 2008.

[174] Q. Tran-Dinh, C. Savorgnan, and M. Diehl. Adjoint-based predictor-corrector sequential convex programming for parametric nonlinear optimization. *SIAM Journal on Optimization*, 22(4):1258–1284, 2012.

[175] A. van den Bos. *Parameter Estimation for Scientists and Engineers*. John Wiley & Sons, 2007.

[176] R. Vilanova and A. Visioli, editors. *PID Control in the Third Millennium*. Advances in Industrial Control. Springer, 2012.

[177] A. Visioli. *Practical PID Control*. Advances in Industrial Control. Springer, 2006.

[178] V. Volterra. Variazioni e fluttuazioni del numero d'individui in specie animali conviventi. *Memorie della Reale Accademia Nazionale dei Lincei*, VI-2, 1926.

[179] G. Wachsmuth. On LICQ and the uniqueness of Lagrange multipliers. *Operations Research Letters*, 41:78–80, 2013.

[180] A. Wächter. *An Interior Point Algorithm for Large-Scale Nonlinear Optimization with Applications in Process Engineering*. PhD thesis, Carnegie Mellon University, 2002.

[181] J.H. Wilkinson. *The Algebraic Eigenvalue Problem*. Clarendon Press, Oxford, 1965.

[182] D.I. Wilson, M. Agarwal, and D.W.T. Rippin. Experiences implementing the extended Kalman filter on an industrial batch reactor. *Computers & Chemical Engineering*, 22:1653–1672, 1998.

[183] R.B. Wilson. *A simplicial algorithm for concave programming*. PhD thesis, Harvard University, 1963.

[184] L. Wirsching. An SQP algorithm with inexact derivatives for a Direct Multiple Shooting method for optimal control problems. Diploma thesis, Universität Heidelberg, 2006.

[185] L. Wirsching, J. Albersmeyer, P. Kühl, M. Diehl, and H.G. Bock. An adjoint-based numerical method for fast nonlinear model predictive control. In M.J. Chung and P. Misra, editors, *Proceedings of the 17th IFAC World Congress, Seoul, Korea, July 6–11, 2008*, volume 17, pages 1934–1939. IFAC-PapersOnLine, 2008.

[186] L. Wirsching, H.G. Bock, and M. Diehl. Fast NMPC of a chain of masses connected by springs. In *Proceedings of the 2006 IEEE International Conference on Control Applications (CCA)*, pages 591–596, 2006.

[187] L. Wirsching, H.J. Ferreau, H.G. Bock, and M. Diehl. An online active set strategy for fast adjoint based nonlinear model predictive control. In *Proceedings of the 7th Symposium on Nonlinear Control Systems (NOLCOS), Pretoria*, 2007.

[188] I.J. Wolf and W. Marquardt. Fast NMPC schemes for regulatory and economic NMPC – a review. *Journal of Process Control*, 44:162 – 183, 2016.

[189] I.J. Wolf, H. Scheu, and W. Marquardt. A hierarchical distributed economic NMPC architecture based on neighboring-extremal updates. In *American Control Conference (ACC), 2012*, pages 4155–4160. IEEE, 2012.

[190] L. Würth, R. Hannemann, and W. Marquardt. Neighboring-extremal updates for nonlinear model-predictive control and dynamic real-time optimization. *Journal of Process Control*, 19(8):1277–1288, 2009.

[191] X. Yang and L.T. Biegler. Advanced-multi-step nonlinear model predictive control. *Journal of Process Control*, 23(8):1116–1128, 2013.

[192] V.M. Zavala and L.T. Biegler. The advanced–step NMPC controller: optimality, stability and robustness. *Automatica*, 45(1):86–93, January 2009.