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

vorgelegt von
M. Sc. Conrad Leidereiter
aus Prenzlau

Tag der mündlichen Prüfung:

12. Februar 2018

Numerical Methods for
Scenario Tree Nonlinear Model Predictive Control

Gutachter:

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

Danksagung

Meine tiefe Dankbarkeit gilt an dieser Stelle allen, die mich bei dieser Arbeit inspiriert und unterstützt haben. Ich danke meinen Lehrern und Mentoren der Fakultät für Mathematik und Informatik, Hans Georg Bock, Christian Kirches, Ekaterina Kostina, Johannes Schlöder und Andreas Potschka, für das Teilen ihres umfangreichen Wissens und für das Vertrauen in meine Arbeit.

Für die anregende Umgebung danke ich dem Interdisziplinären Zentrum für Wissenschaftliches Rechnen der Ruprecht-Karls-Universität Heidelberg, deren Lehre, Forschung und Entwicklung die Basis dieser Arbeit formen. Ich bedanke mich bei allen Kollegen der Arbeitsgruppen Simulation und Optimierung, Numerische Optimierung, Model-Based Optimizing Control, Optimization of Uncertain Systems und Optimum Experimental Design für die freundliche und kooperative Atmosphäre, für die fachlichen und die überfachlichen Gespräche.

Der Graduiertenschule HGS MathComp als Struktur aus der Exzellenzinitiative danke ich für ihre Angebote, die ich während der Promotion gern genutzt habe. Für die erfolgreiche Zusammenarbeit im Kreis der Officer des Heidelberg Chapter of SIAM danke ich Anja Bettendorf, Dominik Cebulla, Diana-Patricia Danciu, Jürgen Gutekunst, Robert Kircheis, Ole Klein, Felix Lenders, Robert Scholz, Ruobing Shen und Christoph Weiler.

Dem DFG-Cluster Optimizing Control for Uncertain Systems und dem ERC Grant Model-Based Optimizing Control gebühren Dank für die finanzielle Unterstützung.

Meinen Dank für die Unterstützung in organisatorischen Angelegenheiten richte ich an die Verwaltung der Arbeitsgruppe Simulation und Optimierung, Abir Al-Laham, Margret Rothfuß, Anastasia Valter und Anja Vogel, sowie an die HGS Verwaltung, Ria Hillenbrand-Lynott und Sarah Steinbach. Für die Rahmenbedingungen der Promotion danke ich der Fakultät für Mathematik und Informatik der Ruprecht-Karls-Universität Heidelberg und für die Unterstützung in formalen Promotionsangelegenheiten danke ich Dorothea Heukäufer vom Dekanat.

Schließlich danke ich von Herzen meinen Eltern und meiner Schwester, dass sie immer für mich da sind.

Abstract

In this thesis we propose new methods in the field of numerical mathematics and stochastics for a model-based optimization method to control dynamical systems under uncertainty. In model-based control the model-plant mismatch is often large and unforeseen external influences on the dynamics must be taken into account. Therefore we extend the dynamical system by a stochastic component and approximate it by scenario trees. The combination of Nonlinear Model Predictive Control (NMPC) and the scenario tree approach to robustify with respect to the uncertainty is of growing interest. In engineering practice scenario tree NMPC yields a beneficial balance of the conservatism introduced by the robustification with respect to the uncertainty and the controller performance. However, there is a high numerical effort to solve the occuring optimization problems, which heavily depends on the design of the scenario tree used to approximate the uncertainty. A big challenge is then to control the system in real-time. The contribution of this work to the field of numerical optimization is a structure exploiting method for the large-scale optimization problems based on dual decomposition that yields smaller subproblems. They can be solved in a massively parallel fashion and additionally their discretization structure can be exploited numerically. Furthermore, this thesis presents novel methods to generate suitable scenario trees to approximate the uncertainty. Our scenario tree generation based on quadrature rules for sparse grids allows for scenario tree NMPC in high-dimensional uncertainty spaces with approximation properties of the quadrature rules. A further novel approach of this thesis to generate scenario trees is based on the interpretation of the underlying stochastic process as a Markov chain. Under the Markovian assumption we provide guarantees for the scenario tree approximation of the uncertainty. Finally, we present numerical results for scenario tree NMPC. We consider dynamical systems from the chemical industry and demonstrate that the methods developed in this thesis solve optimization problems with large scenario trees in real-time.

Zusammenfassung

Diese Arbeit stellt neue Methoden aus dem Bereich der numerischen Mathematik und der Stochastik für ein modellbasiertes Optimierungsverfahren zur Regelung unsicherheitsbehafteter dynamischer Systeme vor. Bei der Anwendung modellbasierter Regelungsansätze ist die Diskrepanz zwischen dem mathematischen Modell und der zu steuernden Anlage oft groß. Auch unvorhersehbare externe Einflüsse müssen berücksichtigt werden. Daher erweitern wir das dynamische System um eine stochastische Komponente und approximieren diese durch einen Szenarienbaum. Nichtlineare modellprädiktive Regelung (NMPC) in Kombination mit dem Szenarienbaumansatz als Robustifizierung gegen die Unsicherheit stößt auf wachsendes Interesse in der Regelungstechnik, denn in der Praxis hat sich gezeigt, dass Szenarienbaum-NMPC eine gute Balance zwischen dem Konservatismus von Robustifizierungsmethoden und der Performanz des Reglers schafft. Der numerische Aufwand für die Lösung der auftretenden Optimierungsprobleme hängt stark von der Struktur des Szenarienbaumes ab. Dieser wiederum soll die Unsicherheit möglichst gut approximieren. Eine große Herausforderung von Szenarienbaum-NMPC ist die Lösung der auftretenden Optimierungsprobleme in Echtzeit. Der Beitrag dieser Arbeit im Forschungsfeld der numerischen Optimierung ist ein strukturausnutzendes Verfahren, welches die auftretenden Optimierungsprobleme mit Hilfe von dualer Dekomposition in kleinere Teilprobleme zerlegt. Die Teilprobleme können parallel gelöst werden unter zusätzlicher numerischer Ausnutzung ihrer Diskretisierungsstruktur. Desweiteren stellt diese Arbeit neuartige Verfahren vor, die passende Szenarienbäume generieren. Unsere Szenarienbaumgenerierung basierend auf Quadraturformeln mit dünnen Gittern ermöglicht Szenarienbaum-NMPC in mehrdimensionalen Unsicherheitsräumen mit den Approximationseigenschaften der Quadraturformeln. Ein weiterer neuartiger Ansatz der Arbeit zum Generieren von Szenarienbäumen basiert auf der Interpretation des zugrundeliegenden stochastischen Prozesses als Markovkette. Unter der Markov-Annahme geben wir Garantien, wie gut die Unsicherheit durch den Szenarienbaum approximiert wird. Schließlich präsentieren wir in der Arbeit numerische Resultate für Szenarienbaum-NMPC. Wir betrachten dynamische Systeme aus industriellen Anwendungen der chemischen Verfahrenstechnik und belegen, dass mit den entwickelten Methoden dieser Arbeit Optimierungsprobleme mit großen Szenarienbäumen in Echtzeit gelöst werden können.

Contents

Chapter 1

Introduction

Mathematical optimization under uncertainty is a growing field of research as the awareness of the uncertain influence has increased. Steering real-world systems into a desired state by dynamical model-based optimization approaches requires to make decisions within a specified time. Full information that can serve as basis for the decision is hardly ever available. Therefore uncertainty occurs in a natural way and has to be considered in dynamical model-based optimization methods. Robert K. Greenleaf, a pioneer of modern management, leadership, organizational development and education approaches wrote in his essay [49]:

> ”On an important decision one rarely has 100 % of the information needed for a good decision no matter how much one spends or how long one waits. And, if one waits too long, he has a different problem and has to start all over.”

We emphasize three insights from the citation that originates from a management context but fits perfectly in our framework. First, uncertainty is present in decision making. Second, the uncertainty does not vanish completely over time. And third, the problem that requires decisions, will change over time. Especially the third insight will lead us later to the fast feedback principle.

Scenario tree Nonlinear Model Predictive Control accounts for the uncertain influence and generates decisions to control a dynamical system in a robust sense - robust against the uncertainty. Nonlinear Model Predictive Control (NMPC) is a model-based mathematical optimization framework to obtain feedback control inputs for a dynamical system. The framework has been applied in various fields such as physics, chemistry, biology, engineering, social sciences and psychology, overviews of NMPC applications can be found for example in [85, 95]. Techniques to numerically address the mathematical optimization problems in NMPC have been developed e.g. in [17, 109, 4, 11] and continuously enhanced by real-time and multi-level components [27, 15, 114, 67, 44]. However, the mismatches

between the real-world system and the mathematical model as well as disturbances are often large. Therefore NMPC can only be successful when the feedback controls are computed in a robust sense as [10] pointed out. Otherwise, in the case of a chemical plant for example, violations of system constraints can cause severe safety-critical situations.

Robust NMPC techniques treat model-plant mismatch and disturbances as probabilistic perturbations to a nominal model. One technique is the game-theoretic worst case approach, mathematically a bilevel optimization problem [26]. It suffers from high conservatism, because it safeguards against all possible realizations of the perturbation at the same time. Lucia et al. [84] have shown that using scenario tree NMPC as proposed in [23] can efficiently reduce this conservatism while remaining feasible with high probability.

The main assumption of the scenario tree approach is of stochastic nature. We assume that the uncertainty can be approximated by a discrete and finite tree process. Usually the scenario tree construction starts with choosing a finite number of realizations of the uncertainty. Then a number of decision points in time are specified. At these points the realizations are allowed to change. The sequences of those realizations are regarded as scenarios that must be coupled according to the tree structure. Figure 1.1 illustrates the classical tree construction. There are alternatives to generate the scenario trees. In this thesis we develop tree generation algorithms based on the investigation of the underlying stochastic process.

The major challenge of scenario tree NMPC is the exponential growth of the full tree in the number of decision points. This yields a high demand for fast structure-exploiting numerical methods. Especially in high dimensional uncertainty spaces the classical tree construction itself must be questioned. How should the uncertainty space be discretized and how should we choose a finite number of realizations? Are there alternative tree structures? This thesis aims to provide answers to the questions based on a mathematical investigation of the stochastic process behind the scenario tree and practical implementations of demanding case studies. Contributions of the thesis to scenario tree NMPC are named in the next section.

1.1 Aims and Contributions of this Thesis

The overarching goal is to control dynamic processes under uncertain influence in real-time using scenario tree NMPC. This thesis contributes to the field of applied mathematics by the following topics.

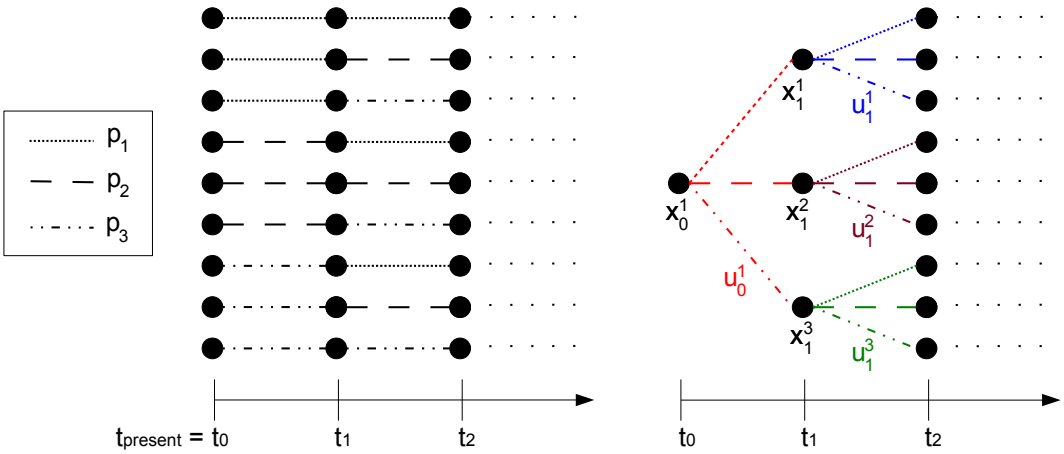

Figure 1.1: Single scenarios (left) and scenario tree (right) with three realizations of the uncertainty and two decision points. The three realizations are represented by the dotted, the dashed and the two-dot-one-dash line styles. We have two decision points t_0 and t_1 where the realization can change. Between the time points t_0 and t_1 (stage 1) and between t_1 and t_2 (stage 2) the parameter realizations stay constant respectively. The nine scenarios on the left are all combinations of the parameter realizations for the first two stages. We couple the scenarios due to a tree-specific stochastic principle and arrive at the scenario tree structure on the right. The colors indicate the controls we have at hand for optimization according to the stochastic tree process. For the continuation of the scenarios beyond t_2 there are various possibilities indicated by dots. Most commonly the realizations stay constant over the rest of the considered time horizon.

Formulation of stochastic properties of the scenario tree process

Scenario tree NMPC is located at the intersection of the fields numerics, optimization, process control and stochastics and has been investigated by researchers of all those fields. However, the stochastic process behind scenario trees requires a rigorous formulation. This work contributes a mathematical tree process formulation and rigorous investigation of a special process property, the non-anticipativity.

Fast structure-exploiting numerical methods for scenario tree NMPC

For fast NMPC with scenario trees the exploitation of the discretization structure, tree structure and state-of-the art methods to deal with real-time requirements are inevitable. In this thesis we contribute a fast numerical method for the solution of large-scale tree-structured problems based on a dual decomposition approach that exploits the inherent tree structure. In combination with discretization structure exploitation and the Real-Time Iteration or Multi-Level Iteration scheme we can speed up the necessary computations of the process feedback.

Quadrature-based scenario tree generation

In high-dimensional uncertainty spaces the major challenge of generating scenario trees is the choice of finitely many realizations of the uncertainty. We present a contribution of scenario tree generation based on sparse grid quadrature rules, a commonly used tool in the field of uncertainty quantification. With the approach the number of realizations can be significantly reduced and there is still a reasonable approximation quality determined by error formulars of sparse grid quadrature.

Scenario tree generation based on Markov chains

In the classical tree construction the scenarios are determined by the finitely many realizations and the decision points where the tree can branch. As a consequence the number of scenarios grows exponentially in the number of decision points. Hence, for practical computations the number of decision points must be restricted to a robust horizon in the beginning. This thesis contributes an alternative scenario tree generation approach that is based on the stochastic properties of the tree process and does not require the choice of a robust horizon. We provide a fast implementation of the alternative tree generating algorithm that builds on interpreting scenarios as realizations of Markov chains over time and pruning the most unlikely scenarios.

Implementation of scenario tree NMPC

We contribute a software implementation of the structure-exploiting numerical algorithms of the thesis to demonstrate the effectiveness and efficiency of the methods. The STMLI package (Scenario Tree Multi Level Iteration package) is the scenario tree extension of the MLI software (Multi Level Iteration software) that implements the Multi Level Iteration idea [16, 15].

Demanding case studies

One important contribution of this work is to apply the developed methods to challenging problems. First, we study a basic chemical reactor, a continuous stirred tank reactor. Then we continue with a demanding real-world industrial problem, a biochemical batch reactor. The plant model is provided by BASF. As a third example we consider a pharmaceutical problem on Penicillin production. All problems have in common that there is uncertainty present in the dynamical system model. We showcase the performance and robustness of scenario tree NMPC with respect to the uncertainty.

Published contributions

The scenario tree generating approach based on sparse grids has been published in

> [74]: C. Leidereiter, A. Potschka, and H. G. Bock. Quadrature-based scenario tree generation for Nonlinear Model Predictive Control. In *Proceedings of the 19th IFAC World Congress*, volume 47, pages 11087–11092, 2014.

To demonstrate performance and robustness of NMPC with scenario trees generated from sparse grids we study the control of a simulated distillation column with three-dimensional uncertainty space on a Monte-Carlo testbed and statistically evaluate the results.

The scenario tree structure exploiting numerical method based on dual decomposition has been published in

> [75]: C. Leidereiter, A. Potschka, and H. G. Bock. Dual decomposition for QPs in scenario tree NMPC. In *Proceedings of the European Control Conference (ECC15)*, pages 1608–1613, 2015.

We present the structure-exploiting numerics and computational results for NMPC with large scenario trees.

Scenario tree generation by the Markov chain approach will be published in

[73]: C. Leidereiter, D. Kouzoupis, M. Diehl, and A. Potschka. Pruning for scenario tree NMPC with uncertainties described by Markov chains. In preparation.

In the above publication we will discuss numerical results for an uncertainty that has an intrinsic Markovian property.

1.2 Organization of this Thesis

Chapter 1 is this introduction. In Chapter 2 of problem classification we put scenario tree NMPC into context and view the main object of this thesis from a numerical, optimization, process control and stochastic perspective. The latter focuses on properties of the stochastic tree process, especially non-anticipativity. In the three subsequent chapters we present numerical methods for scenario tree NMPC. We discuss the Direct Multiple Shooting discretization including the exploitation of discretization structure in Chapter 3. It serves as the basis for fast feedback NMPC, that we survey in Chapter 4. We propose our tree structure exploiting numerical method in Chapter 5. The dual decomposition approach with non-smooth Newton method is the main contribution of numerical structure exploitation in this thesis. The contributions based on the stochastic nature of the scenario tree process follow in Chapter 6 with quadrature-based scenario tree generation and in Chapter 7 with Markov chain scenario tree pruning. In Chapter 8 we describe the software implementation of the numerical methods. With numerical results for demanding applications we demonstrate the efficiency of our methods in Chapter 9. In the last Chapter 10 we draw conclusions and give an outlook on topics that arise from the results of this thesis.

Chapter 2

Problem Classification

In this chapter we start with the mathematical formulation of the scenario tree optimization problem, also known as multi-stage optimization problem. We approach the core object from various perspectives. The first perspective is to consider the dynamical aspects and focus on the optimal control formulation. We continue with the optimization perspective and focus on the discretized finite nonlinear optimization problem in the second section. The next perspective, the dynamical process control, underlines the requirement of fast feedback methods. At the end of this chapter we present the stochastic properties of the scenario tree process. Theorem, proof and examples in the last section are contributions of this thesis.

2.1 The Scenario Tree Optimization Problem

Generally, optimization with scenario trees is a systematic and efficient approach to deal with uncertainty in nonlinear model predictive control (NMPC) as we have pointed out in the introductory chapter. At this point we already state the scenario tree optimal control problem as one core optimization problem in scenario tree NMPC being aware that background from different perspectives in the next sections is required to fully explain it.

We set $I := [t_0, t_f] \subset \mathbb{R}$ as the underlying time domain, define the states $x : I \to \mathbb{R}^{n_x}$ and the controls $u : I \to \mathbb{R}^{n_u}$. In this section we denote the possibly uncertain parameters simply as $p \in \mathbb{R}^{n_p}$. A further investigation and rigorous definition follows later on. We consider a finite number $S \in \mathbb{N}$ of scenarios. From now on the elements of the set of scenarios $\mathcal{S} = \{1, \ldots, S\}$ have index j. We assign a weight $w_j \in [0,1]$ to all scenarios $j \in \mathcal{S}$. Furthermore, we define the set $\mathcal{K} = \{0, \ldots, M-1\}$. The time interval I is partitioned by

$$t_0 < t_1 < \cdots < t_{M-1} < t_M = t_f$$

into intervals $[t_k, t_{k+1}]$, $k \in \mathcal{K}$, the so-called stages that are eponymous for the multi-stage

approach.
The sufficiently smooth functions

$$f : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_p} \to \mathbb{R}^{n_x}, \qquad \Phi : \mathbb{R} \times \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_p} \to \mathbb{R}$$

describe the system dynamics and the cost function. Furthermore we require sufficiently smooth descriptions of the feasible regions $\mathcal{X}$ and $\mathcal{U}$ for the states and controls. At the current system state $x(t_0) = x_0$ the optimal control problem reads

$$\min_{x(t),u(t)} \quad \sum_{j\in\mathcal{S}} w_j \int_{t_0}^{t_f} \Phi(t, x_j(t), u_j(t), p_j(t))\mathrm{d}t \tag{2.1a}$$

$$\text{s.t.} \quad \dot{x}_j(t) = f(x_j(t), u_j(t), p_j(t)), \qquad t \in [t_0, t_f],\ j \in \mathcal{S}, \tag{2.1b}$$

$$x_j(t_0) = x_0, \qquad j \in \mathcal{S}, \tag{2.1c}$$

$$x_j(t) \in \mathcal{X}, \qquad t \in [t_0, t_f],\ j \in \mathcal{S}, \tag{2.1d}$$

$$u_j(t) \in \mathcal{U}, \qquad t \in [t_0, t_f],\ j \in \mathcal{S}, \tag{2.1e}$$

$$u_i(t) = u_j(t), \qquad t \in [t_k, t_{k+1}],\ (i,j) \in \mathcal{C}_k,\ k \in \mathcal{K}. \tag{2.1f}$$

In the above formulation every scenario has its own controls. We define the set $\mathcal{C}_k$ for constraint (2.1f) to encode the tree structure,

$$\mathcal{C}_k := \left\{ (i,j) \in \mathcal{S}^2 \,\middle|\, p_i(t) = p_j(t) \text{ for all } t \in [t_0, t_k] \right\}.$$

The set $\mathcal{C}_k$ is required for the scenario-wise view of the tree structure. For every stage k, the set $\mathcal{C}_k$ contains tuples of scenarios with common history of parameter values on all previous stages. The respective control variables of the scenario tuples in $\mathcal{C}_k$ are coupled on stage k according to (2.1f). We emphasize that the constraints (2.1f) are the only coupling of the scenarios. Problem (2.1) without the linear coupling constraints completely decouples into S scenario optimal control problems. To sum up, we minimize a weighted sum of scenario objectives with respect to dynamic constraints, an initial value, upper and lower variable bounds and linear scenario coupling constraints.

2.2 Dynamical System Perspective: Optimal Control

The evolution of processes that change in time can be modelled by dynamical systems. Transferring the observations from a physical, biological or chemical process to a dynamical model and applying mathematical tools helps to gain more insight into the process behavior. This applies as well to processes from economical, social and life sciences. For abstract dynamical systems we refer to [52]. The ordinary differential equation (ODE) of the form

$$\dot{x} = f(t,x)$$

corresponds in a natural way to a dynamical system. The function $f : D \to \mathbb{R}^d$ is a time-dependent continuous vector field on $D \subset \mathbb{R} \times \mathbb{R}^d$. In the autonomous case f does not depent on time. But we can transform every ODE to an autonomous ODE by introducing an extra state τ that represents the time. We then replace all t-dependencies of f by τ-dependencies and add the equations $\dot{\tau} = 1$ and $\tau(0) = t_0$ to the ODE.

For an ODE $\dot{x} = f(t,x)$ and $(t_0,x_0) \in D$ we can solve initial value problems (IVPs). In the following we state the fundamental existence and uniqueness theorem of IVP solutions.

Theorem 2.1 (Existence and uniqueness of solutions)**.** *Let $f : D \to \mathbb{R}^d, D \subset \mathbb{R} \times \mathbb{R}^d$ be a vector field that is continous and Lipschitz continuous in the second argument. Now consider the ODE $\dot{x} = f(t,x)$. Then for all initial values $(t_0,x_0) \in D$ there exists a unique solution $x(t;t_0,x_0)$ to the ODE with initial condition $x(t_0) = x_0$ on a maximum existence interval $I_{t_0,x_0} \subset \mathbb{R}$ with $t_0 \in I_{t_0,x_0}$.*

The existence and uniqueness theorem is usually proven by the Banach contraction mapping theorem. Often the theorem is referred to Picard and Lindelöf.

Ordinary differential equations implicitly define the dynamical evolution constraint of optimal control problems. At this point we state an optimal control problem (OCP) in a general form to optimize a process with underlying dynamical system. We transfer the assumptions of Theorem 2.1 to all following OCPs with underlying ODE.

The general optimal control problem reads

$$\min_{x,u} \quad \int_{t_0}^{t_f} \Phi(t,x(t),u(t),p(t))\mathrm{d}t \tag{2.2a}$$

$$\text{s.t.} \qquad \dot{x}(t) = f(x(t),u(t),p(t)), \qquad t \in [t_0,t_f], \tag{2.2b}$$

$$0 = x(t_0) - x_0, \tag{2.2c}$$

$$0 \leq r(t,x(t),u(t),p(t)), \qquad t \in [t_0,t_f]. \tag{2.2d}$$

Our goal is to minimize the objective over the states $x : [t_0,t_f] \to \mathbb{R}^{n_x}$ and the controls $u : [t_0,t_f] \to \mathbb{R}^{n_u}$. We do not regard the possibly uncertain parameters $p \in \mathbb{R}^{n_p}$ as optimization variables as we later interpret them stochastically.

The objective function (2.2a) is a Lagrange term

$$\int_{t_0}^{t_f} \Phi(t, x(t), u(t), p(t)) \mathrm{d}t$$

with $\Phi : [t_0, t_f] \times \mathbb{R}^{n_x+n_u+n_p} \to \mathbb{R}$. The function Φ is assumed to be twice continuously differentiable.

The dynamical evolution of the system is implicitly given by the constraint (2.2b) and the initial value of the dynamical evolution is set by (2.2c).

Additionally, we have path constraints (2.2d) described by a twice continuously differentiable $r : [t_0, t_f] \times \mathbb{R}^{n_x+n_u+n_p} \to \mathbb{R}^{n_r}$. The so-called box constraints of the form

$$\underline{x}(t) \leq x(t) \leq \overline{x}(t)$$
$$\underline{u}(t) \leq u(t) \leq \overline{u}(t)$$

are a special case of (2.2d).

Further possible components of optimal control problems such as interior point constraints are not considered in this thesis. Objective formulations that depend on $(t_f, x(t_f))$ can be reformulated to fit (2.2a).

Now the question arises how to solve the OCP (2.2) as the optimization variables are functions and we therefore have an infinite-dimensional problem. For an in-depth introduction to functional analysis and the relation to optimal control and calculus of variations we refer to the textbook [21].

Since the middle of the 20th century indirect approaches to solve optimal control problems emerged from Pontryagin's maximum principle [91]. The maximum principle states necessary conditions of optimality which can be solved analytically for certain examples. We first transform the optimal control problem to a boundary value problem (BVP) by the maximum principle and then solve the BVP numerically by collocation or shooting methods.

In contrast to the indirect approach we choose direct methods - we first discretize the infinite-dimensional optimization problem and then optimize. The direct multiple shooting method [17] and the direct collocation method [109, 4, 11] yield finite dimensional optimization problems from optimal control problems. For an efficient and fast solution the resulting discretized problems must be treated with tailored numerical methods. The direct multiple shooting discretization and a related discretization structure exploiting method is explained in Chapter 3.

We conclude the section with the remark that the scenario tree problem (2.1) is a special case of the OCP (2.2). It has to be emphasized that the methods developed for problems of the form (2.2) are fully applicable to the scenario tree problem (2.1). Furthermore we can

understand the OCP (2.2) as problem (2.1) with only one scenario and weight $w_1 = 1$. The so-called nominal problem corresponding to a scenario tree optimization problem is such a one-scenario problem with the nominal parameter realization.

2.3 Optimization Perspective: Nonlinear Programming

In Section 2.2 we have stated that the OCP (2.2) or its special case with tree structure (2.1) can be transferred to a finite dimensional optimization problem by direct methods. Those finite dimensional optimization problems are the mathematical objects of interest in the field of numerical optimization. The general constrained nonlinear optimization problem (NLP) in $\mathbb{R}^n$ is defined as follows.

Definition 2.1 (Nonlinear optimization problem)**.**

$$\min_{z\in\mathbb{R}^n} \quad F(z) \tag{2.3a}$$

$$\text{s.t.} \quad c_i(z) = 0, \quad i \in \mathcal{E} \tag{2.3b}$$

$$c_i(z) \geq 0, \quad i \in \mathcal{I}. \tag{2.3c}$$

The functions F and c_i are assumed to be twice continuously differentiable functions mapping a subset of $\mathbb{R}^n$ to $\mathbb{R}$. We call F the objective function, $c_i, i \in \mathcal{E}, |\mathcal{E}| = ce$ equality constraints and $c_i, i \in \mathcal{I}, |\mathcal{I}| = ci$ inequality constraints. In this formulation the sets $\mathcal{E}$ and $\mathcal{I}$ are disjoint. If the objective function and all the constraint functions are linear, (2.3) becomes a linear programming problem (LP). The quadratic programming problem (QP) is (2.3) with linear constraint functions and an objective function of the form

$$F(z) = \frac{1}{2} z^T H z + g^T z.$$

The symmetric matrix $H \in \mathbb{R}^{n\times n}$ is addressed as the Hessian of the QP and the vector $g \in \mathbb{R}^n$ as the QP gradient. In the following we state only the most important definitions and theorems of NLP theory as this is not the main focus of thesis. A comprehensive introduction to numerical optimization is for example the book [88].

Definition 2.2 (Feasible)**.** *A point $\bar{z} \in \mathbb{R}^n$ is called feasible point of the problem* (2.3) *if it satisfies*

$$c_i(\bar{z}) = 0 \qquad \text{for all } i \in \mathcal{E},$$

$$c_i(\bar{z}) \geq 0 \qquad \text{for all } i \in \mathcal{I}.$$

Definition 2.3 (Local optimum)**.** *A point $z^* \in \mathbb{R}^n$ is called locally optimal point of the problem* (2.3) *if for $\varepsilon > 0$ there exists an open ball $\mathcal{B}_\varepsilon(z^*)$ with $\varepsilon > 0$ such that*

$$F(z) \geq F(z^*)$$

for all feasible $z \in \mathcal{B}_\varepsilon(z^*)$.
If additionally $F(z) > F(z^*)$ holds for all feasible $z \neq z^*$, then z^* is a strict local optimum.

Definition 2.4 (Active set)**.** *For a feasible point $\bar{z}$ we call an inequality constraint c_i active for some $i \in \mathcal{I}$ if $c_i(\bar{z}) = 0$. Equality constraints are always active in a feasible point. The active set associated to a feasible point $\bar{z}$ is the set of indices of all active constraints*

$$\mathcal{A}(\bar{z}) = \{i | c_i(\bar{z}) = 0\}.$$

Constraint qualifications (CQs) are important to ensure that the feasible set in a neighborhood of a feasible point is well-behaved. There is a whole theory on how to derive CQs for certain types of NLPs [55]. We often assume the set of active constraint gradients to be linearly independent.

Definition 2.5 (Linear independence constraint qualification)**.** *Given the feasible point $\bar{z}$ and its active set $\mathcal{A}(\bar{z})$ according to Definition 2.4. We say that the linear independence constraint qualification (LICQ) holds if the set of active constraint gradients $\{\nabla c_i(\bar{z}), i \in \mathcal{A}(\bar{z})\}$ is linearly independent.*

Definition 2.3 characterizes a locally optimal point. However, this characterization is computationally expensive to check. Therefore we state optimality conditions that can be handled more conveniently from a computational perspective. For this purpose we need the Lagrangian function.

Definition 2.6 (Lagrangian function)**.** *The function $L : \mathbb{R}^{n_z} \times \mathbb{R}^{n_{ce}} \times \mathbb{R}^{n_{ci}} \to \mathbb{R}$,*

$$L(z, \mu, \lambda) := F(z) - \sum_{i \in \mathcal{E}} \lambda_i c_i(z) - \sum_{i \in \mathcal{I}} \mu_i c_i(z)$$

with Lagrangian multipliers $\lambda \in \mathbb{R}^{n_{ce}}$ and $\mu \in \mathbb{R}^{n_{ci}}$ is called the Lagrangian function of the NLP (2.3)*.*

Theorem 2.2 (1st order neccesary conditions of optimality)**.** *Let $z^* \in \mathbb{R}^n$ be a point satisfying LICQ and a local minimum of* (2.3)*. Then there exist $\lambda^* \in \mathrm{R}^{n_{ce}}$ and $\mu^* \in \mathbb{R}^{n_{ci}}$ such that (z^*, λ^*, μ^*) satisfies the following:*

Stationarity:

$$\nabla_z L(z^*, \lambda^*, \mu^*) = \nabla F(z^*) - \sum_{i \in \mathcal{E}} \lambda_i^* \nabla c_i(z^*) - \sum_{i \in \mathcal{I}} \mu_i^* \nabla c_i(z^*) = 0 \tag{2.4}$$

Feasibility:

$$c_i(z^*) = 0 \qquad \text{for } i \in \mathcal{E}, \tag{2.5}$$

$$c_i(z^*) \geq 0 \qquad \text{for } i \in \mathcal{I}, \tag{2.6}$$

Dual feasibility:

$$\mu^* \geq 0 \tag{2.7}$$

Complementarity:

$$\mu_i^* c_i(z^*) = 0, \qquad \text{for } i \in \mathcal{I}. \tag{2.8}$$

Theorem 2.2 is often referred as Karush-Kuhn-Tucker theorem, named after [64] and [71]. A point (z^*, λ^*, μ^*) satisfying the conditions of Theorem 2.2 is called KKT-point. We notice that Theorem 2.2 provides necessary conditions for optimality that can be checked computationally. To distinguish between minimum, maximum and saddlepoints we state 2nd order conditions. Therefore we require the following two sets representing cones for a feasible point $z \in \mathbb{R}^n$ satisfying LICQ:

$$\hat{T}(z) = \{p \in \mathbb{R}^n |\ \nabla c_i(z)p = 0 \text{ for } i \in \mathcal{E}(x), \nabla c_j(z)^\top p = 0 \text{ for } j \in \mathcal{I}(x)\},$$
$$T(z) = \{p \in \mathbb{R}^n |\ \nabla c_i(z)p = 0 \text{ for } i \in \mathcal{E}(x), \nabla c_j(z)^\top p = 0 \text{ for } j \in \mathcal{I}(x) \text{ and } \mu_j > 0\}.$$

Furthermore,

$$\nabla_{zz} L(z^*, \lambda^*, \mu^*) = \nabla^2 F(z^*) - \sum_{i \in \mathcal{E}} \lambda_i^* \nabla^2 c_i(z^*) - \sum_{i \in \mathcal{I}} \mu_i^* \nabla^2 c_i(z^*)$$

is the Hessian matrix of the Lagrangian function with respect to z.

Theorem 2.3 (2nd order neccesary conditions of optimality)**.** *Let $z^* \in \mathbb{R}^n$ be a local minimum satisfying LICQ and let λ^*, μ^* be Lagrange multipliers such that (z^*, λ^*, μ^*) is a KKT point. Then*

$$p^\top \nabla_{zz} L(z^*, \lambda^*, \mu^*) p \geq 0 \quad \text{for all } p \in \hat{T}(z^*).$$

Theorem 2.4 (2nd order sufficient conditions of optimality)**.** *Let $z^* \in \mathbb{R}^n$ be a local minimum satisfying LICQ and let λ^*, μ^* be Lagrange variables such that (z^*, λ^*, μ^*) is a KKT point. If*

$$p^\top \nabla_{zz} L(z^*, \lambda^*, \mu^*) p > 0 \quad \text{for all } p \in T(z^*)$$

holds, then z^* is a strict local minimum.

In the broad field of nonlinear optimization various concepts have been developed to solve (2.3). We mention three algorithmic classes and refer the interested reader to [88].

Interior-point methods

The interior-point or barrier methods are a class of efficient NLP algorithms. We treat the combinatorial complexity introduced by inequality constraints (2.3c) through smoothing and obtain iterates that are strictly in the interior of the set of feasible points. A comprehensive overview is in [112]. Interior-point method tailored to tree-structured NLPs can be found in [60].

Augmented Lagrangian methods

In augmented Lagrangian methods we define a minimizer function combining the Lagrangian and a penalty term depending on the constraints. Some aspects of augmented Lagrangian methods are present in the algorithms for dynamic optimization. Especially in the dual-decomposition based algorithm in Chapter 5 a part of the augmented Lagrangian term becomes relevant.

Sequential quadratic programming

In sequential quadratic programming (SQP) methods we pass the combinatorial complexity of determining the correct active set to a simpler subproblem. This simpler problem is often a convex quadratic programming problem (QP). QP subproblems have some favorable properties and characteristics making SQP one of the most successful methods for nonlinear programming. If we need to repeatedly solve similar NLPs for example in the context of real-time optimization, these features have a strong effect, see Chapter 4.

The Tree QP

The tree QP is a special case of a QP, therefore a finite dimensional scenario tree optimization problem. We want to state it here to finish the section on the optimization perspective.

$$\min_{z_1,\ldots,z_S} \quad \sum_{j\in\mathcal{S}} w_j \left(\frac{1}{2} z_j^T H_j z_j + g_j^T z_j\right) \tag{2.9a}$$

$$\text{s.t.} \qquad x_{j,0} = x_0, \qquad j \in \mathcal{S}, \tag{2.9b}$$

$$x_{j,k+1} = A_{j,k}x_{j,k} + B_{j,k}u_{j,k}, \qquad k \in \mathcal{K},\ j \in \mathcal{S}, \tag{2.9c}$$

$$\underline{x} \le x_{j,k} \le \overline{x}, \qquad k \in \mathcal{K}\cup\{M\},\ j \in \mathcal{S}, \tag{2.9d}$$

$$\underline{u} \le u_{j,k} \le \overline{u}, \qquad k \in \mathcal{K},\ j \in \mathcal{S}, \tag{2.9e}$$

$$E^{j+1} z_{j+1} = C^j z_j, \qquad j \in \mathcal{S}\backslash S. \tag{2.9f}$$

The set $\mathcal{K} = \{0,\dots,M-1\}$ contains the time indices, and the set $\mathcal{S} = \{1,\dots,S\}$ the scenario indices. Corresponding scenario weights are $w_j \geq 0$ for all $j \in \mathcal{S}$. State and control variables are grouped in $z_j^\intercal = [x_{j,0}^\intercal, u_{j,0}^\intercal, \dots, x_{j,M-1}^\intercal, u_{j,M-1}^\intercal, x_{j,M}^\intercal]^\intercal \in \mathbb{R}^{n_z}$ for all $j \in \mathcal{S}$. At this point we require box constraints for states and controls. We exploit the inherent tree structure of (2.9) for fast numerical solution methods in Chapter 5.

2.4 Process Control Perspective: Fast Feedback in Real-Time

Up to now we have introduced the optimization problems in the context of scenario tree NMPC. For the online control of an application such as a chemical plant we do not only solve one optimization problem. It is required to solve a sequence of dynamical optimization problems and with that, new challenges arise. In the moving horizon framework [25, 98] we solve problem (2.2) or a scenario tree optimization problem (2.1) for every instant t of a sampling time grid in order to drive a process according to the desired objective. The problems (2.2) and (2.1) depend parametrically on the initial value x_0 representing the current state of the plant that further depends on the current sampling time t, so $x_0 = x_0(t)$. In ideal NMPC and the special case scenario tree NMPC the optimization problem is assumed to be solved instantaneously. The resulting optimal control $u^*(t)$ is immediately fed back to the process. Therefore ideal NMPC can be regarded as a control feedback law $u^*(t, x(t))$ implicitly given by the solution of optimization problems (2.2) or (2.1). The dynamical evolution of the process is then described by the ODE

$$\dot{x}(t) = f(t, x(t), u^*(t, x(t)), p(t)).$$

The practical implementation of NMPC differs from the idealized NMPC by the following main points. First, we do not solve the infinite-dimensional problem (2.2) or (2.1) at every sampling time, but a finite dimensional approximation, and second, the solution $u^*(t)$ is not obtained and applied to the process immediately. There is a delay between receiving the current process state $x_0(t)$ and feeding the control back to the process, determined mainly by the cost of solving the discretized optimization problem. In practice these circumstances affect the performance and stability of the controller [38].

For fast feedback we need to minimize the delay by fast NMPC schemes. We emphasize that fast NMPC is not just NMPC with a fast solver, it rather requires sophisticated methods that we describe in Chapter 4. One algorithmic concept to reduce the time delay between retrieving the process state and the control feedback is the Real-Time Iteration scheme [28] that we focus on in Section 4.2. Major ingredients are the initial value embedding idea and splitting the computation into phases. A further contribution to fast NMPC is the Multi-Level-Iteration scheme [16]. We describe it in Section 4.3. The fast feedback

NMPC methods tackle the model-plant mismatch and disturbances. Further robustification against uncertainty in the process is achieved by stochastic approaches. We continue with the stochastic nature of scenario tree NMPC in the next section.

2.5 Stochastic Perspective: Discrete Tree Process

At this point we focus on the stochastic nature of the uncertainty. In the field of optimization under uncertainty, especially in stochastic finance applications and optimal portfolio selection problems [30, 51, 58], the evolution of a discrete stochastic process is often illustrated as a scenario tree. We adapt this interpretation and regard the uncertain parameter process as a finite stochastic process. In this section we consider two processes and define mathematically rigorously a central property of the tree process. Uncertainty is present in the underlying parameter process. In traditional theory, which concentrates mainly on stability and tractability assertions for robust optimal control, there are two basic approaches to incorporate parameter uncertainty into optimization problems. While in robust optimization the uncertainty model is basically deterministic and set-based, the second perspective builds upon a probabilistic description of the uncertainty. The thesis [61] extensively investigates the effects from the robust and probabilistic optimization perspective. A main focus there lies on optimal control problems driven by stochastic differential equations. The scenario tree approach is located between the two basic approaches. In this section we aim to clarify the stochastic properties of the scenario tree uncertainty model. For this purpose we require basic stochastic definitions. In the following we give a condensed overview. For a detailed introduction and further background in the field we refer to the textbooks [31, 68].

Definition 2.7 (Sigma-Algebra)**.** *Let Ω be a non-empty set and $\mathcal{P}(\Omega)$ its power set. A subset $\mathcal{F} \subset \mathcal{P}(\Omega)$ is a sigma-algebra if it satisfies the following properties:*

1. *$\mathcal{F}$ contains the universal set: $\Omega \in \mathcal{F}$.*
2. *$\mathcal{F}$ is closed under complementation: If $A \in \mathcal{F}$, then $\Omega \setminus A \in \mathcal{F}$.*
3. *$\mathcal{F}$ is closed under countable unions: If $(A_i)_{i\in\mathbb{N}}$ with $A_i \in \mathcal{F}$ for all i, then $\bigcup_{i\in\mathbb{N}} A_i \in \mathcal{F}$.*

We call the elements of $\mathcal{F}$ measurable sets and $(\Omega, \mathcal{F})$ a measurable space.

Definition 2.8 (Generated Sigma-Algebra)**.** *Let $\mathcal{G}$ be any subset of $\mathcal{P}(\Omega)$. Then the smallest sigma-algebra containing $\mathcal{G}$, $\sigma(\mathcal{G}) = \bigcap\{\mathcal{F} | \mathcal{F}$ is sigma-algebra on $\Omega, \mathcal{G} \subset \mathcal{F}\}$, is called the sigma-algebra generated by $\mathcal{G}$.*

Definition 2.9 (Measurable Function)**.** *Let $(\Omega, \mathcal{F})$ and $(\Omega', \mathcal{F}')$ be measurable spaces. A function $f : \Omega \to \Omega'$ is a measurable function if for the preimages it holds that*

$$f^{-1}(A') := \{x \in \Omega |\ f(x) \in A'\} \in \Omega, \quad \text{for all } A' \in \mathcal{F}'.$$

Definition 2.10 (Sigma-algebra generated by a function)**.** *Let Ω be a non-empty set, $(\Omega', \mathcal{F}')$ a measurable space and $f : \Omega \to \Omega'$ a function. Then the sigma-algebra on Ω generated by the function f, denoted as $\sigma(f)$, is the collection of all preimages of the sets in $\mathcal{F}'$: $\sigma(f) := \{f^{-1}(A') |\ A' \in \mathcal{F}'\}$. It is the smallest sigma-algebra such that f is measurable.*

Definition 2.11 (Measure)**.** *A measure μ on a measurable space is a non-negative function $\mu : \mathcal{F} \to [0, \infty[$ such that*

1. *$\mu(\emptyset) = 0$,*
2. *for $(A_i)_{i\in\mathbb{N}}$ with $A_i \cap A_j = \emptyset$ for all $i \neq j \in \mathbb{N}$ the equality*

$$\mu\left(\bigcup_{i=1}^{\infty} A_i\right) = \sum_{i=1}^{\infty} \mu(A_i)$$

holds.

The triple $(\Omega, \mathcal{F}, \mu)$ is called measure space.
If additionally $\mu(\Omega) = 1$, we call $(\Omega, \mathcal{F}, \mu)$ a probability space.

Definition 2.12 (Random Variable)**.** *Let $(\Omega, \mathcal{F}, P)$ be a probability space and $(\Omega', \mathcal{F}')$ a measurable space. A random variable X is a measurable function $X : (\Omega, \mathcal{F}, P) \to (\Omega', \mathcal{F}')$. Every random variable X induces a probability measure μ_X on $(\Omega', \mathcal{F}')$ through $\mu_X(A') := P(X^{-1}(A'))$ for all $A' \in \mathcal{F}'$.*

Definition 2.13 (Stochastic Process)**.** *Let $I \subset \mathbb{R}$. A collection of random variables $X = (X_t, t \in I)$ on a probability space $(\Omega, \mathcal{F}, P)$ with values in a measurable space $(\Omega', \mathcal{F}')$ is a stochastic process with time domain I and state space Ω'.*
If I is discrete or even finite, the process X is named discrete or finite stochastic process. In case of $(\Omega', \mathcal{F}') = (\mathbb{R}, \mathcal{B}(\mathbb{R}))$ ($\mathcal{B}(\mathbb{R})$ the Borel sigma-algebra), X is a real-valued stochastic process.

Definition 2.14 (Filtration)**.** *A collection $\mathbb{F} = (\mathcal{F}_t, t \in I)$ of sigma-algebras with $\mathcal{F}_t \subset \mathcal{F}$ for all $t \in I$ is a filtration, if $\mathcal{F}_s \subset \mathcal{F}_t$ for all $s, t \in I$ with $s \leq t$.*

Definition 2.15 (Adapted to a Filtration)**.** *A stochastic process $X = (X_t, t \in I)$ is adapted to the filtration $\mathbb{F} = (\mathcal{F}_t, t \in I)$ if X_t is measurable with respect to $\mathcal{F}_t$ for all $t \in I$.*

At this point we return to the scenario tree approach. We assume that the parameters representing the uncertainty are constant on every stage. In the following we consider two stochastic processes, the uncertain parameter process and the scenario tree process.

Definition 2.16 (Discrete uncertain parameter process)**.** *The underlying parameter process is the collection of random variables $X = (X_t, t \in \{0, \dots M\}), M \in \mathbb{N}$, $X_t : (\Omega, \mathcal{F}, P) \to (\mathbb{R}, \mathcal{B}(\mathbb{R}))$ for all t.*

Then we define the discrete scenario tree process on the finite subset $E \subset \Omega$, as we have to choose a finite number of parameter realizations.

Definition 2.17 (Scenario tree process)**.** *The scenario tree process is the collection of random variables $Y = (Y_t, t \in \{0, \dots, M\}), M \in \mathbb{N}$, $Y_t : (E, \mathcal{E}, P') \to (\mathbb{R}, \mathcal{B}(\mathbb{R}))$ with $E \subset \Omega$ a finite set, $\mathcal{E} = \sigma(E)$ and P' a probability measure on $\mathcal{E}$.*

Remark 2.1. *The measure P' of the scenario process is chosen either as branch probabilities $P'(Y_t = w_i) = p_i$ for all realizations $w_i \in E$, or transition probabilities depending on the previous realization $P'(Y_{t+1} = w_j | Y_t = w_i) = p_{ij}$ for $w_i, w_j \in E$.*

We now define the most prominent term used in the context of scenario tree NMPC. The definition is mathematically rigorous in the sense that it relates a stochastic process to a filtration containing the probabilistic information of another stochastic process.

Definition 2.18 (Non-anticipativity)**.** *A stochastic process $Y = (Y_t, t \in J)$ is called non-anticipative with respect to the stochastic process $X = (X_t, t \in I)$ if Y is adapted to the filtration $\sigma(X_t, t \in I)$.*

Remark 2.2.
The definition of non-anticipativity is independent of the measures of the two stochastic processes.

Remark 2.3.
Non-anticipativity means that we can observe the realization of the stochastic process Y at times until t if we can observe process X until time t. It does not mean that we can predict the process Y. We want to clearly distiguish between the term non-anticipativity and the term of a predictable stochastic process.
The process $X = (X_n, n \in \mathbb{N}_0)$ is predictable with respect to the Filtration $\mathbb{F} = (\mathcal{F}_n, n \in \mathbb{N}_0)$ if X_0 is constant and it holds for all $n \in \mathbb{N}$ that X_n is $\mathcal{F}_{n-1}$ measurable.

We continue with the central theorem of this section.

Theorem 2.5. *The scenario tree process Y is non-anticipative with respect to the underlying process X.*

Proof.
We recall from the definition of non-anticipativity that we have to show the following statement: The process Y is is adapted to the filtration $\sigma(X_t, t \in I)$ of the underlying parameter process X.

By definition of the stochastic tree process Y, each Y_t is $\mathcal{E}$-measurable for all $t \in \{0, \dots, M\}$. The sample space E of the tree process is a finite subset of Ω, therefore $\mathcal{E} \subset \mathcal{F}$. As both processes X and Y are defined for $t \in \{0, \dots M\}$, we deduce the statement that Y_t is $\mathcal{F}$-measurable for all $t \in \{0, \dots, M\}$.

We now focus on the filtrations $\mathcal{E}_t := \sigma(Y_s, 0 \le s \le t)$ for $s,t \in \{0 \dots M\}$. The random variables Y_t are measurable with respect to the filtration $\mathcal{E}_t$ for all $t \in \{0, \dots, M\}$. Because Y_t is $\mathcal{F}$-measurable for all $t \in \{0, \dots, M\}$, we conclude that Y_t is also measurable with respect to the filtration $\mathcal{F}_t := \sigma(X_s, 0 \le s \le t)$ for all $s,t \in \{0, \dots, M\}$. Then by the definition of adaptivity, Def. 2.15, the scenario tree process Y is adapted to $\sigma(X_t, t \in \{0, \dots, M\})$. □

We close this section on the stochastic perspective with two examples illustrating the concept on non-anticipativity.

Example 2.1 (Anticipativity)**.** *Consider $\Omega = \{A, B\}$ and a discrete constant process X, with $X_t(\omega) = 10$ for all $t \in \mathbb{N}$, $\omega \in \Omega$, and the process Y with*

$$Y_t(\omega) = \begin{cases} 0 & \text{for } \omega = A \\ 1 & \text{for } \omega = B \end{cases} \quad \text{for all } t \in \mathbb{N}.$$

Because X is constant, the generated sigma-algebra of all random variables X_t is trivial: $\sigma(X_t) = \{\emptyset, \Omega\}$ for all $t \in \mathbb{N}$. Hence, the filtrations $\mathcal{E}_t = \mathcal{E} := \sigma(X_t)$ stay constant for all $t \in \mathbb{N}$. The generated sigma-algebra is $\sigma(Y) = \{\emptyset, \{A\}, \{B\}, \Omega\}$, and the filtration is $\mathcal{F}_1 = \mathcal{F} = \sigma(Y)$.

For completeness we specify the probability measures and refer to Remark 2.2 that non-anticipativity is measure-independent. Process X is equipped with measure P_1 on $\mathcal{F}$ such that $P_1(X_t = 10) = 1$. For process Y we can define a measure P_2 on $\mathcal{E}$ such that

$$P_2(\omega) = \begin{cases} 0.6 & \text{for } \omega = A \\ 0.4 & \text{for } \omega = B. \end{cases}$$

The event $(\omega = B)$ is an element of the filtration $\mathcal{F}_1$, because it is possible to observe $Y_1 = 1$. From the process X we can only observe that it has constant value 10. We do not know the elementary random event that is happening at a time t. Regarding non-anticipativity we state formally that the event $(\omega = B)$ is not contained in $\mathcal{E}$, therefore Y_1 is not measurable with respect to $\mathcal{E}$ and Y is not adapted to $\sigma(X_1)$. Hence, the process Y on $(\Omega, \mathcal{E}, P_2)$ is anticipative (not non-anticipative) with respect to the process X on $(\Omega, \mathcal{F}, P_1)$.

Example 2.2 (Events of the tree process)**.** *Let X be the uncertain parameter process from Def. 2.16 and Y the tree process from Def. 2.17. We now consider a few events to explain the non-anticipativity we have proven above.*

In the following we need the filtrations of the underlying process X,

$$\mathcal{F}_t = \sigma(X_s, 0 \leq s \leq t), s,t \in \{0,\dots,M\}$$

and the filtrations of the tree process Y,

$$\mathcal{E}_t = \sigma(Y_s, 0 \leq s \leq t), s,t \in \{0,\dots,M\}.$$

If we look at the first stage, then the event $(Y_1 = p_1)$ is contained in $\mathcal{E}_1 \subset \mathcal{F}_1$. The events $(Y_1 = p_1, Y_2 = p_1)$ and $(Y_1 = p_1, Y_2 = p_2)$, that represent scenarios of length two, are not in $\mathcal{F}_1$. But the considered events $(Y_1 = p_1, Y_2 = p_1)$ and $(Y_1 = p_1, Y_2 = p_2)$ are elements of the filtrations $\mathcal{E}_2 \subset \mathcal{F}_2$. Therefore those two events cannot be distinguished until $t = 2$. Considering the two events as two possible scenarios, we have to ensure later that they coincide until time $t = 2$. We shall reflect this property in the so-called non-anticipativity control constraints for the branch-wise scenario tree formulation to get scenario tree formulation that is consistent with the underlying stochastics.

Up to now, we have introduced non-anticipativity and illustrated the concept in our framework. There are two chapters in this thesis that especially rely on the stochastic nature of the scenario tree approach. In Chapter 6 we present an approach how to generate scenario trees based on quadrature rules and in Chapter 7 we develop an efficient tree reduction algorithm for a process with Markovian properties.

2.6 Summary

The present chapter has introduced the mathematical formalism of the scenario tree approach and classified the arising problems in different fields of applied mathematics. From a dynamical systems perspective we have formulated an optimal control problem with scenario tree structure. In the direct optimization approach the optimal control problem is discretized. We have seen from the optimization perspective that we arrive at a specific structured finite-dimensional constrained scenario tree optimization problem. The process control perspective section has emphasized that fast feedback NMPC methods like the Real-Time Iteration scheme and the Multi-Level-Iteration scheme are required to minimize the feedback delay. In the last section we have defined the scenario tree process from a stochastics perspective and introduced the principle of non-anticipativity. This chapter has set the perspectives and directions to approach scenario tree NMPC. In the following chapters we always start discussing the problem from a specific viewpoint to derive the contributions of this thesis.

Chapter 3

Discretization Structure Exploitation

Numerical algorithms exploiting an inherent problem structure are indispensable for an efficient solution process. In case of optimal control, direct methods such as Direct Multiple Shooting [17] or Direct Collocation [109, 4, 11] have proven to be versatile and efficient whereas the initial value problems in Direct Single Shooting and the boundary value problems resulting from indirect methods suffer from the fact that the trajectories of nonlinear ODE systems do not necessarily exist on a long time period. If they exist, the high nonlinearity can cause a blowup of the solution trajectories such that boundary conditions cannot be satisfied. In this chapter we focus on the direct multiple shooting method, its structure and tailored methods to solve the resulting finite-dimensional optimization problems.

3.1 The Direct Multiple Shooting Method

The foundations of multiple shooting are improved solution methods for boundary value problems [89, 18, 12]. In the thesis [90] supervised by Hans Georg Bock and the contribution [17] the multiple shooting idea has been extended to optimal control problems. The Direct Multiple Shooting Method combines several advantages of other direct approaches. For instance, the state discretization allows to incorporate a priori knowledge of the process for initialization. The underlying dynamics can be computed by fast and also adaptive state-of-the art integrators because we solve initial value problems (IVPs). Moreover, the Direct Multiple Shooting method can cope with highly nonlinear dynamics because we solve IVPs on subintervals of the solution horizon and do not require an all-at-once solution over the whole horizon. For an explanation of the method let us state again the optimal control

problem (2.2) in the following. We assume that all functions are sufficiently smooth.

$$\min_{x,u} \quad \int_{t_0}^{t_f} \Phi(t,x(t),u(t),p(t))\mathrm{d}t \tag{3.1a}$$

$$\text{s.t.} \qquad \dot{x}(t) = f(x(t),u(t),p(t)), \qquad t \in [t_0,t_f], \tag{3.1b}$$

$$0 = x(t_0) - x_0, \tag{3.1c}$$

$$0 \leq r(t,x(t),u(t),p(t)), \qquad t \in [t_0,t_f]. \tag{3.1d}$$

The problem (3.1) is a constrained infinite-dimensional optimization problem on a time horizon $I := [t_0,t_f] \subset \mathbb{R}$ with state variables $x : I \to \mathbb{R}^{n_x}$ and control variables $u : I \to \mathbb{R}^{n_u}$. The dynamical evolution of the states is described by a system of ordinary diffential equations (ODEs) (3.1b) with right hand side $f : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_p} \to \mathbb{R}^{n_x}$ and initial value $x_0 \in \mathbb{R}^{n_x}$. For the ODE right hand side we assume continuity and Lipschitz continuity in x such that local solutions to IVPs are guaranteed by Theorem 2.1. We aim to minimize a performance criterion (3.1a) while satisfying the dynamical evolution constraints (3.1b) with initial value (3.1c) and path constraints (3.1d) with $r : \mathbb{R} \times \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_p} \to \mathbb{R}^{n_r}$. The system behavior is affected by uncertain parameters $p \in \mathbb{R}^{n_p}$. The solution trajectories x and u have infinitely many degrees of freedom. Hence, for numerical computations we must discretize the optimal control problem. After the discretization procedure we arrive at a structured NLP.

Discretization of the optimal control problem

First, the time horizon I is partitioned into a non-necessarily equidistant shooting grid $\{t_k\}$ with

$$t_0 < t_1 < \ldots < t_M = t_f.$$

On each interval $I_k := [t_k,t_{k+1}],\ 0 \leq k \leq M-1$ we use a control discretization

$$u_k(t) = \psi_k(t,q_k)$$

with basis functions $\psi_k : I_k \times \mathbb{R}^{n_{q_k}} \to \mathbb{R}^{n_u}$. The basis functions have local support such that we get separability of the discretized problem. A common and easy variant is the piecewise constant control parameterization. For every I_k we choose $\psi_k = q_k$ with $q_k \in \mathbb{R}^{n_u}$. In other control discretization cases we may require continuity of the control. It can be achieved by adding the constraints

$$\psi_k(t_{k+1},q_k) - \psi_{k+1}(t_{k+1},q_{k+1}) = 0 \quad \text{for } k = 0,\ldots,M-1.$$

These constraints are linear and affect only neighboring controls. Therefore the separability of the discretized problem will not get lost by adding continuity constraints.

We continue with the state discretization. The continuity assumptions on the ODE right hand side of the dynamical evolution constraint (3.1b) ensure the existence of a unique solution in the neighborhood of $(t,x,u) \in I \times \mathbb{R}^{n_x} \times \mathbb{R}^{n_u}$ by Theorem 2.1. Therefore we can write the ODE solution as function of initial value x_0 and control q, namely $x(t;x_0,q)$. The fundamental idea of direct multiple shooting is the splitting of the ODE trajectory computation. We do not integrate the ODE on the whole interval I at once. Instead we divide the integration task into subproblems. We aim to compute IVPs on the intervals I_k for $k = 0,\dots,M-1$. Therefore we introduce the variables $s_k \in \mathbb{R}^{n_x},\ k = 0,\dots,M$ and solve local IVPs of the form

$$\dot{x}(t) = f(t,x(t),\psi_k(t,q_k)) \qquad \text{for } t \in I_k \tag{3.2a}$$

$$x(t_k) = s_k. \tag{3.2b}$$

The solution of (3.2) is denoted by $x^k(t;s_k,q_k)$. Then we concatenate the IVP solutions on the whole interval I to the function

$$x(t) = \begin{cases} x^k(t;s_k,q_k) & \text{for } t \in [t_k,t_{k+1}) \\ s_M & \text{for } t = t_M. \end{cases}$$

We have to ensure continuity of the concatenated function on I, because this is a fundamental property of the ODE solution, which yields matching conditions of the form

$$x^k(t_{k+1};s_k,q_k) - s_{k+1} = 0 \qquad \text{for } j = 0,\dots,M-1$$

as additional constraints. Now the evaluation of $x(t)$ does not require an integration over I at once. Only integration over the subintervals I_k must be performed. The shorter integration intervals reduce the error propagation and yield a better convergence behavior of the multiple shooting method, especially compared to single shooting. In addition to that, the multiple shooting formulation allows a parallel integration of subintervals, yielding computation time reduction on multicore systems.

Finally we discretize the continuous path constraints (3.1d) and require

$$r(t_k,x(t_k;s_k,q),\psi_k(t_k,q_k)) \geq 0 \text{ for } k = 0,\dots,M-1.$$

We define the discretized constraint function $r_k : \mathbb{R} \times \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_r}$ with

$$r_k(t_k,s_k,q_k) := r(t_k,x(t_k;s_k,q),\psi_k(t_k,q_k)) \geq 0 \text{ for } k = 0,\dots,M.$$

Alternatives to treat the path constraints are described in [93].

At this point we state the discretized optimal control problem that is a nonlinear optimization problem as in Definition 2.3. The structured nonlinear direct multiple shooting optimization problem then reads

$$\min_{s,q} \phi(s,q) := \sum_{k=0}^{M-1} \int_{t_k}^{t_{k+1}} \Phi(t, x^k(t; s_k, q_k), \psi_k(t, q_k)) \mathrm{d}t \tag{3.3a}$$

$$\text{s.t.} \quad 0 = x^k(t_{k+1}; s_k, q_k) - s_{k+1} \qquad \text{for } k = 0, \ldots, M-1 \tag{3.3b}$$

$$0 = x(t_0) - x_0 \tag{3.3c}$$

$$0 \leq r_k(t_k, s_k, q_k) \qquad \text{for } k = 0, \ldots, M. \tag{3.3d}$$

The inherent multiple shooting structure of the problem (3.3) can be exploited efficiently due to the separability of the objective function (3.3a) and the constraints with respect to the optimization variables s and q. Only the matching conditions (3.3b) couple unknowns of neighboring shooting nodes linearly. This is the reason why the Hessian of the Lagrangian of the NLP (3.3) exhibits block-diagonal structure.

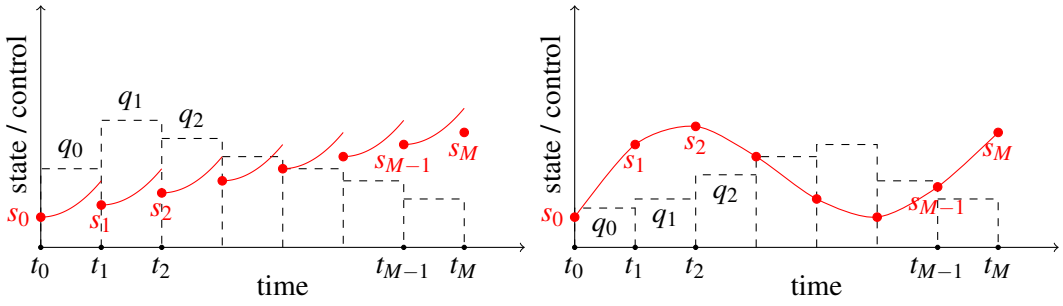

Figure 3.1: Illustration of state and control variables of the direct multiple shooting discretization applied to an optimal control problem. On the left the shooting nodes are initialized linearly and the trajectory violates the matching conditions. On the right the solution of the NLP has converged with feasible matching conditions. Inspired by [65, 41, 101].

We conclude with the remark that the multiple shooting method only solves the discrete approximation (3.3) of the continuous problem (3.1). Investigations on approximation properties and asymptotic behavior can be found in [45].

Control Move Regularization

At this point we introduce the Control Move Regularization (CMR). We formulate the CMR as an addition to the NLP (3.3). The NLP with CMR reads

$$\min_{s,q} \ \phi(s,q) \tag{3.4a}$$

$$\text{s.t.} \quad 0 = x^k(t_{k+1}; s_k, q_k) - s_{k+1} \qquad \text{for } k = 0,\dots,M-1 \tag{3.4b}$$

$$0 = x(t_0) - x_0 \tag{3.4c}$$

$$0 \le r_k(t_k, s_k, q_k) \qquad \text{for } k = 0,\dots,M, \tag{3.4d}$$

$$0 \le \alpha_{\text{CMR}} - (q_{k+1} - q_k) \qquad \text{for } k = 0,\dots,M-1, \tag{3.4e}$$

$$0 \le \alpha_{\text{CMR}} - (q_k - q_{k+1}) \qquad \text{for } k = 0,\dots,M-1. \tag{3.4f}$$

The constraints (3.4e) and (3.4f) state that the differences of subsequent control variables are less than or equal to the CMR parameter $\alpha_{\text{CMR}} \in \mathbb{R}^{n_u}$. All components of α_{CMR} are strictly positive. It is also possible to add an objective term $\phi_{\text{CMR}}(q)$ to (3.4a) penalizing the difference of control variables. For our numerical results in Chapter 9 it is sufficient to use the CMR parameter α_{CMR}. Engineers use CMR extensively as a means to tune NMPC controllers. For instance, let the coolant temperature to control a reactor be 80°C. Assuming that there is a fine sampling grid, it is impossible to apply 60°C at the next sampling point to the plant. Therefore the constraint bounding the difference of subsequent coolant temperature variables is added to the discretized prediction problem in NMPC. We remark that the CMR couples neighboring stages only linearly and that the CMR parameter depends on the discretization grid.

3.2 Structure Exploiting Sequential Quadratic Programming

We employ Sequential Quadratic Programming (SQP) techniques [88] and solve the constrained NLP (3.3) with multiple shooting structure as described in [77, 78]. For notational convenience we write the NLP (3.3) in the more generic form

$$\min_{z} \ \phi(z) \tag{3.5a}$$

$$\text{s.t.} \quad 0 = d(z) + \Lambda x_0 \qquad | \ \lambda \tag{3.5b}$$

$$0 \le h(z) + \overline{h} \qquad | \ \mu \tag{3.5c}$$

with $z = (s_0, q_0, \dots, s_{M-1}, q_{M-1}, s_M)$ and $\Lambda = (\mathbb{I}, 0, 0, \dots) \in \mathbb{R}^{n_x \times (Mn_x + (M-1)n_q)}$. The line (3.5b) comprises the initial value and dynamical evolution constraints. The line (3.5c) comprises all further constraints. Furthermore, the symbols $| \ \lambda$ and $| \ \mu$ on the right represent

the dual variables corresponding to the constraints. The concept is introduced in Definition (2.6) of the Lagrangian function. We emphasize that (3.5) is parametric in x_0.

The standard full-step SQP technique is a Newton-type iterative method. We start with an initial guess of primal and dual NLP variables (z_0, λ_0, μ_0). Within every SQP iteration we compute the primal-dual solution of the structured QP

$$\min_{\Delta z_i} \quad \frac{1}{2}\Delta z_i^\top B_i \Delta z_i + b_i^\top \Delta z_i \tag{3.6a}$$

$$\text{s.t.} \quad 0 = D_i \Delta z_i + d(z_i) + \Lambda x_0 \qquad | \ \lambda_{QP} \tag{3.6b}$$

$$0 \leq H_i \Delta z_i + h(z_i) + \overline{h} \qquad | \ \mu_{QP} \tag{3.6c}$$

to obtain $(\Delta z_i, \lambda_{QP}, \mu_{QP})$. The QP (3.6) is a local quadratic model of the multiple shooting NLP (3.5) at z_i. The matrix B_i denotes an approximation of the Hessian of the Lagrangian of the NLP (3.5), b_i is the objective gradient, D_i is the Jacobian of the function d and H_i is the Jacobian of the function h.

A full step SQP iteration is performed by updating the variables in every iteration according to

$$z_{i+1} = z_i + \Delta z_i, \quad \lambda_{i+1} = \lambda_{QP}, \quad \mu_{i+1} = \mu_{QP}.$$

We use the principle of Internal Numerical Differentiation (IND) to solve the IVPs and to compute the sensitivities. Regarding the whole topic of numerical integration and sensitivity calculation we refer to [13, 2, 1, 7]. For SQP variants, especially the choice of B_i, see [62, 63]. For globalization strategies such as line search SQP or trust-region SQP methods we refer to the textbook [88].

Various structural features as the separable Lagrangian, the block diagonal Hessian, and the block structure of the Jacobians of the matching conditions can be extensively exploited if the control functions, constraints, and multiple shooting variables are discretized on a common grid.

3.3 Condensing

In this section we focus on the multiple shooting structure exploiting method called Condensing. It has been described already in [14, 17, 90, 76] and yields small dense QPs. Solving the tree QP benefits from Condensing because our branchwise formulation preserves the multiple shooting structure. The basic idea of Condensing is a splitting of the optimization variables $z \in \mathbb{R}^{n_z}$ into $(z_1, z_2) \in \mathbb{R}^{n_1+n_2}$ and a structure exploiting elimination of z_2. After Condensing the resulting QP is of much smaller size if n_2 dominates n_1 and can be solved with a state-of-the-art dense QP solver. At first we present the interpretation of Condensing as a partial nullspace approach as described in [92].

We consider a structured QP in the optimization variables $z = (z_1, z_2) \in \mathbb{R}^{n_1+n_2}$ of the form

$$\min_{(z_1,z_2)\in\mathbb{R}^{n_1+n_2}} \quad \frac{1}{2}\begin{bmatrix} z_1 \\ z_2 \end{bmatrix}^\top \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}^\top \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} \tag{3.7a}$$

$$s.t. \quad G_1 z_1 + G_2 z_2 = g, \tag{3.7b}$$

$$D_1 z_1 + D_2 z_2 = d, \tag{3.7c}$$

$$H_1 z_1 + H_2 z_2 \geq h, \tag{3.7d}$$

with matrices $G_1 \in \mathbb{R}^{n_1\times n_1}$, $G_2 \in \mathbb{R}^{n_1\times n_2}$, matrices $D_1 \in \mathbb{R}^{m_2\times n_1}$, $D_2 \in \mathbb{R}^{m_2\times n_2}$ and matrices $H_1 \in \mathbb{R}^{m_3\times n_1}$, $H_2 \in \mathbb{R}^{m_3\times n_2}$.

QP (3.7) is formulated as in [92, 101]. The following theorem from [92, 101] describes how to eliminate variables z_1 that are coupled with variables z_2 by the equality constraints (3.7b). It is known as partial nullspace approach and has been discussed in similar form in [8].

Theorem 3.1. *Let G_1 from QP* (3.7) *be invertible. We introduce the notation*

$$B = \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}, \qquad b = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix},$$

$$Z = \begin{bmatrix} -G_1^{-1}G_2 \\ \mathbb{I} \end{bmatrix}, \qquad B' = Z^\top B Z,$$

$$g' = G_1^{-1} g, \qquad b' = B_{21}g' + b_2 - G_2^\top G_1^{-\top}(B_{11}g' + b_1),$$

$$d' = d - D_1 g', \qquad D' = D_2 - D_1 G_1^{-1} G_2,$$

$$h' = h - H_1 g', \qquad H' = H_2 - H_1 G_1^{-1} G_2.$$

Furthermore let $(z_2^*, \lambda_2^*, \mu^*) \in \mathbb{R}^{n_2+m_2+m_3}$ be a solution of the QP

$$\min_{z_2\in\mathbb{R}^{n_2}} \quad \frac{1}{2} z_2^\top B' z_2 + b'^\top z_2 \tag{3.8}$$

$$s.t. \quad D' z_2 = d',$$

$$H' z_2 \geq h'.$$

Then $(z^*, \lambda^*, \mu^*) := (z_1^*, z_2^*, \lambda_1^*, \lambda_2^*, \mu^*)$ with

$$z_1^* = G_1^{-1}(g - G_2 z_2^*) \text{ and} \tag{3.9a}$$

$$\lambda_1^* = G_1^{-\top}((B_{12} - B_{11}G_1^{-1}G_2)z_2^* + B_{11}g' + b_1 - D_1^\top \lambda_2^* - H_1^\top \mu^*) \tag{3.9b}$$

is a solution of the QP (3.7).

Theorem 3.1 does not specify how to split the optimization variables $z = (z_1, z_2)$. This allows for tailoring the approach to the specific QP structure. We only have to ensure that G_1 is well-conditioned.

In our case the QP originates from the direct multiple shooting discretization and a standard SQP method. We apply Theorem 3.1 to QP (3.6) and focus on the constraint (3.6b). First, we drop Δ and i from the SQP formulation. We call the matrix representing the Jacobian of the matching conditions in this section G. It exhibits block-sparse structure,

$$G = \begin{bmatrix} G_s^0 & G_q^0 & -\mathbb{I} & & & & & \\ & & G_s^1 & G_q^1 & -\mathbb{I} & & & \\ & & & & \ddots & \ddots & \ddots & \\ & & & & & & G_s^{M-1} & G_q^{M-1} & -\mathbb{I} \end{bmatrix}. \tag{3.10}$$

The matrix blocks $G_{s,q}^k$ represent the sensitivities of the IVP solutions on the kth shooting interval for $k = 0, \ldots, M-1$,

$$G_s^k = \frac{\partial}{\partial s} x^k(t_{k+1}; s_k, q_k), \qquad G_q^k = \frac{\partial}{\partial q} x^k(t_{k+1}; s_k, q_k). \tag{3.11}$$

In the following we discuss two established Condensing variants.

Block Gauss Condensing

The first variant is named Block Gauss Condensing. We arrange the variables $z = (z_1, z_2)$ as follows

$$z_1 = (s_1, \ldots, s_M),$$
$$z_2 = (s_0, q_0, q_1, \ldots, q_{M-1}).$$

For the critical constraint matrix G the first variant yields a permutation of its columns. We arrive at the structure

$$G_1 = \begin{bmatrix} -\mathbb{I} & & \\ G_s^1 & \ddots & \\ & \ddots & -\mathbb{I} \\ & & G_s^{M-1} & -\mathbb{I} \end{bmatrix} \quad \text{and} \quad G_2 = \begin{bmatrix} G_s^0 & G_q^0 & & \\ & & G_q^1 & \\ & & & \ddots \\ & & & & G_q^{M-1} \end{bmatrix}.$$

The matrix G_1 is lower-triangular with negative identity matrices on the diagonal. Therefore G_1 is invertible and we can apply Theorem 3.1. As we know the structure of G_1, we can exploit it for the matrix-vector operations with G_1^{-1}. Of course we do not explicitly invert G_1 but perform Block Gauss eliminations that are eponymous for the Condensing variant. This variant has good stability properties for stable systems yielding $\|G_s^i\|_2 < 1$ for all $i \in \{0, \ldots, M-1\}$. In some applications Block Gauss Condensing has shown poor stability properties, an example can be found in [34].

Stable QR Condensing

The origins of the second variant Stable QR Condensing can be found already in [14, 46]. This variant is tailored for QPs that originate from the Direct Multiple Shooting discretization of optimal control problems with unstable ODEs. The QR decomposition is an algorithm to express a matrix $A \in \mathrm{R}^{m\times n}$, $m \geq n$, as product $A = QR$ with an upper triangular matrix $R \in \mathrm{R}^{m\times n}$ and an orthogonal matrix $Q \in \mathrm{R}^{m\times m}$. Variants of QR decompositions employ Householder reflections or Givens rotations. We refer to numerical linear algebra textbooks as [108, 24] and remark that for $\tilde{A} \in \mathrm{R}^{m\times n}$, $m \leq n$, the so-called LQ-decomposition is a QR-decomposition of $\tilde{A}^\top$ with $L := R^\top$. The core of stable QR Condensing is Algorithm 1 for the subblocks of the critical constraint matrix G. After the decomposition according to

Algorithm 1 Decomposition of the constraint matrix

Input: G_s^j, G_q^j for $j = 0,\ldots,M-1$

Output: Q_j, L_j for $j = 0,\ldots,M-1$ and $\tilde{G}_s^j$, $\tilde{G}_q^j$ for $j = 0,\ldots,M-2$

1: $(L_0, Q_0) =$ LQ-decomposition of $\begin{bmatrix} G_s^0 & G_q^0 & -\mathbb{I} \end{bmatrix}$
2: **for** $j = 0,\ldots,M-2$ **do**
3: $\quad \begin{bmatrix} \tilde{G}_s^j & \tilde{G}_q^j & \hat{G}_s^{j+1} \end{bmatrix} = \begin{bmatrix} 0 & 0 & G_s^{j+1} \end{bmatrix} Q_j$
4: $\quad (L_{j+1}, Q_{j+1}) =$ LQ-decomposition of $\begin{bmatrix} \hat{G}_s^{j+1} & G_q^{j+1} & -\mathbb{I} \end{bmatrix}$
5: **end for**

Algorithm 1 we assemble the matrices G_1 and G_2 as

$$G_1 = \begin{bmatrix} L_0 & & & & \\ \tilde{G}_s^0 & L_1 & & & \\ & \tilde{G}_s^1 & \ddots & & \\ & & \ddots & L_{M-2} & \\ & & & \tilde{G}_s^{M-2} & L_{M-1} \end{bmatrix}, \quad G_2 = \begin{bmatrix} 0 & & & \\ \tilde{G}_q^0 & & & \\ & \tilde{G}_q^1 & & \\ & & \ddots & \\ & & & \tilde{G}_q^{M-2} \end{bmatrix},$$

and apply Theorem 3.1. For a detailed description we refer to the thesis [101] and remark that due to the stability properties of the QR decomposition the above Condensing variant is suitable for QPs originating from optimization problems with highly unstable dynamical systems or dynamical systems with stable and unstable components.

Further Condensing variants can be found for example in [3, 66, 105]. These algorithms have quadratic or even linear runtime in the horizon length.

After Condensing the QP with multiple shooting structure we solve it with state-of-the-art dense QP solvers. The size of the QP decreases significantly if the dimension n_2 of $z = (z_1, z_2) \in \mathbb{R}^{n_1+n_2}$ dominates n_1. Condensing as structure-exploitating method is a

computational speedup if the sum of Condensing computation time, dense QP solution time and the time to recover the variables is less than the direct multiple shooting QP solution time.

3.4 Summary

In this chapter we have described the Direct Multiple Shooting discretization method for infinite-dimensional optimal control problems. The discretization yields a finite-dimensional structured NLP that we solve by SQP methods. We have put the focus on the QPs that exhibit an inherent multiple shooting structure and have discussed Condensing variants as part of the QP solution process exploiting the problem structure.

Chapter 4

Optimization in Real-Time

One overarching goal of the development of structure exploiting methods in this thesis is to meet real-time requirements of model-based control of complex systems. In this chapter we first explain the framework of Nonlinear Model Predictive Control, which is a concept to control real world processes online. Then we describe algorithms that ensure fast feedback, meaning that fast corrective actions counteract the observed deviations from the expected process state. The algorithms are tailored to the online solution of optimization problems with underlying dynamical system models for the real process. We describe the Real-Time Iteration scheme and Multi-Level Iteration schemes as state-of-the art methods to deal with real-time requirements.

4.1 Nonlinear Model Predictive Control

We solve the OCP (3.1) with underlying nonlinear dynamical system model to compute optimal controls for a real-world process. In the open-loop approach a precomputed sequence of controls is applied to the system. However, unforeseen disturbances, systematic modeling errors and unmodeled external influences can cause the open-loop system to underperform or - even worse - to violate safety or product requirements. In Nonlinear Model Predictive Control (NMPC) we focus on a closed-loop approach that involves feedback. We repeatedly solve the OCP within a sampling time and feed the control back to the system. The corrective actions counteract the deviations from the expected process state. Furthermore the OCP takes current state and parameter estimations into account. Solving an optimization problem for parameter and state estimation on a moving horizon in the past is called moving horizon estimation. We use the nominal model and minimize the difference between measurements of the current process and the model response. Another possibility for parameter and state estimation is the extended Kalman filter [56, 97]. For the challenge of uncertainty handling

we can on the one hand add safety margins to the deterministic problem formulation or on the other hand use a probabilistic description of the uncertainty. Our probabilistic description is the scenario tree. As NMPC incorporates economic objective functions, constraints, current estimates and uncertainty handling to generate a feedback control online while the real system is running, the principle is one of the most versatile feedback control concepts [40]. The concept is visualized and described in Figure 4.1.

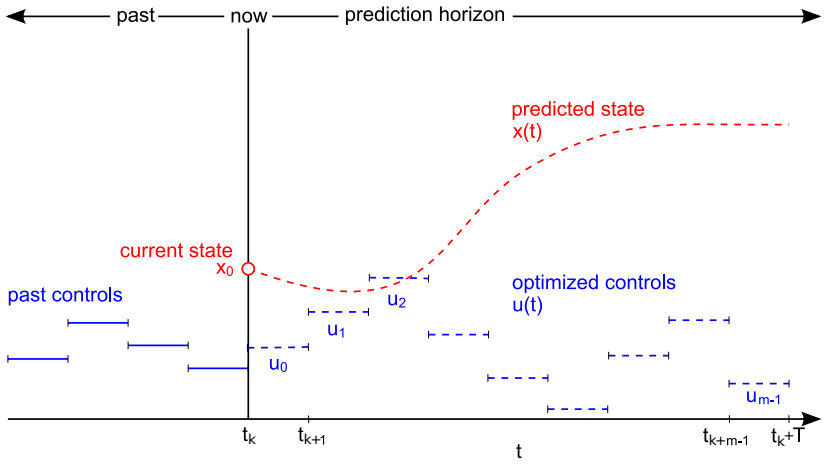

Figure 4.1: At the current time $t = t_k$ we estimate the system state x_0. In NMPC we solve an optimal control problem on the prediction horizon $[t_k, t_k + T]$ yielding the updated predicted state trajectory $x(t)$ realized by optimal controls u_k. We then apply the control u_0 to the system and shift the horizon for one sampling interval. The procedure is repeated on a chosen sampling grid $\{t_k\}$.

At this point we remark another interpretation of NMPC. Ideal NMPC can be regarded as nonlinear control feedback law $\mathbf{u}$ for the closed-loop system

$$x(t) = f(x(t), \mathbf{u}(t, x(t), p(t)), p(t)).$$

The law is implicitly given by the solution of optimization problems on the prediction horizon assuming that the solution exists at least locally. Under certain conditions it is possible to establish stability guarantees of NMPC. We refer to [53, 98] for detailed stability investigations, which exceed the scope of this work.

The scenario tree approach takes the uncertainty in a probabilistic sense into account. We have discussed the stochastic properties of the tree process in Section 2.5 of this thesis. In the NMPC framework the scenario tree approach is an extension because we repeatedly solve scenario tree optimization problems on the prediction horizon as visualized in Figure 4.2. The structure exploiting methods for NMPC, especially those in the following sections

are for better readability described for standard NMPC but the methods are fully applicable to scenario tree NMPC.

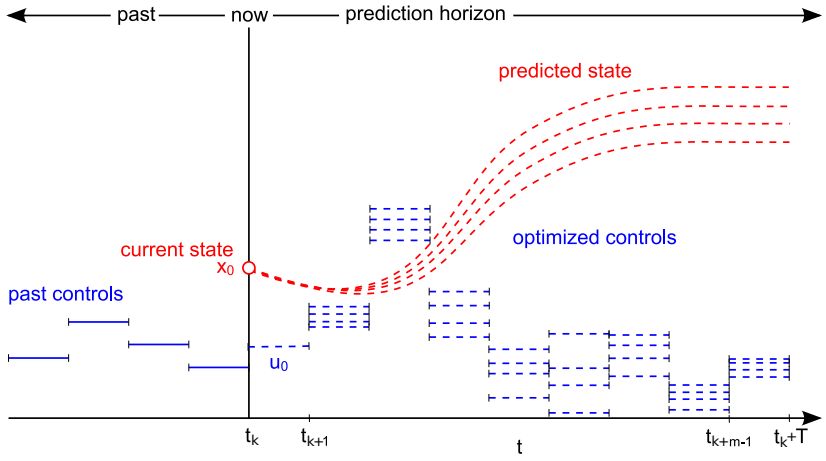

Figure 4.2: Scenario tree NMPC is a variant of NMPC solving scenario tree optimization problems on the prediction horizon $[t_k, t_k + T]$. We depict a tree with one branching point at t_k and four realizations of an uncertain parameter.

4.2 The Real-Time Iteration Scheme

In NMPC the online solution of optimal control problems and the feedback at each sampling time permits to react to unmodeled influences of the real world to the system. This is a powerful concept, but to be successful the actuation delays introduced by the online computations must be minimized. The Real-Time Iteration (RTI) scheme is one algorithmic concept to reduce the required computation time delay between measurement and control feedback. In [25, 28] the RTI scheme is introduced as adaptation of an SQP method to solve NLPs originating from a Direct Multiple Shooting discretization.

An SQP method for the multiple shooting NLP as described in Section 3.2 iteratively solves QPs until a stopping criterion is satisfied. The essential idea of RTI to reduce feedback delay is to perform only one QP solution during each sampling interval. A further reduction of the feedback delay is possible by decoupling of the estimated $\hat{x}(t_k)$ from the QP data. The idea is called initial value embedding. As a result the QP solution can be split into phases. All computations that do not require $\hat{x}(t_k)$ can be performed before the current estimate is available. To explain the RTI scheme let us consider a QP at sampling point t_k that results from an SQP approach to solve (3.3). We aggregate the discretized optimization variables as $z = (x_0, u_0, x_1, u_1, \ldots, x_{M-1}, u_{M-1}, x_M)$. In this chapter we consider a QP of the

form

$$\min_z \ \frac{1}{2} z^\top B z + b^\top z \tag{4.1a}$$

$$\text{s.t. } 0 \leq Az + a, \tag{4.1b}$$

$$0 = x_0 - \hat{x}(t_k), \tag{4.1c}$$

with QP data consisting of the matrices $B \in \mathbb{R}^{n_z \times n_z}$, $A \in \mathbb{R}^{n_c \times n_z}$ and vectors $b \in \mathbb{R}^{n_z}$, $a \in \mathbb{R}^{n_c}$.

The initial value embedding idea means that the estimated $\hat{x}(t_k)$ in (4.1c) decouples from the rest of the QP data. This idea is also possible in the case of Direct Collocation (see, e.g., [114]).

We split the RTI into three phases induced by the availability of the estimated $\hat{x}(t_k)$.

1st phase: Preparation

In the first phase we prepare all the data of the QP (4.1) that does not depend on the current observation. This is possible due to the initial value embedding in (4.1c). We initialize the QP using data from the previous QP. Details about initialization strategies follow below. After computing the sensitivities we can assemble B, b, A, and a.

2nd phase: Feedback

The second phase starts with an incoming estimate $\hat{x}(t_k)$. We embed it into the QP (4.1) by the constraint (4.1c) and then solve the QP. The u_0 part of the QP solution is immediately fed back to system. Therefore the feedback delay of RTI is reduced to the time between the measurement of the system state at t_k and the availability of u_0.

3rd phase: Transition

In the last phase we complete the QP solution, go to the new sampling point t_{k+1} and then repeat the phases of the RTI scheme.

QP initialization

Since the RTI basically depends on the solution of one QP, a good choice of the initializer is crucial for the QP solver. For the first iteration we assume that an initial guess $(x_0, u_0, \ldots x_{M-1}, u_{M-1}, x_M)$ is available. For further real-time iterations there are different strategies for obtaining an initializer from the previous QP solution. We are usually interested in moving prediction horizons meaning that all optimization problems have the same horizon length T and therefore the same number of optimization variables.

For every sampling time we shift the problem instance by one sampling time step as explained for NMPC. Let $z^k = (x_0, u_0, \ldots x_{M-1}, u_{M-1}, x_M)$ be the outcome of the real-time iteration at sampling time t_k. Then we choose the initializer at time t_{k+1} from the following variants.

1. Shift
   We shift the variables such that $z^{k+1} = (x_1, u_1, \ldots x_{M-1}, u_{M-1}, x_M, u_{M-1}^{new}, x_M^{new})$.
   Both terminal control and state are kept from the previous solution, i.e.
   $u_{M-1}^{new} := u_{M-1}$ and $x_M^{new} := x_M$.

2. Extrapolation
   We also shift the variables such that $z^{k+1} = (x_1, u_1, \ldots x_{M-1}, u_{M-1}, x_M, u_{M-1}^{new}, x_M^{new})$.
   At the cost of solving an initial value problem on the last discretization interval we compute x_M^{new}. The required control action can be chosen either as $u_{M-1}^{new} := u_{M-1}$ or from a (not too expensive) control law $u_{M-1}^{new} = \tilde{u}(z^k)$.

3. Reuse
   In case of small sampling intervals and short horizons the solutions of subsequent QPs show similar characteristics. A direct initialization $z^{k+1} := z^k$ results in a further speedup of the RTI scheme. The sampling grid decouples from the discretization grid in this strategy. Therefore we can provide even faster feedback, that again yields QP similarity.

Up to this point we have described initialization strategies for the primal variables. Typically, the dual variables are shifted along the primal variables and kept constant at the terminal stage. For the reuse warm start strategy we initialize $\lambda^{k+1} := \lambda^k$ for all dual variables. In the next section we want to explain why reuse is our preferred initialization approach for the RTI scheme.

Parametric quadratic programming in the RTI scheme

In the reuse initialization strategy it is beneficial to compute the solution of the new QP using the data from the previous solution process. This can be realized by parametric quadratic programming methods. We consider a family of QPs with parameter τ of the form

$$\min_{z(\tau)} \ \frac{1}{2} z(\tau)^\top B z(\tau) + b(\tau)^\top z(\tau) \tag{4.2a}$$

$$\text{s.t. } 0 \leq A z(\tau) + a(\tau), \tag{4.2b}$$

$$0 = x_0 + \tau \hat{x}(t_k) - (1-\tau)\hat{x}(t_{k-1}). \tag{4.2c}$$

We assume that all parameterized functions are piecewise linear in τ. The QP (4.2) depends on two subsequent state estimates $\hat{x}(t_{k-1})$ and $\hat{x}(t_k)$. We aim to find the piecewise affine linear feedback control path $u_0(\tau)$ between t_{k-1} and t_k, parameterized by $\tau \in [0,1]$, to feed u_0 back to the system whenever a new control action is required. Inferring the solution of the QP at t_k from the QP at t_{k-1} speeds up computations for the reuse strategy in order to satisfy strict real-time constraints. We can interrupt the solution process, give feedback immediately to the system and continue from there in the next available time slot. The fast feedback on a fine sampling grid is independent of the discretization grid of the QP. For technical insight into parametric quadratic programming we refer to [35]. The solver qpOASES [36] is an implementation of parametric quadratic programming.

To conclude, the RTI scheme solves the resulting discrete multiple shooting NLPs only approximately and reduces the feedback time further through a splitting of the iteration into a preparation, a feedback and a transition phase. For results on the approximation quality we refer to [25, 29, 27, 65]. RTI in combination with parametric quadratic programming exploits similarities of subsequent QPs, one important aspect of discretization structure exploitation.

4.3 Multi-Level Iteration Schemes

The Multi-Level Iteration (MLI) idea proposed in [16, 15] is an extension of the RTI scheme. Replacing the full preparation phase of RTI by hierarchical updates of the QP data yields even faster feedback. However, there is always a tradeoff between the contraction provided by the QP solution and the computational cost per iteration. A complete update usually provides best contraction but is expensive in time. In the MLI method there are four levels with increasing update expense. The idea behind MLI is that two subsequent QPs of the form (4.1) differ in matrices B, A and vectors b, a only if we choose to recompute them. Otherwise we keep the QP data fixed to accelerate the QP solution.

Level A: Feedback iteration

In level A we assume that QP (4.1) is given with data $\tilde{B}$, $\tilde{A}$, $\tilde{b}$, $\tilde{a}$ and reference solution $(\tilde{z}, \tilde{\lambda})$. We solve the QP

$$\min_z \ \frac{1}{2} z^\top \tilde{B} z + \tilde{b}^\top z \tag{4.3a}$$

$$\text{s.t. } 0 \leq \tilde{A} z + \tilde{a}, \tag{4.3b}$$

$$0 = x_0 - \hat{x}(t_k) \tag{4.3c}$$

for the current estimate $\hat{x}(t_k)$ in order to return the control as fast as possible to the system. The QP solver needs very few iterations only. Essentially, level A is linear model predictive control that operates on the data provided by higher levels of MLI. We save all the ODE integration and sensitivity generation effort.

Level B: Feasibility improvement iteration

For level B we keep the QP data $\tilde{B}$, $\tilde{A}$, a reference gradient $\tilde{b}$ and the reference solution $(\tilde{z}, \tilde{\lambda})$. We evaluate the constraints yielding the current a and update the gradient

$$b = \tilde{b} + B(z - \tilde{z}).$$

With this gradient we solve the QP

$$\min_z \ \frac{1}{2} z^\top \tilde{B} z + b^\top z \tag{4.4a}$$

$$\text{s.t. } 0 \leq \tilde{A} z + a, \tag{4.4b}$$

$$0 = x_0 - \hat{x}(t_k). \tag{4.4c}$$

The additional costs of level B compared to level A are due to constraint evaluation and the extra matrix-vector multiplication to update the gradient. The matrix decompositions inside the solver remain valid. It can be shown that Level B iterations with a fixed $\hat{x}(t_k)$ converge locally to a suboptimal but feasible point of the NLP. For a proof we refer to [15].

Level C: Optimality improvement iteration

In level C we keep the QP matrices $\tilde{B}$, $\tilde{A}$ and the reference solution $(\tilde{z}, \tilde{\lambda})$. We evaluate the constraints yielding the vector a and compute a modified gradient

$$b = b^k + \tilde{A}^\top \lambda^k.$$

We then solve the QP

$$\min_z \ \frac{1}{2} z^\top \tilde{B} z + b^\top z \tag{4.5a}$$

$$\text{s.t. } 0 \leq \tilde{A} z + a, \tag{4.5b}$$

$$0 = x_0 - \hat{x}(t_k). \tag{4.5c}$$

Additional computational costs compared to level B come from the evaluation of the adjoint sensitivities to assemble b. But these costs are less expensive than computing all sensitivities. Again, matrix decompositions inside the solver remain valid. Level C iterations with a fixed $\hat{x}(t_k)$ converge locally to a KKT point of the NLP as proven in [15].

Level D: RTI

Level D is a standard real-time iteration as described in Section 4.2. We evaluate the constraints, gradient, sensitivities and a new Hessian approximation B. Therefore the QP solver needs to refactorize the inherent matrices.

MLI schemes

An MLI scheme is assembled from the levels A-D by specifying how often each level is executed. For instance, the $D^8C^\infty B^4A^1$-scheme means that we solve a fast feedback linear MPC at each sampling time (level A), improving feasibility at every fourth sampling time (level B) and fully recompute the QP data and solve at every eighth sampling time (level D). The ∞-symbol indicates that the level is not performed, therefore we do not use level C in the example.

Modifications of the MLI approach such as mixed-level iterations and fractional-level iterations have been investigated in [67, 44, 41].

4.4 Summary

This chapter has put the focus on one goal of structure exploitation, the real-time feasibility. We have introduced the NMPC principle and iteration schemes to speed up NMPC by splitting the solution process of the optimization problem in different levels. The RTI scheme and its extension MLI as described in this chapter are fully applicable to the scenario tree optimization problems as they also exhibit the structure of the underlying dynamical evolution.

Chapter 5

Scenario Tree Structure Exploitation

In Chapter 3 we have reviewed approaches to exploit the discretization structure of optimization problems with underlying dynamical systems. The present chapter targets the inherent structure of the scenario tree optimization problems. Tree structure can appear on different levels in optimization with scenario trees. Therefore also structure-exploiting methods can be investigated on the OCP, NLP and QP level. We start this chapter with an overview of tree structures on the different levels and relate to existing work that focusses on tree structure. As the main part of the chapter we present the dual decomposition approach for tree-structured QPs. For large scenario trees the QPs cannot be effiently solved without problem tailored methods. On top of that a sequence of similar QPs must be solved in our MLI framework for NMPC, therefore structure exploitation on the QP level yields a high effort with regard to computation times of scenario tree NMPC. From the perspective of numerical optimization methods for scenario tree NMPC the tree structure exploiting QP solution method is the main contribution of this thesis. This main contribution has been published in [75]. In contrast to the rather condensed paper we provide more background and details, especially for the non-smooth Newton method that is required to solve the resulting optimization problem in the dual decomposition approach.

5.1 Tree Structure in Optimization Problems

In the following we state scenario tree optimization problems on the different levels regarding scenario tree NMPC and put a special focus on the QP subproblems because they must be repeatedly solved when applying SQP methods or MLI schemes to the NLP originating from the discretized scenario tree optimal control problem. On the optimal control level we consider the problem (2.1) that we have formulated in Chapter 2. We have seen especially in Chapter 4 on real-time feasibility in optimization that we must discretize it at one point.

Optimal control problem

The tree structure in optimal control problems can be encoded by the non-anticipativity constraints. We have introduced the formulation as problem (2.1) and recall it here.

$$\min_{x(t),u(t)} \quad \sum_{j\in\mathcal{S}} w_j \int_{t_0}^{t_f} \Phi(x_j(t),u_j(t),p_j(t))\mathrm{d}t \tag{5.1a}$$

$$\text{s.t.} \qquad \dot{x}_j(t) = f(x_j(t),u_j(t),p_j(t)), \qquad t\in[t_0,t_f],\ j\in\mathcal{S}, \tag{5.1b}$$

$$x_j(t_0) = x_0, \qquad j\in\mathcal{S}, \tag{5.1c}$$

$$x_j(t) \in \mathcal{X}, \qquad t\in[t_0,t_f],\ j\in\mathcal{S}, \tag{5.1d}$$

$$u_j(t) \in \mathcal{U}, \qquad t\in[t_0,t_f],\ j\in\mathcal{S}, \tag{5.1e}$$

$$u_i(t) = u_j(t), \qquad t\in[t_k,t_{k+1}],\ (i,j)\in\mathcal{C}_k,\ k\in\mathcal{K}. \tag{5.1f}$$

The variable j from the set $\mathcal{S}=\{1,\dots,S\}$ corresponds to a scenario with weight $w_j\in[0,1]$. Indices k in the set $\mathcal{K}=\{0,\dots,M-1\}$ correspond to the time discretization from the partitioned interval I into $t_0<t_1<\cdots<t_{M-1}<t_M=t_f$.

We emphasize that problem (5.1) is formulated branchwise as the variable index j corresponds to a whole scenario. We have a full set of control variables for every scenario that is determined by the sequence of parameter realizations from the tree root to a tree leaf. The tree structure is encoded by posing the conditions (5.1f) on the controls requiring the technical definition

$$\mathcal{C}_k := \left\{(i,j)\in\mathcal{S}^2 \,\middle|\, p_i(t)=p_j(t) \text{ for all } t\in[t_0,t_k]\right\}.$$

Discretizing the OCP (5.1) yields a branchwise oriented nonlinear programming problem, that we focus on later. At this point we continue with a short excursus on the existing tree-structure investigations of the nodewise problem formulation.

Node-oriented nonlinear problem

On the NLP level a node-oriented tree-structured problem is introduced in [59, 60] as nonlinear tree-sparse problem with explicit controls. The problem formulation serves as generalization of the tree sparse convex programs in [106, 107]. In the nodewise formulation every node has its own set of optimization variables. If the scenario tree has many decision points, the advantage of the nodewise formulation is a smaller number of optimization variables than in the branchwise formulation. However, the parallelization of the nodewise formulation is a difficult task due to the coupling structure.

In [105, 59] node-oriented tree-structured NLPs are solved using a primal-dual interior-point method employing filter line-search globalization. The algorithms presented in [59]

exploit the underlying tree topology as inherent structure of the constraint matrices. As a benefit of this node-wise representation, algorithms are designed as traversals of tree nodes that perform the overall solution by a series of node operations. For distributing algorithms of the node operations to parallel architectures as well as for numerical issues such as scaling, problem convexification and matrix regularization we refer to [59].

Branch-oriented nonlinear problem

We return to our branchwise formulation of tree-structured optimization problems. A Direct Multiple Shooting discretization of (5.1) yields discretized system states $x_j^T = [x_{j,0}^T, \ldots, x_{j,M}^T]$ and discretized controls $u_j^T = [u_{j,0}^T, \ldots, u_{j,M-1}^T]$ for $j \in \mathcal{S}$.

The tree-structured NLP (5.2) reads

$$\min_{x,u} \sum_{j\in\mathcal{S}} w_j \phi(x_j, u_j) \tag{5.2a}$$

$$\text{s.t.} \quad x_{j,0} = x_0, \qquad j \in \mathcal{S}, \tag{5.2b}$$

$$x_{j,k+1} = x_j^k(t_{k+1}; x_{j,k}, u_{j,k}), \qquad k \in \mathcal{K},\ j \in \mathcal{S}, \tag{5.2c}$$

$$\underline{x} \le x_{j,k} \le \overline{x}, \qquad k \in \mathcal{K} \cup \{M\},\ j \in \mathcal{S}, \tag{5.2d}$$

$$\underline{u} \le u_{j,k} \le \overline{u}, \qquad k \in \mathcal{K},\ j \in \mathcal{S}, \tag{5.2e}$$

$$u_{i,k} = u_{j,k}, \qquad (i,j) \in \mathcal{C}_k,\ k \in \mathcal{K}. \tag{5.2f}$$

In problem (5.2) we minimize the weighted sum of the scenario objective functions with respect to the following constraints: one initial value for all scenarios (5.2b), system dynamics (5.2c) with nonlinear function x_j^k to evaluate the dynamical evolution at the kth interval of the discretization, state bounds (5.2d), control bounds (5.2e) and non-anticipativity constraints (5.2f). We do not employ an interior point method for the NLP formulation as in [105, 59] for the nodewise formulation. Instead, we employ the SQP-based real-time tailored algorithms that we have discussed in Chapter 4. In the context of the MLI iteration scheme it becomes crucial to investigate the structure on the QP level. Therefore we present in this chapter a tree-structure exploiting algorithm on the QP level.

Branch-oriented quadratic problem

At first we introduce a notation for the branch-oriented tree QP. We denote the discretized system states x_j and discretized controls u_j as variables $z_j \in \mathbb{R}^{n_z}$,

$$z_j^T = [x_{j,0}^T, u_{j,0}^T, x_{j,1}^T, u_{j,1}^T, \ldots, x_{j,M-1}^T, u_{j,M-1}^T, x_{j,M}^T]^T. \tag{5.3}$$

Then the branch-oriented tree QP reads

$$\min_{z} \quad \sum_{j\in\mathcal{S}} w_j \left(\frac{1}{2} z_j^T H_j z_j + g_j^T z_j\right) \tag{5.4a}$$

$$\text{s.t.} \qquad x_{j,0} = x_0, \qquad j \in \mathcal{S}, \tag{5.4b}$$

$$x_{j,k+1} = A_{j,k} x_{j,k} + B_{j,k} u_{j,k}, \qquad k \in \mathcal{K},\ j \in \mathcal{S}, \tag{5.4c}$$

$$\underline{x} \le x_{j,k} \le \overline{x}, \qquad k \in \mathcal{K} \cup \{M\},\ j \in \mathcal{S}, \tag{5.4d}$$

$$\underline{u} \le u_{j,k} \le \overline{u}, \qquad k \in \mathcal{K},\ j \in \mathcal{S}, \tag{5.4e}$$

$$E^{j+1} z_{j+1} = C^j z_j, \qquad j \in \mathcal{S}\backslash\{S\}. \tag{5.4f}$$

As in previous formulations, the index j addresses each of the S scenarios and k is the index for the stages in time. We assume that all $H_j, j \in \mathcal{S}$, in the quadratic objective function (5.4a) are positive definite approximations of the Hessian blocks of the Lagrangian of (5.2). In constraint (5.4b) the initial value is set to x_0 for all scenarios. The dynamical evolution is linearized at z and formulated in constraint (5.4c). The matrices $A_{j,k}$ and $B_{j,k}$ represent the sensitivities of the dynamical evolution description in (5.2c) with respect to the states and controls on stage k. We define the matrices $E^j, C^j \in \mathbb{R}^{n_a \times n_z}$ for $j \in \mathcal{S}$ such that (5.4f) denotes the n_a non-anticipativity constraints for coupled scenarios. To compose C_j, we fill it step by step with blocks of the identity matrix $\mathbb{I}_{n_u}$ according to Algorithm 2.

Algorithm 2 Set scenario tree structure representing matrix

Input: Scenario index $j \in \mathcal{S}\backslash\{S\}$, Tree data: number of realizations m, decision points t_k, number of decision points n_d

Output: C^j

1: $C^j = 0 \in \mathbb{R}^{n_a \times n_z}$
2: **for** $k = 1, \ldots, n_d$ **do**
3: **if** $(j, j+1) \in \mathcal{C}_k$ **then**
4: $i = n_u \sum_{r=0}^{k} (m^r - m^k)$
5: row indices $= i, \ldots, i + n_u - 1$
6: column indices $= kn_x + (k-1)n_u, \ldots, kn_x + kn_u$
7: $C^j(\text{rowindices}, \text{columnindices}) = \mathbb{I}_{n_u}$
8: **end if**
9: **end for**

With the matrices $C^j, j \in \mathcal{S}\backslash\{S\}$, assembled with Algorithm 2 we define

$$E^{j+1} := C^j, j \in \mathcal{S}\backslash\{S\}.$$

For notational convenience the matrices E^1 and C^S are set to $0 \in \mathbb{R}^{n_a \times n_z}$. We assume in the following that a solution $z^* := (z_1^*, \ldots, z_S^*)$ of (5.4) exists and that LICQ is satisfied in this solution.

From the perspective of numerical optimization methods for scenario tree NMPC the structure-exploiting method to solve (5.4) is the main contribution of this thesis. The method we describe in the next section has been presented by the author at the European Control Conference 2015 and published in [75]. Therefore the description of the algorithm is largely based on [75].

5.2 Dual Decomposition

The dual decomposition approach divides the QP solution into subproblems exploiting the separability properties of the QP that are induced by its constraint structure. The approach is most promising if we choose a subset of constraints that couple only a few variables, because dual decomposition then allows to massively parallelize the solution process. Dual decomposition has been used already in the solution for QP subproblems in NMPC by [41, 42, 43]. The authors consider QPs with variables representing stages in time and decouple the variables originating from different stages. A distributed algorithm for more general coupling topologies is proposed in [70]. We consider the branch-oriented QP (5.4) for dual decomposition. In our case the non-anticipativity contraints containing the tree structure couple only a few control variables of subsequent scenarios, so they can serve for the dual decomposition approach. The non-anticipativity constraints on the one hand have more complex tree structure than the non-branching linear structure in [43], on the other hand the tree structure is a special case of the general separable QP structure introduced in [70]. In the following we present the details of the dual decomposition in the non-anticipativity constraints. We start with problem (5.4) from Section 5.1,

$$\min_{z} \quad \sum_{j \in \mathcal{S}} w_j \left(\frac{1}{2} z_j^T H_j z_j + g_j^T z_j \right) \tag{5.5a}$$

$$\text{s.t.} \qquad x_{j,0} = x_0, \qquad j \in \mathcal{S}, \tag{5.5b}$$

$$x_{j,k+1} = A_{j,k} x_{j,k} + B_{j,k} u_{j,k}, \qquad k \in \mathcal{K},\ j \in \mathcal{S}, \tag{5.5c}$$

$$\underline{x} \le x_{j,k} \le \overline{x}, \qquad k \in \mathcal{K} \cup \{M\},\ j \in \mathcal{S}, \tag{5.5d}$$

$$\underline{u} \le u_{j,k} \le \overline{u}, \qquad k \in \mathcal{K},\ j \in \mathcal{S}, \tag{5.5e}$$

$$E^{j+1} z_{j+1} = C^j z_j, \qquad j \in \mathcal{S} \backslash \{S\}. \tag{5.5f}$$

QP (5.5) is separable except for the linear coupling terms of subsequent indices in (5.5f). Our aim is to find a reformulation of QP (5.5) that separates into subproblems. We decom-

pose the QP (5.5) by dualizing the constraints (5.5f). We introduce $\lambda \in \mathbb{R}^{n_a}$ to express (5.5a) and (5.5f) as partial Lagrangian function

$$\tilde{L}(z,\lambda) := \sum_{j\in\mathcal{S}} w_j \left(\frac{1}{2} z_j^T H_j z_j + g_j^T z_j\right) + \sum_{j\in\mathcal{S}\setminus\{S\}} \lambda^T (C^j z_j - E^{j+1} z_{j+1}). \tag{5.6a}$$

The separation of the terms depending on E_j or C_j and an index shift in the E_j terms yield

$$\tilde{L}(z,\lambda) = \sum_{j\in\mathcal{S}} w_j \left(\frac{1}{2} z_j^T H_j z_j + g_j^T z_j\right) + \sum_{j\in\mathcal{S}\setminus\{S\}} \lambda^T C^j z_j - \sum_{j\in\mathcal{S}\setminus\{1\}} \lambda^T E^j z_j. \tag{5.6b}$$

With the notation convention $E^1 = C^S = 0 \in \mathbb{R}^{n_a \times n_z}$ we arrive at

$$\tilde{L}(z,\lambda) = \sum_{j\in\mathcal{S}} \frac{w_j}{2} z_j^T H_j z_j + \left(w_j g_j^T + \lambda^T (C^j - E^j)\right) z_j. \tag{5.6c}$$

By Lagrangian duality as described for example in [9, 88], we can compute the solution of QP (5.5) from

$$\max_{\lambda} \min_{z} \sum_{j\in\mathcal{S}} \left(\frac{w_j}{2} z_j^T H_j z_j + \left(w_j g_j^T + \lambda^T (C^j - E^j)\right) z_j\right) \tag{5.7a}$$

$$\text{s.t.} \quad x_{j,0} = x_0, \quad j \in \mathcal{S}, \tag{5.7b}$$

$$x_{j,k+1} = A_{j,k} x_{j,k} + B_{j,k} u_{j,k}, \quad k \in \mathcal{K},\ j \in \mathcal{S}, \tag{5.7c}$$

$$\underline{x} \le x_{j,k} \le \overline{x}, \quad k \in \mathcal{K} \cup \{k+1\},\ j \in \mathcal{S}, \tag{5.7d}$$

$$\underline{u} \le u_{j,k} \le \overline{u}, \quad k \in \mathcal{K},\ j \in \mathcal{S}. \tag{5.7e}$$

We introduce for $j \in \mathcal{S}$ the matrices and vectors

$$G_j = \begin{pmatrix} -\mathbb{I}_{n_x} & & & & \\ A_{j,0} & B_{j,0} & -\mathbb{I}_{n_x} & & \\ & & A_{j,1} & B_{j,1} & -\mathbb{I}_{n_x} & \\ & & & & \ddots \\ & & & & A_{j,M-1} & B_{j,M-1} & -\mathbb{I}_{n_x} \end{pmatrix}, \; D_j = \begin{pmatrix} \mathbb{I}_{n_z} \\ -\mathbb{I}_{n_z} \end{pmatrix},$$

$z_{0,j}^T = [x_{j,0}^T, 0, \ldots, 0]^T \in \mathbb{R}^{2n_z}$ and $z_{\text{bnd},j}^T = [\overline{x}, \overline{u}, \overline{x}, \ldots, \overline{x}, \overline{u}, \overline{x}, -\underline{x}, -\underline{u}, \ldots, -\underline{u}, -\underline{x}]^T \in \mathbb{R}^{2n_z}$.

Then we formulate (5.7) as

$$\max_{\lambda} \min_{z} \sum_{j\in\mathcal{S}} \left(\frac{w_j}{2} z_j^T H_j z_j + \left(w_j g_j^T + \lambda^T (C^j - E^j)\right) z_j\right) \tag{5.8a}$$

$$\text{s.t.} \quad G_j z_j = z_{0,j}, \quad j \in \mathcal{S}, \tag{5.8b}$$

$$D_j z_j \le z_{\text{bnd},j}, \quad j \in \mathcal{S}. \tag{5.8c}$$

For $j \in \mathcal{S}$, we denote the quadratic objective functions by $F_j : \mathbb{R}^{n_z} \times \mathbb{R}^{n_a} \to \mathbb{R}$,

$$F_j(z_j, \lambda) := \frac{w_j}{2} z_j^T H_j z_j + \left(w_j g_j^T + \lambda^T (C^j - E^j)\right) z_j.$$

This yields the formulation of QP (5.8) as

$$\max_{\lambda} \min_{z} \sum_{j \in \mathcal{S}} F_j(z_j, \lambda) \tag{5.9a}$$

$$\text{s.t.} \quad G_j z_j = z_{0,j}, \quad j \in \mathcal{S}, \tag{5.9b}$$

$$D_j z_j \leq z_{\text{bnd},j}, \quad j \in \mathcal{S}. \tag{5.9c}$$

The optimization problem (5.9) is separable in the variables z_j that refer to scenario j. Thus, we can interchange summation and minimization and write

$$\max_{\lambda} \sum_{j \in \mathcal{S}} \min_{z} F_j(z_j, \lambda) \tag{5.10a}$$

$$\text{s.t.} \quad G_j z_j = z_{0,j}, \quad j \in \mathcal{S}, \tag{5.10b}$$

$$D_j z_j \leq z_{\text{bnd},j}, \quad j \in \mathcal{S}. \tag{5.10c}$$

For fixed λ we define local QPs for all $j \in \mathcal{S}$ corresponding to one scenario by

$$\min_{z} \; F_j(z_j, \lambda) \tag{5.11a}$$

$$\text{s.t.} \; G_j z_j = z_{0,j}, \quad j \in \mathcal{S}, \tag{5.11b}$$

$$D_j z_j \leq z_{\text{bnd},j}, \quad j \in \mathcal{S}. \tag{5.11c}$$

We denote the local QPs (5.11) as $QP_j(\lambda)$. The following assumption guarantees the existence of unique optima z_j^* for all $QP_j(\lambda)$ with optimal objective values $F_j(z_j^*, \lambda)$.

Assumption 5.1. *All local $QP_j(\lambda)$ are feasible and strictly convex.*

With Assumption 5.1 we can reformulate QP (5.10) as an unconstraint optimization problem

$$\max_{\lambda} F(\lambda) := \max_{\lambda} \sum_{j \in \mathcal{S}} F_j(z_j^*, \lambda). \tag{5.12}$$

$F_j(z_j^*, \lambda)$ denotes the optimal objective resulting from the solution of the local subproblems $QP_j(\lambda)$ for fixed λ and all scenarios $j \in \mathcal{S}$.

We are now at the point to state two Lemmata from [37]. Let $F^*(\lambda)$ denote the optimal objective function value with respect to λ.

Lemma 5.1. *Under Assumption 5.1 the optimal solutions z_j^* to the parametric local problems $QP_j(\lambda)$ depend piecewise-affinely and continuously on the parameter λ.*

Lemma 5.2. *If Assumption 5.1 holds, then the function $F^*(\lambda)$ is continuously differentiable, piecewise quadratic, concave in λ.*

Summarizing the dual decomposition approach, we have reformulated the QP (5.5) to (5.12) by exploiting the separable structure. The decoupled local $QP_j(\lambda)$ can be solved in a massively parallel fashion. Furthermore, our local QPs are conventional single scenario QPs. The branchwise problem formulation keeps the dynamical structure of the subproblems $QP_j(\lambda)$. Therefore methods exploiting the dynamical structure can be applied to the subproblems. Especially in case of a direct multiple shooting discretization we can use Condensing as described in Chapter 3.

5.3 Non-smooth Newton Method

The dual decomposition approach for the tree-structured QP (5.4) yields the unconstrained optimization problem (5.12). The smoothness properties of the objective function are stated in Lemma 5.2. According to [37, 70, 41] the consequence of Lemma 5.2 is that the second-order derivatives of the objective function exist almost everywhere and a second-order subderivative exists at active constraints changes of the local problems. We employ a non-smooth Newton method as in [94] to solve (5.12). For every iteration i we require the Newton gradient $\mathcal{G}^i := \left[\frac{dF^*}{d\lambda}(\lambda^i)\right]^T$ and the Newton matrix $\mathcal{M}^i := \left[\frac{d^2F^*}{d\lambda^2}(\lambda^i)\right]$ that exists almost everywhere. We solve in every iteration i the linear system

$$\mathcal{M}^i \Delta\lambda^i = -\mathcal{G}^i \tag{5.13}$$

to obtain the step direction $\Delta\lambda^i$.

After choosing an initial guess λ^0 we iterate

$$\lambda^{i+1} = \lambda^i + \alpha_i \Delta\lambda^i$$

with appropriately chosen stepsize α_i until a stopping criterion is satisfied.

At this point we continue with the computation and properties of $\mathcal{G}^i$ and $\mathcal{M}^i$. For notational convenience we drop the iteration index i of λ^i. We recall that the optimal objective function is separable in j, because $F^*(\lambda) = \max_{\lambda} \sum_{j\in\mathcal{S}} F_j(z_j^*, \lambda)$. Let us focus on the jth component of $F^*(\lambda)$,

$$F_j^*(\lambda) := F_j(z_j^*(\lambda), \lambda). \tag{5.14}$$

It is important to notice in (5.14) that z_j^* depends on λ. We differentiate (5.14) with respect to λ yielding

$$\frac{dF_j^*}{d\lambda}(\lambda) = \frac{\partial F_j}{\partial z_j}(z_j^*(\lambda), \lambda)\frac{dz_j^*(\lambda)}{d\lambda} + \frac{\partial F_j}{\partial \lambda}(z_j^*(\lambda), \lambda). \tag{5.15}$$

The first term in (5.15) vanishes,

$$\frac{\partial F_j}{\partial z_j}(z_j^*(\lambda),\lambda) = 0, \tag{5.16}$$

due to [9, App. C]. For the last term in (5.15) we obtain

$$\frac{\partial F_j}{\partial \lambda}(z_j^*(\lambda),\lambda) = (z_j^*(\lambda))^T (C_j - E_j)^T. \tag{5.17}$$

Therefore we have

$$\frac{dF_j^*}{d\lambda}(\lambda) = (z_j^*(\lambda))^T (C_j - E_j)^T. \tag{5.18}$$

The Newton gradient is the sum of all $j \in \mathcal{S}$ derivative components (5.18),

$$\mathcal{G}^i = \left[\frac{dF^*}{d\lambda}(\lambda)\right]^T = \sum_{j\in\mathcal{S}} \left[\frac{dF_j^*}{d\lambda}(\lambda)\right]^T = \sum_{j\in\mathcal{S}} (C_j - E_j) z_j^*(\lambda). \tag{5.19}$$

For the Newton matrix we formally write

$$\frac{d^2F_j^*}{d\lambda^2}(\lambda) = \frac{d}{d\lambda}\left(\frac{dF_j^*}{d\lambda}(\lambda)\right) = \frac{d}{d\lambda}\left((z_j^*(\lambda))^T (C_j - E_j)^T\right). \tag{5.20}$$

For the existence of the right-hand side term in (5.3), the term $z_j^*(\lambda)$ must be differentiable with respect to λ. Therefore we investigate the local QPs (5.11). We apply the concept of optimality systems and their correspondence to KKT points (c.f. [39]) to the local QPs (5.11).

We assume that a QP solver for the computation of $F(z_j^*,\lambda)$ delivers a working set $\mathcal{A}_j^*$, which is a subset of the set of active constraints and depends on z_j^*, such that the matrix

$$K_j := \begin{pmatrix} H_j & G_j^T & (D_j^{\mathcal{A}^*})^T \\ G_j & 0 & 0 \\ D_j^{\mathcal{A}^*} & 0 & 0 \end{pmatrix}$$

is invertible, where the superscript $\mathcal{A}^*$ selects the rows of D_j that belong to the working set. Introducing $\mu_j \in \mathbb{R}^{n_g}$ and $\nu_j^{\mathcal{A}^*} \in \mathbb{R}^{n^{\mathcal{A}^*}}$, the system of equations

$$K_j \begin{pmatrix} z_j^* \\ \mu_j^* \\ \nu_j^{\mathcal{A}^*} \end{pmatrix} = \begin{pmatrix} -g_j - (C_j - E_j)^T \lambda \\ z_0 \\ z_{\mathrm{bnd},j}^{\mathcal{A}^*} \end{pmatrix} \tag{5.21}$$

holds for optimal z_j^*, μ_j^*, and $\nu_j^{\mathcal{A}^*}$ for $j \in \mathcal{S}$. We derive from (5.21) that

$$\begin{aligned}\begin{pmatrix} z_j^*(\lambda) \\ \mu_j^*(\lambda) \\ \nu_j^{\mathcal{A}^*}(\lambda) \end{pmatrix} &= K_j^{-1} \begin{pmatrix} -g_j \\ z_0 \\ \nu_j^{\mathcal{A}^*} \end{pmatrix} - K_j^{-1} \begin{pmatrix} (C_j - E_j)^T \lambda \\ 0 \\ 0 \end{pmatrix} \\ &= \begin{pmatrix} z_j^*(0) \\ \mu_j^*(0) \\ \nu_j^{\mathcal{A}^*}(0) \end{pmatrix} - K_j^{-1} \begin{pmatrix} \mathbb{I}_{n_z} \\ 0 \\ 0 \end{pmatrix} (C_j - E_j)^T \lambda.\end{aligned}$$

Thus, we arrive at

$$z_j^*(\lambda) = z_j^*(0) - \begin{pmatrix} \mathbb{I}_{n_z} \\ 0 \\ 0 \end{pmatrix}^T K_j^{-1} \begin{pmatrix} \mathbb{I}_{n_z} \\ 0 \\ 0 \end{pmatrix} (C_j - E_j)^T \lambda.$$

For notational convenience we define

$$\tilde{K}_j := \begin{pmatrix} \mathbb{I}_{n_z} & 0 & 0 \end{pmatrix} K_j^{-1} \begin{pmatrix} \mathbb{I}_{n_z} \\ 0 \\ 0 \end{pmatrix}$$

and obtain

$$z_j^*(\lambda) = z_j^*(0) - \tilde{K}_j (C_j - E_j)^T \lambda,$$

which can easily be differentiated with respect to λ yielding

$$\frac{d}{d\lambda} z_j^*(\lambda) = -\tilde{K}_j (C_j - E_j)^T.$$

Equation (5.3) finally delivers

$$\begin{aligned}\frac{d^2 F_j^*}{d\lambda^2}(\lambda) &= \frac{d}{d\lambda} (z_j^*(\lambda))^T (C_j - E_j)^T \\ &= -\left(\tilde{K}_j (C_j - E_j)^T\right)^T (C_j - E_j)^T \\ &= -(C_j - E_j) \tilde{K}_j (C_j - E_j)^T,\end{aligned}$$

and thus

$$\mathcal{M}^i = \frac{d^2 F^*}{d\lambda^2}(\lambda) = \sum_{j \in \mathcal{S}} -(C_j - E_j) \tilde{K}_j (C_j - E_j)^T.$$

As pointed out in [37], $\mathcal{M}$ becomes singular in the case of jointly redundant active constraints in several inner problems $QP_j(\lambda)$ due to the coupling non-anticipativity constraints. In the numerical case studies we indeed sometimes observe zero eigenvalues of $\mathcal{M}$. Note

that the function $F^*(\lambda)$ is piecewise quadratic, differentiable and twice differentiable according to Lemma 5.2. In the computation of $\mathcal{M}$ the matrix $\tilde{K}_j$ depends on the solution $z_j^*(\lambda)$ of the local QPs. The local QPs exhibit active set changes depending on λ that yield the characteristics of $F^*(\lambda)$ [94, 37].

For the solution of the linear Newton system (5.13) in the case of general coupling topologies the paper [70] proposes an iterative method without forming the large Newton system. We employ regularization strategies as for example Levenberg-Marquardt and use a Cholesky decomposition of the regularized $-\mathcal{M}$ to solve the linear system originating from our tree-structured QP. An alternative method [69] solves the singularity problem resulting from (5.13) based on reordering and substitution of constraints in the original QP (5.4). All in all, the non-smooth Newton strategy is summarized in Algorithm 3.

Algorithm 3 Dual Newton strategy

Input: Initial value λ^0, tolerance ε

Output: Optimal solution λ^*, z^*

for $i = 0, 1, \ldots$ **do**
  for $j = 1, \ldots, S$ **do**
    solve $\mathrm{QP}_j(\lambda^i)$ to obtain $z_j^*(\lambda^i)$ **(parallel)**
  end for
  Compute $\mathcal{G}^i$
  if $|\mathcal{G}^i| < \varepsilon$ **then**
    return Optimal solution λ^*, z^*
  end if
  Compute $\mathcal{M}^i$
  Solve (regularized) linear system $\mathcal{M}^i \Delta\lambda^i = -\mathcal{G}^i$
  Compute stepsize α_i
  Update $\lambda^{i+1} = \lambda^i + \alpha_i \Delta\lambda^i$
end for

Globalization is required in Algorithm 3 to determine the stepsize α_i as the objective function is piecewise quadratic. We employ the accelerated bisection line search strategy that is suggested in [43], see Algorithm 4. It is a combination of a fast backtracking line search at the beginning (lines 2–5) and a bisection search for refinement (lines 7–21).

Algorithm 4 Accelerated bisection line search

Input: Current guess λ, current objective $F_c^* = F^*(\lambda)$, search direction $\Delta\lambda$, scaling factor $s \ll 1$, bounds $\alpha_{\min}, \alpha_{\max}$, termination criteria $n_{\max}, \varepsilon$

Output: Stepsize α

1: Solve all scenario QPs for $(\lambda + \alpha_{\max}\Delta\lambda)$ to obtain F^*_{cand} **(parallel)**
2: **while** $F^*_{\text{cand}} < F^*_c$ **do** (Backtracking)
3: $\alpha_{\max} := s \cdot \alpha_{\max}$
4: Solve all scenario QPs for $(\lambda + \alpha_{\max}\Delta\lambda)$ to obtain F^*_{cand} **(parallel)**
5: **end while**
6: $\alpha_{\max} := \min(\frac{\alpha_{\max}}{s}, 1)$
7: **for** $i = 1, \ldots, n_{\max}$ **do** (Bisection)
8: $\alpha := \frac{\alpha_{\max}+\alpha_{\min}}{2}$
9: Solve all scenario QPs for $(\lambda + \alpha\Delta\lambda)$ **(parallel)**
10: Set up the gradient $\frac{d}{d\lambda}F^*(\lambda + \alpha\Delta\lambda)$
11: Compute $F'(\alpha) := \Delta\lambda^T \frac{d}{d\lambda}F^*(\lambda + \alpha\Delta\lambda)$
12: **if** $|F'(\alpha)| \leq \varepsilon$ **then**
13: **return** α
14: **else**
15: **if** $F'(\alpha) < 0$ **then** $\alpha_{\max} := \alpha$
16: **else** $\alpha_{\min} := \alpha$
17: **end if**
18: **end if**
19: **end for**

5.4 Summary

The optimization problems in the context of scenario tree NMPC exhibit a particular structure originating from the considered tree. The tree-structured problems can be formulated nodewise or branchwise. For the branchwise formulation we contribute to the numerical side of the scenario tree approach a structure exploiting method based on dual decomposition in the non-anticipativity constraints. In this chapter we have presented the algorithmic details. Each large-scale tree QP is solved iteratively by a non-smooth Newton method. Within each iteration of the Newton method a multitude of smaller decoupled QPs is solved in parallel. The numerical results in Chapter 9 are all computed using our tree structure exploiting methods.

Chapter 6

Quadrature-based Scenario Tree Generation

A main assumption of the scenario tree approach is that we can represent the uncertain parameter space by a finite number of scenarios. In the case of a high-dimensional uncertainty, i.e. multi-dimensional parameters that enter into the problem, the scenario tree becomes large, even for a two-stage problem. As there is an exponential growth of the number of scenarios in the number of decision points $S = m^{n_d}$, the effort for optimizing the coupled scenarios at the same time becomes prohibitive, even when applying the structure-exploiting methods of the previous chapters. Therefore the question how to choose scenarios in the uncertain parameter space is of major importance for real-time feasibility of scenario tree NMPC. In this chapter we present a method of scenario generation inspired by sparse-grid quadrature rules, which are a commonly used tool in the field of uncertainty quantification. The quadrature-based scenario tree generation method is a contribution of this thesis. It reduces the base m of the exponential growth formula for scenario trees. The chapter is based on the publication [74] presented at the 19th World Congress of the International Federation of Automatic Control.

6.1 Expectation Value of the Objective and Quadrature

To motivate our approach we move back to the nominal optimal control problem

$$\min_{x,u} \quad \int_{t_0}^{t_f} \Phi(t,x(t),u(t),p)\mathrm{d}t \tag{6.1a}$$

$$\text{s.t.} \qquad \dot{x}(t) = f(x(t),u(t),p), \qquad t \in [t_0,t_f], \tag{6.1b}$$

$$0 = x(t_0) - x_0, \tag{6.1c}$$

$$0 \leq r(t,x(t),u(t),p), \qquad t \in [t_0,t_f]. \tag{6.1d}$$

Instead of minimizing the objective function (6.1a) for only one realization of p, we regard p as a random variable. We introduce the notation

$$F_{\text{obj}}(x,u,p) := \int_{t_0}^{t_f} \Phi(t,x(t),u(t),p)\mathrm{d}t,$$

and take the expectation value with respect to p, $\mathbb{E}_p$, of

$$F^*_{\text{obj}}(p) := F_{\text{obj}}(x^*,u^*,p)$$

in the space of uncertain parameters as an objective function. This can be expressed as an integral over a d-dimensional probability space Ω with measure μ and corresponding probability density function f_μ.

$$\begin{aligned}\mathbb{E}_p(F^*_{\text{obj}}(p)) &= \int_\Omega F^*_{\text{obj}}(p)d\mu(p)\\ &= \int_\Omega F^*_{\text{obj}}(p)f_\mu(p)dp\end{aligned}$$

When computing the integral value numerically, we require a reliable quadrature in high dimensions. To this end, we approximate the expectation value with a sum over a finite set $\Gamma \subset \Omega$ according to

$$\begin{aligned}\mathbb{E}_p(F^*_{\text{obj}}(p)) &= \int_\Omega F^*_{\text{obj}}(p)f_\mu(p)dp\\ &\approx \sum_{p\in\Gamma} w(p)F^*_{\text{obj}}(p)f_\mu(p).\end{aligned}$$

We then interpret every $p \in \Gamma$ as one parameter realization. Thus, we have $m = |\Gamma|$. One can argue that a robust scenario tree must contain the combined extreme values for all uncertain parameters. In our setting the identification of extreme values plays a minor role. In the following we consider $F^*_{\text{obj}}f_\mu$ with bounded mixed derivative, cf. (6.2). If we choose for instance f_μ such that the uncertain parameters are normally distributed and the objective function such that the bounded mixed derivative condition holds, an extreme realization of all uncertain parameters at the same time is highly unlikely on the basis of their joint distribution.

Along this line of arguments, we employ quadrature formulas with grid points that resolve the underlying probability space accurately and have benign computation costs in higher dimensions.

6.2 Sparse Grids

In the area of uncertainty quantification sparse grids are the method of choice for the evaluation of high-dimensional integrals. Function evaluations of F^*_{obj} at the nodes are expensive in our case, because we need to simulate a full scenario for each node. Therefore, we aim at reducing the number of grid points compared to a full tensor grid with m^d points without sacrificing accuracy with respect to the expected objective function value on the basis of sparse grids. The accuracy with respect to constraint satisfaction is not covered in a probabilistic sense by the quadrature-based scenario tree generation.

Following [48], we now explain how to approximate the integral of a function $F : \Omega \to \mathbb{R}$ with sparse grid quadrature. We denote the exact value by

$$I^d(F) = \int_\Omega F(x)\mathrm{d}x.$$

Moreover, we consider a sequence of quadrature formulas on level $l \in \mathbb{N}$ with n_l^d underlying points, $n_l^d < n_{l+1}^d$. Then the exact integral can be approximated by

$$\mathcal{Q}_l^d(F) := \sum_{i=1}^{n_l^d} w_{li} F(x_{li})$$

with quadrature weights $w_{li} \in \mathbb{R}$ and node points $x_{li} \in \Omega, i = 1, \ldots, n_l^d$. The underlying quadrature grid on level l is denoted by

$$\Gamma_l^d := \{x_{li} : 1 \le i \le n_l^d\} \subset \Omega.$$

6.2.1 Smolyak's Algorithm

The construction of sparse grids was proposed in [102] for functions with bounded mixed derivatives of order r, denoted by $\mathcal{W}_d^r$,

$$\mathcal{W}_d^r := \left\{ F : \Omega \to \mathbb{R}, \left\| \frac{\partial^{|\mathbf{s}|} F}{\partial x_1^{s_1} \cdots x_d^{s_d}} \right\|_\infty < \infty \text{ for all } s \in \mathbb{N}^d, s_i \le r \right\}, \tag{6.2}$$

with multi-index $\mathbf{s} \in \mathbb{N}^d$ and $|\mathbf{s}| = \sum_{i=1}^d s_i$.

Let $l \in \mathbb{N}$. For $F \in \mathcal{W}_1^r$, $w_{li} \in \mathbb{R}$, and $x_{li} \in \Gamma_l^d$, we consider the one-dimensional quadrature formula

$$\mathcal{Q}_l^1(F) = \sum_{i=1}^{n_l^1} w_{li} F(x_{li}).$$

We then define the difference formulas

$$\Delta_l^1(F) := (\mathcal{Q}_l^1 - \mathcal{Q}_{l-1}^1)F \quad \text{with} \quad \Delta_0^1(F) := 0.$$

The difference formulas are quadrature formulas on the grid $\Gamma_l^1 \cup \Gamma_{l-1}^1$. If the quadrature formulas are nested, that is $\Gamma_{l-1}^1 \subset \Gamma_l^1$, then the underlying grid of the difference formula Δ_l^1 is Γ_l^1.

To lift one-dimensional formulas to d-dimensional formulas for $F \in \mathcal{W}_d^r$, we define the tensor product of quadrature formulas $(\mathcal{Q}_{l_1}^1 \otimes \ldots \otimes \mathcal{Q}_{l_d}^1)$ as the sum over all possible combinations.

$$(\mathcal{Q}_{l_1}^1 \otimes \cdots \otimes \mathcal{Q}_{l_d}^1)(F) := \sum_{i_1=1}^{n_{l_1}^1} \cdots \sum_{i_d=1}^{n_{l_d}^1} w_{l_1 i_1} \cdots w_{l_d i_d} \cdot F(x_{l_1 i_1}, \cdots, x_{l_d i_d}).$$

Smolyak's formula for $F \in \mathcal{W}_d^r$, $l \in \mathbb{N}$, and multi-index $\mathbf{k} \in \mathbb{N}^d$ can then be expressed as

$$\mathcal{Q}_l^d(F) := \sum_{|\mathbf{k}| \leq l+d-1} (\Delta_{k_1}^1 \otimes \ldots \otimes \Delta_{k_d}^1)(F). \tag{6.3}$$

The underlying grid for formula (6.3) is called sparse grid.

Interpretation

Compared to the sparse grid formula (6.3), the full tensor product formula

$$\sum_{j=1}^{d} \sum_{1 \leq k_j \leq l} (\Delta_{k_1}^1 \otimes \ldots \otimes \Delta_{k_d}^1)(F)$$

corresponds to summation over the whole cube of indices $\{\mathbf{k} : k_j \leq l,\ j = 1, \ldots, d\}$.

The sparse grid formula (6.3) sums over a much smaller simplex of indices $\{\mathbf{k} : |\mathbf{k}| \leq l + d - 1\}$ instead.

Expanding the difference formulas $\Delta_{k_j}^1$, we can denote Smolyak's formula in terms of $\mathcal{Q}_{k_j}^1$ by

$$\mathcal{Q}_l^d(F) = \sum_{|\mathbf{k}|=l}^{l+d-1} (-1)^{(l+d-|\mathbf{k}|-1)} \binom{d-1}{|\mathbf{k}|-l} (\mathcal{Q}_{k_1}^1 \otimes \cdots \otimes \mathcal{Q}_{k_d}^1)(F).$$

Sparse grids of levels 0, 1, 2, and 3 as well as a tensor grid in dimension $d = 3$ are depicted in Fig. 6.1 to 6.4.

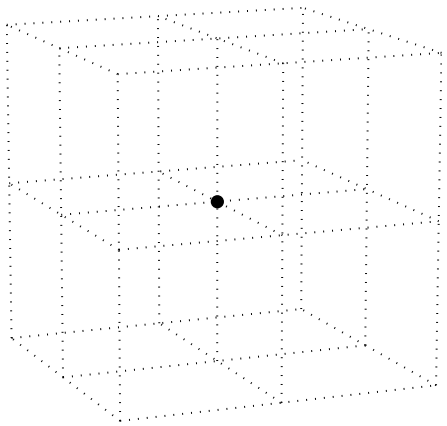

Figure 6.1: The sparse grid of level $l = 0$ in dimension $d = 3$ is a single point.

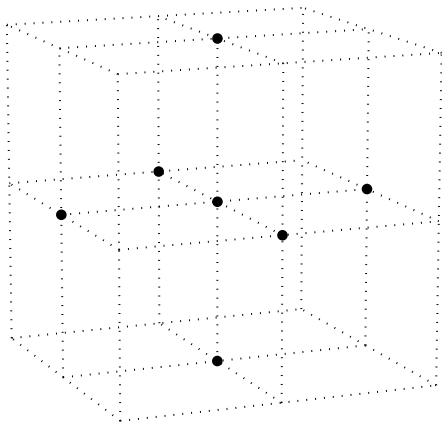

Figure 6.2: The sparse grid of level $l = 1$ in dimension $d = 3$ consists of seven points on the coordinate axes.

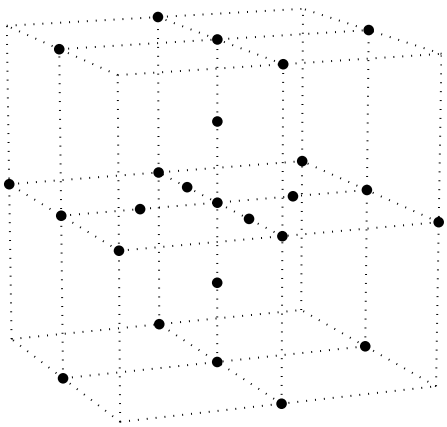

Figure 6.3: The sparse grid of level $l = 2$ in dimension $d = 3$ has 25 points which cluster at the coordinate axes. We remark that the depicted grid is not a subgrid of the tensor grid with $m = 3$.

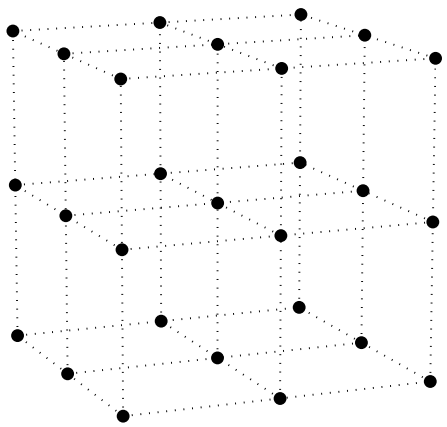

Figure 6.4: The full tensor grid with $m = 3$ in dimension $d = 3$ already consists of 27 points.

6.2.2 Error Bounds

Especially in higher dimensions, the number of underlying quadrature nodes for sparse grid quadrature is much smaller compared to tensor grid quadrature. In the nested case, the number of quadrature points of a sparse grid is

$$n_l^d = \sum_{|\mathbf{k}| \leq l+d-1} n_{k_1}^1 \cdot \ldots \cdot n_{k_d}^1.$$

If we assume $n_l^1 = \mathcal{O}(2^l)$, which is a justified assumption for one-dimensional quadrature rules like the trapezoidal rule or the Clenshaw-Curtis rule, we arrive at $n_l^d = \mathcal{O}(2^l \cdot l^{(d-1)})$. In contrast, the number of grid points for full tensor product rules is $\mathcal{O}(2^{ld})$.

To formulate error bounds for sparse grid quadrature, we start with an error bound $E_l^1(F)$ of the one-dimensional quadrature formulas with positive weights, as for example in the case of the Clenshaw-Curtis rule. If we assume that $f \in C^r$, the approximation

$$|E_l^1(F)| = \mathcal{O}((n_l^1)^{-r})$$

holds. We take such a quadrature formula as a basis for Smolyak's algorithm and additionally assume $F \in \mathcal{W}_d^r$ and $n_l^1 = \mathcal{O}(2^l)$. Then the error of sparse grid quadrature is according to [48]

$$|E_l^d(F)| = \mathcal{O}(2^{-lr} \cdot l^{(d-1)(r+1)}).$$

As stated in [100], the approximation quality of sparse grids even outperforms tensor grid approximation quality. The composition of sparse grid points, which cluster along the axes (Fig. 6.3), is better than the composition of tensor grid points (Fig. 6.4).

6.3 Scenario Tree NMPC with Quadrature-based Scenario Tree Generation

In Chapter 1 we have introduced the steps of scenario tree construction. First we choose a finite number of realizations (m) of the uncertainty, second we define a number of decision points (n_d) in time where the realizations are allowed to change and third, we couple the scenarios as sequences of parameter realizations according to the tree structure. The quadrature-based approach affects the first step. Especially in high-dimensional uncertainty spaces we choose sparse grid nodes as realizations of the multi-dimensional parameter space instead of tensor grid nodes. In consequence the basis m of the exponential growth formula $S = m^{n_d}$ for the number of scenarios can be significantly reduced. The scenario tree is used in every NMPC iteration. Thus computation time is massively reduced by the sparse grid approach as we shall see in the numerical result Chapter 9.

6.4 Summary

We have proposed a method to generate scenario trees for robust NMPC of uncertain systems with randomly distributed parameters in this chapter. The approach is based on high-dimensional quadrature rules that can be efficiently generated a-priori on the basis of sparse grids. One main difference of the resulting trees compared to usually used trees is that the extreme corner points of parameter realizations are not included. In the case of distributed parameters, these corner cases are highly unlikely to realize. We demonstrate the efficiency of our approach in the numerical results chapter considering sparse grid trees with significantly less amount of scenarios compared to a conventional full tensor grid. Quadrature-based tree generation reduces the base of the exponential growth in the number of decision points. Further scenario reduction based on dynamical evolution of the parameter process is discussed in the next chapter.

Chapter 7

Markov Chain Scenario Tree Pruning

The design of a suitable scenario tree is always a trade-off between the coverage of the uncertainty space and the computational cost of large trees. In Chapter 6 the basis of the exponential growth when using the usual scenario tree construction procedure has been successfully reduced. However, the usual approach still exhibits exponential dependence on the robust horizon. In this chapter we demonstrate an alternative scenario tree construction method that does not depend on a robust horizon as the usual scenario tree setup. The following method is based on one main assumption for the dynamical evolution of uncertainty: We assume that the uncertain parameter process approximated by the scenario process in time is a Markov chain. We emphasize that other properties of the stochastic tree process as described in Chapter 2 are not affected by the additional property. The Markov chain scenario tree pruning is a contribution of this thesis to the stochastic side of scenario tree NMPC.

7.1 Markovian Scenario Tree Process

As mentioned above we interpret the uncertain parameter values as realizations of a Markov chain with finite state space. The following definition recalls the main stochastic principle of a Markov chain, namely the dependence of a state only on the previous state.

Definition 7.1 (Markov chain)**.** *Let $E \neq \emptyset$ be a countable set and $\Pi = (\Pi(x,y))_{x,y\in E}$ a stochastic matrix. A sequence of E-valued random variables $X_0, X_1, \ldots$ on a probability space $(\Omega, \mathcal{F}, P)$ is a Markov chain with state space E and*

transition matrix Π*, if for all* $n \geq 0$ *and* $x_0, \ldots, x_{n+1}$ *the equation*

$$\begin{aligned} P(X_{n+1} = x_{n+1} | X_0 = x_0, \ldots, X_n = x_n) \\ &= P(X_{n+1} = x_{n+1} | X_n = x_n) \\ &= \Pi(x_n, x_{n+1}) \end{aligned}$$

holds with $P(X_0 = x_0, \ldots, X_n = x_n) > 0$.

The full combinatorial scenario tree can be interpreted now in the sense that every scenario represents a realization of the sequence of these random variables forming the Markov chain. In Figure 7.1 we illustrate a Markov chain with transition probabilities and in Figure 7.2 its corresponding scenario tree starting from the state 2. We get the probability of a scenario that is a path from the root to a leaf of the tree by multiplying the transition probabilities along the path.

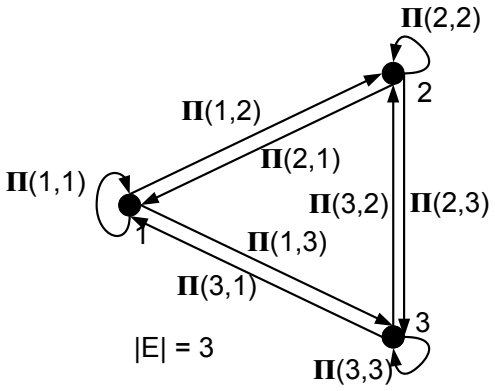

Figure 7.1: Transition graph of a Markov chain with state space $E = \{1,2,3\}$. The three states are represented by the vertices. Possible transitions are represented by the edges. The transition probabilities from i to j are the matrix elements $\Pi(i, j)$.

7.2 Invariant Distribution as Initial Distribution

We can already deduce from the tree depicted in Figure 7.2 that the initial distribution of X_0 influences the scenario probabilities. In our NMPC framework we compute the solution of a mathematical optimization problem depending on the initial state for every NMPC iteration. Translated to the Markov chain scenario tree approach this would mean, that we have a tree for every initial state in the state space E. The resulting forest is undesirable for fast numerical methods exploiting the structure of subsequent optimization problems.

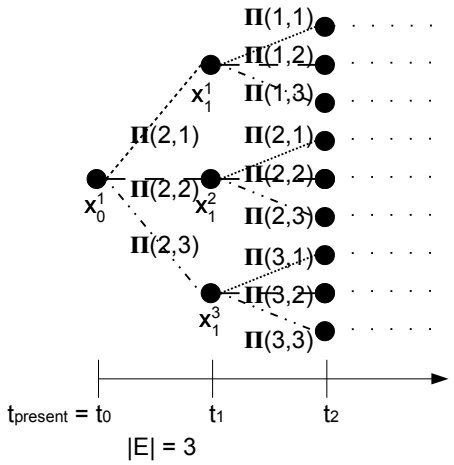

Figure 7.2: A scenario tree with 3 states and 2 decision points. Branch probabilities are the transition probabilities of the Markov chain with transition matrix Π. The tree illustrates all possible sequences of the Markov chain with length 3 and initial distribution $P(X_0 = 2) = 1$.

On the one hand taking into account the whole forest would generate a large forest-size optimization problem that takes longer to solve than a single tree-structured optimization problem. On the other hand switching between trees depending on the current estimate of the state would slow down NMPC schemes that exploit the structure of the optimization problems of subsequent NMPC iterations. Therefore, if the initial distribution or initial state is not determined, we have to make a choice for the initial distribution of the Markov chain before constructing the tree.

A natural choice is the invariant measure of the Markov chain that describes the long-term stochastic behavior and can easily be computed. The following theorem is known from ergodic theory and formulated for example in the Markov chain chapters of [31, 68].

Theorem 7.1. *Let μ be an invariant measure of the Markov chain Π on the finite set E, i.e. for all $x \in E$*

$$\sum_{y \in E} \mu(y)\Pi(y,x) = \mu(x). \tag{7.1}$$

Furthermore, let there exist a $k \geq 1$ with $\Pi^k(x,y) > 0$ for all $x, y \in E$.
Then for all $y \in E$ the limit

$$\lim_{n \to \infty} \Pi^n(x,y) = \mu(y) > 0$$

exists independently of the choice of $x \in E$, and the limit μ is the unique invariant measure of the Markov chain Π on E.

Thus, the invariant measure μ is a natural choice, because it is the limit distribution of the Markov chain. From equation (7.1) we deduce that the invariant measure is easily computed as the left eigenvector of Π to the eigenvalue 1.

7.3 Tree Pruning Algorithm

The full scenario tree, which branches at every stage, grows exponentially in the number of decision points. Hence, we would like to choose a subtree for practical computations. It is desirable to cut out the scenarios with lowest probability such that the scenario probabilities of the remaining tree sum up to a coverage probability p_{coverage} or such that a maximum number of scenarios $k_{\max}$ is reached. The naive approach to get the subtree would be to construct the full combinatorial tree, compute all scenario probabilities and then prune the tree. But due to the combinatorial complexity, this so-called full enumeration approach might become very slow or even impossible for a large number of decision points. We note here that there is a maximum of decision points, the prediction horizon length, because our intention is to use the tree for the optimization problem in the prediction task.

We propose an efficient algorithm to generate a scenario tree from a Markov chain with $|E| = m < \infty$ states in the order of non-increasing scenario probabilities . We are interested in a scenario tree of depth $n_{\text{levels}} \in \mathbb{N}$ without full enumeration of all possible $m^{n_{\text{levels}}}$ scenarios. To describe scenarios, we use a tuple notation for the vertices of the complete scenario tree. For any k-tuple $s = (e_1, \ldots, e_k)$, $e_i \in E$, we denote its length by $|s| = k$. Two tuples over E are concatenated with the $\oplus$-operation according to

$$\left(e_1^1, \ldots, e_{k_1}^1\right) \oplus \left(e_1^2, \ldots, e_{k_2}^2\right) = \left(e_1^1, \ldots, e_{k_1}^1, e_1^2, \ldots, e_{k_2}^2\right).$$

The root of the tree is given by the 1-tuple (o) for some $o \in E$. The full scenario tree of length n_{levels} is denoted by $\mathcal{T}(n_{\text{levels}})$. We characterize $\mathcal{T}(n_{\text{levels}})$ by $(o) \in \mathcal{T}(n_{\text{levels}})$ and the implication

$$\exists s \in \mathcal{T}(n_{\text{levels}}) \text{ with } |s| < n_{\text{levels}} \;\Rightarrow\; s \oplus (e) \in \mathcal{T}(n_{\text{levels}}) \;\forall e \in E.$$

Thus, $\mathcal{T}(n_{\text{levels}})$ is a set of tuples of length less than or equal to n_{levels}, which are exactly the vertices of the complete scenario tree. The labeling with tuples implicitly encodes the connectivity of the vertices, e.g., the tuple $(1,1,3,2)$ encodes that we arrive at the vertex by starting from realization 1, staying in 1, then jumping to 3 and finally to 2. We can assign to each vertex $s = \left(e_1, \ldots, e_{|s|}\right) \in \mathcal{T}(n_{\text{levels}})$ a probability

$$p(s) = \prod_{i=2}^{|s|} \Pi(e_{i-1}, e_i) \tag{7.2}$$

by multiplying the transition probabilities. At this point we formulate Algorithm 5 for the scenario tree generation. It requires the Markov chain transition matrix Π, the scenario length n_{levels}, the coverage probability $p_{\text{coverage}} \in [0,1]$ and maximum number of scenarios $k_{\max} \in \mathbb{N}$ as inputs. The scenario length n_{levels} should be smaller than or equal to the prediction horizon.

Algorithm 5 Scenario tree generation

Input: Π, n_{levels}, p_{coverage}, $k_{\max}$

Output: Set of scenarios with length n_{levels}

1: $\mathcal{R} \leftarrow \{(o)\}, k \leftarrow 0$
2: **while** $\mathcal{R} \neq \varnothing$ **do**
3: Choose $v = \operatorname{argmax}_{s \in \mathcal{R}} p(s)$ (cf. (7.2) using Π)
4: $\mathcal{R} \leftarrow \mathcal{R} \setminus \{v\}$
5: **if** $|v| < n_{\text{levels}}$ **then**
6: $\mathcal{R} \leftarrow \mathcal{R} \cup \{v \oplus (e) \mid e \in E\}$
7: **else**
8: $k \leftarrow k+1, s_k \leftarrow v$
9: **if** $\sum_{i=1}^{k} p(s_i) \geq p_{\text{coverage}}$ **or** $k \geq k_{\max}$ **then**
10: **return** Set of scenarios $\{s_i\}_{i=1,\ldots,k}$
11: **end if**
12: **end if**
13: **end while**

Algorithm 5 successively updates a subtree, given implicitly by its potential leaves $\mathcal{R} \in \mathcal{T}(n_{\text{levels}})$ that are reachable after one step, in a greedy fashion. In the main loop we choose always one additional leaf with highest probability at a time. Scenarios of length n_{levels} will be enumerated as a finite sequence (s_k) as soon as they are encountered . The algorithm terminates after line 9 if the maximum number of scenarios $k \geq k_{\max}$ or the coverage probability

$$\sum_{i=1}^{k} p(s_i) \geq p_{\text{coverage}} \in [0,1]$$

is satisfied. Algorithm 5 can be implemented efficiently by representing the set $\mathcal{R}$ by a heap data structure. This data structure can provide the maximum entry in constant time and perform the removal and insertion of additional elements in logarithmic time complexity. Further extensions of the algorithm are in preparation [73].

7.4 Constraint Satisfaction

In optimization under uncertainty one essential question is how well the proposed method ensures feasibility of the constraints. We recall that that the underlying uncertain parameter process is a discrete Markov process. Then, if we know the transition probabilities of the uncertainty and assume that $k < k_{\max}$, the proposed Algorithm 5 will ensure constraint satisfaction with a probability of at least p_{coverage} for each NMPC subproblem.

Let us start from the full combinatorial tree representing all possible sequences of realizations of the Markov chain up to a certain time horizon. The algorithm yields a subset of the full tree. This subtree consists of scenarios with cumulative probability of at least p_{coverage} if the algorithm does not terminate due to $k \geq k_{\max}$. If the transition matrix Π coincides with the underlying Markov chain of the real uncertainty, Algorithm 5 covers at least p_{coverage} of all possible realizations of the uncertainty up to the time horizon restricted by the tree length. As our numerical methods provide a solution that is feasible for all considered scenarios, the tree resulting from Algorithm 5 ensures a probability of p_{coverage} of constraint satisfaction.

7.5 Constructing a Markov Chain from a Distribution

A discrete-time and finite state Markov process is characterized by the transition matrix and an initial distribution. If both are explicitly indicated by the application, a scenario tree can be immediately constructed according to Algorithm 5. However, often only the uncertain parameter and its probability distribution are specified. In this case the distribution can serve as an initial distribution for the Markov chain but still the transition matrix is required. As there is no obvious principle on how to assemble the matrix, we describe in the following two methods to compute the matrix elements assuming that $|E| < \infty$.

Transition Matrix from the Solution of an Optimization Problem

We regard the elements of the transition matrix Π for the finite number of states as variables and formulate an optimization problem with constraints that model the desired properties of the transition matrix.

First, Π must be a stochastic matrix, that is

$$\sum_j \Pi_{ij} = 1 \text{ for all } i, \tag{7.3}$$

$$\Pi_{ij} \geq 0 \text{ for all } i, j. \tag{7.4}$$

Second, we want to ensure that the given distribution v is an invariant distribution of the chain,

$$v\Pi = v. \tag{7.5}$$

Third, convergence of the Markov chain to the distribution v due to Theorem 7.1 is guaranteed if the Markov chain is aperiodic and irreducible. Formulating these properties as constraints would introduce combinatorial complexity to the problem. Therefore we leave them out and test the properties in the solution. Furthermore, we can formulate sparsity constraints by setting entries of the matrix Π to zero. Finally, in some cases symmetry is desirable,

$$\Pi_{ij} = \Pi_{ji} \text{ for all } j > i. \tag{7.6}$$

For the objective function of the optimization problem we suggest to minimize the linear objective $\sum_{i,j} \Pi_{ij}$ that yields an LP or a quadratic objective $\sum_{i,j} \Pi_{ij}^2$ that yields a QP. Both penalize all entries of the transition matrix. Also nonlinear functions can be considered as objective. Minimizing for example the second eigenvalue of the transition matrix would yield a faster convergence of the Markov chain to the invariant distribution. However, the resulting NLPs are harder to solve than the QPs or LPs and we keep in mind that computation of the transition matrix is just a subproblem before the actual tree creation and application as a scenario tree controller. A cheaper method for transition matrix creation is presented in the following section.

Metropolis Matrix as Transition Matrix

Originating from the field of statistics, Markov Chain Monte Carlo (MCMC) methods are Markov chain based algorithms for sampling from a probability distribution. The considered Markov chain has the target distribution as invariant distribution. The idea behind the statistical methods can be adapted to assemble our transition matrix. In MCMC the state of the chain after a number of steps is used as a sample of the desired distribution. One of the algorithms is Metropolis-Hastings MCMC [57]. The core object is the Metropolis matrix that is constructed according to Algorithm 6.

Algorithm 6 Assembling the Metropolis matrix

Input: Probability density function f, state discretization $x \in \mathbb{R}^m$, reference transition matrix $Q \in \mathbb{R}^{m \times m}$

Output: Metropolis matrix M

$M = 0 \in \mathbb{R}^{m \times m}$
for $i = 1, \ldots, m$ **do**
  for $j = 1, \ldots, m$ **do**
    if $j \neq i$ and $Q(i,j) > 0$ **then**
      $M(i,j) = Q(i,j) \min\left\{1, \frac{f(x(j))Q(j,i)}{f(x(i))Q(i,j)}\right\}$
    end if
  end for
  $M(i,i) = 1 - \sum_{j=1}^{m} M(i,j)$
end for
return M

The matrix M is a stochastic matrix with an invariant distribution that approximates f. Aperiodicity and irreducibility are guaranteed if the reference Markov chain Q is aperiodic and irreducible. Before assigning the matrix elements of Q, we first set the structure of Q that ensures aperiodicity and irreducibility. For instance, a tridiagonal or pentadiagonal structure can be chosen. Then we define Q row-wise. For every row we fill the non-zero entries according to the uniform distribution on exactly these non-zero row entries. After assembling Q we can compute the Metropolis matrix as transition matrix for the Markov chain using Algorithm 6 and construct the scenario tree according to Algorithm 5.

7.6 Scenario Tree Examples

In this section we assemble Markov transition matrices by the ideas presented in the previous sections and then construct scenario trees according to Algorithm 5. We consider three realizations of an uncertain parameter with distribution $v = (0.1, 0.5, 0.4)$. In Table 7.1 we list the resulting transition matrices from Algorithm 6 (Metropolis) with tridiagonal reference matrix

$$Q = \begin{pmatrix} 0.6667 & 0.3333 & 0 \\ 0.3333 & 0.3333 & 0.3333 \\ 0 & 0.3333 & 0.6667 \end{pmatrix}.$$

and the matrices from the LP solution as well as from the QP solution.

Table 7.1: Transition matrices

Assembling type	Transition matrix	Stationary distribution
Metropolis (M)	$\begin{pmatrix} 0.6667 & 0.3333 & 0 \\ 0.0667 & 0.6667 & 0.2667 \\ 0 & 0.3333 & 0.6667 \end{pmatrix}$	$(0.1, 0.5, 0.4)$
LP	$\begin{pmatrix} 0.7264 & 0.2 & 0.0736 \\ 0.0264 & 0.8 & 0.1736 \\ 0.0354 & 0.2 & 0.7646 \end{pmatrix}$	$(0.1, 0.5, 0.4)$
QP	$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$	initial distr. of Markov chain

The metropolis matrix M in the second column of Table 7.1 exhibits tridiagonal structure as the reference matrix Q. The diagonal dominates, the chain stays at every state with probability 0.6667. In the third row the solution of the LP also shows diagonal-dominance, aperiodicity and irreducibility can be verified. Both examples underline that the solution to the problem of assembling the transition matrix is not unique. The invariant distribution is the target distribution v in both cases. The solution to the QP in the fourth table row yields the identity matrix. We remark at this point that imposing sparsity constraints on the LP in order to get a tridiagonal matrix also yields the identity matrix in this example. The Markov chain with identity transition matrix is not irreducible, as it stays always in the current state. Furthermore, the invariant distribution of such a chain depends of the initial distribution. In fact it equals the initialization of the Markov chain. In the following we consider the first and second transition matrix and construct the trees according to Algorithm 5.

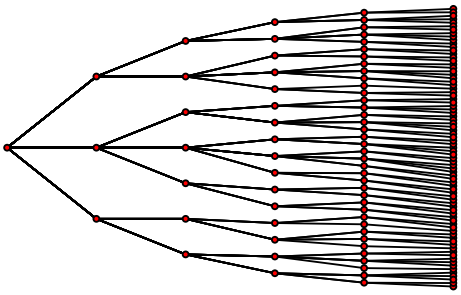

Figure 7.3: The left scenario tree results from 5 with $p_{\text{coverage}} = 0.99$, $n_{\text{levels}} = 5$ and the Metropolis transition matrix as inputs. The tree consists of 81 scenarios covering 99,0814 % of the uncertainty. We save 67 % of scenarios with respect to the full combinatorial tree of 243 scenarios.

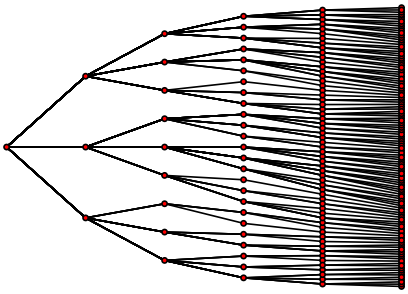

Figure 7.4: The right scenario tree results from 5 with $p_{\text{coverage}} = 0.99$, $n_{\text{levels}} = 5$ and the LP transition matrix as inputs. The tree consists of 122 scenarios covering 99,0036 % of the uncertainty. We save almost 50 % of scenarios with respect to the full combinatorial tree of 243 scenarios.

In Figure 7.3 and 7.4 we depict the scenario trees constructed with Algorithm 5 and parameters $p_{\text{coverage}} = 0.99$, $n_{\text{levels}} = 5$, $k_{\max} = \infty$. On the left the Metropolis transition matrix is used and on the right the LP transition matrix is used. The resulting tree structure depends on the sparsity structure of the transition matrices. In this example the Metropolis approach with sparsity structure predefined by the reference chain saves more scenarios than the LP approach.

7.7 Approximation Error of the Examples

In the last section we investigate the influence of the probability p_{coverage} on the number of scenarios and the approximation error. Therefore we consider the two Markov chain transition matrices from the previous section and construct trees according to Algorithm 5 depending on p_{coverage}. We define the approximation error as the cumulative probability of all pruned scenarios.

Figure 7.5 shows the number of scenarios and the approximation error for $n_{\text{levels}} = 10$. All scenarios of the various trees constructed with probability p_{coverage} have length 10, which is already a large number of stages if we recall that the full combinatorial tree has $3^{10} = 59049$ scenarios. From the left figure we deduce that the number of scenarios decreases gradually with increasing p_{coverage} for both choices of transition matrices with better performance for the Metropolis case in this example. The decrease is on a logarithmic scale implying the large savings and high potential of the scenario tree constructing algorithm. On the right the error decreases with higher probability p_{coverage}. For Figure 7.6 we have

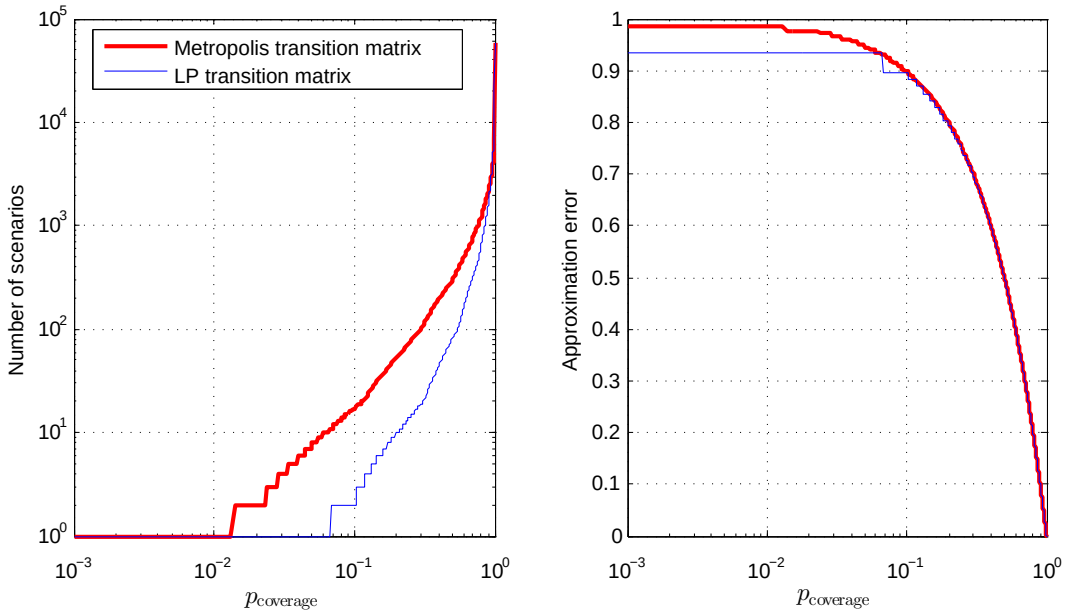

Figure 7.5: Tree pruning algorithm results: Number of scenarios (left) and approximation error (right) for scenario length $n_{\text{levels}} = 10$ and three parameter realizations.

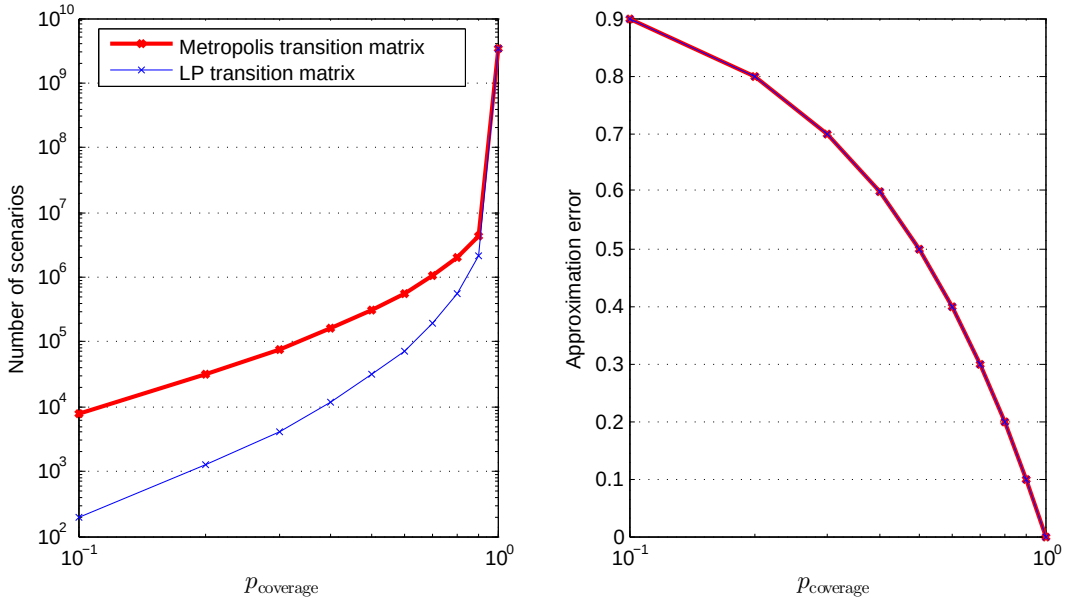

Figure 7.6: Tree pruning algorithm results: Number of scenarios (left) and approximation error (right) for scenario length $n_{\text{levels}} = 20$ and three parameter realizations.

increased the scenario length to $n_{\text{levels}} = 20$ and computed 9 trees for $p_{\text{coverage}} = 0.1, \ldots, 0.9$. Using trees of 20 stages corresponds to a state-of-the-art prediction horizon of 20 for scenario tree NMPC. The full combinatorial tree would consist of 3^{20} scenarios, equal to more than 3 billion scenarios. Such a large number exceeds the power of numerical methods for scenario tree NMPC. In the left figure again the number of scenarios decreases gradually with increasing probability p_{coverage} on a logarithmic scale. A probability $p_{\text{coverage}} = 0.5$ for example yields trees with thousands of scenarios. This is a significant saving compared to the full tree with billions of scenarios. Constructing such large trees can be done offline before starting the actual scenario tree NMPC. The approximation error on the right again decreases with higher probability p_{coverage} for both transition matrices.

7.8 Summary

In this chapter we have presented a scenario tree generation algorithm based on Markov chains. The algorithm generates the tree without full enumeration of all possible scenarios. Due to the Markov assumption the resulting tree depends on the initial distribution and transition matrix of the Markov chain. If only a distribution of the uncertain parameter is provided, we regard it as the invariant distribution of a Markov chain and compute the transition matrix with a Metropolis approach or as a solution of an optimization problem. We have illustrated the methods with example trees and discussed the high potential of our contribution by staging the extraordinary savings in the number of scenarios and the approximation error induced by the coverage.

Chapter 8

Implementation

In this chapter we present the software implementation of the numerical methods discussed in this thesis. The STMLI package (Scenario Tree Multi Level Iteration package) is the scenario tree extension of the MLI software (Multi Level Iteration software). The scenario tree NMPC methods have been implemented by the author as part of the doctoral studies to demonstrate the effectiveness and efficiency of the methods. This chapter shall serve as an overview of STMLI. A discussion of numerical results using the software follows in Chapter 9.

8.1 Design Decisions

We have implemented the structure exploiting methods for scenario tree NMPC in the numerical computing environment MATLAB by Mathworks [86]. The software MLI provides the NMPC part, especially the Multi-Level Iteration scheme [15], and STMLI extends it by the scenario tree related features. A short overview of MLI follows in the next section.

MLI

In MLI there is on the one hand a simulator that acts as the real plant and propagates a specified model of the reality. On the other hand, there is the optimizer that is based on the Multi-Level Iteration scheme. Prediction as well as estimation tasks along the NMPC paradigm can be performed with MLI. An overview of the architecture of MLI is depicted in Figure 8.1.

We emphasize that the models of optimizer, estimator and simulator are independent and the effects of model-plant mismatch can be investigated. The data of simulator, optimizer and estimator are also stored separately, even each phase of the MLI scheme operates on its own data. Communication between the phases and between optimizer, estimator and

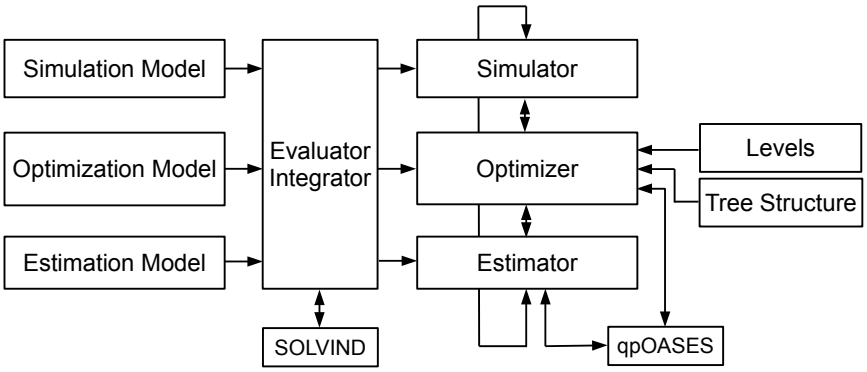

Figure 8.1: MLI has a modular architecture. The Tree Structure module is the essential module required for the STMLI part of the optimizer that runs the non-smooth Newton method from the dual decomposition approach.

simulator is implemented according to the NMPC and MLI scheme. In this section we focus primarily on the optimizer as this is the module where the scenario tree approach acts to robustify against the uncertainty. Depending on the current phase the optimizer assembles and solves QPs and/or gives immediate feedback. The most time critical operations such as sensitivity calculations and dense QP solutions are performed by the software SolvIND and qpOASES written in C/C++.

SolvIND

SolvIND is a suite of ODE/DAE Solvers capable of generating arbitrary order forward and adjoint derivatives of IVPs using the principle of internal numerical differentiation [5, 6, 1]. The SolvIND suite contains the integrator DAESOL-II based on BDF methods for stiff ODEs and index 1 DAEs [22, 47]. Furthermore, the integrator family RKFSWT based on Runge-Kutta-Fehlberg methods of different order [99, 72, 19, 32, 33] is part of SolvIND. Model function derivatives are computed using the principle of Automatic Differentiation via built-in ADOL-C (Automatic Differentiation by OverLoading in C) [50]. ADOL-C uses operator overloading to differentiate C and C++ programs. SolvIND comes with a MATLAB interface that we use to compute derivatives and sensitivities during the assembly of the optimization problems in scenario tree NMPC.

qpOASES

The software qpOASES [36] is an open-source implementation of the online active set strategy [35]. It is inspired by observations from the field of parametric quadratic programming. In qpOASES the expectation that the optimal active set does not change much from one

sampling time to the next is exploited. Thus, the online active set strategy is particularly suited for model predictive control applications. The developer of qpOASES provide a MATLAB interface that we use to solve the local scenario QPs within the dual decomposition procedure.

STMLI

The extension module of MLI by the scenario tree robustification approach is called STMLI. It implements the dual decomposition presented in Chapter 5. All design decisions of MLI are passed to STMLI. The data of STMLI for scenario tree NMPC consists of the uncertain parameter specification, the tree topology and algorithmic parameters of the dual decomposition and non-smooth Newton method. Our implementation follows the branchwise view on the scenario tree. We store the data for each scenario and the coupling structure that forms the tree.

8.2 Numerical Methods

From the methodological perspective we perform in STMLI the dual decomposition approach with a non-smooth Newton method. For the decoupled inner QPs we employ the online active set strategy. As they are representing one scenario in the branchwise view we employ standard condensing. For integration and sensitivity computation we can switch between methods based on backward difference formulas or Runge-Kutta-Fehlberg formulas. We point out that here the nodewise view of the tree is advantegous for all dynamical evolution computations.

Furthermore it is possible to use positive definite approximations of the inner QP Hessians by Limited Memory BFGS updates [87, 80]. The resulting strictly convex QPs are required for the dual decomposition approach. Another feature is the Control Move Regularization (CMR) to penalize the difference of subsequent controls.

8.3 Setup of Problems

The MLI software is available from the Model-Based Optimizing Control group at IWR, Heidelberg University. It requires a MATLAB installation, the SolvIND package and the software qpOASES.

For the setup of a NMPC problem we provide first a SolvIND model formulated in C++ that is used for integration and sensitivity calculation. Then the simulator, the controller and the scenario tree are initialized as described in the following.

Initializing the simulation

In `scenario_initialize` we define the initial values for the simulation, the SolvIND model for the simulator and an integrator supported by SolvIND. Furthermore, we set the sampling grid of the NMPC scheme and set flags to turn on or off the controller and the estimator.

Initializing the controller

For NMPC we need to set up the optimization environment in `controller_initialize`. We specify the prediction horizon and the discretization grid, especially the number of multiple shooting points. For the QP solution the initial values, bounds, scaling factors and the condensing flag are required. The Hessians can be calculated using second-order derivatives or L-BFGS updates. Algorithmic parameters for the control move regularization are also specified in `controller_initialize`. Furthermore, we define the MLI scheme.

Initializing the scenario tree

The extension module STMLI requires a specification of the scenario tree structure in `tree_initialize`. We define the finite set of uncertain parameters as basis of the tree, the robust horizon and weights for all scenarios. Then we specify the parameters of the non-smooth Newton method and specify how to manage the resulting data of the scenario tree optimization problems.

Initializing further features

We set further the plant simulation behavior (`user_scen_update`) and the interaction between optimizer, estimator and simulator regarding states and parameters in `user_opt_x_from_scen, user_opt_p_from_scen, user_est_p_from_scen`. In case of state or parameter estimation the estimator settings are in `estimator_initialize`. For visualization, the figures and plotting data is set via `user_plot, user_plot_init` with the full flexibility of the MATLAB plotting tools.

Running MLI

Finally, running the problem requires only a function call `runMLI()` or `runSTMLI()` in MATLAB with the specific problem name as argument.

8.4 Summary

We have introduced STMLI as an extension of the MLI software that is executable in the numerical computing environment MATLAB. The tree module is essentially based on the dual decomposition strategy of Chapter 5. We emphasize that all features of MLI are passed to STMLI, especially the data storage, discretization technique and interfacing to software that performs time critical operations such as sensitivity computation and QP solution.

Chapter 9

Numerical Results

In this chapter we demonstrate effectiveness and efficiency of the numerical methods developed in the previous chapters. For this purpose we apply our theory and algorithms to challenging problems. First, we consider a Continuous Stirred Tank Reactor. Our powerful structure exploiting numeric methods yield successful performance of scenario tree NMPC with up to 1001 scenarios. The second problem is an industrial optimization problem, a Biochemical Batch Reactor model provided by BASF. Already for small trees the optimization problems on the prediction horizon become large-scale and hard to solve due to the highly nonlinear system model. We demonstrate how the sparse grid approach reduces the scenario tree size without sacrificing controller performance. In the third section we consider a pharmaceutical problem from Penicillin production, and point out how we cope with special properties of the nonlinear dynamical system. All problems have in common that there is an uncertainty present in the dynamical system model. Our results do not only illustrate the performance and robustness of scenario tree NMPC against the uncertainty, but also emphasize that numerical structure exploitation in the form of our dual decomposition approach and the generation of suitable scenario trees like the quadrature-based sparse trees are required for real-time feasible scenario tree NMPC.

9.1 Continuous Stirred Tank Reactor

We consider a nonlinear, continuous-time stirred tank reactor (CSTR) problem. Inside a tank with cooling system the chemical A reacts to the chemical B. We control the reaction by the cooling temperature and aim to track the setting point. The CSTR dynamics have been studied already in [110], an extended version in [20] and the CSTR in context of NMPC under uncertainty in [113].

Dynamical system model

In the following we describe the model representing a first-order, irreversible, exothermic reaction $A \rightarrow B$ of the CSTR. The mass and energy balances of the reaction are described by two nonlinear differential equations.

$$\frac{dc_A}{dt} = \frac{q}{V}(c_{\mathrm{Af}} - c_A) - k_0 \exp\left(-\frac{E}{RT}\right) c_A \tag{9.1a}$$

$$\frac{d\theta}{dt} = \frac{q}{V}(\theta_f - \theta) - \frac{\Delta H k_0}{\rho C_p} \exp\left(-\frac{E}{R\theta}\right) c_A + \frac{UA}{VC_p}(\theta - \theta_C) + \tilde{\theta} \tag{9.1b}$$

The two state variables are reactant concentration c_A and reactor temperature θ. The control variable is the cooling water temperature θ_C. All other symbols are listed in Table 9.1.

Table 9.1: CSTR model parameters, bounds, initial values

k_0	7.2e10	1/min	Reaction rate constant
E	72.752e3	J/mol	Activation energy
R	8.314	J/mol/K	Ideal gas constant
ΔH	-5e4	J/mol	Heat of reaction
ρ	1000	g/l	Mass density
C_p	0.239	J/g/K	Heat capacity
$U \cdot A$	5e4	J/mol	Overall heat transfer coefficient $\times$ Reaction area
q	100	l/min	Feed stream flow rate
θ_f	350	K	Actual feed temperature
V	100	l	Reactor volume
c_{Af}	1.0	mol/l	Feed concentration
$\overline{\theta_C}$	370	K	Coolant upper bound
$\underline{\theta_C}$	280	K	Coolant lower bound
c_A^0	0.4	mol/l	Initial concentration
θ^0	360	K	Initial reactor temperature

Uncertainty model

The disturbance of the temperature by unmodeled influences is regarded as a probabilistic uncertainty. We model the uncertainty by the term $\tilde{\theta}$ in (9.1b). The unit of $\tilde{\theta}$ is

Kelvin per minute, therefore it describes an uncertain temperature change. We assume that $\tilde{\theta}$ is normally distributed with zero mean and standard deviation of 0.1. The uncertain parameter $\tilde{\theta}$ is approximated by a scenario tree. For our numerical experiments with scenario tree NMPC we consider different robust horizons and numbers of scenarios. The main focus of the CSTR example is to show that our numerical methods can cope with large scenario trees. Therefore we consider scenario trees consisting of up to 1001 scenarios.

Simulation Setup

The simulation model is the dynamical system (9.1). We simulate 60 minutes of the reaction. For the uncertain parameter $\tilde{\theta}$ we draw samples from a normal distribution with zero mean and standard deviation of 0.1 and concatenate them to a trajectory, see Table 9.2. Every 10 minutes the parameter jumps. We remark that the controller does not have this information about the uncertain parameter. Therefore we can test the robustness.

Table 9.2: Sampled parameter trajectory of the CSTR

Time [min]	0	10	20	30	40	50
Parameter value $\tilde{\theta}$ [K/min]	-0.0840	0.0319	0.0124	0.0056	0.0294	0.0293

The initial values for the simulation are a reactant concentration of $c_A^0 = 0.4$ mol/l and a reactor temperature of $\theta^0 = 360$ K.

Controller Setup

The performance criterion is of tracking type with setpoint $c_A^{\text{set}} = 0.5$ mol/l, $\theta^{\text{set}} = 350$ K, and $\theta_C^{\text{set}} = 300$ K. Note that the setpoint differs from the initial value of the simulation. The objective function reads

$$\int_{t_0}^{t_f} (c_A - c_A^{\text{set}})^2 + (\theta - \theta^{\text{set}})^2 + (\theta_C - \theta_C^{\text{set}})^2 dt. \tag{9.2}$$

We aim to minimize the term (9.2) with respect to the dynamics (9.1). For NMPC we solve the optimization problem by a direct multiple shooting discretization with 10 nodes on a prediction horizon of 2 minutes. We perform one RTI per NMPC sampling time of 6 seconds and solve the tree QP with the dual decomposition approach. The QP is assembled using second order derivative information for the Hessian and every branch QP is condensed.

Scenario Tree Setup

For the uncertain parameter $\tilde{\theta}$ we consider first the realizations $\{-1.0, -0.5, 0.0, 0.5, 1.0\}$. The size of the scenario trees is then determined by the robust horizon n_d. We start from $n_d = 0$, which means nominal NMPC with the nominal realization $\tilde{\theta} = 0$ and continue until $n_d = 3$. In the end we carry out scenario tree NMPC with 1001 equidistant realizations in the interval $[-1, 1]$ and robust horizon $n_d = 1$. This is a very dense approximation of the continuous uncertainty within the bounds $\mu - 10\sigma$ and $\mu + 10\sigma$ since we assume $\tilde{\theta} \sim \mathcal{N}(0, 0.1)$. The probability of the tails of this distribution that lie outside the considered interval $[-1, 1]$ are much smaller than machine precision, because we already have $P(\tilde{\theta} \notin [-0.8, 0.8]) = 1.2 \cdot 10^{-15}$. We summarize the scenario tree setup and the resulting tree QP size in Table 9.3.

Table 9.3: CSTR: Scenario tree data

Scenario tree label	0	1	2	3	4
Robust horizon	0	1	2	3	1
Number of scenarios	1	5	25	125	1001
Non-anticipativity constraints	0	4	44	344	1000
Tree QP size:					
Number of variables	43	215	1075	5375	43043
Number of equality constraints	33	169	869	4469	34033
Number of inequality constraints	80	400	2000	10000	80080

Tree Structure of the Prediction Problem

The size of the tree-structured QP that is solved in every NMPC iteration grows with the number of scenarios as we can see in Table 9.3. When we look at the solution of one tree-structured QP, the branching structure becomes visible, see Figure 9.1. Due to the non-anticipativity constraints on the controls we can see for Tree 2 in the second column of Figure 9.1 one control at the first, 5 controls on the second and 25 controls on the third stage. In the last column of Figure 9.1 (Tree 4) the scenarios cannot be distinguished. It underlines the very dense parameter realization approximation to the continuous parameter distribution in case of Tree 4.

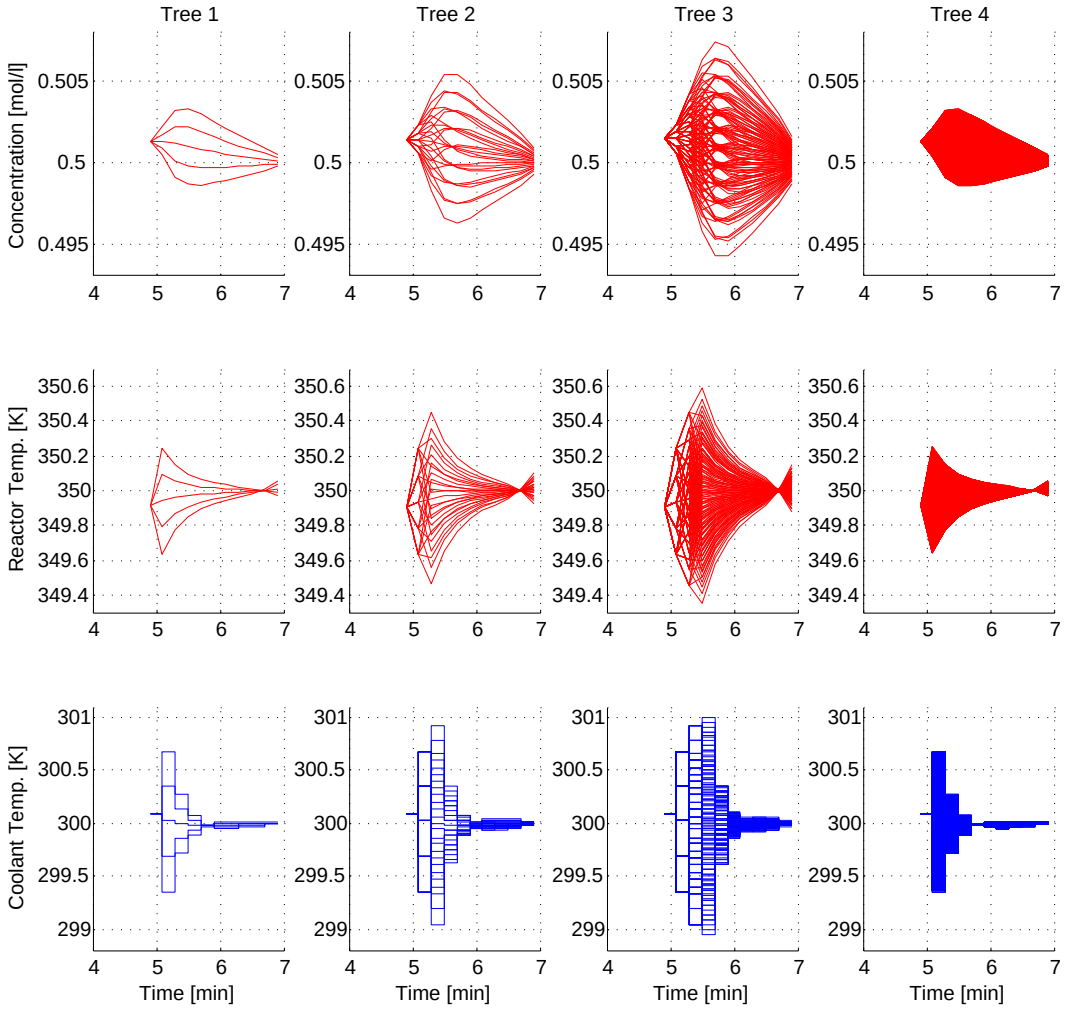

Figure 9.1: Solutions of the prediction problem: At time $t = 4.9$ min the solutions of the prediction problem exhibit the typical branching structure of the trees due to the non-anticipativity constraints.

Resulting Trajectories

The resulting trajectories of the scenario tree NMPC controllers for Tree 0 to Tree 4 are depicted in Figure 9.2. The concentration of reactant A starts for all trees at the initial value c_A^0 and is kept at the setpoint c_A^{set} after 5 minutes. All the 5 different controllers drive the reactor temperature from the initial θ^0 to the setpoint temperature θ^{set}. The control trajectory starts for all scenario trees at the lower bound 280 K and increases to the coolant temperature setpoint 300 K. The trajectories are feasible at all times and the setpoint is well tracked although the uncertain parameter jumps.

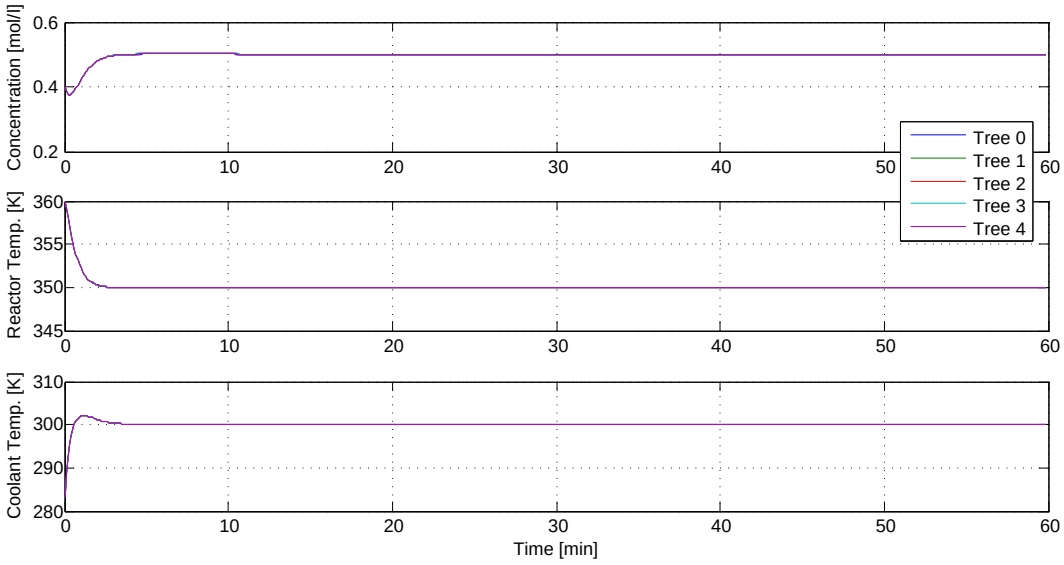

Figure 9.2: CSTR trajectories resulting from scenario tree NMPC: For all scenario trees the concentrations and temperature of the controlled reactor start from their initial values and arrive at the setpoint after 5 minutes. Although the uncertain parameter jumps, the setpoint is tracked nicely by the different controllers.

Non-smooth Newton Iterations

We solve every tree QP with the dual decomposition approach and employ a non-smooth Newton method with accelerated bisection line search. The number of iterations per NMPC iteration is crucial for fast feedback as we solve one tree QP per NMPC iteration according to the RTI scheme. We have chosen to perform one NMPC iteration every 6 seconds. Due to this fast sampling time we subsequently solve similar QPs. Therefore almost always one non-smooth Newton iteration suffices for all trees. The number of non-smooth Newton

iterations per NMPC iteration over the whole control horizon of one hour is depicted in Figure 9.3. The number of non-smooth Newton iterations for the controllers with different scenario trees are stacked along the y-axis for better visibility. Almost always we see one iteration, sometimes even zero iterations for the smallers trees. This happens if the stopping criterion for the Newton loop is already met at the beginning. The norm of the Newton gradient is already below the given ε. We can clearly observe that the uncertain parameter jump every 10 minutes, i.e. every 100 NMPC iterations, influences the number of iterations. If we were at 0 iterations for the smaller trees, we again require one iteration to react to the changed system. At the 400th iteration the largest tree requires 2 iterations. However, these iteration numbers are in general very low due to the fast NMPC sampling time. Therefore the fine sampling grid enables us to give fast feedback.

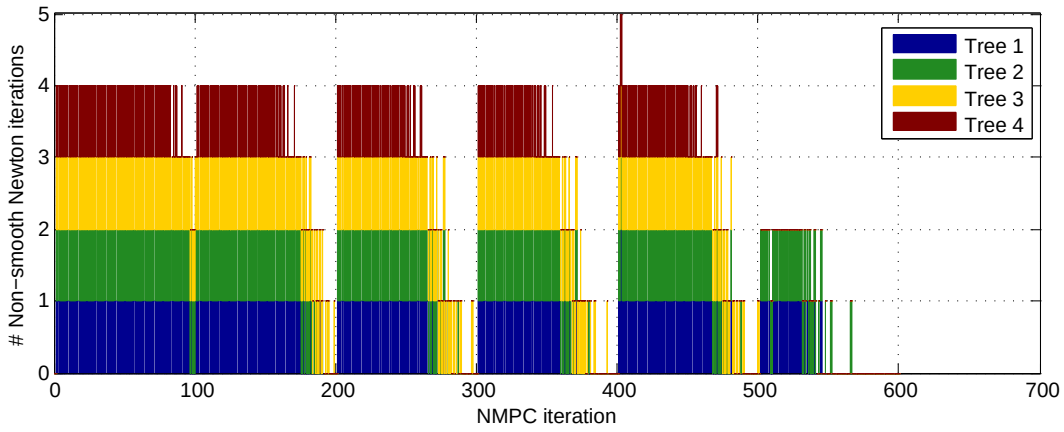

Figure 9.3: Number of non-smooth Newton iterations per NMPC iteration for the CSTR: The number of non-smooth Newton iterations is almost always one or even zero for all NMPC iterations, independent of the scenario tree.

Real-Time Feasibility

We require a real-time feasible controller to steer the chemical system, meaning that the computational time of one NMPC iteration remains below the sampling time. In our case the crucial time is 6 seconds. In Figure 9.4 the computation times per NMPC iteration are shown. Clearly the largest tree with 1001 scenarios requires the highest computation time and the single nominal scenario the lowest for all NMPC iterations. Along the x-axis we see a similar behavior for all trees. At the beginning the computation time decreases with increasing NMPC iteration. As the trajectories get closer to the setpoint, the computation time decreases until a certain level. Temporary increases can be observed as small kinks in

the graphs. They occur at NMPC iterations directly after the uncertain parameter jumps.

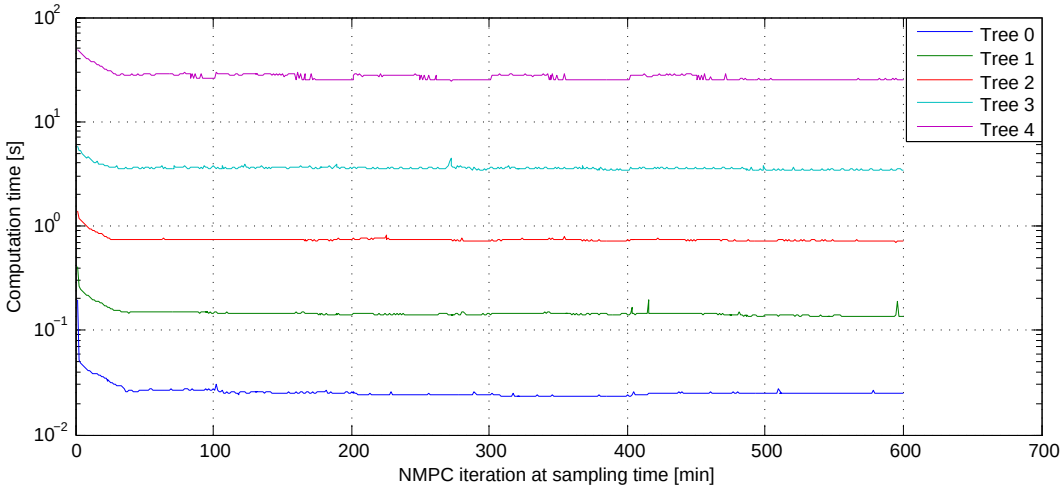

Figure 9.4: CSTR: For the example the computation time per NMPC iteration grows with the number of scenarios. For all trees the computation time decreases at the beginning until a certain level. Times increase temporarily after the uncertain parameter jumps.

Table 9.4 consists of two parts. The minimum, the median and the maximum computation times per NMPC iteration for all considered scenario trees are depicted in the upper part. Most NMPC iteration times lie around the median time. The maximum times are insofar important as they decide about the real-time feasibility of the controller. In the lower part of the table we find the maximum CPU times of the most expensive numerical operations. The times are measured from a serial implementation, therefore they represent the sum of the computation time for all scenarios.

From the upper part of the table we deduce that the choice Tree 0 - Tree 3 for NMPC yields real-time feasible controller. The lower part of the table underlines that the evaluation time consisting of integration and sensitivity computation dominates the computation time of an NMPC loop. Due to the early branching, it is necessary to evaluate every scenario separately. If we look at the evaluation times per scenario, all trees are at the same order of magnitude of 10^{-2} seconds. Scaling this with the number of scenarios yields the table row of evaluation times. When picking one of the trees, the maximum QP solution time and condensing time are of the same order of magnitude. With growing tree size the dominance of the evaluation time over the QP operation times becomes apparent.

Table 9.4: CSTR: Computational performance of scenario tree NMPC (serial implementation), times in seconds, sampling time is 6 seconds

Scenario tree label	0	1	2	3	4
NMPC loop: Minimum	0.023	0.134	0.704	3.408	24.736
NMPC loop: Median	0.025	0.144	0.734	3.578	25.988
NMPC loop: Maximum	0.194	0.406	1.381	5.751	48.698
Evaluation: Maximum	0.045	0.218	1.028	4.463	37.914
QP solution: Maximum	0.028	0.047	0.081	0.358	4.973
Condensing: Maximum	0.020	0.037	0.118	0.567	4.032

Evaluating whole scenarios in parallel is a possible computation time reduction strategy. A discussion about parallelization strategies follows for the industrial batch reactor example in Section 9.2. At this point we emphasize that our methods are capable to solve the large-scale optimization problems for 1001 scenarios.

9.2 A Biochemical Batch Reactor

The optimization of batch processes has attracted attention mainly because in the face of growing competition, methods for mathematical model-based optimizing control have been shown to be a viable choice for reducing production costs, improving product quality, meeting safety requirements and environmental regulations. We present in this section an industrial batch polymerization reactor that has been investigated in [82, 81]. The model is provided by BASF SE, which shows that it is of significant relevance for industry. As illustrated in Figure 9.5 the reactor is equipped with a jacket and an external heat exchanger (EHE).

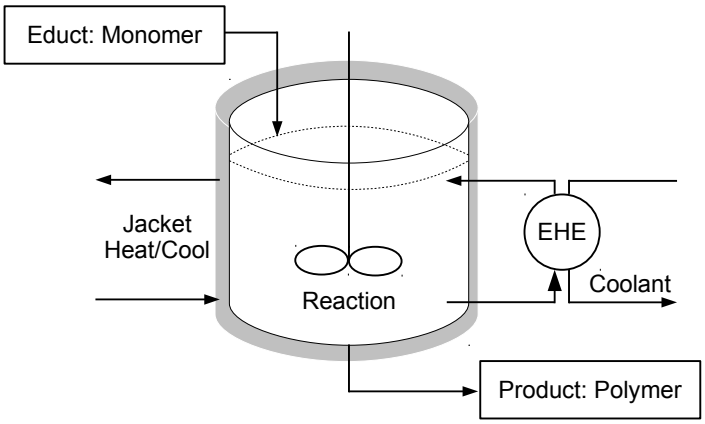

Figure 9.5: The scheme of the biochemical batch reactor shows the connections of the components reactor, vessel, jacket and the two cooling systems.

Dynamical system model

The dynamical system model consists of three mass balance equations for the water (m_W), the educt (m_A) ((9.3a) – (9.3c)) and the product (m_P), and five energy balance equations for the temperatures of the reactor (θ_R), the vessel (θ_S), the jacket (θ_M), the external heat exchanger (θ_{EK}) and the coolant leaving the external heat exchanger (θ_{AWT}) ((9.3d)-(9.3h)). In order to compute the ODE right hand side it is required to evaluate formulas (9.3i) – (9.3n). They describe the polymer-monomer ratio U in the reactor, the total mass m_{ges}, the reaction rate inside the reactor k_{R1} and the reaction rate in the external heat exchanger k_{R2}. Furthermore, the total heat transfer coefficient of the mixture inside the reactor is denoted as k_K and the current amount of monomer inside the reactor is $m_{A,R}$. The control inputs of the system (9.3) are the feed flow $\dot{m}_F$, the coolant temperature at the inlet of the jacket θ_M^{IN} and the coolant temperature at the inlet of the external heat exchanger $\theta_{\mathrm{AWT}}^{\mathrm{IN}}$. The dynamical

system (9.3) requires the model parameters listed in Table 9.5. They are provided by the BASF and published by [82].

Table 9.5: Biochemical batch reactor model parameters

R	8.314	kJ/kmol/K	Ideal gas constant
$c_{p,W}$	4.2	kJ/kg/K	Specific heat capacity of the coolant
$c_{p,S}$	0.47	kJ/kg/K	Specific heat capacity of the steel
$c_{p,F}$	3.0	kJ/kg/K	Specific heat capacity of the feed
$c_{p,R}$	5.0	kJ/kg/K	Specific heat capacity of the reactor contents
k_{WS}	4800	W/m^2/K	Heat transfer coefficient water-steel
θ_F	298.15	K	Feed temperature
A	338.15	K	Heat exchange surface of the jacket
$m_{\mathrm{M,KW}}$	5000	m^2	Mass of coolant in the jacket
m_S	3.9e4	kg	Mass of reactor steel
m_{AWT}	200	kg	Mass of the product in the EHE
$m_{\mathrm{AWT,KW}}$	1000	kg	Mass of the coolant in the EHE
$\dot{m}_{\mathrm{M,KW}}$	3e5	kg/h	Coolant flow of the jacket
$\dot{m}_{\mathrm{AWT,KW}}$	1e5	kg/h	Coolant flow of the EHE
$\dot{m}_{\mathrm{AWT}}$	2e4	kg/h	Product flow to the EHE
E_a	8500	kJ/kmol	Activation energy
ΔH_R	950	kJ/kg	Specific reaction enthalpy
k_0	7	1	Specific reaction rate
k_{U1}	32	1	Reaction parameter 1
k_{U2}	4	1	Reaction parameter 2
$w_{W,F}$	0.333	1	Mass fraction of water in the feed
$w_{A,F}$	0.667	1	Mass fraction of monomer in the feed
k_{AS}	1000	W/m^2/K	Heat transfer coefficient monomer-steel
k_{PS}	100	W/m^2/K	Heat transfer coefficient product-steel
α	3.6e6	1/s	Experimental coefficient

The dynamical reactor model is described by the ODE system

$$\frac{dm_W}{dt} = \dot{m}_F w_{W,F}, \tag{9.3a}$$

$$\frac{dm_A}{dt} = \dot{m}_F w_{A,F} - k_{R1} m_{A,R} - k_{R2} m_{\mathrm{AWT}} \frac{m_A}{m_{\mathrm{ges}}}, \tag{9.3b}$$

$$\frac{dm_P}{dt} = k_{R1} m_{A,R} + k_{R2} m_{\mathrm{AWT}} \frac{m_A}{m_{\mathrm{ges}}}, \tag{9.3c}$$

$$\frac{d\theta_R}{dt} = \frac{\dot{m}_F c_{p,F}(\theta_F - \theta_R) + \Delta H_R k_{R1} m_{A,R} - k_K A(\theta_R - \theta_S) - \dot{m}_{\mathrm{AWT}} c_{p,R}(\theta_R - \theta_{\mathrm{EK}})}{c_{p,R} m_{\mathrm{ges}}}, \tag{9.3d}$$

$$\frac{d\theta_S}{dt} = \frac{k_K A(\theta_R - \theta_S) - k_K A(\theta_S - \theta_M)}{c_{p,S} m_S}, \tag{9.3e}$$

$$\frac{d\theta_M}{dt} = \frac{\dot{m}_{\mathrm{M,KW}} c_{p,W}(\theta_M^{\mathrm{IN}} - \theta_M) + k_K A(\theta_S - \theta_M)}{c_{p,W} m_{\mathrm{M,KW}}}, \tag{9.3f}$$

$$\frac{d\theta_{\mathrm{EK}}}{dt} = \frac{\dot{m}_{\mathrm{AWT}} c_{p,W}(\theta_R - \theta_{\mathrm{EK}}) - \alpha(\theta_{\mathrm{EK}} - \theta_{\mathrm{AWT}}) + k_{R2} m_{\mathrm{AWT}} \Delta H_R \frac{m_A}{m_{\mathrm{ges}}}}{c_{p,R} m_{\mathrm{AWT}}}, \tag{9.3g}$$

$$\frac{d\theta_{\mathrm{AWT}}}{dt} = \frac{\dot{m}_{\mathrm{AWT,KW}} c_{p,W}(\theta_{\mathrm{AWT}}^{\mathrm{IN}} - \theta_{\mathrm{AWT}}) - \alpha(\theta_{\mathrm{AWT}} - \theta_{\mathrm{EK}})}{c_{p,W} m_{\mathrm{AWT,KW}}}, \tag{9.3h}$$

with

$$U = \frac{m_P}{m_A + m_P}, \tag{9.3i}$$

$$m_{\mathrm{ges}} = m_W + m_A + m_P, \tag{9.3j}$$

$$k_{R1} = k_0 \exp\left(-\frac{E_a}{R(\theta_R + 273.15)}\right)(k_{U1}(1-U) + k_{U2}U), \tag{9.3k}$$

$$k_{R2} = k_0 \exp\left(-\frac{E_a}{R(\theta_{\mathrm{EK}} + 273.15)}\right)(k_{U1}(1-U) + k_{U2}U), \tag{9.3l}$$

$$k_K = \frac{m_W k_{WS} + m_A k_{AS} + m_P k_{PS}}{m_{\mathrm{ges}}}, \tag{9.3m}$$

$$m_{A,R} = m_A\left(1 - \frac{m_{\mathrm{AWT}}}{m_{\mathrm{ges}}}\right). \tag{9.3n}$$

The BASF adds to the model an important safety feature, which is incorporated to avoid failures of the equipment. Here the maximum temperature the reactor could reach in case of a cooling failure is added to the model and constrained to be below 382.15 K. The differential equation for the additional state θ_{adiab} reads

$$\frac{d\theta_{\mathrm{adiab}}}{dt} = \frac{\Delta H_R \dot{m}_A}{m_{\mathrm{ges}} c_{p,R}} - (\dot{m}_W + \dot{m}_A + \dot{m}_P)\frac{m_A \Delta H_R}{m_{\mathrm{ges}}^2 c_{p,R}} + \dot{\theta}_R. \tag{9.4}$$

At this point we motivate a further additional state m_{acc}. The maximum amount of material that can be fed into the reactor is 30,000 kg. After all the material is fed, the reactor feeding phase ends and the reaction continues with the holding phase. To avoid the

switching between different models for the feeding and the holding phase, we introduce the additional state for the accumulated material that has been fed. The state m_{acc} is then constrained to the maximum amount of material. The differential equation for m_{acc} reads

$$\frac{dm_{\text{acc}}}{dt} = \dot{m}_F. \tag{9.5}$$

The bounds on all states and controls, as well as the initial values are listed in Table 9.6.

Table 9.6: Biochemical batch reactor: Variable bounds and initial values

Variable	Initial value	Lower bnd.	Upper bnd.	Unit	Description
m_W	10000.0	0	-	kg	Mass of water
m_A	853.0	0	-	kg	Mass of the educt
m_P	26.5	0	-	kg	Mass of the product
m_{acc}	300.0	0	30000.0	kg	Accumulated mass of feed
θ_R	363.15	361.65	364.65	K	Reactor temperature
θ_S	363.15	273.15	373.15	K	Vessel temperature
θ_M	363.15	273.15	373.15	K	Jacket temperature
θ_{EK}	308.15	273.15	373.15	K	Temperature of the EHE
θ_{AWT}	308.15	273.15	373.15	K	Temp. of coolant leaving EHE
θ_{adiab}	378.05	288.15	382.15	K	Adiabatic temperature
$\dot{m}_F$	-	0	30000.0	kg/h	Feed flow
θ_M^{IN}	-	333.15	373.15	K	Coolant temp. at inlet of jacket
$\theta_{\text{AWT}}^{\text{IN}}$	-	333.15	373.15	K	Coolant temp. at inlet of EHE

Uncertainty model

We consider two of the most critical parameters of the model as uncertain but constant during the reaction. In particular, the specific reaction rate k_0 and the specific reaction enthalpy ΔH_R differ from their nominal values listed in Table (9.5). We assume a Gaussian distribution of $(k_0, \Delta H_R)$ with mean $\mu = (7, 950)$ and variance/covariance matrix

$$\Sigma = \begin{pmatrix} 0.25 & 0 \\ 0 & 5 \end{pmatrix}.$$

Simulation Setup

The simulation model consists of the ODE system (9.3), but with the paramount difference that the uncertain parameters are realized at $k_0 = 6.9$ and $\Delta H_R = 955$. The controller is not

aware of the model-plant mismatch, therefore we can study how robust the scenario tree approach performs in the presence of uncertainty. For the initial values see Table 9.6.

Controller Setup

The goal of the controlled batch reactor is to synthesize a batch of polymer product while satisfying safety constraints and constraints on the quality of the product. The objective function reads

$$\int_{t_0}^{t_f} -m_P + 10^4(\theta_R - \theta_{set})^2 dt. \tag{9.6}$$

We control the reactor maximizing the mass of product that is synthesized within the prediction horizon. There is also a strong influence of the reaction temperature on the properties of the resulting product. Therefore the temperature of the reactor has to be maintained between 361.65 K and 364.65 K around the reaction temperature set point $\theta_{set} = 363.15$ K. Furthermore, we employ a control move regularization with $\alpha_{\text{CMR}} = (10^4, 5, 5)^T$. We minimize the term (9.6) with respect to the dynamical system (9.3), (9.4), (9.5), the bounds and initial values specified in Table 9.6. We solve the prediction problem discretized by the direct multiple shooting method with 21 nodes on a prediction horizon of 1000 seconds. The NMPC sampling time is 50 seconds. Furthermore, we assemble the QP using limited memory BFGS updates for the Hessian approximation. We employ the RTI scheme and solve the tree QP with the dual decomposition approach and Condensing of the local branch QPs.

Scenario Tree Setup

The main design decisions for the scenario tree are the realizations of the uncertain parameter and the robust horizon. In the present example of the industrial batch reactor the uncertain parameter space is two-dimensional, thus suitable to compare the full tensor grid approximation with the sparse grid approximation from Chapter 6. For the tensor grid we consider the realizations $\mu + 2d_{\text{tensor}}\Sigma$ with all 9 directions in two dimensions

$$d_{\text{tensor}} \in \{(0,0),(0,1),(0,-1),(1,0),(1,1),(1,-1),(-1,0),(-1,1),(-1,-1)\}.$$

The sparse grid approach only uses 5 directions within a diamond around the nominal parameter realization. Therefore the 5 realizations $\mu + 2d_{\text{sparse}}\Sigma$ for the sparse grid are

$$d_{\text{sparse}} \in \{(0,0),(0,1),(0,-1),(1,0),(-1,0)\}.$$

The sizes of the scenario trees are dependent on the robust horizon n_d. We start from $n_d = 0$, which is nominal NMPC with the nominal realization. As one focus of this work lies on tree design, we evaluate the performance of scenario tree NMPC of the batch reactor example

with four trees. Robust horizons up to $n_d = 2$ for both sparse and tensor grid form the trees. The scenario tree used in [82] corresponds to our tensor grid tree with robust horizon of one. We summarize the scenario tree setup and the resulting tree QP size in Table 9.7.

Table 9.7: Industrial batch reactor: Scenario tree data

Scenario tree label	0	1	2	3	4
Robust horizon	0	1	1	2	2
Sparse or **T**ensor grid	-	S	T	S	T
Number of scenarios	1	5	9	25	81
Non-anticipativity constraints	0	4	8	44	152
Tree QP size:					
Number of variables	291	1455	2619	7275	23571
Number of equality constraints	231	1167	2103	5907	19167
Number of inequality constraints	60	300	540	1500	4860

Resulting Trajectories

The results of all five controllers are depicted in three figures. Figure 9.6 and Figure 9.7 show the ten system states and Figure 9.8 shows the three controls. All controllers have successfully operated the plant in the presence of systematic model-plant mismatch. The critical reactor temperature stays within its bounds and the mass of product increases during the reaction. We observe that the performance depends on the scenario tree. Most interestingly, the sparse grid controllers Tree 1 and Tree 3 yield the highest amount of product m_P at the end. Therefore robustness at the edges of the parameter realizations that are not taken into acount by the sparse grid diamond comes at the cost of less product. We continue how the dual decomposition performs and then discuss a major point, the real-time feasibility.

Non-smooth Newton Iterations

In the present reactor example we have solved every tree QP with the dual decomposition approach and non-smooth Newton method. The number of iterations per NMPC iteration has a large influence on the solution time of the tree QP we solve in every NMPC iteration according to the RTI scheme. We depict the number of non-smooth Newton iterations per NMPC iteration over the control horizon of two hours in Figure 9.9. As in the previous example the number of non-smooth Newton iterations for the controllers with different

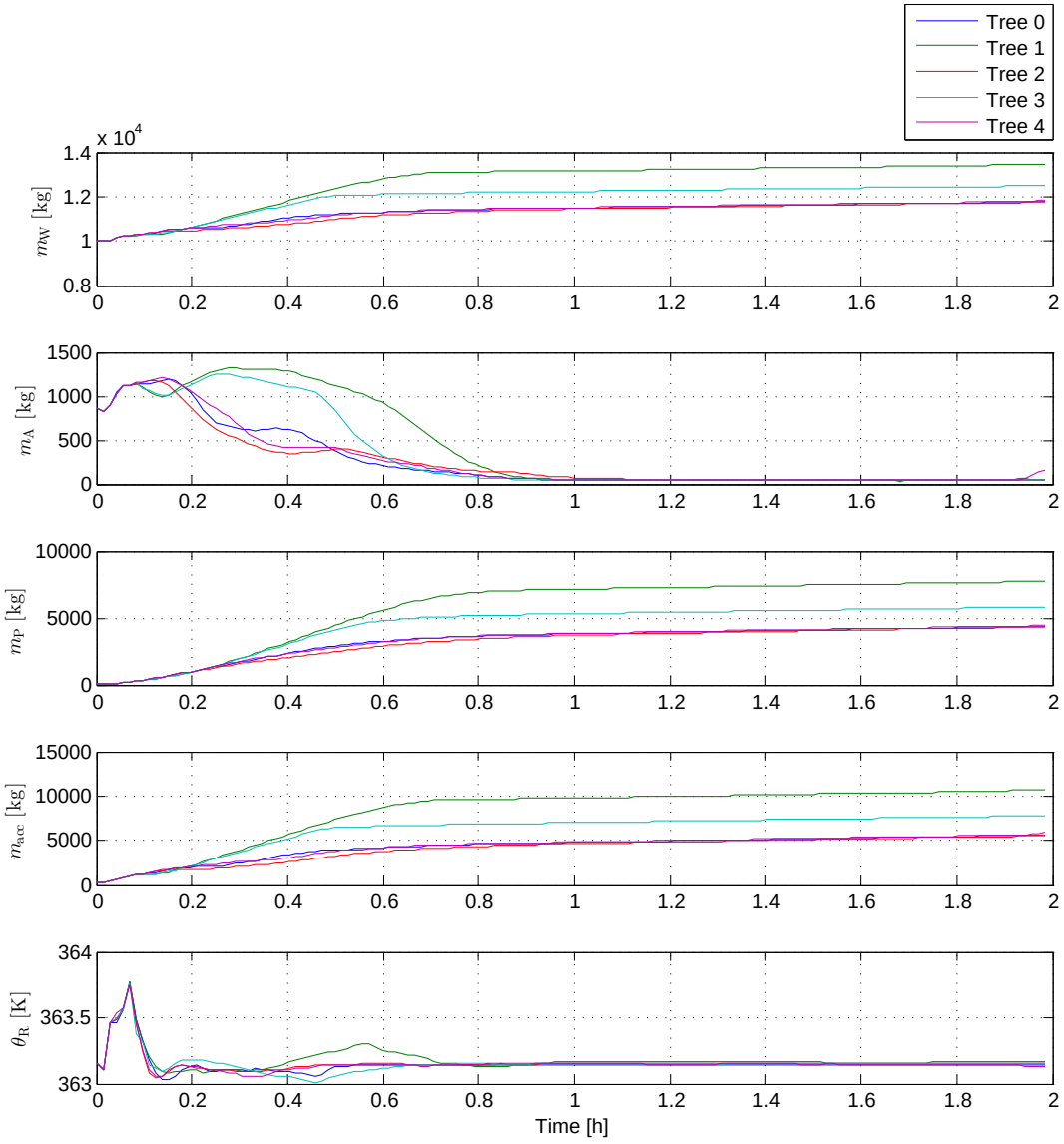

Figure 9.6: Resulting trajectories of the reactor system states (part 1): For all scenario trees the mass of product and water increase whereas the mass of educt decreases. The reactor temperature has a peak at the beginning. Due to the cooling systems it can track the temperature set point of 363.15 K. The accumulated amount of feed material increases until the educt is completely converted.

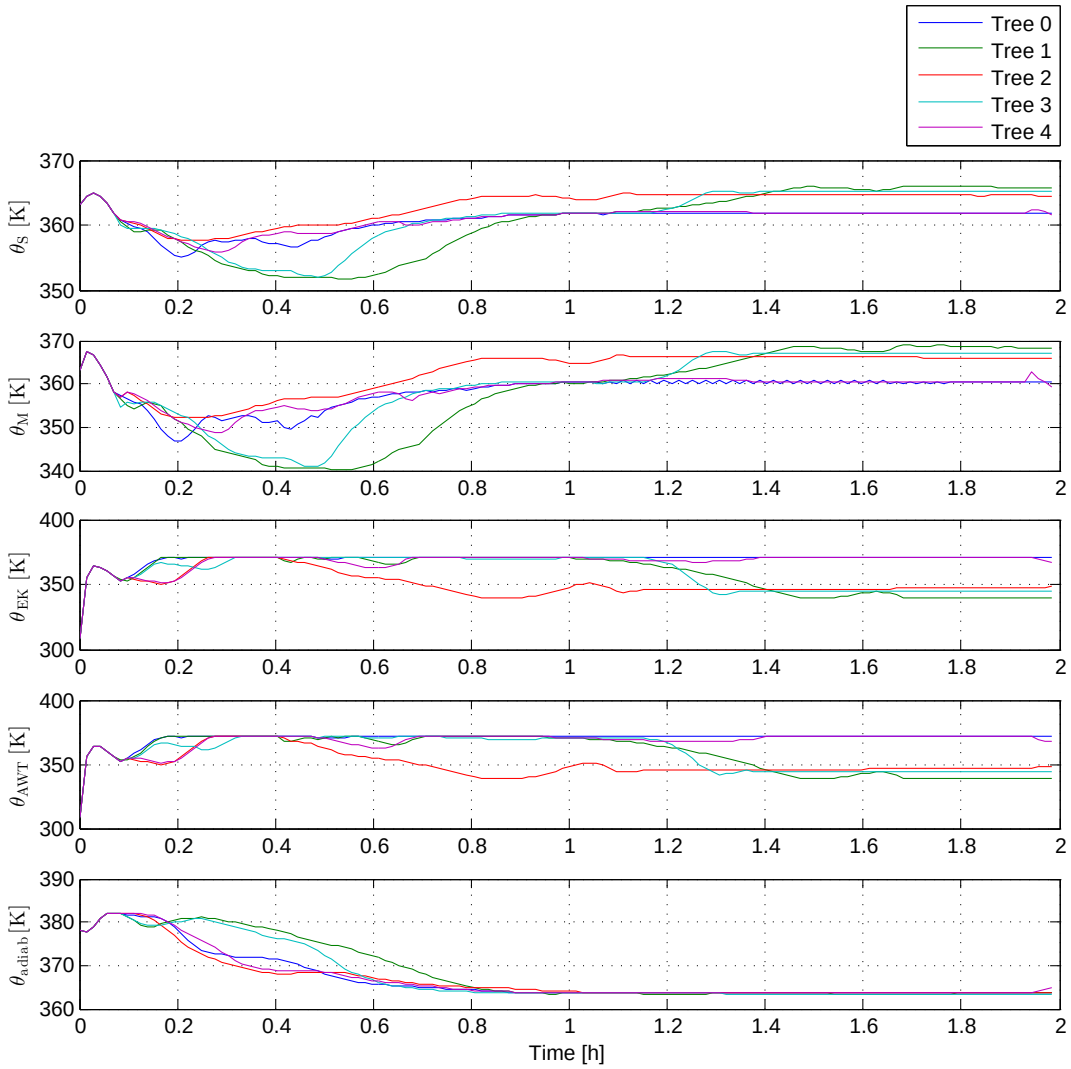

Figure 9.7: Resulting trajectories of the reactor system states (part 2): The four temperature variables of the cooling system stay within their bounds. The adiabatic temperature, the indicator for the cooling failure behavior, touches the upper bound only at the beginning of the reaction.

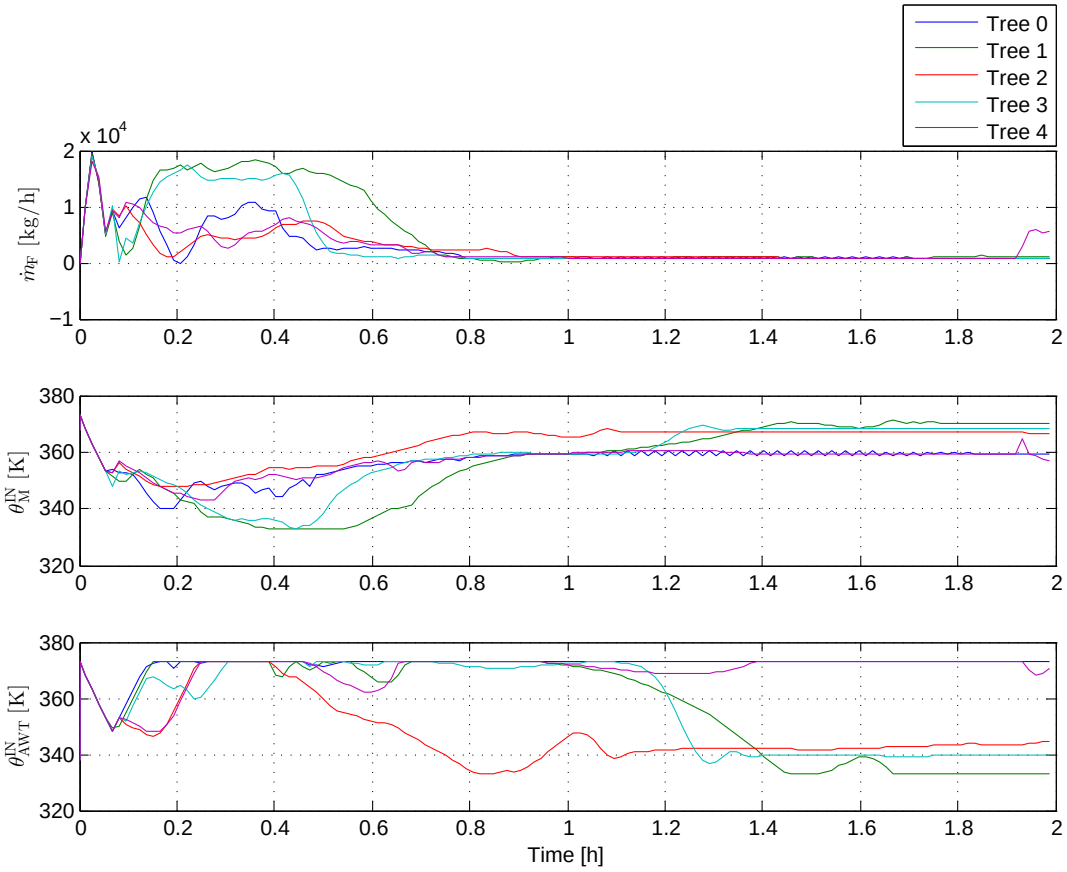

Figure 9.8: Resulting reactor control variables: The mass feed increases at the beginning up to a peak of 20000 kg/h, then decreases to not overheat the reactor. Depending on the controller a second increase follows until the mass of the educt approaches zero. Afterwards, for all controllers the feed stays constant on a low level. Both inlet temperatures of the cooling system act within their bounds and show at later times a different behavior for the scenario trees, because the different scenario trees consider different sequences of uncertain parameter realizations.

scenario trees are stacked along the y-axis. Most of the time the number of non-smooth Newton iterations lies in order of magnitude of ten. Shortly after the beginning the larger trees require one order of magnitude more iterations due to many active set changes. Especially the values of variables originating from the discretized adiabatic temperature that safeguards against cooling failure change from the upper bound to values close to the upper bound depending on the scenario. Another crucial point when the cumulative number of iterations reaches even over 1000 iterations is at the 15th NMPC iteration. At this point the cooling system is at bound to react to the massive feed in the first reaction phase. Another observation is that Tree 4, the largest tree, required most iterations and has even more critical NMPC iterations with more than 100 non-smooth Newton iterates. The more scenarios we consider the more active set changes regarding subsequent iterations happen.

Real-Time Feasibility

The real-time feasibility of the controller heavily depends on the number of scenarios of the tree. Here the computational time of one NMPC iteration must remain below the sampling time of 50 seconds. From Figure 9.10 we deduce that nominal NMPC and Tree 1 as well as Tree 2 with 9 scenarios fulfill the real-time requirement. With robust horizon of 2 the number of scenarios increases. The controller with Tree 3 (25 scenarios) is real-time feasible except for some critical NMPC iterations. The computation time of the Tree 4 controller with 81 scenarios is not real-time feasible. However, the depicted times result from a serial implementation. We have pointed out in Chapter 5 that dual decomposition has a large parallelization potential. In Figure 9.11 we depict the percentage of various operations that are performed in every NMPC iteration. A large percentage of computation time is spend on evaluation.A virtual parallelization of evaluation and branch QP solution yields the computation times in Figure 9.12. They are computed by the formula

$$t_{\text{loop}}^{\text{virt.parallel}} = t_{\text{loop}}^{\text{serial}} - \sum_{j=1}^{S} t_{\text{Evaluation}}^{j} - \sum_{j=1}^{S} t_{\text{QP}}^{j} + \left\lceil \frac{S}{n_{\text{Cores}}} \right\rceil \left(\max_j t_{\text{Evaluation}}^{j} + \max_j t_{\text{QP}}^{j} \right)$$

with S as number of scenarios, $t_{\text{Evaluation}}^{j}$ as the evaluation time of scenario j, the branch QP time t_{QP}^{j} within the dual decomposition and the number of parallel threads of a multi-core system n_{Cores}. As a result most NMPC iterations of Tree 4 with 81 scenarios are real-time feasible, underlining the high parallelization potential. Critical iterations with more than 50 seconds of NMPC loop time are those that required more than 100 non-smooth Newton iterations.

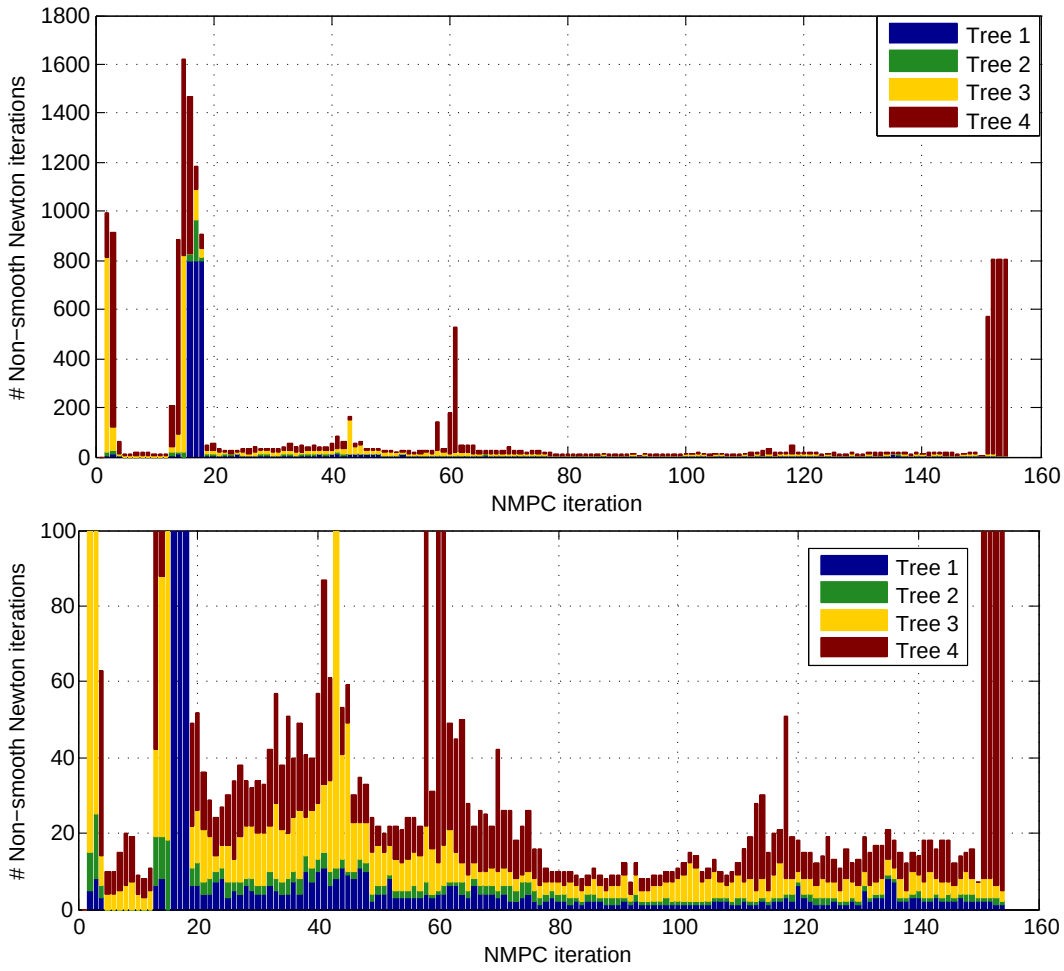

Figure 9.9: Number of non-smooth Newton iterations for the industrial batch reactor. Top: Full stacked bar graph. Bottom: Close-up of the graph up to 100 on the y-axis. The number of non-smooth Newton iterations per NMPC iteration for the different trees is horizontally stacked. The numbers are most of the time in the order of magnitude of ten, critical iterations at the beginning and at the 15th iteration become well visible due to the stacking.

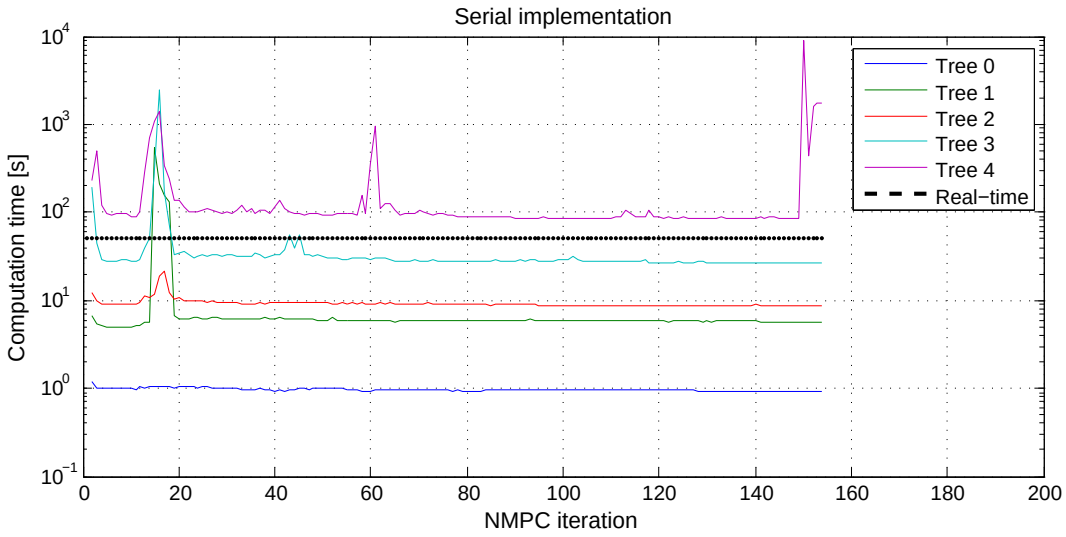

Figure 9.10: Computation times of the batch reactor for the serial implementation of dual decomposition: The computation time per NMPC iteration in a serial implementation grows with the number of scenarios. Tree 4 with 81 scenarios exceeds the real-time barrier.

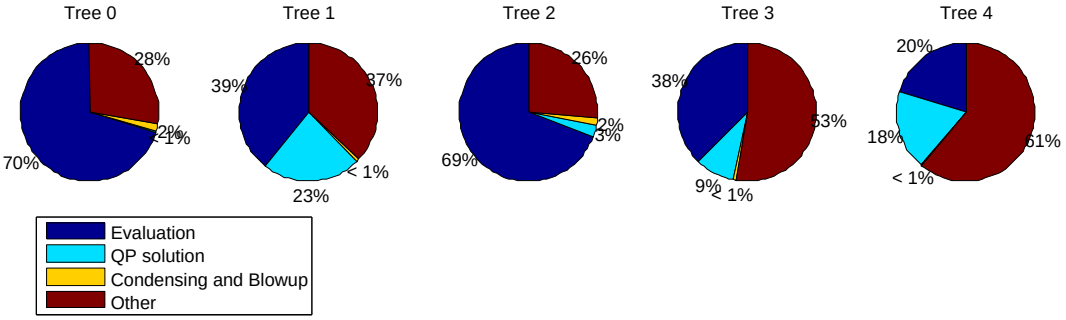

Figure 9.11: Percentage of time spent on operations during NMPC: The computation time for every operation is cumulated over the whole control horizon for the specified scenario tree controller and then divided by the total time of the controller. The evaluation time and the part for other operations dominate the pie charts. Therefore a parallelization of the evaluation part has high potential to save computation time.

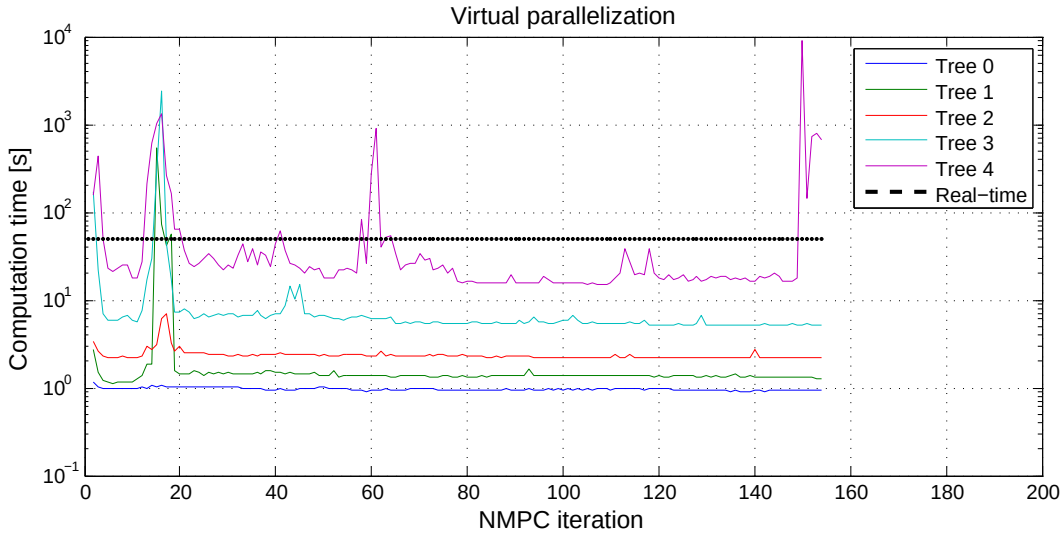

Figure 9.12: Computation times of the batch reactor for virtually parallelized NMPC loops: The virtual parallelization with $n_{\text{Cores}} = 8$ yields real-time feasibility for all controllers for almost all NMPC iterations, thus highlighting the parallelization potential.

Sparse Grid versus Tensor Grid

At this point we discuss the impact of scenario reduction by using a sparse grid approximation of the uncertain parameter space instead of the full tensor grid approach. To this end we compare the scenario tree controllers Tree 1 with Tree 2 and Tree 3 with Tree 4. The robust horizon and resulting number of scenarios are in the upper part of Table 9.8. In the middle part of Table 9.8 we list the overall computation times. The bottom part of Table 9.8 evaluates the the median of computation times per NMPC loop for all controllers.

In the middle column of Table 9.8 we compare the sparse grid of robust horizon 1 and the tensor grid of robust horizon 1. By using sparse grids we can consider 44.4 % less scenarios, without sacrificing feasibility or performance as the trajectories in Figures 9.6, 9.7 and 9.8 show. Comparing the computation times we save all in all 16.2 % by the reduced grid. The median computation time per NMPC loop of the sparse Tree 1 is even 34.7 % less than the median computation time per NMPC loop of the tensor grid Tree 2. An even stronger effect of scenario reduction by sparse grids can be observed for robust horizon 2. In the right column we note that all operations for 69.1 % of the scenarios of the full tensor grid can be saved. For the overall computation time the sparse grid yields a reduction of more than 75 %, the median time per NMPC loop can also be reduced by 69.9 %. We emphasize that in case of Tree 3 and Tree 4 as in the previous case neither feasibility nor performance are

Table 9.8: Industrial batch reactor: Savings by using a sparse grid instead of a tensor grid

Robust horizon	1	2
Scenario tree labels	1,2	3,4
Number of scenarios		
Sparse grid	5	25
Tensor grid	9	81
Saving of sparse grid w.r.t tensor grid (%)	44.4	69.1
Overall computation time [s]		
Sparse grid	1187.6	7602.9
Tensor grid	1416.5	33688.5
Saving of sparse grid w.r.t tensor grid (%)	16.2	77.5
NMPC loop: Median computation time [s]		
Sparse grid	5.9	28.2
Tensor grid	8.9	93.7
Saving of sparse grid w.r.t tensor grid (%)	34.7	69.9

sacrificed, see Figures 9.6, 9.7 and 9.8. Another important point is that the data of Table 9.8 stems from the serial implementation. Therefore a combination of scenario reduction like the sparse grid approach and parallelization would be most beneficial. All in all scenario tree NMPC is a versatile method and capable to control the industrial batch reactor.

9.3 Penicillin Production

A variety of pharmaceutical substances are often produced in batch operations. In the following we study the biochemical synthesis of the antibiotic Penicillin with the molecular structure depicted in Figure 9.13.

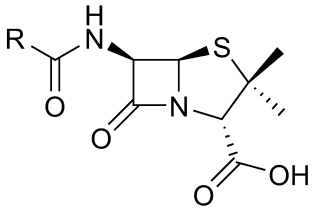

Figure 9.13: Chemical structure of the common part of the Penicillins. The part R determines the specific Penicillin molecule.

The reaction model is studied in [103], a more general version also in [104]. The potential of scenario tree NMPC has been pointed out by [83] with simulation studies.

Dynamical system model

The ODE model of the Penicillin synthesis describes the dynamical evolution of the concentrations during the reactions Substrate $\rightarrow$ Biomass and Biomass $\rightarrow$ Penicillin. Model parameters are listed in Table 9.9.

Table 9.9: Penicillin production model parameters

μ_m	0.02	l/h	Reaction rate constant
K_m	0.05	g/l	Activation energy
K_i	5	g/l	Ideal gas temperature
Y_x (nominal)	0.4	g/g	Heat of reaction
Y_p	1.2	g/g	Mass density
ν	0.004	l/h	Heat capacity
S_{in} (mean)	200	g/l	Overall heat transfer coefficient $\times$ reaction area

The dynamical system model reads

$$\frac{dX}{dt} = \mu(S)X - \frac{u}{V}X, \tag{9.7a}$$

$$\frac{dS}{dt} = -\frac{\mu(S)X}{Y_x} - \frac{\nu X}{Y_p} + \frac{u}{V}(S_{\text{in}} - S), \tag{9.7b}$$

$$\frac{dP}{dt} = \nu X - \frac{u}{V}P, \tag{9.7c}$$

$$\frac{dV}{dt} = u, \tag{9.7d}$$

with

$$\mu(S) = \frac{\mu_m S}{K_m + S + \frac{S^2}{K_i}}. \tag{9.7e}$$

In the model (9.7) X represents the concentration of biomass, S the concentration of substrate, P the concentration of the product Penicillin and V the volume. The control input u is the feed flow rate of the substrate. In addition to the ODEs for the four states we compute the specific growth rate $\mu(S)$ according to (9.7e). Bounds and initial values can be found in Table 9.10.

Table 9.10: Penicillin production bounds and initial values

$\overline{X}$	3.7	g/l	Upper bound of biomass concentration
$\overline{u}$	1	l/h	Upper bound of feed flow rate
$\underline{u}$	0	l/h	Lower bound of feed flow rate
X^0	1	g/l	Initial biomass concentration
S^0	0.5	g/l	Initial substrate concentration
P^0	0	g/l	Initial product concentration
V^0	150	l	Initial volume

Uncertainty model

The inlet substrate concentration S_{in} and the specific growth rate Y_x are regarded as uncertain parameters. We assume for S_{in} a normal distribution with mean 200 and standard deviation 25 as in [83]. The parameter Y_x is assumed to be constant during the whole reaction and within the interval $[0.3, 0.5]$ as in [83]. In consequence, the uncertainty space is two dimensional.

Simulation Setup

For the simulation we run the model (9.7) until the final time of 150 hours is reached. The control input is provided by the scenario tree NMPC controller. The simulator starts from the initial values from Table 9.9 and keeps the uncertain parameters on their nominal values.

Controller Setup

The objective is of economic type, the goal is to maximize the amount of Penicillin at the end. The objective function reads

$$\int_{t_0}^{t_f} -P \, dt. \tag{9.8}$$

The prediction horizon is 50 minutes. We perform phase D in every NMPC sampling time of 12.5 minutes and solve the tree QP with the dual decomposition approach. During assembly of the QP we use L-BFGS updates on the Hessian matrix approximation and condense each branch QP. We further make use of the control move regularization with $\alpha_{\text{CMR}} = 0.05$.

We remark for the controller setup that the dynamical system has special properties requiring restarts of the homotopy-based controller.

Scenario Tree Setup

For the numerical experiments we first consider nominal NMPC with $(S_{\text{in}}, Y_x) = (200, 0.4)$ as Tree 0. Then we regard only S_{in} as uncertain and carry out scenario tree NMPC with robust horizon 1. The 3 scenarios represent the nominal value and the 2σ variations for S_{in}. Therefore Tree 1 consists of the scenarios $(S_{\text{in}}, Y_x) \in \{(150, 0.4), (200, 0.4), (250, 0.4)\}$. Finally we consider a scenario tree with 9 scenarios representing the two-dimensional uncertainty space. The scenarios of Tree 3 are

$$\begin{aligned}(S_{\text{in}}, Y_x) \in \{&(150, 0.3), (200, 0.3), (250, 0.3), (150, 0.4), (200, 0.4), (250, 0.4),\\ &(150, 0.5), (200, 0.5), (250, 0.5)\}.\end{aligned}$$

Results

In contrast to the previous numerical examples we first summarize the scenario tree NMPC performance and then put our main focus on a special system property that we observed.

The resulting trajectories for the nominal NMPC controller (Tree 0) and the scenario tree NMPC controller that take into account one uncertain parameter (Tree 1) and both uncertain parameters (Tree 2) are depicted in Figure 9.14. We observe an increase of the biomass concentration up to a certain level in the feeding phase. In the nominal case the level is the upper bound 3.7 g/l. With increasing number of scenarios the feeding phase

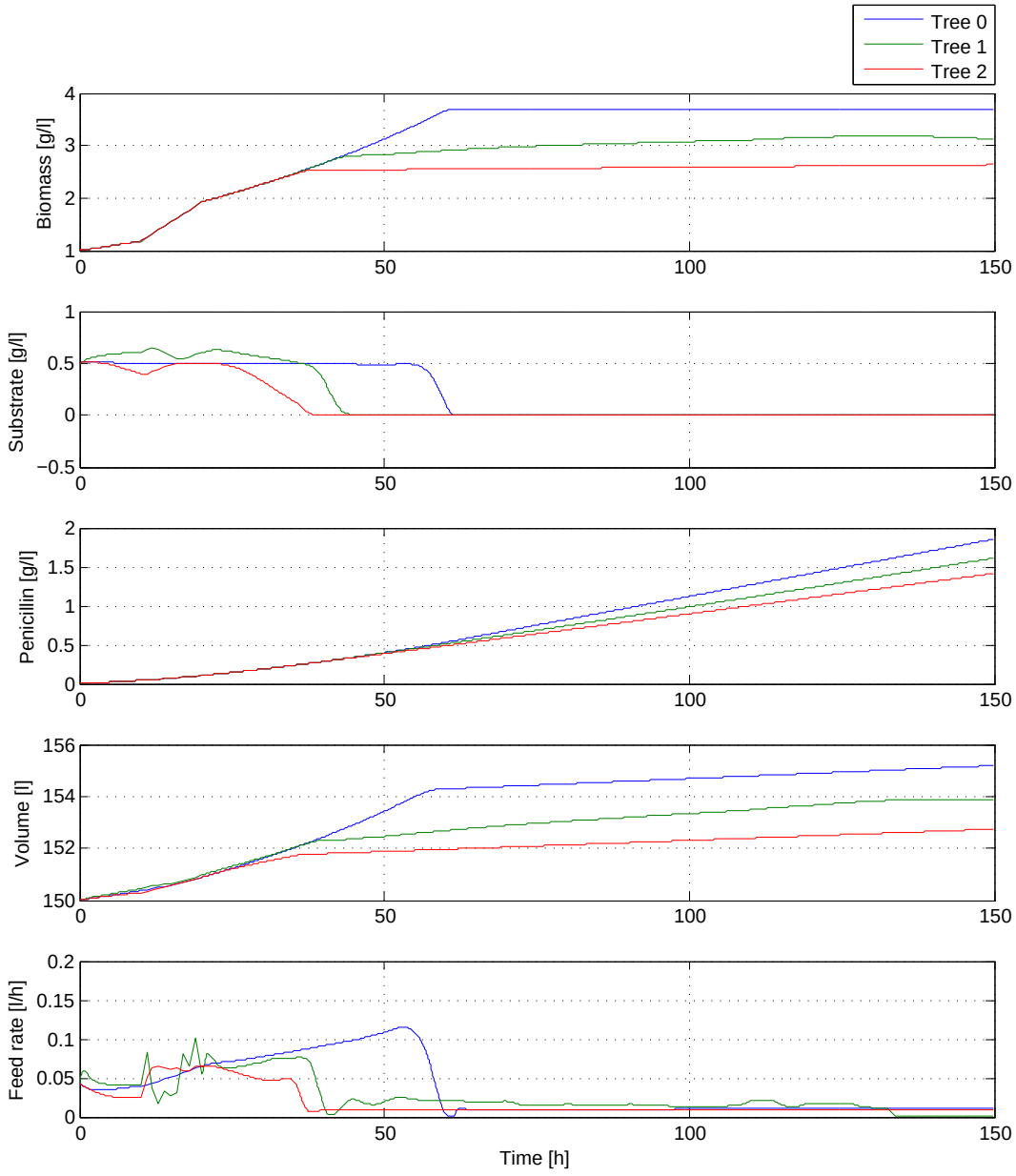

Figure 9.14: Scenario tree NMPC results for Penicillin production: The trajectories of the Penicillin example illustrate for all trees two phases: A quick feeding phase and an equilibrium phase. The key performance index is the produced amount of Penicillin in the middle.

stops earlier because the scenario tree controller takes into account the uncertainty and safeguards against it. The subproblems in all NMPC iterations must stay feasible for all scenarios on the whole prediction horizon in order to return a valid solution. The substrate concentration starts for all trees from 0.5 g/l, stays there and decreases to 0 g/l shortly after the tree-dependent end of the feeding phase is reached. From the economic perspective the amount of produced Penicillin is most important. For all trees the concentration increases monotonically. After running the reactor for 150 hours, controller Tree 0 yields 1.83 g/l, Tree 1 yields 1.61 g/l and Tree 2 yields 1.41 g/l Penicillin. Apparently more scenarios yield a more conservative controller from performance point of view. But the higher performance comes at the price of infeasibility. The nominal controller Tree 0 does not safeguard against the uncertain parameter. This agrees with the simulation studies in [83]. We continue to explain the depicted trajectories. The volume increases during the reaction and also shows a dependence on the different trees to the feeding phase end. In the bottom subplot the feed flow rate is depicted. The increase at the beginning is the feeding phase when the biomass also increases. As the growth of biomass must be interrupted to not exceed the upper biomass bound, the control decreases drastically. Then, until the end it stays constant at an equilibrium level above 0 l/h.

Non-smooth Newton Iterations

In our NMPC framework we solve one tree-structured optimization problem per NMPC iteration. We apply the dual decomposition approach and observe for the Penicillin example a maximum number of non-smooth Newton iteration of 3 for Tree 1 and 9 for Tree 2. The maximum iteration numbers are attained at NMPC iteration 2 for both trees. However, the RTI scheme requires that the homotopy always continues with the evolution of the closed-loop dynamical system. For nonlinear dynamics this cannot be guaranteed. The dynamical model (9.7) is nonlinear and we observe terminating homotopy paths. Therefore we must adjust the controller for the Penicillin example to cope with the terminating homotopy paths.

Homotopy Termination / Bifurcation in the Dynamical System

The issue we address in this section is an inherent property of the closed-loop dynamical system. It is not caused by the scenario tree approach, the effects must be handled also in the common nominal case. Due to nonlinearities there is a whole class of bifurcations that can occur. We refer to standard literature such as [54]. For our NMPC framework we put the focus on how to adjust the controller. As far as now we have presented scenario tree NMPC with real-time iterations. In every NMPC iteration we solve one tree-structured QP by the dual decomposition approach. An important role within the dual decomposition play

the branch QPs. For every scenario we initialize the branch QP with the solution of the respective branch QP of the previous iteration. This yields a fast homotopy-based method. However, in nonlinear dynamical systems the homotopy path can terminate or split due to bifurcations. Additionally, the termination or splitting depends on the uncertain parameter realization. In the computations of the Penicillin example this dependence is reflected by branch QPs that become infeasible due to homotopy termination, even if neighboring branch QPs have a feasible and stationary point. Based on the observations we suggest an fallback strategy. We adjust Algorithm 3 to cope with possibly infeasible local QPs.

Fallback strategy

In line 3 of Algorithm 3, the dual Newton strategy, we solve local $\mathrm{QP}_j(\lambda)$ to obtain $z_j^*(\lambda)$. If we observe that $\mathrm{QP}_j(\lambda)$ is infeasible for at least one $j \in \mathcal{S}$, we stop the dual decompostition approach with non-smooth Newton strategy and solve the whole tree-structured NLP that originates from a Direct Multiple Shooting discretization of the scenario tree optimization problem (2.1). We have implemented the solution of the tree-structured NLP in the MLI software using an interface to IPOPT [111]. The fallback strategy can also be applied to the general nominal NMPC case.

Furthermore we can interprete the fallback strategy as hierarchical controller. Solving the complete NLP can be regarded as a high-rank phase in the MLI framework. Our approach represents then a communication between the RTI phase and the phase solving the complete NLP. Additionally, the complete NLP solution phase is a very good instrument to compute an initial value for the system offline before the first NMPC iteration.

Real-Time Feasibility

At the end the paramount question remains if the controller with fallback strategy still yields real-time feasibility for the Penicillin example. The upper part of Table 9.3 the maximum and median computation times of the NMPC iterations with tree-QP solution are listed for all considered trees. In the lower part we list the maximum and median computation times of those NMPC iterations with solution of the full NLP. The sampling time is 12.5 minutes, or equivalently 750 seconds. For all trees the maximum times lie below the sampling time, therefore real-time feasibility is ensured. It is noticable that the full NLP solution takes much more time than the QP solution because the structure-exploiting numerics and the RTI pay off.

To summarize, our discussion underlines the result of [83] that scenario tree NMPC controls the biochemical Penicillin reaction and robustifies against the uncertainty. Using RTI and fast sampling times for NMPC reduces the feedback delay considerably. The nonlinear

Table 9.11: Computation times per NMPC iteration in seconds for the Penicillin example

Scenario tree label	0	1	2
NMPC iteration with solution of tree-QP:			
Maximum	0.24	0.57	7.36
Median	0.20	0.41	1.52
NMPC iterations with solution of full NLP:			
Maximum	56.84	293.09	462.36
Median	8.81	0.40	1.54

dynamical system of the Penicillin example yields terminating homotopy paths requiring a fallback strategy. We point out that the issue of bifurcations is presumably not observed in [83] because their sampling time is one hour and always the full NLP is solved. Therefore the critical bifurcation points were not hit as in our case with sampling time of 12.5 minutes. All in all the controller adjustment comes at the cost of computation time, in the Penicillin case still below the real-time barrier. But most important, the fallback strategy safeguards against the critical controller failure due to homotopy termination.

9.4 Summary

We have demonstrated that our contribution to the numerical computation side of scenario tree NMPC, the dual decomposition approach, is an efficient method to solve large-scale tree-structured QPs. Combining the RTI scheme and structure exploiting numerics yields real-time feasibility of scenario tree NMPC. Furthermore, when uncertainty is present in more than one parameter, the scenario trees generated by sparse grid quadrature nodes yield a reduction of computational effort. The sparse grid trees have significantly less scenarios than the full tensor grid trees. We have demonstrated the impact of this tree generation method on controlling an industrial batch reactor. Moreover, our examples showcase the relevance of scenario tree NMPC for chemical engineering. We summarize the remarkable aspects from the three chemical reactor studies in the following.

- CSTR: We demonstrate the power of our structure exploitation methods as the dual decomposition approach and solve optimization problems originating from large trees with up to 1001 scenarios. A certain cost of computational and objective performance lies in the nature of robustification against uncertainty. Scenario tree NMPC is a good balance between performance and feasibility with high probability.

- Industrial Batch Reactor: Scenario tree NMPC is capable of controlling real-world applications and thus has a relevance for the process industry. Furthermore the scenario reduction by sparse grids saves up to 70 % of computation time without sacrificing feasibility or controller performance.
- Penicillin production: There are limitations of the RTI, in this case caused by the dependence on continuous homotopy paths, which cannot be guaranteed for highly nonlinear systems. We have described a fallback strategy for the online optimization situation.

The results of this chapter constitute a major part of the general conclusions of this thesis that follow in the next chapter.

Chapter 10

Conclusion and Outlook

Dynamical optimization under uncertainty is enriched by scenario tree NMPC as versatile method to robustify against the uncertainty that is present in the underlying system. Novel methods for scenario tree NMPC in this thesis make computations faster by exploiting the problem structure. The dual decomposition approach contributes to the numerical side and allows to perform scenario tree NMPC online with hundreds of scenarios. On the stochastic side our development of scenario tree generation based on sparse grid quadrature rules opens the doors to apply scenario tree NMPC in high-dimensional uncertainty spaces. As a further contribution of this thesis, the tree generation method based on a Markov chain interpretation of scenario trees yields alternative lean tree structures. The novel approach guarantees a coverage of the uncertainty space with significantly less scenarios than the usual tree generation procedure. Model-based optimizing control of demanding applications can be performed by scenario tree NMPC in real-time. Investigation of the topic has lead to the following future research directions. We start with stochastic research directions, which go deep into the fundamentals of the field. Then we pose optimization directions, numerical questions and an outlook for process control.

Approximation of a continuous parameter distribution by a discrete parameter distribution

The main assumption of the scenario tree approach is that we can approximate the uncertainty by a finite number of realizations. Often the uncertainty space is continuos and the uncertain parameter is assumed to be distributed according to a continous probability density function. Approximation results beyond the quadrature approximation results for discretization of the uncertainty space are a future research direction. The Markov chain interpretation and existing results for approximation of a continuos state space Markov chain by a discrete space Markov chain would be a possible starting point.

Approximation of a continuous time process by a discrete time process

Up to this point we have considered a discrete time stochastic process as underlying uncertain parameter process. Approximation results with respect to a continuos time process go in the direction of measure convergence in the sense of [96]. Investigations of the topic lead to stochastic differential equations and their solution approximations.

Dual decomposition for the node-wise scenario tree formulation

At the heart of our numerical tree-structure exploitation is the dual decomposition approach yielding subproblems that can be solved in a massively parallel fashion. The discretization structure of the single scenario QP subproblems is exploited by condensing. A possible future research direction is to consider the tree in the nodewise formulation and to decouple also the stages. We can then compare the dual decomposition in the non-anticipativity constraints and condensing with the dual decomposition in the tree nodes.

Analysis of highly nonlinear dynamical systems

We have observed bifurcations in the dynamical system of the Penicillin example. The analysis of nonlinear dynamical systems, conclusions for the control of such systems and the development of hierarchical controllers are worth investigating. The topic touches the fields of nonlinear dynamics and chaos.

Online scenario tree adaptation based on state and parameter estimation

In this thesis we generate the scenario tree offline before running the NMPC scheme online. By the estimation task during NMPC we can get more information about the uncertainty. This could be reflected by an online adaptation of the scenario tree. In our Markov framework we have seen that the tree depends on the initial state. We remark that an expansion of scenario tree NMPC by online scenario tree adaptation must stay real-time feasible.

All in all we have shown that scenario tree NMPC is a real-time feasible model-based control method. There is a high potential for future research directions for the scenario tree approach in the field of optimization under uncertainty.

Bibliography

[1] 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.

[2] 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: Proc. of the International Conference on High Performance Scientific Computing, March 2006, Hanoi*, pages 15–24. Springer, 2008.

[3] J. Andersson, J. Frasch, M. Vukov, and M. Diehl. A condensing algorithm for nonlinear mpc with a quadratic runtime in horizon length. *Optimization Online*, (5837), 2017.

[4] V. Bär. Ein Kollokationsverfahren zur numerischen Lösung allgemeiner Mehrpunktrandwertaufgaben mit Schalt- und Sprungbedingungen mit Anwendungen in der optimalen Steuerung und der Parameteridentifizierung. Diploma thesis, Universität Bonn, 1983.

[5] I. Bauer. *Numerische Verfahren zur Lösung von Anfangswertaufgaben und zur Generierung von ersten und zweiten Ableitungen mit Anwendungen bei Optimierungsaufgaben in Chemie und Verfahrenstechnik*. PhD thesis, Ruprecht-Karls-Universität Heidelberg, 1999.

[6] I. Bauer, H.G. Bock, S. Körkel, and J.P. Schlöder. Numerical Methods for Initial Value Problems and Derivative Generation for DAE Models with Application to Optimum Experimental Design of Chemical Processes. In F. Keil, W. Mackens, H. Voß, and J. Werther, editors, *Scientific Computing in Chemical Engineering II*, pages 282–289. Springer, 1999.

[7] D. Beigel. *Efficient goal-oriented global error estimation for BDF-type methods using discrete adjoints*. PhD thesis, Ruprecht-Karls-Universität Heidelberg, 2012.

[8] M. Benzi, G.H. Golub, and J. Liesen. Numerical solution of saddle–point problems. *Acta Numerica*, 14:1–137, 2005.

[9] D. Bertsekas and J.N. Tsitsiklis. *Parallel and distributed computation: Numerical methods*. Prentice Hall, 1989.

[10] D. Bertsimas, D. Brown, and C. Caramanis. Theory and applications of robust optimization. *SIAM Review*, 53:464–501, 2011.

[11] L.T. Biegler. Solution of dynamic optimization problems by successive quadratic programming and orthogonal collocation. *Computers & Chemical Engineering*, 8:243–248, 1984.

[12] H.G. Bock. Numerische Optimierung zustandsbeschränkter parameterabhängiger Prozesse mit linear auftretender Steuerung unter Anwendung der Mehrzielmethode. Diploma thesis, Universität zu Köln, 1974.

[13] H.G. Bock. Recent advances in parameter identification techniques for ODE. In P. Deuflhard and E. Hairer, editors, *Numerical Treatment of Inverse Problems in Differential and Integral Equations*, pages 95–121. Birkhäuser, Boston, 1983.

[14] H.G. Bock. *Randwertproblemmethoden zur Parameteridentifizierung in Systemen nichtlinearer Differentialgleichungen*, volume 183 of *Bonner Mathematische Schriften*. Universität Bonn, 1987.

[15] H.G. Bock, M. Diehl, E.A. 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.

[16] 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.

[17] 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.

[18] R. Bulirsch. Die Mehrzielmethode zur numerischen Lösung von nichtlinearen Randwertproblemen und Aufgaben der optimalen Steuerung. Technical report, Carl-Cranz-Gesellschaft, Oberpfaffenhofen, 1971.

[19] J.C. Butcher. Coefficients for the study of Runge-Kutta integration processes. *Journal of the Australian Mathematical Society*, 3:185–201, 1963.

[20] H. Chen, A. Kremling, and F. Allgöwer. Nonlinear predictive control of a benchmark CSTR. In *Proceedings of the European Control Conference (ECC95)*, pages 3247–3252, Rome, 1995.

[21] F. Clarke. *Functional Analysis, Calculus of Variations and Optimal Control*. Springer, 2013.

[22] C.F. Curtiss and J.O. Hirschfelder. Integration of stiff equations. *Proceedings of the National Academy of Sciences of the USA*, 38:235–243, 1952.

[23] K. Dadhe and S. Engell. Robust nonlinear model predictive control: A multi-model nonconservative approach. In *Book of Abstracts, International Workshop on NMPC, Pavia*, page 24, 2008.

[24] P. Deuflhard and A. Hohmann. *Numerische Mathematik: Eine algorithmisch orientierte Einführung*. Edition de Gruyter, 4th edition, 2008.

[25] M. Diehl. *Real-Time Optimization for Large Scale Nonlinear Processes*. PhD thesis, Ruprecht-Karls-Universität Heidelberg, 2001.

[26] M. Diehl, H.G. Bock, and E. Kostina. An approximation technique for robust nonlinear optimization. *Mathematical Programming*, B 107:213–230, 2006.

[27] 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.

[28] 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.

[29] M. Diehl, R. Findeisen, F. Allgöwer, J.P. Schlöder, and H.G. Bock. Stability of nonlinear model predictive control in the presence of errors due to numerical online optimization. In *Proc. 43th IEEE Conf. Decision Contr.*, pages 1419–1424, Maui, Hawaii, 2003.

[30] J. Dupačová, G. Consigli, and S. W. Wallace. Scenarios for multistage stochastic programs. *Annals of Operations Research*, 100:25–53, 2000.

[31] R. Durrett. *Probability: Theory and Examples*, volume 4. Cambridge University Press, 2010.

[32] E. Fehlberg. Klassische Runge-Kutta-Formeln fünfter und siebenter Ordnung mit Schrittweiten-Kontrolle. *Computing*, 4:93–106, 1969.

[33] E. Fehlberg. Klassische Runge-Kutta-Formeln vierter und niedrigerer Ordnung mit Schrittweiten-Kontrolle und ihre Anwendung auf Wärmeleitungsprobleme. *Computing*, 6:61–71, 1970.

[34] M. Felis. *Modeling Emotional Aspects in Human Locomotion*. PhD thesis, Ruprecht-Karls-Universität Heidelberg, 2015.

[35] 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.

[36] 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.

[37] H.J. Ferreau, A. Kozma, and M. Diehl. A parallel active-set strategy to solve sparse parametric quadratic programs arising in mpc. In *Proceedings of the 4th IFAC NMPC Conference, Noordwijkerhout*, 2012.

[38] R. Findeisen, L. Imsland, F. Allgöwer, and B. A. Foss. Output feedback stabilization of constrained systems with nonlinear predictive control. *International Journal of Robust and Nonlinear Control*, 13(3-4):211–227, 2003.

[39] R. Fletcher. *Practical Methods of Optimization*. Wiley, Chichester, 2nd edition, 1987.

[40] G.F. Franklin, J.D. Powell, and A. Emami-Naeini. *Feedback Control of Dynamic Systems*. Prentice Hall, 7th edition, 2015.

[41] J. V. Frasch. *Parallel Algorithms for Optimization of Dynamic Systems in Real-Time*. PhD thesis, Otto-von-Guericke Universität Magdeburg, 2014.

[42] J. V. Frasch, S. Sager, and M. Diehl. A parallel quadratic programming method for dynamic optimization problems. *Mathematical Programming Computation*, 7:289–329, 2015.

[43] J. V. Frasch, M. Vukov, H.J. Ferreau, and M. Diehl. A dual Newton strategy for the efficient solution of sparse quadratic programs arising in SQP-based nonlinear MPC. Technical report, 2013. Optimization Online 3972.

[44] 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.

[45] J. Frehse. Existence of Optimal Controls I. *Operations Research Verfahren*, 31:213–225, 1979.

[46] J.V. Gallitzendörfer. *Parallel Algorithms for Optimization Boundary Value Problems in DAE*. PhD thesis, Ruprecht-Karls-Universität Heidelberg, 1994.

[47] C.W. Gear. *Numerical Initial Value Problems in Ordinary Differential Equations*. Prentice-Hall, 1971.

[48] T. Gerstner and M. Griebel. Numerical integration using sparse grids. *Numerical Algorithms*, 18:209–232, 1998.

[49] R. K. Greenleaf. *The Servant as Leader*. The Greenleaf Center for Servant Leadership, Westfield, IN, 1991.

[50] A. Griewank, D. Juedes, and J. Utke. ADOL-C: A package for the automatic differentiation of algorithms written in C/C++. *ACM Transactions on Mathematical Software*, 22(2):131–167, 1996.

[51] N. Gröwe-Kuska, H. Heitsch, and W. Römisch. Scenario reduction and scenario tree construction for power management problems. In *Proceedings of the IEEE Power Tech Conference Bologna*, 2003.

[52] L. Grüne and O. Junge. *Gewöhnliche Differentialgleichungen - Eine Einführung aus der Perspektive der dynamischen Systeme*. Springer, Berlin, Heidelberg, 2nd edition, 2016.

[53] L. Grüne and J. Pannek. *Nonlinear Model Predictive Control*. Springer, 2011.

[54] J. Guckenheimer and P. J. Holmes. *Nonlinear Oscillations, Dynamical Systems, and Bifurcations of Vector Fields*. Springer, 1st edition, 1983.

[55] M. Guignard. Generalized Kuhn–Tucker conditions for mathematical programming problems in a Banach space. *SIAM Journal on Control*, 7(2):232–241, 1969.

[56] E.L. Haseltine and J.B. Rawlings. Critical Evaluation of Extended Kalman Filtering and Moving-Horizon Estimation. *Industrial and Engineering Chemistry Research*, 44(8):2451–2460, 2005.

[57] W. K. Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):97–109, 1970.

[58] H. Heitsch and W. Römisch. Scenario tree modeling for multistage stochastic programs. *Mathematical Programming Series A*, 118:371–406, 2009.

[59] J. Hübner. *Distributed Algorithms for Nonlinear Tree-Sparse Problems*. PhD thesis, Universität Hannover, 2016.

[60] J. Hübner, M. Schmidt, and M.C. Steinbach. Optimization techniques for tree-structured nonlinear problems. *Optimization Online*, (5845), 2017.

[61] T. Huschto. *Numerical Methods for Random Parameter Optimal Control and the Optimal Control of Stochastic Differential Equations*. PhD thesis, Ruprecht-Karls-Universität Heidelberg, 2014.

[62] D. Janka. *Sequential quadratic programming with indefinite Hessian approximations for nonlinear optimum experimental design for parameter estimation in differential-algebraic equations*. PhD thesis, Ruprecht-Karls-Universität Heidelberg, 2015.

[63] D. Janka, C. Kirches, S. Sager, and A. Wächter. An sr1/bfgs sqp algorithm for nonconvex nonlinear programs with block-diagonal hessian matrix. *Mathematical Programming Computation*, 2016.

[64] W. Karush. Minima of functions of several variables with inequalities as side conditions. Master's thesis, Department of Mathematics, University of Chicago, 1939.

[65] C. Kirches. *Fast numerical methods for mixed-integer nonlinear model-predictive control*. PhD thesis, Ruprecht-Karls-Universität Heidelberg, 2010.

[66] C. Kirches, H.G. Bock, J.P. Schlöder, and S. Sager. Complementary Condensing for the Direct Multiple Shooting Method. In H.G. Bock, H.X. Phu, R. Rannacher, and J.P. Schlöder, editors, *Modeling, Simulation, and Optimization of Complex Processes. Proceedings of the Fourth International Conference on High Performance Scientific Computing, March 2-6, 2009, Hanoi, Vietnam*, pages 195–206, Heidelberg Dordrecht London New York, 2012. Springer Verlag.

[67] 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.

[68] A. Klenke. *Wahrscheinlichkeitstheorie*. Springer, Berlin, Heidelberg, 3rd edition, 2013.

[69] D. Kouzoupis, E. Klintberg, M. Diehl, and S. Gros. A dual newton strategy for scenario decomposition in robust multi-stage mpc. *Optimization Online*, (5858), 2017.

[70] A. Kozma, J.V. Frasch, and M. Diehl. A distributed method for convex quadratic programming problems arising in optimal control of distributed systems. In *IEEE 52nd Annual Conference on Decision and Control (CDC)*, 2013.

[71] 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, Berkeley, 1951. University of California Press.

[72] W. Kutta. Beitrag zur näherungsweisen Integration totaler Differentialgleichungen. *Zeitschrift für Mathematik und Physik*, 46:435–453, 1901.

[73] C. Leidereiter, D. Kouzoupis, M. Diehl, and A. Potschka. Pruning for scenario tree NMPC with uncertainties described by Markov chains. In preparation.

[74] C. Leidereiter, A. Potschka, and H. G. Bock. Quadrature-based scenario tree generation for Nonlinear Model Predictive Control. In *Proceedings of the 19th IFAC World Congress*, volume 47, pages 11087–11092, 2014.

[75] C. Leidereiter, A. Potschka, and H. G. Bock. Dual decomposition for QPs in scenario tree NMPC. In *Proceedings of the European Control Conference (ECC15)*, pages 1608–1613, 2015.

[76] D.B. Leineweber. Analyse und Restrukturierung eines Verfahrens zur direkten Lösung von Optimal-Steuerungsproblemen. Diploma thesis, Ruprecht-Karls-Universität Heidelberg, 1995.

[77] 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.

[78] 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 (Parts I and II). *Computers & Chemical Engineering*, 27:157–174, 2003.

[79] W. Li and J. Swetits. A new algorithm for solving strictly convex quadratic programs. *SIAM Journal of Optimization*, 7(3):595–619, 1997.

[80] D. C. Liu and J. Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical Programming*, 45(3):503–528, 1989.

[81] S. Lucia, J. Andersson, H. Brandt, A. Bouaswaig, M. Diehl, and S. Engell. Efficient robust economic nonlinear model predictive control of an industrial batch reactor. In *Proceedings of the 19th IFAC World Congress*, pages 11093–11098, 2014.

[82] S. Lucia, J. Andersson, H. Brandt, M. Diehl, and S. Engell. Handling uncertainty in economic nonlinear model predictive control: A comparative case study. *Journal of Process Control*, 24(8):1247–1259, 2014.

[83] S. Lucia and S. Engell. Robust nonlinear model predictive control of a batch bioreactor using multi-stage stochastic programming. In *Proceedings of the European Control Conference (ECC13)*, pages 4124–4129, 2013.

[84] S. Lucia, T. Finkler, and S. Engell. Multi-stage nonlinear model predictive control applied to a semi-batch polymerization reactor under uncertainty. *Journal of Process Control*, 23(9):1306–1319, 2013.

[85] 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.

[86] The Mathworks, Inc., Natick, Massachusetts. *MATLAB version 9.2 (R2017a)*, 2017.

[87] J. Nocedal. Updating quasi-newton matrices with limited storage. *Mathematics of Computation*, 35:773–782, 1980.

[88] J. Nocedal and S.J. Wright. *Numerical Optimization*. Springer Verlag, Berlin Heidelberg New York, 2nd edition, 2006.

[89] M.R. Osborne. On shooting methods for boundary value problems. *Journal of Mathematical Analysis and Applications*, 27:417–433, 1969.

[90] K.J. Plitt. Ein superlinear konvergentes Mehrzielverfahren zur direkten Berechnung beschränkter optimaler Steuerungen. Diploma thesis, Rheinische Friedrich–Wilhelms–Universität Bonn, 1981.

[91] L. S. Pontryagin, V. G. Boltyanskii, R. V. Gamkrelidze, and E. F. Mishchenko. *The Mathematical Theory of Optimal Processes*. Interscience, English Translation, 1962.

[92] A. Potschka. *A direct method for the numerical solution of optimization problems with time-periodic PDE constraints*. PhD thesis, Ruprecht-Karls-Universität Heidelberg, 2011.

[93] 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(2):237–252, 2009.

[94] L. Qi and J. Sun. A nonsmooth version of Newton's method. *Mathematical Programming*, 58:353–367, 1993.

[95] 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.

[96] S. T. Rachev. *Probability metrics and the stability of stochastic models*. Wiley, 1991.

[97] C. V. Rao, J. B. Rawlings, and D. Q. Mayne. Constrained state estimation for nonlinear discrete-time systems: Stability and moving horizon approximations. *IEEE Transactions on Automatic Control*, 48(2):246–258, 2003.

[98] J.B. Rawlings and D.Q. Mayne. *Model Predictive Control: Theory and Design*. Nob Hill, 2012.

[99] C. Runge. über die numerische Auflösung von Differentialgleichungen. *Math. Ann.*, 46:167–178, 1895.

[100] C. Schillings. *Optimal aerodynamic design under uncertainties*. PhD thesis, Universität Trier, 2011.

[101] R. Scholz. Stabiles Condensing für Optimale Steuerung. Master thesis, Universität Heidelberg, 2016.

[102] S.A. Smolyak. Quadrature and interpolation formulas for tensor products of certain classes of functions. *Doklady Akademii Nauk SSSR*, 4:240–243, 1963.

[103] B. Srinivasan, D. Bonvin, E Visser, and S. Palanki. Dynamic Optimization of Batch Processes: II. Role of Measurements in Handling Uncertainty. *Computers & Chemical Engineering*, 27(1):27–44, 2002.

[104] B. Srinivasan, S. Palanki, and D. Bonvin. Dynamic Optimization of Batch Processes: I. Characterization of the Nominal Solution. *Computers & Chemical Engineering*, 27:1–26, 2003.

[105] M.C. Steinbach. *Fast recursive SQP methods for large-scale optimal control problems*. PhD thesis, Ruprecht-Karls-Universität Heidelberg, 1995.

[106] M.C. Steinbach. Recursive direct algorithms for multistage stochastic programs in financial engineering. Technical report, ZIB, 1998.

[107] M.C. Steinbach. Tree-Sparse Convex Programs. *Math. Methods Oper. Res.*, 56(3):347–376, 2002.

[108] J. Stoer and R. Bulirsch. *Introduction to Numerical Analysis*. Springer, 3rd edition, 2002.

[109] T.H. Tsang, D.M. Himmelblau, and T.F. Edgar. Optimal control via collocation and non-linear programming. *International Journal on Control*, 21:763–768, 1975.

[110] A. Uppal, W.H. Ray, and A.B. Poore. On the dynamic behavior of continuous stirred tank reactors. *Chemical Engineering Science*, 29(4):967 – 985, 1974.

[111] A. Wächter and L. T. Biegler. On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57, 2006.

[112] S.J. Wright. *Primal-Dual Interior-Point Methods*. SIAM Publications, Philadelphia, 1997.

[113] F. Wu. LMI-based robust model predictive control and its application to an industrial CSTR problem. *Journal of Process Control*, 11:649–659, 2001.

[114] V.M. Zavala and L.T. Biegler. The advanced–step NMPC controller: optimality, stability and robustness. *Automatica*, 45(1):86–93, 2009.

List of Acronyms

BDF	Backward Differentiation Formula
BFGS	Broyden-Fletcher-Goldfarb-Shanno
BVP	Boundary Value Problem
CMR	Control Move Regularization
CQ	Constraint Qualification
CSTR	Continuous Stirred Tank Reactor
DAE	Differential-Algebraic Equation
EHE	External Heat Exchanger
IND	Internal Numerical Differentiation
IVP	Initial Value Problem
KKT	Karush-Kuhn-Tucker
LICQ	Linear Independence Constraint Qualification
LP	Linear Programming Problem
MCMC	Markov Chain Monte Carlo
MLI	Multi-Level Iteration
NLP	Nonlinear Programming Problem
NMPC	Nonlinear Model Predictive Control
OCP	Optimal Control Problem
ODE	Ordinary Differential Equation
QP	Quadratic Programming Problem
RTI	Real-Time Iteration
SQP	Sequential Quadratic Programming