EFFICIENT SOLUTION APPROACH TO NONLINEAR OPTIMAL CONTROL PROBLEMS AND APPLICATION TO AUTONOMOUS DRIVING

D I S S E R T A T I O N

zur Erlangung des akademischen Grades
Doktoringenieur (Dr.-Ing.)

vorgelegt der
Fakultät für Informatik und Automatisierung der Technische Universität Ilmenau

von
M. Sc. Evgeny Lazutkin

geboren am 11. September 1989 in Moskau, Russland

Gutachter:

1. Univ.-Prof. Dr.-Ing. habil. Pu Li, Technische Universität Ilmenau
2. Prof. Dr.-Ing. habil. Thomas Rauschenbach, Fraunhofer IOSB-AST Ilmenau
3. Univ.-Prof. Dr.-Ing. habil. Rudibert King, Technische Universität Berlin

Tag der Einreichung : 02.07.2018
Tag der wissenschaftlichen Aussprache: 06.11.2018

urn:nbn:de:gbv:ilm1-2018000504

I dedicate this doctoral thesis to my family, especially my beloved wife Yauheniya and our wonderful daughter Valerie.

Acknowledgements

Firstly, I would like to thank my supervisor, Prof. Dr.-Ing. habil. Pu Li for his patience, encouragement and advices. During my master program he gave me an inspiration to dedicate my research activity in the area of applied dynamic optimization. Besides my supervisor, I would like to thank my co-supervisor Dr. rer. nat. Abebe Geletu for his support, permanent inspiration and helpful discussions. Without their guidance, constructive and collegial cooperation, this research could not have been successfully conducted.

I am eternally grateful to my loving parents Tatiana and Yuri for their unreserved love and support throughout the years of study. Without their support I could not finish my thesis. I would like to express my profound gratitude to my sincere brother Dmitry for his advices. I am especially thankful to my beloved wife Yauheniya for her every day support, love and patience.

I would like also to thank all my friends whom I met in Ilmenau, especially Oleksandr and Maria, Vasilii and Anna, Dzmitry and Daria, Dmitrii and Dasha for being wonderful company during all these years.

I want to express my gratefulness to the stuff members of the SOP group for their constant support during my research work, exclusively Dr.-Ing. Siegbert Hopfgarten, Prof. Dr.-Ing. habil. Horst Puta, Mr. Björn Töpper, Dipl.-Ing. (FH) Jens Hollandmoritz, M. Sc. Shih-Jan Lin and M. Sc. Xujiang Huang. It was an honor to me to work with you within our research group. I wish them all the best and good luck.

Some parts of this research work were conducted within the Model Driven Physical Systems Operation (MODRIO) project under the financial support of the ITEA2 and German Ministry of Education and Research (BMBF). This support is gratefully acknowledged. In addition, I would like to thank the Audi AG for the equipment and technical support during the Audi Autonomous Driving Cup 2017 competition, where our team won the first place.

Finally, I am highly grateful for the reviewers of this thesis and for the promotion committee.

Abstract

This thesis deals with the numerical solution of dynamic nonlinear optimization problems and the development of new methods for their analysis in order to increase the efficiency of calculations. The operation of many natural and technical processes can be formulated as a nonlinear optimal control problem with constraints. Because of the increasing complexity, the solution of such a problem becomes challenging, in particular if it has to be obtained in real-time. The approach of combined multiple-shooting with collocation is efficient for solving such problems even if they contain fast dynamics. Thus, the first target of this work is to further improve its computational performance by providing an analytical Hessian and realizing a parallel-computing scheme. First, the formulas for computing the second-order sensitivities for the combined approach were derived. Using multiple-shooting, the solutions of model equations and evaluations of both first-order and second-order sensitivities can be provided independently for each time interval. Therefore, the second contribution is dedicated to the realization of a parallel computing scheme. As a result, a high speedup factor is attained through parallelization leading to reduction of computational expenses. As a third contribution, a novel control-variable correlation analysis was introduced, which indicates the necessity of employing the analytical Hessian instead of its approximation to efficiently solve an optimization problem. The numerical performance of these three contributions was demonstrated through challenging dynamic optimization problems including optimal control of a large-scale problem containing more than one thousand dynamic variables.

The combined method converts the continuous dynamic optimization problem into a nonlinear programming problem using a given number of time intervals. However, there have been no comprehensive rules to properly choose this number. Therefore, the fourth target of this work is devoted to the analysis of the underlying optimization problem with the special focus on the number of discrete time intervals. From the application point of view, the number of time intervals should be selected to simultaneously achieve the balance between the numerical accuracy and the computation load for solving the discretized optimization problem. Moreover, it is imperative to find the minimum number of time intervals to guarantee this balance. Thus, in the context of collocation on finite elements, a novel bilevel approach was proposed, where the outer loop is responsible for finding the minimum number of time intervals and the inner loop evaluates an upper limit of the approximation error by solving an error maximization problem by manipulating the control variables. In this way, a minimum number of time intervals can be determined guaranteeing a user defined error tolerance. Moreover, the impact of the initial conditions on the maximum approximation error is taken into account so that the determined number of intervals is valid for varying initial conditions and thus can be applied to nonlinear model predictive control (NMPC). Several case studies were conducted to demonstrate the efficacy of the proposed approach. Both theoretically developed methods as well as the combined

approach were implemented using open-source software as a generalized framework for testing purposes.

Finally, the developed methods were applied to autonomous driving in the NMPC framework. Autonomous driving is the current trend in the automotive industry with the aim of designing and producing fully automated or self-driving vehicles. Control design and field operation of autonomous vehicles impose several challenges and thus extensive as well as intensive research studies need to be made to cover the growing industrial demand. In this work, the vehicle motion was modeled as a dynamic optimization problem which is efficiently solved on-line. The successful test of the NMPC with two model vehicles (with scale of 1:5 and 1:8 to real vehicles) demonstrated the effectiveness of the developed approach.

Keywords: Multiple-shooting, collocation, interior-point method, nonlinear model predictive control, autonomous driving, error estimation

Zusammenfassung

Diese Arbeit beschäftigt sich mit der numerischen Lösung von dynamischen nichtlinearen Optimierungsaufgaben und der Entwicklung neuer Methoden für deren Analyse, um die Effizienz der Berechnungen zu erhöhen. Der Betrieb vieler natürlicher und technischer Prozesse kann als nichtlineares Optimierungsproblem mit Beschränkungen formuliert werden. Aufgrund der steigenden Komplexität wird die Lösung eines solchen Problems zu einer Herausforderung, insbesondere wenn das Problem in Echtzeit gelöst werden muss. Der Ansatz des kombinierten Mehrfachschießverfahren mit Kollokation ist effizient, um solche Probleme zu lösen, auch wenn sie eine schnelle Dynamik aufweisen. So ist das erste Ziel dieser Arbeit die weitere Verbesserung der Rechenleistung durch die Bereitstellung einer analytischen Hesse-Matrix und die Realisierung eines Parallelberechnungs-Schemas. Zunächst wurden die Formeln zur Berechnung der Sensitivitäten zweiter Ordnung für den kombinierten Ansatz abgeleitet. Mit Hilfe des Mehrfachschießverfahrens können die Lösungen von Modellgleichungen und Auswertungen von Sensitivitäten erster und zweiter Ordnung für jedes Zeitintervall unabhängig voneinander berechnet werden. Der zweite Beitrag widmet sich daher der Realisierung eines parallelen Rechenschemas. Dadurch wird ein hoher Beschleunigungsfaktor durch Parallelisierung erreicht, der zu einer Reduzierung des Rechenaufwands führt. Als dritter Beitrag wurde eine neuartige Korrelationsanalyse der Steuergrößen eingeführt, die auf die Notwendigkeit hinweist, die analytische Hesse-Matrix anstelle seiner Approximation einzusetzen, um ein Optimierungsproblem effizient zu lösen. Die numerische Leistung dieser drei Beiträge wurde mit Hilfe von herausfordernden dynamischen Optimierungsproblemen einschließlich der optimalen Steuerung eines großen Problems mit mehr als tausend dynamischen Variablen demonstriert.

Die kombinierte Methode wandelt das Problem der kontinuierlichen dynamischen Optimierung in ein nichtlineares Programmierungsproblem mit einer vorgegebenen Anzahl der Zeitintervalle um. Es gibt jedoch keine umfassenden Regeln, um diese Anzahl der Zeitintervalle passend zu wählen. Daher widmet sich das vierte Ziel dieser Arbeit der Analyse der zugrunde liegenden Optimierungsprobleme mit dem besonderen Fokus auf der Anzahl der diskreten Zeitintervalle. Aus Anwendungssicht sollte die Anzahl der Zeitintervalle so gewählt werden, dass gleichzeitig die Bilanz zwischen der numerischen Genauigkeit und der Rechenlast zur Lösung des diskreten Optimierungsproblems erreicht wird. Darüber hinaus ist es unerlässlich, die Mindestanzahl an Zeitintervallen zu finden, um diese Genauigkeit zu gewährleisten. So wurde im Rahmen der Kollokation auf finiten Elementen ein neuartiger Bilevel-Ansatz vorgeschlagen, bei dem die äußere Schleife für die Ermittlung der minimalen Anzahl von Zeitintervallen zuständig ist und die innere Schleife eine obere Grenze des Approximationsfehlers auswertet, indem sie ein Fehlermaximierungsproblem durch Manipulation der Steuergrößen löst. Auf diese Weise kann eine Mindestanzahl von Zeitintervallen festgelegt werden, die eine benutzerdefinierte Fehlertoleranz gewährleistet. Außerdem wird der Einfluss der Anfangsbedingungen auf den maximalen Approximationsfehler berücksichtigt, so dass

die ermittelte Anzahl von Intervallen für unterschiedliche Anfangsbedingungen gilt und somit für die nichtlineare modellprädiktive Regelung (engl.: nonlinear model predictive control (NMPC)) angewendet werden kann. Mehrere Fallstudien wurden verwendet, um die Wirksamkeit des vorgeschlagenen Ansatzes zu demonstrieren. Sowohl die theoretisch entwickelten Methoden als auch der kombinierte Ansatz wurden mit Hilfe von Open-Source-Software als allgemeines Framework für Testzwecke implementiert.

Schließlich wurden die entwickelten Methoden auf das autonome Fahren im NMPC-Framework angewendet. Autonomes Fahren ist der aktuelle Trend in der Automobilindustrie mit dem Ziel, vollautomatisierte oder selbstfahrende Fahrzeuge zu entwickeln und zu produzieren. Reglerentwurf und -betrieb von autonomen Fahrzeugen stellen mehrere Herausforderungen dar, weshalb umfangreiche und intensive Forschungsarbeiten notwendig sind, um den wachsenden industriellen Bedarf abzudecken. Die Fahrzeugbewegung wurde als ein dynamisches Optimierungsproblem dargestellt, das online effizient gelöst wird. Der erfolgreiche Test der NMPC mit zwei Modellfahrzeugen (im Maßstab 1:5 und 1:8 im Vergleich zum realen Fahrzeug) zeigte die Effizienz des entwickelten Ansatzes.

Schlüsselwörter: Mehrfachschießverfahren, Kollokationsverfahren, Innere-Punkte-Methode, Nichtlineare modellprädiktive Regelung, Autonomes Fahren, Fehlerschätzung

List of Abbreviations

GPS	Global positioning system
C2C	Car-to-car (communication)
C2I	Car-to-infrastructure (communication)
CMSC	Combined multiple-shooting with collocation (method)
AH	Analytical Hessian
BFGS	Broyden-Fletcher-Goldfarb-Shanno (method)
NMPC	Nonlinear model predictive control
NOCP	Nonlinear optimal control problem
ODE	Ordinary differential equation
DAE	Differential algebraic equation
CVP	Control vector parameterization
BVP	Boundary value problem
PID	Proportional-Integral-Derivative (controller)
PI	Proportional-Integral (controller)
IPM	Interior-point method
KKT	Karush-Kuhn-Tucker (optimality conditions)
NLP	Nonlinear programming (problem)
NS	Newton solver
SC	Sensitivity computation
ADTF	Automotive data and time-triggered framework
IPOPT	Interior-point optimizer
MPC	Model predictive control

List of Symbols

In the list below the most important symbols are gathered together.

$x(t)$	Vector of differential state variables
$z(t)$	Vector of algebraic variables
$u(t)$	Vector of control variables
p	Vector of time independent parameters
t	Time
J	Objective function
M	Mayer term
L	Lagrange term
x_0	Initial values of state variables
t_0	Initial time
t_f	Final time
h_p	Path constraints
h_t	Terminal constraints
N	Number of time intervals
Δt	Length of the time interval
n_x	Number of state variables
n_z	Number of algebraic variables
n_u	Number of control variables
n_{pc}	Number of path constraints
n_t	Number of terminal constraints
G	Nonlinear equation system
X^c	Vector of state and algebraic variables at collocation points in single time interval
X^p	Vector of parameterized state variables in single time interval
V	Vector of parameterized control variables in single time interval
N_{xz}	Total number of variables at collocation points in single time interval
Φ	Vector function of the second-order sensitivities with respect to parameterized state variables
Ψ	Vector function of the second-order sensitivities with respect to parameterized control variables
N_S^X	Dimension of the Φ function
N_S^U	Dimension of the Ψ function
N_{pc}	Total number of path constraints
N_c	Number of collocation points
g_i	i-th equality constraint
h_j	j-th inequality constraint
ω	Vector of the optimization variables
∇	Nabla operator
λ_i	Lagrange multiplier for the i-th equality constraint
μ_j	Lagrange multiplier for the j-th inequality constraint

s^s	Slack variables
H	Hessian matrix of the Lagrangian function
α_k	Step-size in k-th iteration
$\alpha_{i,j}$	Correlation coefficient between i-th and j-th column
Θ	Angle between two columns in sensitivity matrix
$x_N(t)$	Numerically computed state profile using the Lagrange polynomials
$x_{N,k}$	Value of the state trajectory at k-th collocation point
$\tau_{i,j}$	j-th collocation point in i-th time interval
$\hat{t}_i$	Noncollocation point in i-th time interval
$e(\hat{t}_i)$	Value of the error function at noncollocation point
$e_{max}^{(q)}$	Maximum approximation error of the q-th state variable in the i-th interval
$x^{(q)}(\hat{t}_i)$	Analytical solution of the q-th state variable at the noncollocation point $\hat{t}_i$
$x_N^{(q)}(\hat{t}_i)$	Numerical solution of the q-th state variable at the noncollocation point $\hat{t}_i$
$e_{max}^{(l)}$	Maximum approximation error in l-th iteration
ΔN^-	Decrement of the number of time intervals
ΔN^+	Increment of the number of time intervals
ε	User-defined error tolerance
ε_N	Tolerance of the Newton method
ε_I	Tolerance of the IPOPT
ε_A	Tolerance of the bilevel approach
E	Expected value (operator)
A	State matrix
B	Control matrix
R	Measurement error covariance matrix
Q	Process noise covariance matrix
P_k^-	A priori estimate error covariance
P_k	A posteriori estimate error covariance
c_α	Tire cornering stiffness
J_I	Rotational moment of inertia
P_x	Weighting factor for the final longitudinal coordinate
P_y	Weighting factor for the final lateral coordinate
q_x	Weighting factor for the longitudinal coordinates
q_y	Weighting factor for the lateral coordinates
r_v	Weighting factor for the velocity
r_δ	Weighting factor for the steering angle
$\beta(t)$	Sideslip angle
$\psi(t)$	Heading angle
$v(t)$	Vehicle velocity
$\delta(t)$	Steering angle
x_{abs}^R	Absolute longitudinal coordinate of the robot
y_{abs}^R	Absolute lateral coordinate of the robot
x_{abs}^H	Absolute longitudinal coordinate of the obstacle
y_{abs}^H	Absolute lateral coordinate of the obstacle

x_{rel}	Relative longitudinal coordinate of the obstacle
y_{rel}	Relative lateral coordinate of the obstacle

List of Figures

List of Tables

List of Publications

Journal articles:

I. **E. Lazutkin**, A. Geletu, and P. Li: An approach to determining the number of time intervals for solving dynamic optimization problems, Ind. Eng. Chem. Res., 12(57), pp. 4340-4350, 2018.

II. **E. Lazutkin**, A. Geletu, S. Hopfgarten, and P. Li: An analytical Hessian and parallel-computing approach for efficient dynamic optimization based on control-variable correlation analysis, Ind. Eng. Chem. Res., 48(54), pp. 12086-12095, 2015.

Conference articles:

III. E. Drozdova, S. Hopfgarten, **E. Lazutkin**, and P. Li: Autonomous driving of a mobile robot using a combined multiple-shooting and collocation method, 9^{th} IFAC symposium on intelligent autonomous vehicles, 15(49), pp. 193-198, 2016.

IV. **E. Lazutkin**, S. Hopfgarten, A. Geletu, and P. Li: A toolchain for solving dynamic optimization problems using symbolic and parallel computing, Proceedings of the 11^{th} International Modelica Conference, pp. 311-320, 2015.

V. **E. Lazutkin**, A. Geletu, S. Hopfgarten, and P. Li: Modified multiple shooting combined with collocation method in JModelica.org with symbolic calculations, Proceedings of the 10^{th} International Modelica Conference, pp. 999-1006, 2014.

Other contributions to conferences, workshops, etc.:

VI. S.-J. Lin, **E. Lazutkin**, X. Huang, W. Zhang, N.-T. Pham, P. Li: Autonomes Fahren mit modellgestützter Echtzeitoptimierung und künstlicher Intelligenz, Audi Autonomous Driving Cup 2017 Competition, Ingolstadt, Germany, 2017.

VII. **E. Lazutkin**, P. Li: Verbesserter Ansatz des kombinierten Mehrfachschießverfahrens mit Kollokation zur dynamischen Optimierung und Anwendung auf autonomes Fahren, 51. Regelungstechnisches Kolloquium, Boppard, Germany, 2017.

VIII. **E. Lazutkin**, A. Geletu, S. Hopfgarten, P. Li: Sensitivity computation based on CasADi for dynamical optimization, 16^{th} European Workshop on Automatic Differentiation, Jena, Germany, 2014.

Contents

Chapter 1

Introduction

1.1 Motivation

Autonomous driving is the current trend in the automotive industry with the aim of designing and producing fully automated or self-driving vehicles. The deployment and operation of autonomous vehicles in realistic road scenarios brings several challenges and thus extensive as well as intensive research studies need to be made to cover the growing industrial demand. In the past decade, significant progress has been made in the research and application of driving assistance features, e.g., emergency braking and lane keeping assistance systems. These features augment existing active safety systems such as electronic stability program. Autonomous driving has attracted a lot of attention both from the public and the government due to the potential to transform the automobile industry and transportation system. The main interest lies in the dramatic increase of the road safety. Autonomous vehicle is a developing technology which may prove to be the next big revolution in overall transportation. As of now, several major companies including Audi, BMW and Mercedes-Benz are developing and testing prototype vehicles with plans to eventually release the technology to the market in the near future. Autonomous cars are no longer just an unrealistic element of futuristic science-fiction films. This is a real technology. Within the ever-increasing ongoing process of globalization, people have become more dependent on the transportation system. Enormous demand on the efficient solution approaches is required due to several major reasons: (i) more than 90% of all traffic accidents are due to human error (ii) traffic jams are often caused by distracted drivers (iii) air pollution and (iv) finding the optimal routes. Therefore, autonomous driving technologies can solve these issues and are predicted to be one of the most important innovations within the automotive industry.

When designing and implementing an autonomous driving vehicle, there exist three major goals. The first place is dedicated to the efficiency which leads to the increasing of the maximum throughput on highways and reduction of the air pollution. The second goal is safety, which is responsible for elimination of the mortality due to car accidents. The third goal is the affordability. Hence, the designed control system should be reliable and inexpensive. These three goals require advances in computational power and sensor technology. Thus, the development of autonomous vehicle is concerned with the design of an intelligent system with highly advanced control algorithms.

Nowadays, there are six formal definitions of levels of automation from the Society of Automotive Engineers (SAE) within the norm SAE J3016. The meaning of these

levels is summarized in Figure 1.1. The levels span from no automation where the driver operates the vehicle, to fully automated vehicle where the system controls the entire driving. One major difference in the SAE definition: who is monitoring the driving environment? At the lower levels 0, 1 and 2 the monitoring is made by the human driver while in the higher levels 3, 4 and 5 the vehicle will have total control of driving during specific circumstances. In the zero level, the full-time performance is conducted

Level	Name	Execution of steering and acceleration/braking	Monitoring of driving environment	Fallback performance of driving task	Driving modes
0	No automation	Driver	Driver	Driver	Not available
1	Driving assistance	Driver and system	Driver	Driver	Some driving modes
2	Partial automation	System	Driver	Driver	Some driving modes
3	Conditional automation	System	System	Driver	Some driving modes
4	High automation	System	System	System	Some driving modes
5	Full automation	System	System	System	All driving modes

Figure 1.1: Levels of automation from SAE

by the driver, including all aspects of the dynamic driving task. In the next level of automation, some driving modes are supported by the system. The execution by a driver assistant system of either steering or velocity control using information about the driving environment is conducted with the expectation that the driver performs all remaining tasks. In addition, the driver should be prepared to perform the task completely by himself. The difference between first and second level is that the system is capable to control both steering and velocity. However, the remained operational condition are still the same. In the third level, an automated driving system of all aspects of the dynamic driving task is introduced, with the condition that the driver will respond appropriately to a request to intervene. The fourth level of automation, the system must react by itself, even if a driver does not respond appropriately to a request to intervene the driving task. The last level of automation means completely autonomous vehicle, where the system takes all aspects of the dynamic driving task under all roadway and environmental conditions. In this level, the steering wheel or pedals are not even needed. The current state-of-the-art in the automobile industry is the third level. Therefore, further research towards level five is highly required.

Highly automated driving requires complex control mechanisms that make optimal decisions based on the current measurements and information. The tactical level of the decision-making process implements this intelligence through a sophisticated hierarchy of a high-level behavioral strategy and a low-level maneuver planning. The

determined driving decisions will be passed to the subsequent trajectory planning. It plays a central role in the automated driving function and has a great impact on the driving comfort and the overall traffic safety. The framework should allow diversity in the decision-making for various traffic situations and modular expandability of the control system. It means that the existing control framework should be easily extended by new behavioral strategies to deal with further traffic situations. The

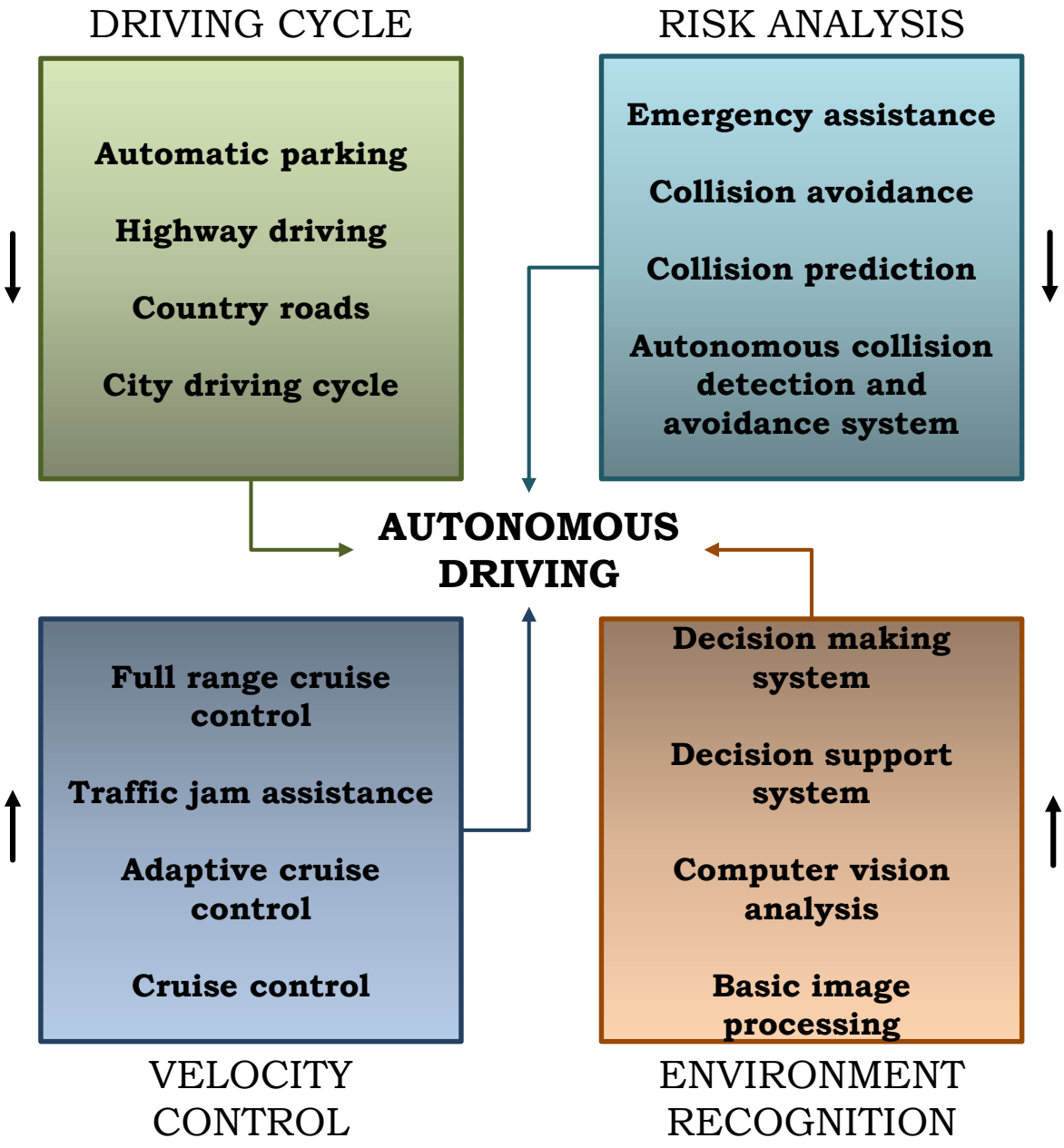

Figure 1.2: The way to autonomous driving

important parts are indicated in Figure 1.2, where the major blocks describe the way

to the autonomous driving concept. The side arrows in Figure 1.2 indicate the rising technological complexity. Based on the Figure 1.2, there are several aspects that need to be discussed.

Primarily, to be able to respond to all different kinds of traffic conditions, the car must be able to detect other cars, pedestrians and information signs in all types of weather conditions and for all types of road situations. This is a prerequisite for the car to know how to act according to the situation on the road. Depending on the information collected through the sensors, the car makes decision on how to handle each specific scenario, depending on the programmed rules and procedures. This enables continuous improvement of the system in the future development process.

Secondly, autonomous cars navigate by using a combination of several systems such as radars, virtual maps, satellites and other sensors. It is possible to drive autonomously if having access to a high precision 3D map of the surrounding environment. The construction of such maps is a highly complicated and expensive task. Unfortunately, the navigation systems used today based on GPS does not always guarantee sufficient precision and can therefore not be used by itself for the navigation. To be able to drive safely the car needs to know its position on the road in the accuracy of centimeters. Among the different approaches and techniques, following hardware may be found: (i) laser scanner (ii) lidar (iii) radar (iv) high-precision GPS (v) computer vision. An automated car uses a combination of hardware and software to locate itself in the real world.

Laser scanner is a device that emits light through a process of optical amplification based on the stimulated emission of electromagnetic radiation. Lidar is a remote sensing technology that measures distance by illuminating a target with a laser and analyzing the reflected light. Lidar uses ultraviolet, visible or near infrared light to image objects. A narrow laser-beam can map physical features with very high resolutions. Radar is an object-detection system that uses radio waves to determine the range, altitude, direction, or speed of objects. The radar antenna transmits pulses of radio waves or microwaves that bounce off any object in their path. The object returns a tiny part of the wave's energy to antenna that is usually located at the same site as the transmitter. The GPS is a space-based satellite navigation system that provides location and time information in all weather conditions, anywhere on or near the Earth where there is an unobstructed line of sight to four or more GPS satellites. The computer vision is any form of signal processing for which the input is an image, such as a photograph or video frame. Most image-processing techniques involve treating the image as a two-dimensional signal and applying standard signal-processing techniques to it.

The objective of automated driving is approached mainly from two different directions. On the one hand, the automotive industry tries to reach this goal from the practical and economical points of view. On the other hand, the robotic researchers, who are working on algorithms dealing with environment perception and artificial intelligence, try to realize different applications for autonomous systems besides driver-less transportation systems. One of the big challenges nowadays is to combine the best of these two directions. Moreover, the reliable operation of autonomous vehicles is still

a challenge and a major barrier in the large scale acceptance and deployment of this technology.

Driving intelligence is such a huge complicated task due to all of the processes that are happening in the background. Autonomous vehicles face essentially five challenges: (i) technology integrity (ii) infrastructure investment (iii) consumer acceptance (iv) legislation and (v) business. To help overcoming these challenges, autonomous vehicles should be capable of exchanging information among themselves to increase the performance of the control system by gathering more information for the underlying algorithms. For instance, vehicles can communicate in real-time with each other − car-to-car communication (C2C) and with the infrastructure − car-to-infrastructure communication (C2I). Under infrastructure one can understand the communication with, e.g., mobile phones or the road itself. Decentralizing the vehicle perception would decrease the need of standalone solutions, complex sensing devices, and artificial intelligence technology, while it would increase the redundancy of the system.

However, many critical challenges still need to be overwhelmed. From the technical perspective, the autonomous driving related technologies are not so reliable. From the cost perspective, the highly precise sensors are still expensive in comparison to the cost of the vehicles. The reliable operation of autonomous vehicles is still a challenge and a major barrier in the large scale acceptance and deployment of the technology. Right now, autonomous vehicles are still in a research and development phase. Based on theories of innovation and design, this doctoral thesis will analyze the development and improvement of efficient solution methods for autonomous vehicles with the main focus on the real-time optimal control strategies. This research will examine essential issues of autonomous driving technology that are the basic foundation of the driving intelligence.

The current development stage of the autonomous driving is at a stage of progress that it will soon put this concept into reality. Undoubtedly, driving requires forecasting (looking-ahead), which can be highly uncertain in some driving scenarios. The simplest way to deal with uncertain forecasts in autonomous driving is to reduce the vehicle speed and wait until the uncertainty becomes negligible. Nevertheless, this is not the preferred driving mode. For such scenarios, control design for autonomous driving is a real challenge. The use of classical controllers would entail several heuristics to assess the criticality of the situation and accordingly activate a control strategy which is finely tuned for that particular situation. It has been a standard approach in the automotive industry with regards to the development of active safety systems. However, this approach does not safely extend to the domain of self-driving cars. Therefore, further studies are needed to provide the efficient, safe and stable operations, which makes this doctoral work actual and important.

1.2 Objectives and contributions

The first objective is to investigate the numerical solution of the optimal control problem to provide fast numerical computation with the focus on the application to the

autonomous vehicles. Although much progress has been made in improving computation efficiency, further studies need to be carried out to explore the potential of the existing approaches. Therefore, the aim is to extend the combined multiple-shooting with collocation (CMSC) method in such a way that the computation efficiency can be enhanced in comparison to those of the existing approaches. This extension leads to following novel contributions. First, the formulas for analytical second-order sensitivities are derived in the context of the CMSC method. As a result, an analytical Hessian (AH) can be used to improve the solution performance. However, the computational cost for an AH is higher than that for a numerical approximation, such as the Broyden-Fletcher-Goldfarb-Shanno (BFGS) method. Therefore, as a second contribution, a new method is introduced for evaluating the profitability of using an AH by analyzing correlations between control (independent) variables of dynamic optimization problems. Correlation relationships are analyzed analytically by checking linear dependencies of the columns of the sensitivity matrix and numerically by calculating the angles between the corresponding columns. The latter is achieved by a simulation step with control signals in the form of a pseudorandom binary sequence. Consequently, the degree of difficulty for solving a dynamic optimization problem under consideration can be examined a priori. Third, by employing the decomposed structure of the CMSC method, parallel processing is implemented for computing state values and both first- and second-order sensitivities using automatic differentiation. Through case studies, this parallel-computing strategy can speed up the computation considerably, particularly for large-scale problems [61].

The second objective is dedicated to investigation of the numerical solution of the differential equations within the optimal control problem. Of course, many previous studies have devoted considerable efforts on the analysis and improvement of the accuracy of numerical methods for the solution of both differential equations and dynamic optimization problems. However, most available approaches are principally a posteriori methods, i.e., to ensure a given error tolerance, a number of time intervals is defined and updated during the solution of the discretized optimization problem. The shortcoming of an a posteriori method lies in the fact that the error estimation is made under a fixed operating condition, i.e., with given control profiles and a given initial condition. In fact, besides the choice of a discretization scheme, control profiles and initial states have a significant influence on the numerical error. Therefore, a novel bilevel approach for a priori error estimation by taking the effect of both the discretization scheme and the operating condition into account is presented. In the context of collocation on finite elements, an upper limit of the approximation error will be gained by formulating and solving an error maximization problem. The proposed bilevel approach is based on error analysis of state profiles at specific noncollocation points considering controls and initial states as decision variables. In this way, a minimum number of time intervals will be determined, which guarantees a user-defined error tolerance for solving the original dynamic optimization problem. As a result, the a priori determined number of time intervals is valid for varying operating conditions and thus can be used for nonlinear model predictive control (NMPC) [62].

The third objective is to implement the theoretically developed methods from the

contributions above into a generalized framework for the solution and analysis of the dynamic optimization problems [60, 63], the generality and effectiveness of which are demonstrated via several applications.

The fourth objective is focused on addressing the fundamental problem of autonomous driving, because it is perhaps the hottest topic in the automotive industry right now. The design and implementation of real-time control is the most important task for autonomous vehicle. Therefore, this dissertation focuses on the challenges associated with the application of control strategies for fully autonomous driving coupled with prediction strategies of vehicle behavior. To achieve this goal, the driving task is described as a nonlinear optimal control problem [34]. From the motivation described in the previous section, the following objectives for this doctoral thesis can be identified. Once the mathematical vehicle model is available, it will be firstly investigated through the proposed framework using the developed control-variable correlation analysis and bilevel approach. Aftermath, the closed-loop solution of the problem will be realized in the form of NMPC. The implemented real-time framework for the autonomous driving will be tested using two different model vehicles and several road situations.

1.3 Outline of the thesis

A schematic of this dissertation is depicted in Figure 1.3 and serves as a pictorial guide of how the chapters are linked. The contributions of the individual chapters are highlighted below.

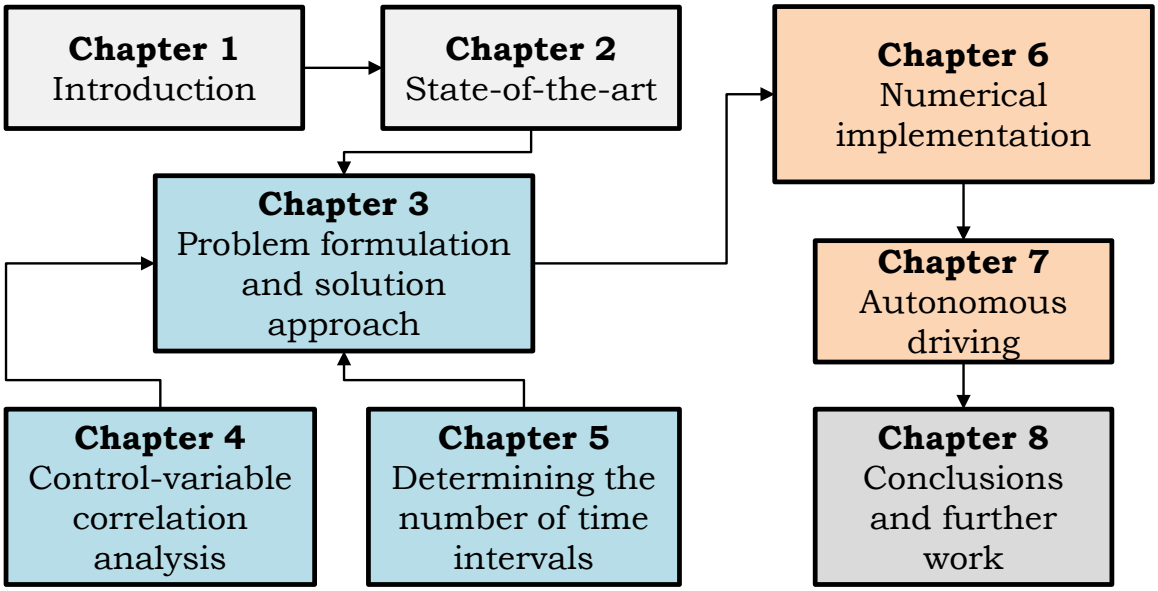

Figure 1.3: Thesis structure

Chapter 1 describes the motivation of this thesis and highlights important aspects and problems encountered in the autonomous driving.

Chapter 2 introduces the state-of-the-art and the progress achieved by the research work in order to demonstrate that the results of this thesis are novel and important.

Chapter 3 formulates the nonlinear optimal control problem (NOCP) that is addressed in this dissertation. Since this thesis deals with a direct solution approach of continuous NOCP, a detailed treatment of this problem with the help of CMSC method is presented, which leads to the so-called nonlinear programming (NLP) problem. Furthermore, it is shown how the resulted NLP problem is solved using the primal-dual interior-point method within the NMPC framework. Moreover, the extension of the CMSC method to handle the second-order analytical sensitivities is presented.

However, it is not always required to use analytical second-order sensitivities. Therefore, Chapter 4 presents the control-variable correlation analysis to investigate the necessity of the employment of second-order sensitivities. Several case studies are used to demonstrate this issue.

Because the NOCP is solved using direct solution approach, the first step is to discretize the problem using a certain number of time intervals. However, until now there were no comprehensive rules. Hence, Chapter 5 presents an approach to determining the number of time intervals. The numerical implementation issues are discussed in Chapter 6.

Chapter 7 demonstrates the realization of the NMPC scheme using two model vehicles using different task in the underlying NOCP, which are highly relevant for the enhancement of the autonomous driving technology. The obtained results and further research directions are thoroughly discussed in Chapter 8. This doctoral thesis ends with an appendix, bibliography and index.

Chapter 2
State-of-the-art

The autonomous driving task in this work is considered as a dynamic optimization problem which includes the model equations of vehicle motion. The solution of dynamic optimization problems with ordinary differential equations (ODEs) or differential-algebraic equations (DAEs) still poses a great challenge especially for online implementations such as nonlinear model predictive control and real-time parameter estimation [86, 101, 50]. Although many efficient approaches have been developed to address large-scale dynamic optimization problems, further studies are needed to enhance the efficiency of existing solution algorithms in the context of discretization, sensitivity calculation, and parallel computing.

Two major and widely used discretization methods are collocation on finite elements [27, 16, 17] and the multiple-shooting strategy [19, 64, 58]. In the collocation approach, the DAE system is fully discretized, leading to a large number of optimization variables in the nonlinear programming (NLP) formulation along with a corresponding number of equality constraints. On the other hand, the multiple-shooting approach decomposes the dynamic optimization problem into smaller problems over individual time intervals, thus allowing for the easy implementation of parallel computing [65].

The control vector parametrization (CVP) method is known as a sequential approach [98, 99]. Its basic idea is to discretize control variables over time intervals and obtain the states by solving DAEs by an integration scheme. A quasi-sequential approach was proposed by Hong et al. [49] in which only control variables are treated as decision variables and collocation on finite elements is used for discretization, leading to a two-stage solution strategy. An adaptive collocation scheme for this approach was introduced by Bartl et al. [11].

Recently, a combined multiple-shooting and collocation (CMSC) approach was proposed to exploit the advantages of both collocation and multiple-shooting [92]. In this approach, the discretization of model equations is based on the collocation method for individual time intervals, whereas optimality and state continuity are ensured in the resulting NLP in the context of the multiple-shooting method. It has been demonstrated that this approach is highly efficient for solving dynamic optimization problems [92, 60].

In this study, the advantages of the CMSC approach are further exploited by introducing an analytical Hessian (AH) and implementing parallel computing. It is known that the use of exact second-order sensitivities to provide an AH is beneficial for the solution of the resulting NLP problem. The results obtained using CVP indicate that exact second-order sensitivities, obtained by deriving and solving a differential equation system over the whole time horizon, can enhance robustness and convergence rate [97, 9, 10]. An extension of this method was made in the framework of collocation

on finite elements and considering piecewise constant controls [13, 12]. Moreover, there is wide interest in automatic sensitivity computation [67, 78]. However, the computation of an AH is known to be highly expensive, which might counterbalance the computational advantages. Therefore, a method to examine the profitability of using an AH instead of a numerical approximation is highly desired.

Another way to enhance the efficiency of solving dynamic optimization problems is to decompose the resulting NLP problem and implement a parallel-computing strategy. The single-shooting method was implemented using parallel computation of first-order sensitivities [45]. Parallel computing was also implemented in the context of multiple-shooting involving DAE integration and sensitivity computation [65]. In the context of collocation on finite elements, parallel computing was also realized by a Schur-complement decomposition within an interior-point algorithm [104]. This decomposition method introduces coupling variables to ensure the continuity of state trajectories, leading to high computational costs. This approach was further improved by Kang et al. [53] by utilizing the block structure of the resulting NLP. In the work of Zavala et al. [109], the optimizer was also adapted to the Schur-complement method to provide parallel computing for solving parameter estimation problems. More recently, Washington and Swartz [103] implemented a parallel-computing scheme in the context of multiple-shooting, but state trajectories in each time interval were computed using a numerical integrator [48]. In addition, the CMSC method was also implemented using parallel computing in an open-source environment, but with numerically approximated second-order derivatives [8].

To numerically solve a dynamic optimization problem, the model equations need to be discretized over a time horizon. Having chosen a discretization method, one needs to decide the number of time intervals in the time horizon under consideration. However, there have been no comprehensive rules for this purpose. Therefore, it is imperative to enhance the computational efficiency for the solution of such problems. In principle, the decision is made to achieve a compromise between the numerical accuracy of the discretization and the computation load for solving the discretized optimization problem. A larger number of time intervals will lead to, on the one hand, a shorter length of the time intervals and thus a lower discretization error. On the other hand, the dimension of the resulting NLP problem will be higher, thus demanding a greater computational expense. In contrast, if the problem is discretized with a small number of time intervals, the resulting state trajectory will be highly inaccurate, which may cause constraint violations. Therefore, it is desired to choose a minimum number of time intervals which can guarantee a user-specified error tolerance. The current situation is that the number of time intervals is usually chosen intuitively or empirically in practice.

There have been many studies on error estimation in solving ODEs and DAEs. The early work of Ahmed and Wright [1] considered a numerical solution of linear ODEs by means of collocation. In further studies, Wright et al. [106] introduced a subdivision-criterion-function for an adaptive mesh selection for boundary value problems (BVPs). Ascher and Petzold [5] discussed the solution of DAEs using projected implicit Runge-Kutta methods for linear problems in terms of stability, oscillation properties, and

accuracy. In Auzinger et al. [6] an error estimate based on the ideas of Zadunaisky [108] and Stetter [91] for the global error of collocation methods was presented. Furthermore, the authors discussed a mesh selection strategy aiming at equidistribution for an approximation of the numerical solution of singular BVPs computed by a collocation approach [7]. Guo and Wang [43] proposed two algorithms with the accuracy of the Legendre-Gauss collocation method. Their later work [102] described the solution of linear ODEs based on the Legendre-Gauss-Radau interpolation and adaptation of this approach to a multistep version. Moreover, Koch [59] presented a posteriori error estimate of singular BVPs and proposed a representation of the numerical error. In addition, Wright [105] investigated various adaptive methods for piecewise polynomial collocation of ODEs and compared two techniques, namely interval subdivision and mesh redistribution. Moreover, a local error estimation of a class of two-step Runge-Kutta methods for the numerical solution of ODEs was investigated [28].

Many previous studies have also been made to estimate the numerical error in solving the discretized optimization problem. A large portion of the work was in the context of collocation on finite elements (time intervals). Cuthrell and Biegler [26] developed a strategy in which each element length is adjusted by the NLP solver to detect the discontinuities in control profiles using a slightly modified method of de Boor [32, 30, 31]. Logsdon and Biegler [69] explored a discretization and NLP formulation by considering stability and error properties of implicit Runge-Kutta methods for DAEs to obtain appropriate error constraints. Furthermore, Vasantharajan and Biegler [96] introduced a direct error enforcement in the NLP solution, where the authors remarked that simultaneous handling of element placing and nonlinear error constraints can be very difficult. In the work of Seferlis and Hrymak [89] the accuracy was improved by adaptively placing the breakpoints between elements so that the approximation error can be equally distributed among the finite elements. Furthermore, Tanartkit and Biegler [93, 94] introduced a bilevel strategy in which the outer problem determines the number of required elements to fulfill a defined tolerance of state profiles and the inner problem solves the dynamic optimization problem. This strategy was further improved and investigated by Biegler et al. [17].

Huntington and Rao [51] investigated the numerical accuracy by modifying the number of elements and collocation points. Recently, a method to distribute the mesh points using various density functions was introduced by Zhao and Tsiotras [110]. In addition, Darby et al. [29] investigated an *hp*-adaptive pseudospectral method for the numerical solution of dynamic optimization problems. The approach consists of iteratively determining a number of required finite elements, the length of each element, and a polynomial degree to obtain the solution for a user-defined tolerance. Unfortunately, this approach requires iterative solution of the NLP, which takes additional computational burdens; therefore, it will be difficult to use this approach for online applications, especially for large-scale optimization problems.

In addition, the error estimation in the framework of control vector parametrization (CVP) has also been studied. Binder et al. [18] proposed a method in which a hierarchy of successively refined finite dimensional optimization problems are solved. The adaptation is built on a multiscale setting involving wavelets. Schlegel et al.

[88] presented an extension and modification of the work by Binder et al. [18]. An adaptive CVP method was also introduced in the study of Mehrpouya et al. [76], for determining the control structure and the number of switching points using the homotopy continuation technique.

Moreover, in the context of the quasi-sequential approach proposed by Hong et al. [49], Bartl et al. [11] presented a technique for adaptive mesh selection during solving the NLP problem with a variable interval length. The movable elements are combined with the concept of co-simulations, which improves the tolerance of state profiles by consequently dividing a given time interval into multiple subintervals until the required state profile tolerance is guaranteed. Note that the original number of discrete intervals in the NLP formulation does not change. In addition, the authors [11] proposed an empirical method for error estimation based on first-order derivative analysis.

Furthermore, Paiva and Fontes [79] recently proposed an adaptive time-mesh algorithm considering different levels of refinement. The information on the adjoint multipliers was used for interval refinement. The necessary condition of optimality in the form of the Maximum Principle of Pontryagin [15] was applied. Nevertheless, the permanent mesh refinement during solving the optimization problem increases the computation time and may negatively influence its online applications.

Above all, in digital signal processing, one of the most applied methods for determining suitable sampling frequency (i.e., the length of time intervals) is to apply the Nyquist-Shannon theorem [90], especially for linear systems. However, because the theorem gives no explicit relationship between the sampling frequency and the discretization accuracy, in practical applications, the sampling frequency is chosen much higher than what is suggested by the Nyquist-Shannon theorem. This leads to a conservative decision for the discretization, i.e., the number of time intervals used is much higher than necessary.

In the last past decades, the enormous interest in dynamic optimization was concentrated in the area of autonomous vehicle driving. In recent years, obstacle detection and avoidance as well as lane-keeping have got major attention in studies. MPC has proved as a promising control strategy with many desired features. Unlike pre-mission planning and offline design of nominal references, MPC provides adaptive control strategies based on the actual and spontaneous traffic situation, leading to a higher level of system autonomy and robustness. There are varieties of MPC approaches to autonomous vehicle steering, differing in used vehicle models, application purposes, optimization problem formulations, numerical solution techniques, applied software, etc.

A predictive control approach was reported by Keviczky et al. [56] where a nonlinear bicycle model with constant tire forces and a tire model considering the interaction between tractive force and cornering force in a combined braking and steering. An Euler method was employed to discretize the optimal control problem where the steering angle was used as the control variable. The problem was solved with a commercial software in different simulation scenarios.

Kim et al. [57] proposed an MPC-based path tracking algorithm including steer-

ing actuator dynamics. The problem was solved by a quadratic programming method. Various scenarios of simulation results concerning prediction and control horizon lengths, model order of the steering system and speed were reported.

Using an extended bicycle model with lagged tire force for better prediction accuracy, the work of Choi and Choi [24] addressed electronic stability control relying on MPC. Based on the nonlinear model, a reference trajectory is generated to maintain the vehicle yaw stability. To avoid the computational burden in satisfying state inequalities, the reference strategy is followed applying a linear MPC scheme which can easily be obtained in a closed form. Furthermore, Schildbach and Borelli [87] presents a new algorithm for detecting the safety of lane changes on highways and for computing safe lane change trajectories.

MPC exploits the model of the system dynamics to predict the future system evolution and to accordingly select the best control action with respect to a specified performance criterion. In addition, inequality constraints on control and state variables can be satisfied by MPC. Therefore, using MPC represents a promising control strategy for autonomous driving and outperforms control schemes based on the proportional-integral-differential (PID) regulator.

Most previous MPC applications have considered simple vehicle models, simple discretization techniques (i.e., the Euler method) and consider only simulative investigations [20, 14, 70]. In Cairano et al. [23] a switched MPC controller was proposed, where different local MPC controllers were used depending on the tire force conditions. Similar investigations were conducted by Gao et al. [39]. In Frasch et al. [36] multiple-shooting was applied for the discretization to solve the dynamic optimization problem based on a bicycle model. A linear MPC was analyzed in Katriniok et al. [55] based on linearized longitudinal and lateral dynamics and a highly precise GPS. A tube-based robust NMPC approach was suggested by Gao et al. [38] for lane-keeping and obstacle avoidance. The approach is based on a control law where nominal states and controls are obtained from a nominal NMPC and an offline calculated robust invariant set. Both simulation and experimentation results were given for obstacle avoidance. The robustness of the proposed nonlinear MPC algorithm was investigated by introducing tube-based constraints. In Yu et al. [107] a simulative investigation was conducted where a kinematic vehicle model was considered and the state estimation was conducted by an unscented Kalman filter.

Based on the literature review, the autonomous driving within the NMPC framework represents a great challenge and is considered as a hot topic in academia and in industry. According to the high requirements for autonomous vehicles, existing algorithms and methods should be significantly improved in order to guarantee the reliability of the overall concept.

CHAPTER 3

Problem formulation and solution approach

3.1 Problem statement

This work considers nonlinear dynamic optimization problems of the form

$$
\begin{aligned}
& \min_{u(t)} \left\{ J = M\left(x(t_f)\right) + \int_{t_0}^{t_f} L(x(t), z(t), u(t), t)dt \right\} \\
\text{subject to:} \quad & \dot{x}(t) = f(x(t), z(t), u(t), p, t),\ x(t_0) = x_0, \\
& 0 = g(x(t), z(t), u(t), p, t), \\
& h_p(x(t), z(t), u(t)) \leq 0,\ h_t(x(t_f)) = 0, \\
& x_{min} \leq x(t) \leq x_{max},\ z_{min} \leq z(t) \leq z_{max}, \\
& u_{min} \leq u(t) \leq u_{max},\ t \in [t_0, t_f],
\end{aligned}
\tag{3.1}
$$

where $x(t) \in \Re^{n_x}$ represents the vector of state variables, $z(t) \in \Re^{n_z}$ denotes the vector of algebraic variables and $u(t) \in \Re^{n_u}$ is the control vector. The time-independent parameters are defined as $p \in \Re^{n_p}$. The initial state $x(t_0)$ is supposed to be known and fixed. The final state $x(t_f)$ may be fixed or free. The time horizon is defined as $t \in [t_0, t_f]$, with initial time t_0 and terminal time t_f. The functions $f : \Re^{n_x} \times \Re^{n_z} \times \Re^{n_u} \times \Re^{n_p} \times [t_0, t_f] \rightarrow \Re^{n_x}$ and $g : \Re^{n_x} \times \Re^{n_z} \times \Re^{n_u} \times \Re^{n_p} \times [t_0, t_f] \rightarrow \Re^{n_z}$ in the model equations as well as the objective function J, consisting of the Mayer term $M : \Re^{n_x} \rightarrow \Re$ and the Lagrange term $L : \Re^{n_x} \times \Re^{n_z} \times \Re^{n_u} \times [t_0, t_f] \rightarrow \Re$, are assumed to be twice differentiable. The differential states, algebraic and control variables are bounded as indicated by the box-constraints. The functions $h_p : \Re^{n_x} \times \Re^{n_z} \times \Re^{n_u} \rightarrow \Re^{n_{pc}}$ and $h_t : \Re^{n_x} \rightarrow \Re^{n_t}$ denote path and terminal constraints, respectively.

3.2 Nonlinear model predictive control

Section 3.1 describes only the general form of the continuous dynamic optimization problem. The important question is how to use this problem formulation for controlling the real physical system. Recently, a variety of control strategies are available. Depending on the type of application and system configurations one can choose a proper design of the controller. For instance, classical controllers, such as PID or PI, are successfully applied for regulation and tracking of given setpoints. However, when

state and control constraints are taken into account, these controllers become unsuitable. In addition, in the presence of disturbances, classical controllers do not guarantee the stability of the system, especially in the presence of derivative components in the controller. To overcome these limitations, an advanced strategy called NMPC can be applied to guarantee the constraints imposed on the system. This control strategy was firstly introduced in 1963 [80] but has gained more interest after almost 20 years, after its successful application in the process industry [82, 40]. However, this control technique was applied for systems with slow dynamics and relatively large sampling times. Nevertheless, in the last two decades, a growing interest in the application of the NMPC on fast dynamic systems with high sampling rates has been raised, leading to an intensive research work for developing and implementing new and efficient methods.

The core of the NMPC controller is the solution of state and control constrained dynamic optimization problem on prediction horizons. The model equations are used to predict the possible future behavior of the system and decide the most suitable control signals for leading the system to a desired behavior. The constraints in the optimization problem are formulated considering physical limitations in the actuators and the states and control ranges of operation or imposed by environment (e.g., in autonomous driving). Practically, a general control scheme of the real plant can be described as a set of devices that are integrated together in order to enforce a desired plant behavior. A block diagram describing the general structure of a control system is shown Figure 3.1, where the gray dashed box indicates the physical parts of the system.

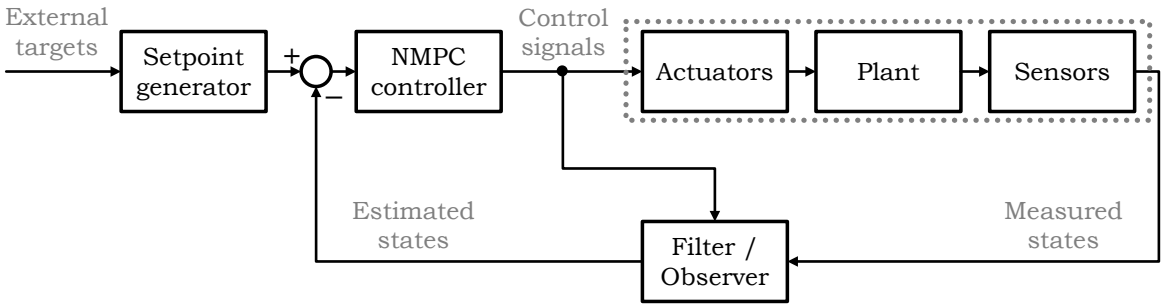

Figure 3.1: General NMPC framework block diagramm

The main components of the NMPC controller can be implemented on an external computer or embedded system. The controller computes a suitable control strategy and applies it to actuators, which realize it on a physical system, which is called the plant. Using appropriate sensors, the system states can be measured. However, it is not always possible to measure all the system states directly. To compute them, special computational units, like filter or observer, can be applied. Afterwards, the estimated states are compared with given or computed reference state trajectories by the setpoint generator. This control strategy is known as feedback and allows the controller to give

a proactive response to uncertainties, which generally can arise from a combined effect of external disturbances, measurement noise and plant-model mismatch. A scheme of the NMPC technique is shown in Figure 3.2.

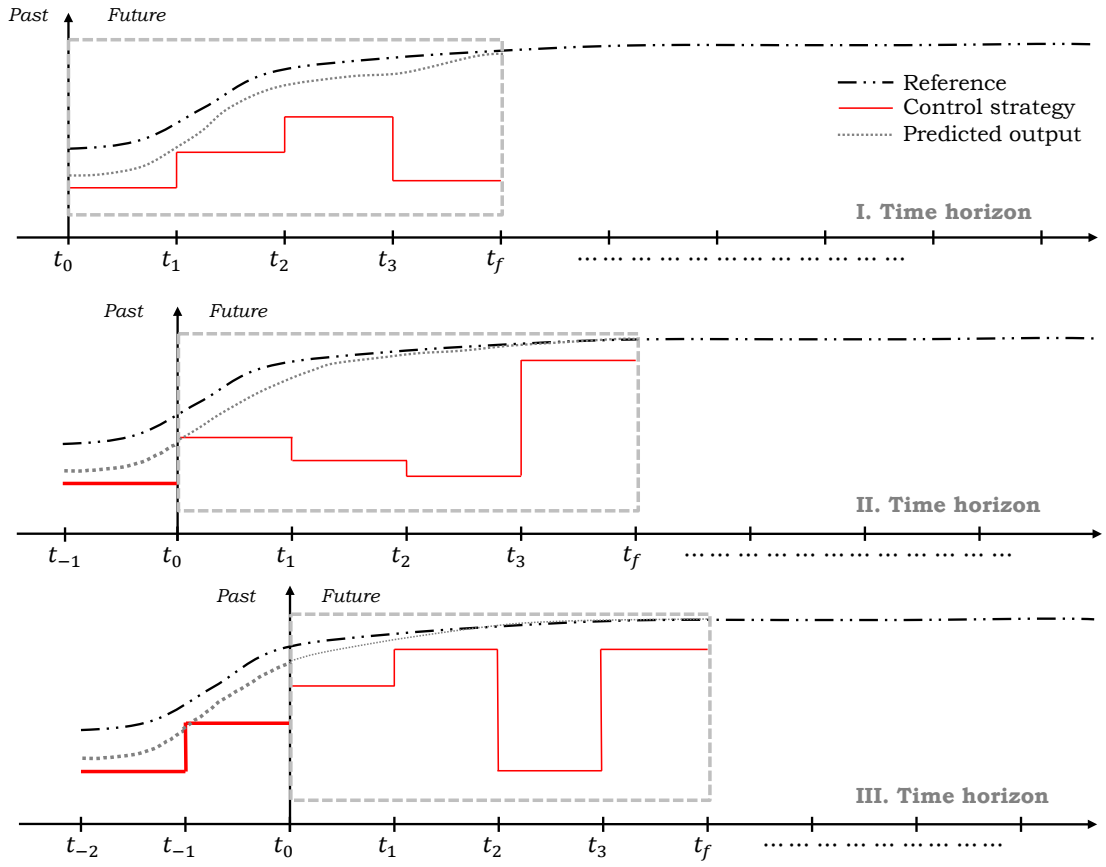

Figure 3.2: NMPC working principle

In general, NMPC works by repeatedly solving (i.e., every sampling time) a finite dimensional NLP problem considering the current state as the initial state value x_0 of the problem. The prediction horizon is always relative to the current state and recedes away from the initial point as the dynamic moves forward. The predicted states and controls are continually updated to take into account of the most recent target and measured data. Since data measurement are taken into account in every feedback loop, predictive control has the advantage of feedback controller, achieving stability and good control performance.

For each prediction, the solution of the NMPC problem is a sequence of optimal control inputs, which minimizes an objective function and leads the dynamic towards the desired behavior. At time $t = t_0$ the NMPC problem is solved and the actuator applies only the first element of the optimal control input computed for the time $t = [t_0, t_1]$. After that, the prediction horizon is receded, the current state is measured (or estimated) using sensors (or observers), and the NMPC problem is solved again

using the measured (or estimated) state as initial state. This process is repeated iteratively while the system is in operation.

The main idea of such control scheme is very simple, but it is important to analyze the key components, which are crucial for obtaining an efficient and reliable control strategy. First, the objective function is a real-valued performance criteria for the best control signal which leads to the desired behavior. Since the optimization is performed online, the complexity of the objective function should be set according to the application and preferably defined as simple as possible. Typically, a quadratic objective function is used because it provides a well-conditioned optimization problem with smooth behavior. Second, there is no standard rule for selecting the prediction horizon N, but it is advisable to use such value of N and interval length which allow a prediction beyond the key dynamic of the system (e.g., transient part of dynamics) or at least big enough to consider the possible critical constraints that could exist in such way that the controller can manage them. Therefore, it should be large enough for obtaining a good performance but simultaneously it should consider the corresponding complexity in solving the optimization problem. Third, the core part of the predictive controller is the prediction and, thus, a model is required. Defining an appropriate model is a key point in the controller design. The prediction model should be descriptive to capture the most significant dynamics of the system and simple for solving the optimization problem. In practice, it is not beneficial to spend excessive effort improving model accuracy, which can result in high order models but may have little impact on the systems behavior. Moreover, the feedback property of the NMPC scheme generally corrects small modeling errors. Thus, the dynamic model should be as simple as possible to reduce complexity in the solution of the NMPC problem, but must consider the inherent characteristics of the system to achieve a good performance.

The necessity of the application of the NMPC controller can be explained in several ways. In the presence of unknown disturbances or unmodeled system dynamics, it is necessary to use a feedback-like controller which solves the optimization problem repeatedly in real-time. Suppose the optimal control problem with prediction horizon N has been solved. Then, it would be sufficient to apply the whole optimal control sequence computed by the optimizer and not only the first element, as shown in Figure 3.2. However, this is possible only if the process model is exact, there are no external disturbances, and if the control input is applied instantaneously to the process. Unfortunately, in the real applications, these conditions are never satisfied because of the model-plant mismatch due to the complexity of the system. Likewise, unknown disturbances are likely to occur, as well as noise in the measurements, which means that the initial state cannot be correctly determined. Moreover, actuators need some time to react (i.e., so-called dead time) and thus, there exist deviations between the optimal and the applied control. Nevertheless, predictive control has shown to be very efficient for controlling very complex systems and has outperformed typical control strategies that have been used for many years in the industry. Compared to traditional control techniques such as conventional controllers, predictive controllers offer the following set of advantages. It is possible to specify the desired limitations in the process (considering control and state constraints), as well as the desired behav-

ior through the objective function employed in the formulation of the optimal control problem. It can handle large-scale control problems of dynamic systems with multiple inputs and outputs. The propagation of measurement noise through the control signal is reduced. Disturbance compensation is also achieved due to the feedback feature given by the receding horizon technique. In addition, the initialization of successive optimization problems is simple, because the structure of the NLP problem does not change between computations. Moreover, NMPC is based on a well-established theoretical background. Different studies about stability and robustness support the use of this control technique. The extensive studies on mathematical optimization tools used to solve optimal control problems have increased in the last decades, giving, as a result, different robust solvers that can be used in general NMPC applications [42].

Besides these positive characteristics and advantages, the application of the NMPC can become very challenging in systems where time is a crucial factor, such as autonomous driving. In particular, MPC becomes challenging due to the following reasons. If the model equations are nonlinear and the optimization problem is nonconvex, the following potential problems could arise. It might not be possible to obtain a global optimal solution. Instead, many sub-optimal local solutions can be found. This turns the problem very difficult to solve, which implies more computational effort for obtaining the true optimal solution. Systems with fast or even slow dynamics require the solution of the optimization problem in real-time, i.e., within the single time interval (in the range of milliseconds). Thus, it is necessary to compute the solution of the problem as fast as possible in order to obtain the optimal control input. Therefore, the main bottleneck when using NMPC for controlling dynamic systems with high sampling rates is the computational burden for obtaining the optimal solution at each sampling time. Dealing with this problem is the one of several main focuses of the work presented in this thesis. Since the problem (3.1) is continuous, it is difficult to directly solve this problem within NMPC framework. Therefore, this problem is converted into an NLP problem with help of CMSC method, which is discussed in the next subsection.

3.3 Combined multiple-shooting with collocation

3.3.1 Transformation to an NLP problem

To transform the dynamic optimization problem (3.1) into a finite-dimensional NLP problem, the combined multiple-shooting with collocation method [61, 60, 92] is applied. The time horizon $[t_0, t_f]$ is divided into N intervals, and in each interval, the state variables are treated using the multiple-shooting method. It should be noted that, in this work, equidistant time intervals are considered. Over each time interval the state variables are parametrized as initial values; that is, $\mathbf{x}_p = [\mathbf{x}_{p,0}, \mathbf{x}_{p,1}, ..., \mathbf{x}_{p,N}]$. Due to its simplicity of implementation and practical applicability, the piecewise constant form for control variable parameterization $\mathbf{u} = [\mathbf{u}_0, \mathbf{u}_1, ..., \mathbf{u}_{N-1}]$ is chosen. Considering $\mathbf{x}_p \in \Re^{n_x \cdot (N+1)}$ and $\mathbf{u} \in \Re^{n_u \cdot N}$ as decision variables, problem (3.1) is now transformed

to the following NLP problem

$$
\begin{aligned}
& \min_{\mathbf{x}_p, \mathbf{u}} \left\{ M(\mathbf{x}_{p,N}) + \sum_{i=0}^{N-1} L_i(\mathbf{x}_{p,i}, \mathbf{u}_i) \right\} \\
\text{subject to:} \quad & \mathbf{x}_{p,0} - \mathbf{x}_0 = 0, \\
& \mathbf{x}_{p,i} = \mathbf{P} \cdot \hat{\mathbf{x}}_i \left(\mathbf{x}_{p,i-1}, \mathbf{u}_{i-1} \right), \ i = 1, \ldots, N, \\
& H_p(\hat{\mathbf{x}}_i \left(\mathbf{x}_{p,i-1}, \mathbf{u}_{i-1} \right), \mathbf{u}_{i-1}) \leq 0, \\
& H_t(\mathbf{x}_{p,N}) = 0, \\
& \mathbf{x}_{min} \leq \mathbf{x}_p \leq \mathbf{x}_{max}, \\
& \mathbf{u}_{min} \leq \mathbf{u} \leq \mathbf{u}_{max},
\end{aligned}
\tag{3.2}
$$

where $\mathbf{x}_0$ denotes the initial values of the state variables. The equality constraints $\mathbf{x}_{p,i} = \mathbf{P} \cdot \hat{\mathbf{x}}_i \left(\mathbf{x}_{p,i-1}, \mathbf{u}_{i-1} \right)$ are introduced to ensure continuity of the differential states between neighboring intervals. However, the state vector $\hat{\mathbf{x}}_i$ consists of all state and algebraic variables at collocation points. Therefore, the projection matrix $\mathbf{P}$ selects only the differential states at last collocation point from the vector $\hat{\mathbf{x}}_i \left(\mathbf{x}_{p,i-1}, \mathbf{u}_{i-1} \right)$, since this state vector includes state and algebraic variables at all collocation points in each interval and has dependency only on the parameterized states $\mathbf{x}_{p,i-1} \in \Re^{n_x}$ and controls $\mathbf{u}_{i-1} \in \Re^{n_u}$. Note, that the model equations are not directly involved in the NLP problem formulation (3.2), because in each time interval, they as well as the required (at least) first-order sensitivities are solved at the collocation points in the simulation layer.

Based on the multiple-shooting discretization, in contrast to the pure collocation, the model equations are not considered directly in the NLP problem (3.2). Therefore, the problem dimension is highly reduced and time intervals are independent, which allows parallel computing. On the other hand the model equations are discretized using collocation method, which guarantees high approximation accuracy. Thus, the combined approach gathers together the advantages of both methods, which plays a crucial role for further practical engineering applications.

3.3.2 Computation of first-order sensitivities

In the CMSC method, for each time interval, a collocation method is employed. The state and algebraic variables are approximated inside each time interval by a linear combination of the Lagrange polynomials using the Radau scheme with N_c collocation points. For the sake of brevity, the index for time intervals is dropped here and therefore the decision variables for the individual time interval are denoted as X^p and V for parameterized state and control variables, respectively. After this, the state and algebraic variables X^c at all collocation points have implicit dependency on the decision variables X^p and V. Therefore, the discretized dynamic model equations for each time interval can be written in the compact form

$$
\left[G(X^c(X^p, V), X^p, V) \right]_{N_{xz} \times 1} = \mathbf{0}, \tag{3.3}
$$

where $N_{xz} = (n_x + n_z) \cdot N_c$ and $X^c \in \Re^{N_{xz}}$. To obtain the first-order sensitivities for individual time intervals, two differentiation operators $\frac{\partial}{\partial X^p}$ and $\frac{\partial}{\partial V}$ are applied to equation (3.3), which yields

$$\left[\frac{\partial G}{\partial X^c}\right]_{N_{xz}\times N_{xz}} \cdot \left[\frac{\partial X^c}{\partial X^p}\right]_{N_{xz}\times n_x} = -\left[\frac{\partial G}{\partial X^p}\right]_{N_{xz}\times n_x}, \tag{3.4}$$

$$\left[\frac{\partial G}{\partial X^c}\right]_{N_{xz}\times N_{xz}} \cdot \left[\frac{\partial X^c}{\partial V}\right]_{N_{xz}\times n_u} = -\left[\frac{\partial G}{\partial V}\right]_{N_{xz}\times n_u}. \tag{3.5}$$

The construction of the linear equation systems for the first-order sensitivities can be done in a straightforward manner using automatic differentiation. Because of the CMSC method, equations (3.4) and (3.5) have a sparse structure and thus can be solved for $\frac{\partial X^c}{\partial X^p}$ and $\frac{\partial X^c}{\partial V}$ by a linear sparse algebra algorithm. Note that sensitivity equations can be combined together since the left-hand-side matrix is the same.

3.3.3 Computation of second-order sensitivities

In this subsection, the formulas for computing second-order sensitivities for the CMSC method are derived. In general, there are two types of second-order sensitivities: derivatives of functions with respect to all variables (both states and controls) in the context of simultaneous approaches and derivatives of state variables with respect to controls in the context of (quasi-) sequential approaches. The CMSC method is a kind of simultaneous approach, and thus, we need to compute the second-order derivatives of states with respect to controls and parameterized state variables. In comparison with the approaches of previous studies [97, 12], where sensitivities have to be transferred from one time interval to the next (so-called global sensitivities) in the context of single shooting, we employ the advantage of the CMSC method that the computations of the second-order sensitivities in different time intervals are independent. The second-order sensitivities of the state continuity condition (i.e., the equality constraints of problem (3.2) are obtained simply as those on the last collocation point of the time intervals.

In each time interval, the first-order sensitivities $\left[\frac{\partial X^c}{\partial X^p}\right]$ and $\left[\frac{\partial X^c}{\partial V}\right]^k$ solved from equations (3.4) and (3.5) have implicit dependency on the decision variables of only this interval. To clearly explain the derivation, the first-order sensitivity (matrix) equation (3.4) is reformulated as a vector of linear equations. This is made by multiplying the matrix $\frac{\partial G}{\partial X^c}$ with the vector $\left[\frac{\partial X^c}{\partial X^p}\right]^k$ and the result is added with the vector $\left[\frac{\partial G}{\partial X^p}\right]^k$, where $k = 1, ..., n_x$ is the column number. This procedure is also done for equation (3.5), where $k = 1, ..., n_u$. As a result, equations (3.4) and (3.5) can be rewritten in the compact form

$$\left[\Phi(X^c(X^p, V), X^p, V, \frac{\partial X^c}{\partial X^p}(X^p, V))\right]_{N_S^X \times 1} = \mathbf{0}, \tag{3.6}$$

$$\left[\Psi(X^c(X^p, V), X^p, V, \frac{\partial X^c}{\partial V}(X^p, V))\right]_{N_S^U \times 1} = \mathbf{0}, \tag{3.7}$$

where $N_S^X = N_{xz} \cdot n_x$ and $N_S^U = N_{xz} \cdot n_u$. Applying both differentiation operators $\frac{\partial}{\partial X^p}$ and $\frac{\partial}{\partial V}$ to equations (3.6) and (3.7), the equations for second-order sensitivities can be expressed as follows

$$\left[\frac{\partial \Phi}{\partial(\frac{\partial X^c}{\partial X^p})}\right]_{N_S^X \times N_S^X} \left[\frac{\partial^2 X^c}{\partial X^{p,2}}\right]_{N_S^X \times n_x} = -\left[\frac{\partial \Phi}{\partial X^c}\right]_{N_S^X \times N_{xz}} \cdot \frac{\partial X^c}{\partial X^p} - \left[\frac{\partial \Phi}{\partial X^p}\right]_{N_S^X \times n_x}, \quad (3.8)$$

$$\left[\frac{\partial \Phi}{\partial(\frac{\partial X^c}{\partial X^p})}\right]_{N_S^X \times N_S^X} \left[\frac{\partial^2 X^c}{\partial X^p \partial V}\right]_{N_S^X \times n_u} = -\left[\frac{\partial \Phi}{\partial X^c}\right]_{N_S^X \times N_{xz}} \cdot \frac{\partial X^c}{\partial V} - \left[\frac{\partial \Phi}{\partial V}\right]_{N_S^X \times n_u}, \quad (3.9)$$

$$\left[\frac{\partial \Psi}{\partial(\frac{\partial X^c}{\partial V})}\right]_{N_S^U \times N_S^U} \left[\frac{\partial^2 X^c}{\partial V \partial X^p}\right]_{N_S^U \times n_x} = -\left[\frac{\partial \Psi}{\partial X^c}\right]_{N_S^U \times N_{xz}} \cdot \frac{\partial X^c}{\partial X^p} - \left[\frac{\partial \Psi}{\partial X^p}\right]_{N_S^U \times n_x}, \quad (3.10)$$

$$\left[\frac{\partial \Psi}{\partial(\frac{\partial X^c}{\partial V})}\right]_{N_S^U \times N_S^U} \left[\frac{\partial^2 X^c}{\partial V^2}\right]_{N_S^U \times n_u} = -\left[\frac{\partial \Psi}{\partial X^c}\right]_{N_S^U \times N_{xz}} \cdot \frac{\partial X^c}{\partial V} - \left[\frac{\partial \Psi}{\partial V}\right]_{N_S^U \times n_u}. \quad (3.11)$$

It can be seen that the matrix dimensions grow rapidly as the numbers of state and control variables increases, as indicated in equations (3.8) - (3.11). In these equations, the partial derivative matrices can be generated by applying automatic differentiation. Note that equation (3.10) can be eliminated, because the sensitivities $\frac{\partial^2 X^c}{\partial X^p \partial V}$ and $\frac{\partial^2 X^c}{\partial V \partial X^p}$ are exactly the same. Then, the second-order derivatives can be calculated by solving the above linear matrix equations by a sparse linear solver, and the results provide an AH for the CMSC method.

However, there are two major obstacles to using an AH, especially for solving large-scale problems. First, in general, the generation of an AH is complicated. Nevertheless, it is quite straightforward using our approach as described above. Second, the computation time for each NLP iteration of an AH can be expensive; specifically, the computation time for each NLP iteration using an AH is higher than that using a BFGS approximation. Hence, an improvement in computation time for solving an NLP can be achieved only if the difference between the numbers of NLP iterations required when using an AH approach and a BFGS formula is sufficiently large. Therefore, in the next chapter, a method based on a priori simulation is proposed to examine whether an AH is beneficial for solving dynamic optimization problems.

3.3.4 Handling of path and terminal constraints

Similarly to equation (3.3), the discretized form of the path constraints can be written for the individual time intervals as follows

$$[H_p(X^c(X^p, V), V)]_{N_{pc} \times 1} \leq \mathbf{0}, \quad (3.12)$$

where $N_{pc} = n_{pc} \cdot N_c$. For the sake of brevity, the index for time intervals is dropped here. To obtain the first-order sensitivities for individual time intervals, the same

differentiation operators $\frac{\partial}{\partial X^p}$ and $\frac{\partial}{\partial V}$ are applied to equation (3.12), which yield

$$\left[\frac{\partial H_p}{\partial X^p}\right]_{N_{pc}\times n_x} = \left[\frac{\partial H_p}{\partial X^c}\right]_{N_{pc}\times N_{xz}} \cdot \left[\frac{\partial X^c}{\partial X^p}\right]_{N_{xz}\times n_x}, \tag{3.13}$$

$$\left[\frac{\partial H_p}{\partial V}\right]_{N_{pc}\times n_u} = \left[\frac{\partial H_p}{\partial X^c}\right]_{N_{pc}\times N_{xz}} \cdot \left[\frac{\partial X^c}{\partial V}\right]_{N_{xz}\times n_u} + \left[\frac{\partial H_p}{\partial V}\right]_{N_{pc}\times n_u}, \tag{3.14}$$

where numerical values of the first-order sensitivities with respect to parameterized state and control variables are available from the solution of the linear equations systems (3.4) and (3.5) and the structure of the derivative matrices $\frac{\partial H_p}{\partial X^c}$ and $\frac{\partial H_p}{\partial V}$ should be precomputed and further evaluated using numerical values of state and algebraic variables at collocation points. If the AH is used, the second-order sensitivities can be computed in the same way as written in the previous section.

The discretized form of the terminal constraints can be directly constructed as follows, because of the additional state parameterization at final time

$$H_t(X^p) = 0, \tag{3.15}$$

where $H_t \in \Re^{n_t}$. The first and second-order sensitivities can be directly computed using numerical values of state variables from the last time interval obtained by the optimizer.

3.4 Primal-dual interior-point method

Without loss of generality, the optimization problem defined in (3.2) can be rewritten in more compact form. The standard notation of an optimization problem is given by

$$\begin{aligned} & \min_{\omega} f(\omega) \\ \text{subject to:} \quad & g_i(\omega) = 0,\ i = 1,\ldots,p\,, \\ & h_j(\omega) \leq 0,\ j = 1,\ldots,m\,, \end{aligned} \tag{3.16}$$

where $\omega \in \Re^d$ is the vector of optimization variables, $f : \Re^d \to \Re$ is the objective function to be minimized, $g(\omega) = [g_1(\omega),\ldots,g_p(\omega)]^T$ is the set of equality constraints, and $h(\omega) = [h_1(\omega),\ldots,h_m(\omega)]^T$ is the set of inequality constraints. The goal consists in finding an optimal value ω^* that minimizes an objective function $f(\omega)$, while satisfying the constraints. Our special interest belongs to special class of NLP problems, namely convex optimization. The NLP problems can be classified as convex if $f(\omega)$ and $h_j(\omega)$ are convex functions of ω and the equality constraints $g_i(\omega)$ are affine. The most important property in convex problems is that if there exists a local minimum, it is also a global minimum. Many applications in different fields can be described as convex problems, making convex optimization very attractive.

Nowadays, interior-point methods (IPMs) are the most widely used numerical methods for solving (3.16) optimization problem. The very first version was proposed

by Karmarkar [54] and currently many researches have been focused on IPM and its applications.

Starting from the initial guess for the optimal solution, IPM iteratively generate steps towards the minimum point inside the feasible region described by constraints. The most popular IPM are primal-barrier, primal-dual and its derivative predictor-corrector. This thesis is oriented only on the primal-dual interior-point method, which is more effective than primal-barrier method, especially when high accuracy is required [21]. An efficient version of IPM has been developed in FORTRAN around 1992 [75]. Since then the source code has been significantly improved and was made available in many programming languages, such as C++ or Python. The IPOPT solver [100] is programmed to work fine with large-scale constrained optimization problems, which implies solution of large-scale sparse matrices, which can be exploited by advanced linear space algebra solvers. The standard version on IPOPT includes free HSL-MA27 [81] linear solver, which is based on direct method based on a sparse Gaussian elimination approach. Furthermore, the source-code can be compiled with several other linear algebra solvers.

The main idea in primal-dual interior-point method is the solution of the Karush-Kuhn-Tucker (KKT) optimality conditions by introducing a slack variable in the inequality constraints (3.16) to transform them into equality constraints so that the Newton method can be applied to solve the optimization problem. In general, the KKT optimality conditions are given as follows

$$\nabla f(\omega) + \sum_{i=1}^{p} \lambda_i \cdot \nabla g_i(\omega) + \sum_{j=1}^{m} \mu_j \cdot \nabla h_j(\omega) = 0\,, \tag{3.17}$$

$$g_i(\omega) = 0\,, \tag{3.18}$$

$$h_j(\omega) + s_j^s = 0\,, \tag{3.19}$$

$$\mu_j \cdot s_j^s = 0\,, \tag{3.20}$$

$$\mu_j \geq 0\,, \tag{3.21}$$

where $j = 1, \ldots, m$, $i = 1, \ldots, p$, $s^s \in \Re^m$ are the slack variables for the inequality constraints, $\lambda \in \Re^p$ is the vector of dual variables for the equality constraints and $\mu \in \Re^m$ the vector of dual variables for the inequality constraints. However, in order to use the Newton method for the KKT system above, it is necessary to provide the following modifications. The complementary condition (3.20) is relaxed by making the equation equal to γ (which allows a modification of the right hand side at each iteration), and the variables $\mu \geq 0$ and $s^s \geq 0$. A search direction is obtained by linearizing the modified system and solving the following linear system

$$\begin{bmatrix} H(\omega,\lambda,\mu) & \nabla g(\omega)^T & \nabla h(\omega)^T & 0 \\ g(\omega) & 0 & 0 & 0 \\ \nabla h(\omega) & 0 & 0 & I \\ 0 & 0 & S & Y \end{bmatrix} \begin{bmatrix} \Delta\omega \\ \Delta\lambda \\ \Delta\mu \\ \Delta s \end{bmatrix} = - \begin{bmatrix} R_\omega \\ R_\lambda \\ R_\mu \\ R_s \end{bmatrix}, \tag{3.22}$$

where $S = diag(s_1^s, \ldots, s_m^s)$, $Y = diag(\mu_1, \ldots, \mu_m)$, $H(\omega, \lambda, \mu)$ is the Hessian of the Lagrangian function, and residuals in the right-hand side R_ω, R_λ, R_μ, R_s defined by left-hand sides of equations (3.17) - (3.20). In addition, the Hessian matrix of Lagrangian function is technically defined as

$$H(\omega, \lambda, \mu) = \nabla^2 f(\omega) + \sum_{i=1}^{p} \lambda_i \nabla^2 g_i(\omega) + \sum_{j=1}^{m} \mu_j \nabla^2 h_j(\omega). \tag{3.23}$$

Unfortunately, the exact Hessian matrix is in most cases hard to calculate. Therefore, in general the L-BFGS [77] (where "L" stands for "Limited-memory") method for the Hessian approximation is applied. In addition, the residual vector in the left-hand side of (3.22) is defined as follows

$$\begin{bmatrix} R_\omega \\ R_\lambda \\ R_\mu \\ R_s \end{bmatrix} = \begin{bmatrix} \nabla f(x) + \nabla g(\omega)^T \cdot \lambda + \nabla h(\omega)^T \cdot \mu \\ g(\omega) \\ h(\omega) + s^s \\ SY - \gamma \mathbf{1} \end{bmatrix}, \tag{3.24}$$

where $\mathbf{1} = [1, 1, \ldots, 1]^T$.

Furthermore, this method uses standard iteration scheme defined as follows

$$(\omega^{k+1}, \lambda^{k+1}, \mu^{k+1}, s^{s,k+1}) = (\omega^k, \lambda^k, \mu^k, s^{s,k}) + \alpha^k (\Delta\omega^k, \Delta\lambda^k, \Delta\mu^k, \Delta s^{s,k}), \tag{3.25}$$

where α^k can be computed using, e.g., Armijo-rule [77]. The algorithm stops when the norm of the residual vector is less than a given small tolerance ϵ. In this thesis, the primal-dual interior-point algorithm is employed for the solution of resulted NLP problem within IPOPT software [100]. The IPOPT solver modifies the objective function by adding penalization of slack variables, so that

$$\hat{f}(\omega) = f(\omega) + \Theta \cdot \sum_{j=1}^{m} \ln(s_j^s), \tag{3.26}$$

where Θ serves for controlling of slack variables and therefore at the end of convergence $\Theta \to 0$. The complementarity condition (3.20) are relaxed by γ, as well as Θ. But this condition will be satisfied after the algorithm converges with $\gamma \to 0$.

In order to use IPOPT software to obtain the numerical solution of the NLP, it is required to provide important information, such as the number of optimization variables, number of equalities and inequalities constraints, the definition of the cost function, constraints, and bounds on the decision variables and box-constraints for constraints. To start the search process, it is also necessary to compute at least the first-order derivatives, i.e., the gradient of the objective function, Jacobian matrix of equality and inequality constraints. As it was mentioned before, to compute the Hessian matrix at each iteration, it is possible to employ the IPOPT's approximation using L-BFGS method. However, if the analytical Hessian is available, it can be easily

supplied directly to the algorithm. Further information about primal-dual interior-point method can be found in [95, 100].

CHAPTER 4

Correlation analysis of control variables

4.1 Idea of the correlation analysis

The basic idea of correlation analysis comes from parameter estimation problems where a model under consideration will be non-identifiable if there are strong correlations between model parameters [25, 74, 66]. In such situations, the NLP solver will experience convergence difficulties to an optimal point, because the resulting sensitivity matrix tends to be singular or ill-conditioned. Based on these considerations, a novel approach to determine the use of an analytical or approximate Hessian by analyzing control-variable correlations of the dynamic optimization problem is proposed [61]. This is done through a priori simulation using proper input control profiles. As in parameter estimation, the convergence to an optimal point will be slow if there are strong correlations between control variables. Such correlations can arise from improper modeling of the system. Now, we investigate correlations of control variables in a dynamic systems described by

$$\dot{x}(t) = f\left(x(t), u(t), t\right), x(t_0) = x_0, \tag{4.1}$$

where $x(t) \in \Re^{n_x}$, $u(t) \in \Re^{n_u}$ are state and control vectors, respectively. The known initial state vector is given by $\mathbf{x}_0$. The decision variables are $\mathbf{u}(t)$ and have to be determined by the optimization method.

Mathematically, a correlation between u_i and u_j (U-U correlation) means that the two control variables have a functional relationship. The physical meaning of such a correlation is that the effects of u_i and u_j on the system will be compensated. A straightforward way to identify U-U correlations in a system described by equation (4.1) is to analyze the state sensitivity matrix to the controls $\left(\frac{\partial \mathbf{x}}{\partial \mathbf{u}}\right)$. If there are linearly dependent columns in this matrix, then the corresponding controls are correlated.

Based on equation (4.1), the sensitivities of states to the control variables can be expressed as

$$\frac{d}{dt}\left(\frac{\partial \mathbf{x}}{\partial \mathbf{u}}\right) = \left(\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right) \cdot \left(\frac{\partial \mathbf{x}}{\partial \mathbf{u}}\right) + \left(\frac{\partial \mathbf{f}}{\partial \mathbf{u}}\right). \tag{4.2}$$

If the initial state of the system $\mathbf{x}_0$ is a steady-state, the initial condition of equation

(4.2) can be written as

$$\left(\left.\frac{\partial \mathbf{x}}{\partial \mathbf{u}}\right|_{t=t_0}\right) = \left(\left.\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right|_{t=t_0}\right)^{-1} \cdot \left(\left.\frac{\partial \mathbf{f}}{\partial \mathbf{u}}\right|_{t=t_0}\right). \tag{4.3}$$

The solution of (4.2) can be similarly obtained as in [66]. Based on this solution, $\left(\frac{\partial \mathbf{x}}{\partial \mathbf{u}}\right)$ has a linear (integral) relation with $\left(\frac{\partial \mathbf{f}}{\partial \mathbf{u}}\right)$ from t_0 to t. Consequently, the columns in $\left(\frac{\partial \mathbf{x}}{\partial \mathbf{u}}\right)$ will be linearly dependent, i.e., the control variables are correlated, if at any time the columns of $\left(\frac{\partial \mathbf{f}}{\partial \mathbf{u}}\right)$ are linearly dependent. As a result, U-U correlations can be identified by analyzing the linear dependence of the columns of the function sensitivity $\left(\frac{\partial \mathbf{f}}{\partial \mathbf{u}}\right)$ which is easy to achieve. If the following equality holds true

$$\left(\frac{\partial \mathbf{f}}{\partial u_i}\right) = \alpha_{i,j} \cdot \left(\frac{\partial \mathbf{f}}{\partial u_j}\right), \tag{4.4}$$

then there is a correlation between u_i and u_j. If the coefficient $\alpha_{i,j}$ is a constant, it is a *structural* correlation, that is, this correlation does not depend on control variables. If the coefficient $\alpha_{i,j}$ is a function of control variables, $\alpha_{i,j}(u_i, u_j)$, it is a *practical* correlation and can be resolved and remedied by selecting proper control profiles. In this way, the dependence of a U-U correlation of the controls can be identified. As a simple example, consider $\dot{x} = -(u_1 + u_2) \cdot x$. Then $\frac{\partial f}{\partial u_1} = \frac{\partial f}{\partial u_2}$; that is, u_1 and u_2 have a structural correlation. If we have $\dot{x} = -u_1 \cdot u_2 \cdot x$, then $\frac{\partial f}{\partial u_1} = \frac{u_2}{u_1} \cdot \frac{\partial f}{\partial u_2}$, which means that u_1 and u_2 are practically correlated. It should be noted that this correlation analysis is not limited to ordinary differential equations. In fact, it can be easily extended to handle DAE systems.

However, the method of correlation analysis presented above can only explicitly determine whether there is or is not a correlation in the system under consideration, that is, it provides a result that two control variables are either 0% or 100% correlated. Nevertheless, control variables in a system are usually correlated between 0% (weak correlation) and 100% (strong correlation). Therefore, an index is needed to quantify the degree of a correlation between two controls. Such an index can be readily obtained by calculating the angles of the corresponding columns in the matrix of $\left(\frac{\partial \mathbf{x}}{\partial \mathbf{u}}\right)$. If the control profiles are given, one can obtain this matrix through simulation. The construction of this matrix in straightforward. The resulting sensitivities from each time interval are vertically concatenated together.

Considering two columns in the matrix $\left(\frac{\partial \mathbf{x}}{\partial u}\right)$, $v = [v_1, \ldots, v_n]^T, w = [w_1, \ldots, w_n]^T$, the angle between the two columns can be calculated as

$$\Theta = \arccos\left(\frac{\sum_{i=1}^{n} v_i \cdot w_i}{\sqrt{\sum_{i=1}^{n} v_i^2 \cdot \sum_{i=1}^{n} w_i^2}}\right). \tag{4.5}$$

From the viewpoint of optimality conditions, a U-U correlation negatively affects the regularity conditions, that is, the Mangasarian-Fromovitz constraint qualification

[77] (MFCQ), when the angle describing the collinearity of two columns is close to 0°. Then, the Jacobian matrix of the equality constraints in the NLP tends to be ill-conditioned, and hence, the NLP solver will need more iterations to converge. It is clear that the presence of correlated control variables will also negatively affect the convergence rate when the BFGS method is used, that is, the approximated Hessian might be insufficient and thus causes slow convergence of the optimization algorithm. Therefore, an AH is desired to accelerate the convergence if there is a U-U correlation. Because an a priori simulation is needed to perform the correlation analysis, proper profiles of control inputs need to be selected. Here, the pseudorandom binary sequence (PRBS) is proposed as a stimulating signal. The PRBS has been widely used for system identification [68, 44], since it exhibits many preferred properties (e.g., persistently exciting, being white-noise-like, and having maximum power for a limited amplitude) [73, 85]. These properties are also advantageous for the purpose of identifying control-variable correlations, because the input signals should include a broad frequency band and both persistently and sufficiently excite the system within a limited time horizon.

4.2 Demonstration examples

The problem below was firstly introduced by Luus [71] and further analyzed by Balsa-Canto et al. [9]. The purpose of this optimization problem is to determine four control variables of the continuous stirred-tank reactor (CSTR) to maximize the economic benefit. The system dynamics is represented by several simultaneous chemical reactions. The dynamic optimization problem is formulated as [9]:

$$
\begin{aligned}
& \max_{u(t)} x_8(t_f) \\
\text{subject to:}\quad & \dot{x}_1(t) = u_4(t) - q \cdot x_1(t) - 17.6 \cdot x_1(t) \cdot x_2(t) - 23 \cdot x_1(t) \cdot x_6(t) \cdot u_3(t), \\
& \dot{x}_2(t) = u_1(t) - q \cdot x_2(t) - 17.6 \cdot x_1(t) \cdot x_2(t) - 146 \cdot x_2(t) \cdot x_3(t), \\
& \dot{x}_3(t) = u_2(t) - q \cdot x_3(t) - 73 \cdot x_2(t) \cdot x_3(t), \\
& \dot{x}_4(t) = -q \cdot x_4(t) + 35.2 \cdot x_1(t) \cdot x_2(t) - 51.3 \cdot x_4(t) \cdot x_5(t), \\
& \dot{x}_5(t) = -q \cdot x_5(t) + 219 \cdot x_2(t) \cdot x_3(t) - 51.3 \cdot x_4(t) \cdot x_5(t), \\
& \dot{x}_6(t) = -q \cdot x_6(t) + 102 \cdot x_4(t) \cdot x_5(t) - 23 \cdot x_1(t) \cdot x_6(t) \cdot u_3(t), \\
& \dot{x}_7(t) = -q \cdot x_7(t) + 46 \cdot x_1(t) \cdot x_6(t) \cdot u_3(t), \qquad (4.6) \\
& \dot{x}_8(t) = 5.8 \cdot (q \cdot x_1(t) - u_4(t)) - 3.7 \cdot u_1(t) - 4.1 \cdot u_2(t) + \\
& \quad q \cdot (23 \cdot x_4(t) + 11 \cdot x_5(t) + 28 \cdot x_6(t) + 35 \cdot x_7(t)) - 5 \cdot u_3^2(t) - 0.09, \\
& q = (u_1(t) + u_2(t) + u_4(t)), \\
& x(t_0) = [0.1883, 0.2507, 0.0467, 0.0899, 0.1804, 0.1394, 0.1046, 0.0]^T, \\
& 0 \le u_1(t) \le 20, \quad 0 \le u_2(t) \le 6, \\
& 0 \le u_3(t) \le 4, \quad 0 \le u_4(t) \le 20, \\
& t_0 \le t \le t_f, \\
& t_0 = 0, \ t_f = 0.2.
\end{aligned}
$$

The Jacobian of the right-hand side $\mathbf{f}$ of the model equations with respect to the control variables $\mathbf{u} = [u_1, u_2, u_3, , u_4]^T$ is given by

$$\frac{\partial \mathbf{f}}{\partial \mathbf{u}} = \begin{bmatrix} -x_1(t) & -x_1(t) & -23 \cdot x_1(t) \cdot x_6(t) & 1 - x_1(t) \\ 1 - x_2(t) & -x_2(t) & 0 & -x_2(t) \\ -x_3(t) & 1 - x_3(t) & 0 & -x_3(t) \\ -x_4(t) & -x_4(t) & 0 & -x_4(t) \\ -x_5(t) & -x_5(t) & 0 & -x_5(t) \\ -x_6(t) & -x_6(t) & -23 \cdot x_1(t) \cdot x_6(t) & -x_6(t) \\ -x_7(t) & -x_7(t) & 46 \cdot x_1(t) \cdot x_6(t) & -x_7(t) \\ 3.7 + \xi & 4.1 + \xi & -10 \cdot u_3(t) & 5.8 + \xi \end{bmatrix}, \tag{4.7}$$

where $\xi = 5.8 \cdot x_1(t) + 23 \cdot x_4(t) + 11 \cdot x_5(t) + 28 \cdot x_6(t) + 35 \cdot x_7(t)$. It can be seen that the first, second, and fourth column are nearly linearly dependent. This means that the matrix $\frac{\partial \mathbf{f}}{\partial \mathbf{u}}$ tends to be ill-conditioned, which will cause convergence difficulties during the solution of the resulting NLP problem.

Now, we calculate the angles between these columns using simulation, based on four PRBS signals as control profiles as shown in Figure 4.1. In order to compute state trajectories and corresponding sensitivities the original dynamic optimization problem (4.6) is discretized using CMSC method by applying 60 equidistant time intervals.

The calculated angles are given in Table 4.1, showing that there are strong U-U correlation pairs, namely, u_1-u_2, u_1-u_4, and u_2-u_4. Consequently, it will be favorable

Table 4.1: Correlation values between controls for CSTR

	u_1	u_2	u_3	u_4
u_1	0.0°	12.1°	123.6°	14.5°
u_2	12.1°	0.0°	123.4°	14.2°
u_3	123.6°	123.4°	0.0°	123.1°
u_4	14.5°	14.2°	123.1°	0.0°

to solve this problem using an AH instead of the BFGS method.

As predicted from the correlation analysis, the solution of this problem indicates that the analytical Hessian is indeed much more efficient than the BFGS method. It can be seen that the convergence is very slow (1687 NLP iterations) using the BFGS method, whereas a significant improvement (22 NLP iterations) is achieved using the AH, see Figure 4.2A. This leads to a significant reduction in computation time. The convergence rate in terms of the objective-function value and the primal and dual infeasibilities (reported by IPOPT) is shown in Figure 4.2, where only the first 100 iterations are plotted for the BFGS method. It can be clearly seen from Figure 4.2B,C that a feasible solution is difficult to achieve using the BFGS method.

The second problem is an optimal control of a rigid satellite [52, 83] initially undergoing a tumbling motion. The system dynamics is described by seven ordinary

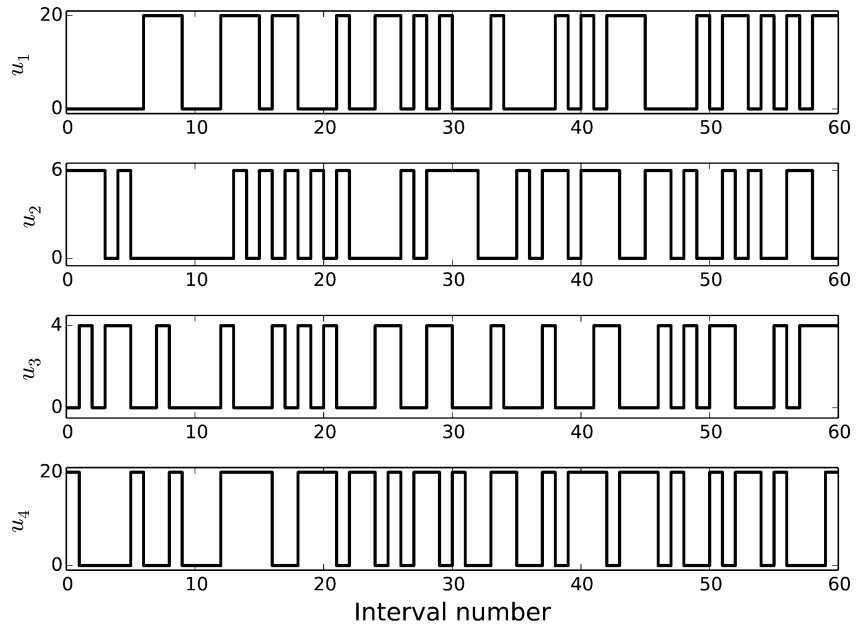

Figure 4.1: Control profiles generated from PRBSs for the CSTR problem

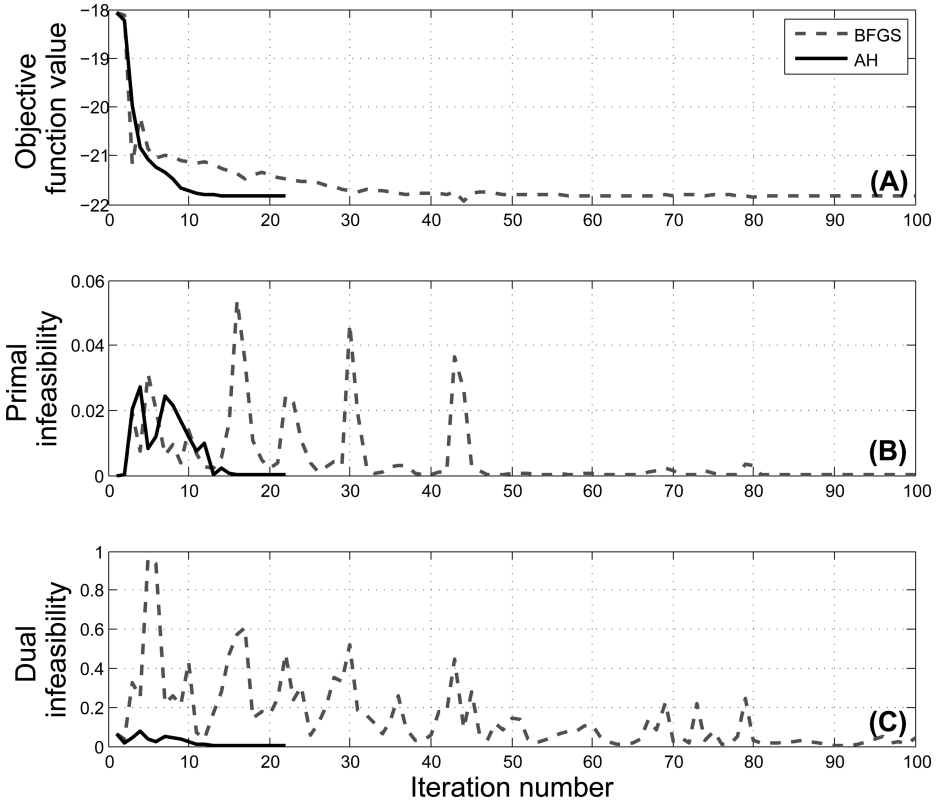

Figure 4.2: Convergence profiles during solution of the CSTR problem

differential equations. The optimal control problem is formulated as follows

$$
\begin{aligned}
&\min_{u(t)} \left\{ \|x(t_f) - x_f\|^2 + 0.5 \cdot \int_{t_0}^{t_f} \|u\|^2 dt \right\} \\
&\text{subject to:} \\
&\dot{x}_1 = 0.5 \cdot (x_5 x_4 - x_6 x_3 + x_7 x_2)\,, \\
&\dot{x}_2 = 0.5 \cdot (x_5 x_3 + x_6 x_4 - x_7 x_1)\,, \\
&\dot{x}_3 = 0.5 \cdot (-x_5 x_2 + x_6 x_1 - x_7 x_4)\,, \\
&\dot{x}_4 = -0.5 \cdot (x_5 x_1 + x_6 x_2 + x_7 x_3)\,, \\
&\dot{x}_5 = ((I_2 - I_3)\, x_6 x_7 + T_{1s} u_1) \cdot I_1^{-1}, \\
&\dot{x}_6 = ((I_3 - I_1)\, x_7 x_5 + T_{2s} u_2) \cdot I_2^{-1}, \\
&\dot{x}_7 = ((I_1 - I_2)\, x_5 x_6 + T_{3s} u_3) \cdot I_3^{-1}, \\
&x(t_0) = [0, 0, 0, 1, 0.01, 0.005, 0.001]^T, \\
&x_f = [0.70106, 0.0923, 0.56098, 0.43047, 0, 0, 0]^T, \\
&t_0 \le t \le t_f,\ t_0 = 0,\ t_f = 100.
\end{aligned}
\tag{4.8}
$$

where $I_1 = 10^6, I_2 = 833333, I_3 = 916677$ represent principal moments of inertia and $T_{1s} = 550, T_{2s} = 50, T_{3s} = 550$ are the corresponding time constants. The Jacobian of the right-hand-side functions of model equations with respect to the control variables is represented by

$$
\frac{\partial \mathbf{f}}{\partial \mathbf{u}} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ \frac{T_{1s}}{I_1} & 0 & 0 \\ 0 & \frac{T_{2s}}{I_2} & 0 \\ 0 & 0 & \frac{T_{3s}}{I_3} \end{bmatrix}. \tag{4.9}
$$

From equation (4.9), it can be clearly seen that the columns are linearly independent, and thus, the controls are not correlated with each other. The angles between the columns calculated through simulation using PRBS signals as control profiles are given in Table 4.2, indicating very weak correlations between the controls. Consequently, it is expected that the problem can be efficiently solved using the BFGS method.

Table 4.2: Correlation values between controls for SAT

	u_1	u_2	u_3
u_1	0.0°	98.3°	95.9°
u_2	98.3°	0.0°	102.7°
u_3	95.9°	102.7°	0.0°

The time horizon is divided into 60 equidistant time intervals. The IPOPT default

convergence tolerance 10^{-8} is used in solving the resulting NLP problem. The convergence rate is indicated in Figure 4.3. The numbers of iterations required using the BFGS method and using AH are equal to 13 and 8 iterations, respectively. However, the computation cost is less than that required using an AH. Therefore, using the BFGS method is preferred.

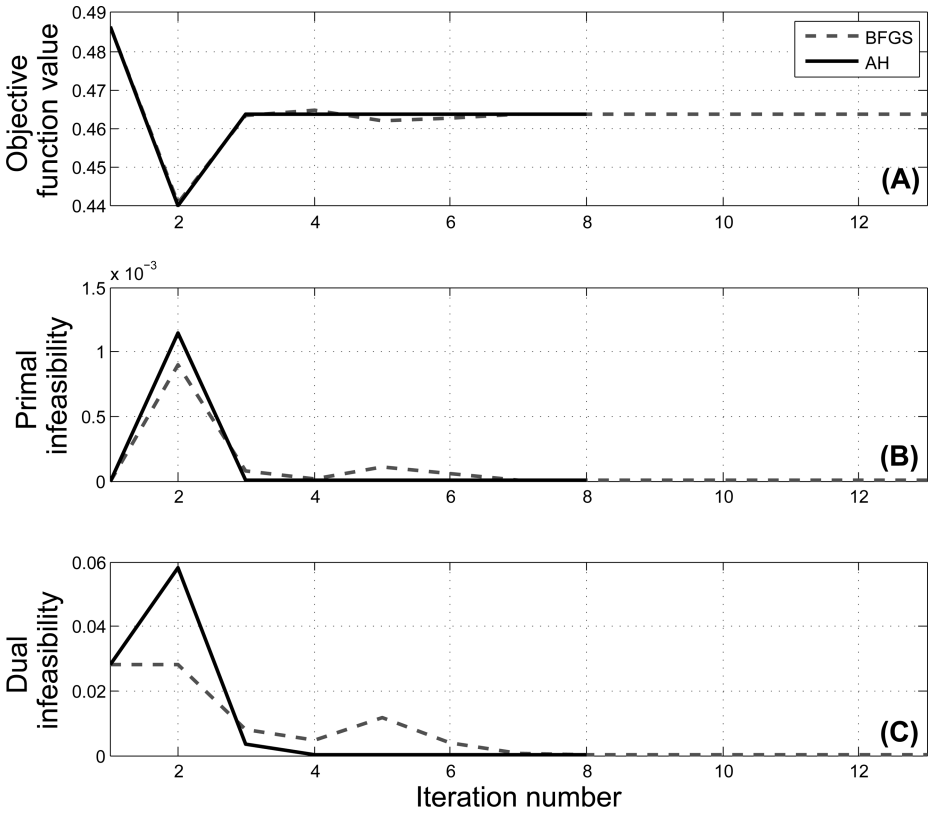

Figure 4.3: Convergence profiles during solution of the SAT problem

Through both practical engineering examples the viability of the proposed control-variable correlation analysis was demonstrated. Furthermore, this approach will be applied for the a priori investigation of the vehicle models in the design of control strategies for autonomous driving.

Chapter 5

An approach to determining the number of time intervals

5.1 Error estimation problem

The CMSC method, on the one hand, allows us to convert a time-dependent dynamic optimization problem (3.1) into an NLP problem (3.2) and provides fast numerical solution [61]. On the other hand, the numerical solution is only an approximation of the analytical solution of the original problem. The numerical accuracy of the approximation plays an important role in the realizability of the solution of the optimization problem. Therefore, in this study, the focus is on the investigation of the impact as well as the determination of the number of time intervals on the approximation errors of the numerically computed state trajectories.

For the analysis of the approximation error, three factors should be taken into account in solving the NLP problem described above: the discretization method together with the order of polynomials used in the algorithm; the number of time intervals used for the discretization; and the variation of the state trajectories over the time horizon which is caused by the operating condition. In this study, we assume that a discretization scheme is already selected with a fixed order of polynomials. Thus, the interest here is to analyze the impact of the number of time intervals for the collocation and the operating conditions (i.e., control profiles as well as the initial state values) on the numerical error.

Applying the CMSC method, collocation on finite elements is used for the discretization of state trajectories inside each time interval. For simplicity of the analysis, the number of Radau collocation points is chosen as $n_c = 3$. Thus, in each time interval, a state trajectory $x(t)$, namely its analytical solution, will be approximated by a numerically computed profile $x_N(t)$ using Lagrange polynomials

$$x_N(t) = \sum_{k=0}^{n_c} \prod_{\substack{j=0 \\ j \neq k}}^{n_c} \frac{t - t_j}{t_k - t_j} \cdot x_{N,k}, \tag{5.1}$$

where $x_{N,k}$ is the value of the state trajectory at the k-th collocation point. Based on the principle of the collocation methods, the numerical and analytical state values at the collocation points are nearly identical, within the tolerance of the solution method used in the simulation layer [92, 61]. Thus, $x(t_k) = x_{N,k}$, where t_k denotes the time position of the k-th collocation point. Therefore, it is not necessary to investigate

approximation errors at the collocation points. Instead, we need to analyze the errors at noncollocation points.

Theoretically, the number of noncollocation points is infinite, which makes an error estimation overcomplicated. To overcome this difficulty, we choose a single noncollocation point in each time interval. Consider the i-th time interval $[t_{i-1}, t_i]$ with corresponding Radau collocation points $\{t_{i-1}, \tau_{i,1}, \tau_{i,2}, \tau_{i,3}\}$. Hence, $\hat{t}_i = \tau_{i,1} + 0.5 \cdot (\tau_{i,2} - \tau_{i,1})$ is a noncollocation point, since this point is farther away from two neighboring collocation points than any other noncollocation points between $[\tau_{i,1}, \tau_{i,2}]$. The state profile between the collocation points is approximated by the Lagrange polynomial. Intuitively, a maximum error will be at the midpoint between two adjacent collocation points. Therefore, we chose this point as the noncollocation point which is also employed for error estimation in the work of Bartl et al. [11]. It is noted that this noncollocation point is not the midpoint of the time interval as mentioned in the work of Vasantharajan and Biegler [96]. Nevertheless, the analysis of the approximation errors can be easily extended by analyzing additional noncollocation points. Such an extension does not change the idea of the proposed approach, which is described in the next section.

Furthermore, the error function $e(\hat{t}_i)$ at the specified noncollocation points over the time horizon $[t_0, t_f]$ is introduced and defined as

$$e(\hat{t}_i) = |x(\hat{t}_i) - x_N(\hat{t}_i)|, \; i \in \{1, \ldots, N\}\,. \tag{5.2}$$

Hence, the absolute value of the numerical error $e(\hat{t}_i)$, defined in (5.2), will be increased if the difference between $x(\hat{t}_i)$ and $x_N(\hat{t}_i)$ is increased, pointwise. Since the form of the function $x_N(t)$ is fixed by the selected discretization scheme, the difference between the two functions will be high if $x(t)$ has a high oscillating behavior over $t \in [t_i, t_{i+1}]$. As a result, the numerical error depends on the amplitude of this oscillating function, i.e., the greater the amplitude, the larger the numerical error will be. Such a behavior depends on the operating conditions (i.e., the initial condition and the control profiles) of the system.

Considering equidistant time intervals, the length of each time interval is determined as $\Delta t = (t_f - t_0)/N$. From the numerical point of view, the smaller the value of Δt, the less will be the resulting approximation error. However, the number of intervals will be larger, which causes higher computational cost because the dimension of the NLP problem (3.2) will be higher. Hence, it is required to find a compromise, in terms of the number of time intervals, between the approximation error over the time horizon $[t_0, t_f]$ and the computational load.

Therefore, the main goal of this study is to find a minimum number of time intervals in a given fixed time horizon so that the maximum approximation error at the noncollocation points will be less or equal to a user-specified tolerance ε of the state accuracy. For this purpose, the NLP problem (3.2) is reformulated by considering the number of time intervals N as an additional degree of freedom, and by introducing an inequality constraint for controlling the maximum approximation error at the noncollocation points. Hence, the error maximization problem is formulated as

$$
\begin{aligned}
& \max_{\mathbf{x}_p, \mathbf{u}, \tilde{\mathbf{x}}_0, N} \left\{ e_{max}^{i,q} \right\} \\
\text{subject to:} \quad & \mathbf{x}_{p,0} - \tilde{\mathbf{x}}_0 = 0, \\
& \mathbf{x}_{p,i} = \hat{\mathbf{x}}_i \left(\mathbf{x}_{p,i-1}, \mathbf{u}_{i-1} \right), \ i = 1, \ldots, N, \\
& \mathbf{x}_{min} \leq \mathbf{x}_p \leq \mathbf{x}_{max}, \\
& \mathbf{u}_{min} \leq \mathbf{u} \leq \mathbf{u}_{max}, \\
& e_{max}^{i,q} - \varepsilon \leq 0.
\end{aligned}
\tag{5.3}
$$

The variable $e_{max}^{i,q}$ denotes the maximum approximation error of the q-th state variable, $q \in \{1, \ldots, n_x\}$ in the i-th interval, $i \in \{1, \ldots, N\}$ and technically it is equal to

$$
e_{max}^{i,q} = \max_{i \in \{1,\ldots,N\}} \left\{ |x^{(q)}(\hat{t}_i) - x_N^{(q)}(\hat{t}_i)| \right\}, \tag{5.4}
$$

where the analytical and numerical solutions of the q-th state variable at the noncollocation point $\hat{t}_i$ are defined as $x^{(q)}(\hat{t}_i)$ and $x_N^{(q)}(\hat{t}_i)$, respectively. Since the number of time intervals is an integer variable, problem (5.3) can be classified as a mixed-integer nonlinear program (MINLP).

In addition, as indicated in problem (3.1), the system dynamics is described by a set of differential equations f with a given initial condition x_0 of the state variables. Thus, the solution of the model equations has a strong dependence on the initial condition. Moreover, if problem (3.1) is considered in a NMPC framework, the initial conditions x_0 cannot be considered as fixed values anymore, because in every prediction horizon they will be correspondingly updated. Furthermore, the resulting behavior of state trajectories is not only associated with the initial condition but also has the strong dependency on the control profile $u(t)$ in the given time horizon $[t_0, t_f]$. Thus, the combined impact on the resulting approximation errors, in the numerically computed state trajectories concerning variable initial values of the state variables and control profiles, has to be examined by solving (5.3).

In particular, the main focus is on the maximum of the approximation error to determine the associated minimum number of time intervals in order to balance the computation expense and the numerical tolerance of the state accuracy. Furthermore, for online applications such as NMPC, it is important to find the maximum approximation error a priori, such that the determined minimum number of intervals ensures the error tolerance for any operating conditions.

5.2 Bilevel problem formulation

Since the MINLP problem (5.3) is hard to solve directly, we transform it into a more convenient bilevel form, where the discretized (originally continuous) variables $\mathbf{x}_p, \mathbf{u}$ and the integer variable N are treated separately from each other. The inner loop determines the state trajectories with corresponding approximation errors to be es-

timated with the continuous independent variables. The outer loop determines the number of time intervals N in such a manner that the given approximation error tolerance is guaranteed. Therefore, both problems are combined together and have bilateral dependencies. It should be noted that the problem formulated is in a general form and it is independent of the chosen discretization method. The proposed bilevel scheme is given in Figure 5.1 .

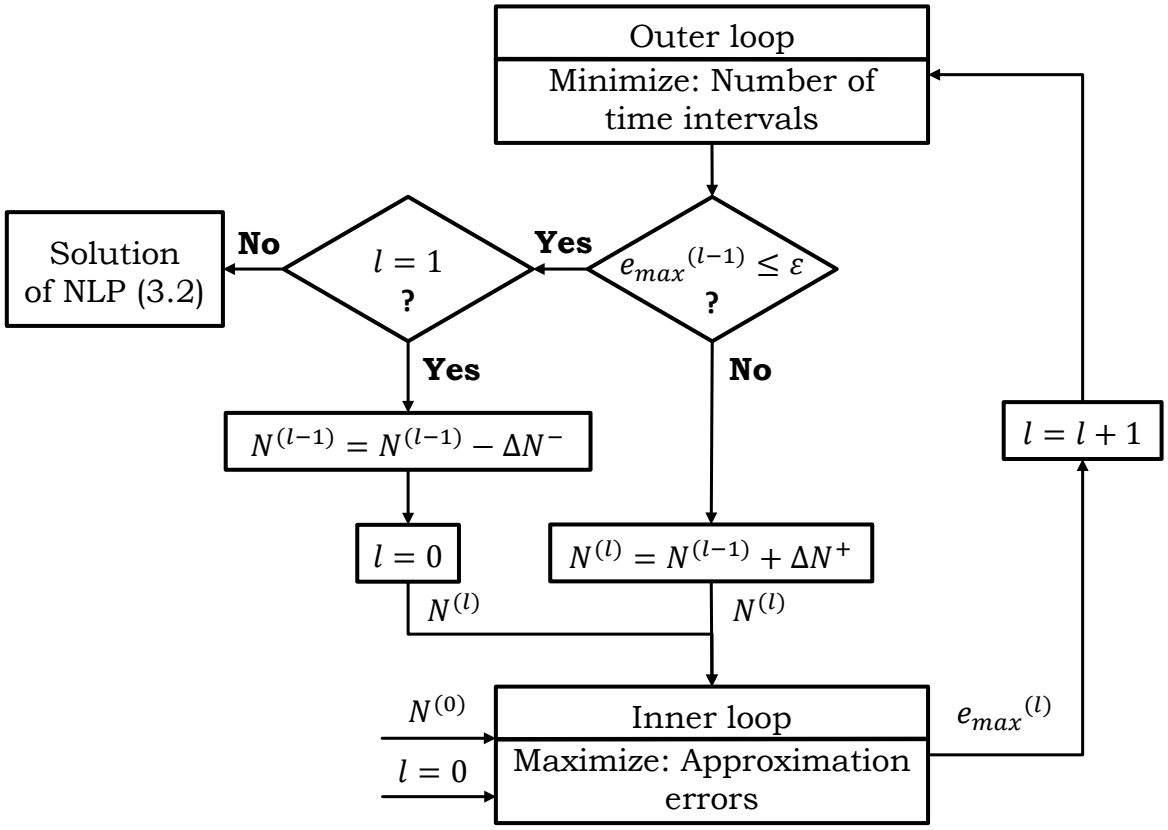

Figure 5.1: The bi-level solution framework

Starting from the initial guess $N^{(0)}$ for a number of time intervals and setting the iteration counter l to zero, the inner loop determines in each l-th iteration the maximum approximation error $e_{max}^{(l)}$ for the given fixed number of intervals from the outer loop by adjusting the control variables $u(t)$ and the initial condition x_0 of the state variables. Based on the results from the inner loop, the outer loop updates the number of time intervals.

On the one hand, if it is found that $e_{max}^{(l)}$, already in the first iteration ($l = 1$), is equal or less than the predefined tolerance ε, the number of time intervals will be reduced, using a decrement ΔN^- and again supplied to the inner loop. The iteration counter will be reset to zero. On the other hand, if the determined maximum approximation tolerance is larger than the predefined tolerance ε, the number of time intervals will be enlarged, i.e., using an increment ΔN^+, and again supplied to the inner loop.

The computation will terminate based on fulfillment of two conditions: $l \neq 1$ and $e_{max}^{(l-1)} \leq \varepsilon$. The resulting number of time intervals is the minimum number and can be applied for the solution of the NLP problem (3.2). It should be noted, however, that if the original problem (3.2) is an offline optimization problem (e.g., optimal operation of batch distillation), the initial condition is fixed. In this case, the problem formulated in (5.3) needs to be modified; i.e., the independent variables should not include the initial state values.

The major difference between the proposed bilevel approach and the one suggested by Tanartkit and Biegler [93, 94] lies in the fact that the objective function of the inner loop in our approach considers an error maximization, but in the inner loop of Tanartkit and Biegler the objective function of the original NLP problem was used. In addition, the bi-level strategy is a priori analysis and to be carried out offline, and thus the computation time is not a critical issue.

5.2.1 The outer loop

The outer loop is responsible for finding a minimum number of time intervals so that the upper limit of approximation errors over the fixed time horizon $[t_0, t_f]$ can be guaranteed, i.e.,

$$\begin{aligned} & \min_{N^{(l)}} \ \{N^{(l)}\} \\ \text{subject to:} \quad & e_{max}^{(q)} \leq \varepsilon, \end{aligned} \tag{5.5}$$

where $l = 0, 1, \ldots$ means the (global) iteration index of the bilevel approach.

Problem (5.5) is solved in a heuristic manner, i.e., if the maximum approximation error provided from the inner loop violates the predefined tolerance ε, the number of time intervals will be increased using an increment ΔN^+. In the case that already in the first iteration the error constraint is fulfilled, the number of time intervals will be decreased using a decrement ΔN^-, as indicated in Figure 5.1. Practically, the values of the increment and decrement can simply be chosen to be equal to 1. Nevertheless, other update strategies can also be used. However, a large increment cannot guarantee that the found number of time intervals is a minimum. Therefore, other update strategies will be much more complicated than the proposed one. As a result, its new value will be supplied to the inner loop, leading to an improved maximum approximation error. If the maximum approximation error is smaller than the tolerance, the corresponding number of time intervals is determined as the minimum number of time intervals. The final result can be applied for the discretization of the original problem (3.1).

5.2.2 The inner loop

As described above, the inner loop solves the error maximization problem with the updated number of time intervals provided by the outer loop. Taking into account the model equations and box-constraints imposed on state and control variables in

problem (3.1), but ignoring the original objective function, the NLP problem in the inner loop is posed as

$$
\begin{aligned}
\text{(Bilevel-I)} \quad & \max_{\mathbf{x}_p, \mathbf{u}, s} \; s \\
\text{subject to:} \quad & \mathbf{x}_{p,0} - \tilde{\mathbf{x}}_0 = 0, && (5.6)\\
& \mathbf{x}_{p,i} = \hat{\mathbf{x}}_i \left(\mathbf{x}_{p,i-1}, \mathbf{u}_{i-1} \right), \; i = 1, \ldots, N, && (5.7)\\
& \max_{i \in \{1,\ldots,N\}} \{ e^q(\hat{t}_i) \} - s = 0, && (5.8)\\
& x_{min} \leq \mathbf{x}_{p,i} \leq x_{max}, \; i = 0, \ldots, N, && (5.9)\\
& u_{min} \leq \mathbf{u}_i \leq u_{max}, \; i = 0, \ldots, N-1, && (5.10)\\
& s_{min} \leq s \leq s_{max}, && (5.11)
\end{aligned}
$$

where $s \in \Re^1$ defines the maximum approximation error and has the same meaning as the selected q-th state variable, $q \in \{1, \ldots, n_x\}$, under consideration. Equation (5.8) is introduced to evaluate the approximation error in the time horizon and thus, selects the maximum value from all calculated approximation errors in the time intervals $i \in \{1, \ldots, N\}$. The term $e^q\left(\hat{t}_i\right)$ describes the approximation error at the noncollocation point $\hat{t}_i$ in the i-th interval and can be computed using equation (5.2). Inequalities (5.9) – (5.10) define box constraints on the parametrized states and controls. These bounds are the same as in the original dynamic optimization problem formulation (3.1). The lower and upper bounds (5.11) of s are chosen as $s_{min} = 0$ and $s_{max} = 1$, respectively. The value of s_{max} is chosen empirically. Problem bilevel-I allows us to find the maximum approximation error with controls as decision variables. The initial condition $\tilde{\mathbf{x}}_0$ (the values of state trajectories), defined in the equality constraint (5.6), is fixed in this problem, which represents the case of an offline optimal control.

For an online application (as in the case of NMPC), the initial condition will vary from time to time. Thus, the impact of the initial condition on the approximation error needs to be considered. To include the initial values of states as decision variables, the corresponding NLP problem is formulated as follows

$$
\begin{aligned}
\text{(Bilevel-II)} \quad & \max_{\mathbf{x}_p, \mathbf{u}, s} \; s \\
\text{subject to:} \quad & \underline{\mathbf{x}}_{p,0} \leq \mathbf{x}_{p,0} \leq \overline{\mathbf{x}}_{p,0}, && (5.12)\\
& \text{constraints } (5.7) - (5.11),
\end{aligned}
$$

where, in contrast to equation (5.6), in inequality (5.12), the initial states are allowed to vary within a lower bound $\underline{\mathbf{x}}_{p,0}$ and an upper bound $\overline{\mathbf{x}}_{p,0}$. Thus, bilevel-II is concerned with the simultaneous impact of both the initial condition and control variables. Therefore, the solution of this problem (i.e., the minimum number of intervals) will be valid for an online application with a guaranteed state profile accuracy for varying initial conditions. In the bilevel-II problem, the bounds for the initial state values should be defined. Otherwise initial state values can be infinite due to the maximization of the approximation error, which is a contradiction to a physical process. Therefore,

one can investigate the related process to constrain the initial state values, i.e., to determine proper values of $\underline{\mathbf{x}}_{p,0}$ and $\overline{\mathbf{x}}_{p,0}$. For instance, state variables of chemical processes are usually mole fractions of components of a mixture. To prevent violations of the physical bounds on the mole fractions additional constraints representing the sum of all mole fractions equal to one should be introduced. The use of component balance equation makes it unnecessary to introduce such constraints in the optimization problem formulation. However, in the formulation of the bilevel-II problem, these constraints should be explicitly posed for limiting the initial state values.

The implementation details, considering the solution of bilevel-I and bilevel-II problems, are given in the next subsection.

5.3 Implementation details

One difficulty in solving the NLP problems bilevel-I and bilevel-II is the evaluation of the approximation error. In the NLP problem (3.2) the model equations are not directly involved in the problem formulation. The discretized model equations $\mathbf{G}_i$ for the i-th time interval are solved in the simulation layer and can be written in the following compact form

$$\left[\mathbf{G}_i\left(\mathbf{x}_i\left(\mathbf{x}_{p,i}, \mathbf{u}_i\right), \mathbf{x}_{p,i}, \mathbf{u}_i\right)\right]_{n_g} = \mathbf{0}, \tag{5.13}$$

where $n_g = n_x \cdot n_c$ and $i = 0, \ldots, N-1$. Thus, the values of the state variables at the collocation points can be obtained by solving the nonlinear equations $\mathbf{G}_i$ for $\mathbf{x}_i$ using a Newton solver. Based on these results, the value of the q-th state $x_N^{(q)}(\hat{t}_i)$ at a noncollocation point $\hat{t}_i$, required in equation (5.4), can be computed by an interpolation using the Lagrange polynomials (5.1).

Furthermore, the analytical (exact) value $x^{(q)}(\hat{t}_i)$ at the noncollocation points for the error evaluation should be made available . For this purpose, an additional solution of the model equations with a reduced interval length is employed. The reduced length of time interval is so determined that the last collocation point lies exactly at the noncollocation point $\hat{t}_i$. Since the state values at this point are also achieved by the Newton method, they are quasi-analytical. To illustrate this aspect, the following two simple differential equations are considered:

$$\dot{x}_1(t) = u_1(t) - x_1(t), \tag{5.14}$$

$$\dot{x}_2(t) = -x_2(t) \cdot u_2(t), \tag{5.15}$$

where $t \in [0, 1], x_1(0) = x_2(0) = 1.0$ and control variables are defined as constants, i.e $u_1(t) = u_2(t) = 2.0$. To demonstrate the proposed error estimation approach, equations (5.14) and (5.15) are solved both analytically and numerically, using a single time interval. Consequently, the errors between analytical/quasi-analytical and interpolated state values are compared with each other. It is shown that, for the equations (5.14) and (5.15) the differences between the real approximation error and that computed by the proposed approach are equal to $3.58 \cdot 10^{-7}$ and $1.47 \cdot 10^{-6}$, respectively,

which supports the reliability of the proposed approach.

It is worth noting that this error estimation approach is straightforward and does not require additional computations as in the residual approach [96, 11]. Moreover, if the state profiles are nonsmooth or if a state trajectory changes dramatically inside an interval or between two neighboring intervals, the residual approach may be insufficient [84, 96]. Nevertheless, our method here can determine the approximation error more precisely even in such situations, because the quasi-analytical values of state variables satisfy the model equations.

To solve the inner NLP problem, the sensitivities of the approximation error are required and calculated as follows. According to the continuity condition (5.7) the state variables $\mathbf{x}_i$ at the collocation points in the i-th interval have an implicit dependency on the decision variables $\mathbf{x}_{p,i}$ and $\mathbf{u}_i$. Hence, to obtain the first-order sensitivities in individual time intervals, two differentiation operators $\partial/\partial\mathbf{x}_{p,i}$ and $\partial/\partial\mathbf{u}_i$ are applied to equation (5.13), which yields

$$\left[\frac{\partial \mathbf{G}_i}{\partial \mathbf{x}_i}\right]_{n_g \times n_g} \cdot \left[\frac{\partial \mathbf{x}_i}{\partial \mathbf{x}_{p,i}}\right]_{n_g \times n_x} = -\left[\frac{\partial \mathbf{G}_i}{\partial \mathbf{x}_{p,i}}\right]_{n_g \times n_x}, \tag{5.16}$$

$$\left[\frac{\partial \mathbf{G}_i}{\partial \mathbf{x}_i}\right]_{n_g \times n_g} \cdot \left[\frac{\partial \mathbf{x}_i}{\partial \mathbf{u}_i}\right]_{n_g \times n_u} = -\left[\frac{\partial \mathbf{G}_i}{\partial \mathbf{u}_i}\right]_{n_g \times n_u}. \tag{5.17}$$

The above linear equation systems can be generated in a straightforward manner using, e.g., automatic differentiation. In the CMSC method, equations (5.16) and (5.17) have a sparse structure and thus can be solved for $\frac{\partial \mathbf{x}_i}{\partial \mathbf{x}_{p,i}}$ and $\frac{\partial \mathbf{x}_i}{\partial \mathbf{u}_i}$ by a sparse linear algebra algorithm.

Moreover, because of the equality constraint (5.8), the sensitivities of the quasi-analytical state variables are computed in the same way as in equations (5.16) and (5.17) based on the solution of the nonlinear equation system (5.13) with the reduced interval length. Furthermore, the state value computed by the interpolation at the noncollocation point $x_N^{(q)}(\hat{t}_i)$ has the same dependency on the decision variables $\mathbf{x}_{p,i}$ and $\mathbf{u}_i$ in the i-th interval. For instance, the sensitivities with respect to the control variable $\mathbf{u}_i$, at a noncollocation point can be obtained by applying the same derivative operator $\partial/\partial\mathbf{u}_i$ on the equation (5.1) for the q-th state variable in the i-th interval as

$$\frac{\partial x^{(i,q)}\left(\mathbf{x}_{p,i}, \mathbf{u}_i; \hat{t}_i\right)}{\partial \mathbf{u}_i} = \sum_{d=0}^{N_c} \prod_{\substack{j=0 \\ j \neq d}}^{N_c} \frac{\hat{t}_i - t_j}{t_d - t_j} \cdot \frac{\partial x_d^{(i,q)}\left(\mathbf{x}_{p,i}, \mathbf{u}_i\right)}{\partial \mathbf{u}_i}. \tag{5.18}$$

The sensitivities with respect to the parametrized initial conditions of state variables are similarly computed using the $\partial/\partial\mathbf{x}_{p,i}$ operator.

In addition, the constraint (5.8) is not smooth, because it contains maximum and absolute operators. However, both of them can be approximated using a softmax function [41], which is differentiable and smooth. For instance, the maximum between

two absolute values of numbers α and β can be computed in the following way:

$$\max\{|\alpha|, |\beta|\} \approx \ln\left(\exp\left(|\alpha|\right) + \exp\left(|\beta|\right)\right), \tag{5.19}$$

with

$$|\alpha| = \max\{\alpha, -\alpha\} \approx \ln\left(\exp\left(\alpha\right) + \exp\left(-\alpha\right)\right), \tag{5.20}$$
$$|\beta| = \max\{\beta, -\beta\} \approx \ln\left(\exp\left(\beta\right) + \exp\left(-\beta\right)\right). \tag{5.21}$$

From the numerical experiments, it is found that the proposed bilevel approach converges locally because of the nonlinear and nonconvex model equations. Thus, during the numerical tests the solution is initialized multiple times using random values (i.e., with uniform distribution) for the control variables and the initial state values to obtain a global maximum approximation error.

5.4 Illustrative examples

To demonstrate the effectiveness of the proposed approach, this section presents the solution of two dynamic optimization problems, namely a bifunctional catalyst blend in a tubular reactor [72] (BCBTR) and a nonlinear continuous stirred tank reactor [71] (NCSTR). In the BCBTR problem the resulting optimal control profile depends on the initial guess which leads to multiplicity of optimal solutions [72]. Therefore, the maximum approximation error obtained by different control profiles is also different. The challenge in the NCSTR problem is due to the correlations between three of the four control variables [61].

During the solution of the bilevel approaches, the NLP problems in the inner loop were solved 300 times using randomized initial guesses for each given number of time intervals from the outer loop. The computation time is not critical because an a priori result is to be gained. Nevertheless, since the multiple runs are independent, the computational burdens can be significantly reduced through parallel computing.

In addition, one state variable needs to be selected for evaluating the approximation error. This state variable can be selected based either on the state dynamics, e.g., larger amplitudes cause higher approximation errors, or on the importance, because in large-scale problems only a small number of state variables have higher priority. Nevertheless, after the maximum approximation error for a defined state is found, numerical errors of all other state variables can be easily computed, using a simulation step with the resulting control profile.

The tolerance 10^{-8} in IPOPT was used for solving the NLP problem in the two examples. For the computation of the state values at collocation and noncollocation points the Newton method was applied with the tolerance 10^{-12}. This tolerance plays a major role within the bilevel approach. The value of it should be low enough in order to compute quasi-analytical state values at noncollocation points closer to the real unknown analytical values. Moreover, if this tolerance is high, the IPOPT solver will experience convergence difficulties due to the continuity conditions for the state

variables formulated by the multiple-shooting discretization. In general, the relation between the Newton tolerance ε_N, the IPOPT tolerance ε_I and the algorithm tolerance ε_A can be described as $\varepsilon_N \leq \varepsilon_I \leq \varepsilon_A$.

In the Tables 5.1 and 5.2 below, the first column shows the predefined error tolerance ε and the second column presents the resulting minimum number of time intervals and the maximum approximation error by solving the bilevel-I problem. The last column shows the maximum error by solving the original dynamic optimization problem discretized with the number of time intervals listed in the second column. Table 5.3 describes the results by solving the bilevel-II problem with predefined error tolerance ε given in the first column. The second and third columns show the results of the bilevel approach using different lengths of the prediction horizon.

The BCBTR problem is formulated as follows [72]

$$
\begin{aligned}
& \max_{u(t)} x_7(t_f) \\
\text{subject to:} \quad & \dot{x}_1(t) = -k_1 \cdot x_1(t), \\
& \dot{x}_2(t) = k_1 \cdot x_1(t) - (k_2 + k_3) \cdot x_2(t) + k_4 \cdot x_5(t), \\
& \dot{x}_3(t) = k_2 \cdot x_2(t), \\
& \dot{x}_4(t) = -k_6 \cdot x_4(t) + k_5 \cdot x_5(t), \\
& \dot{x}_5(t) = k_3 \cdot x_2(t) + k_6 \cdot x_4(t) - (k_4 + k_5 + k_8 + k_9) \cdot x_5(t) \\
& \qquad\quad + k_7 \cdot x_6(t) + k_{10} \cdot x_7(t), \\
& \dot{x}_6(t) = k_8 \cdot x_5(t) - k_7 \cdot x_6(t), \\
& \dot{x}_7(t) = k_9 \cdot x_5(t) - k_{10} \cdot x_7(t), \\
& x(t_0) = [1, 0, 0, 0, 0, 0, 0]^T, \\
& t_0 \leq t \leq t_f, \ t_0 = 0, \ t_f = 2000, \\
& 0.6 \leq u(t) \leq 0.9,
\end{aligned}
\tag{5.22}
$$

The chemical reaction is described by seven differential equations, where $x_i, i = 1, \ldots, 7$ are the mole fractions of different chemical components [72]. Each rate constant $k_i, i = 1, \ldots 10$ is expressed as a cubic function of the catalyst blend $u(t)$, that is

$$
k_i = c_{i,1} + c_{i,2} \cdot u(t) + c_{i,3} \cdot u^2(t) + c_{i,4} \cdot u^3(t), \tag{5.23}
$$

where the coefficients $c_{i,j}, j = 1, \ldots 4$ are experimentally obtained [72].

Without loss of generality, for the a priori determination of the minimum number of time intervals, the state variable $x_7(t)$ is selected for the error analysis. This state variable describes the mole fraction of the product (benzene), which is to be maximized at the exit of the tubular reactor. First, the original optimization problem (5.22) is converted into the bilevel-I problem. The results are given in Table 5.1.

As shown in Table 5.1, as the error tolerance decreases, the required number of time intervals increases. The resulting maximum error is less than the corresponding tolerance value. Then, using the minimum number of intervals determined by bilevel-I, the BCBTR problem is solved. Because of the nonconvexity of the problem, multiple

Table 5.1: Obtained results for the **BCBTR** problem

	Bilevel-I		Original NLP
ε	N	e_{max}	$\hat{e}_{max}$
$2 \cdot 10^{-5}$	10	$1.927 \cdot 10^{-5}$	$1.911 \cdot 10^{-5}$
$1 \cdot 10^{-5}$	21	$9.556 \cdot 10^{-6}$	$9.044 \cdot 10^{-6}$
$5 \cdot 10^{-6}$	37	$4.948 \cdot 10^{-6}$	$3.564 \cdot 10^{-6}$

optimal solutions for the same number of time intervals are obtained, depending on the initial guess of the control values. Therefore, problem (5.22) is transformed into problem (3.2) and solved 200 times using random initializations. The largest value of the approximation error obtained by solving the problems is chosen to compare with the maximum error obtained by the bilevel approach. As shown in the last column of Table 5.1, the numerical error resulted by solving the original optimal control problem is less than that by solving the bilevel-I problem, i.e., the a priori determined minimum number of time intervals can guarantee the predefined error tolerance. Moreover, considering the tolerance $\varepsilon = 2 \cdot 10^{-5}$, 10 time intervals are required, as shown in Table 5.1. As a test, if we apply 9 time intervals instead for solving the original optimization problem, then $\hat{e}_{max} = 2.051 \cdot 10^{-5}$ which violates the predefined error tolerance. To formulate the constraints of the initial state values in the bilevel-II problem of this case study, we introduce the lower and upper bound for each initial state being equal to 0 and 1, respectively, and an equality constraint that the sum of all the mole fractions is equal to 1. In addition, different lengths of the time horizon are used to demonstrate the simultaneous impact of the initial condition and control profiles. The relations between the length of time horizon, error tolerance and number of time intervals are shown in Figure 5.2.

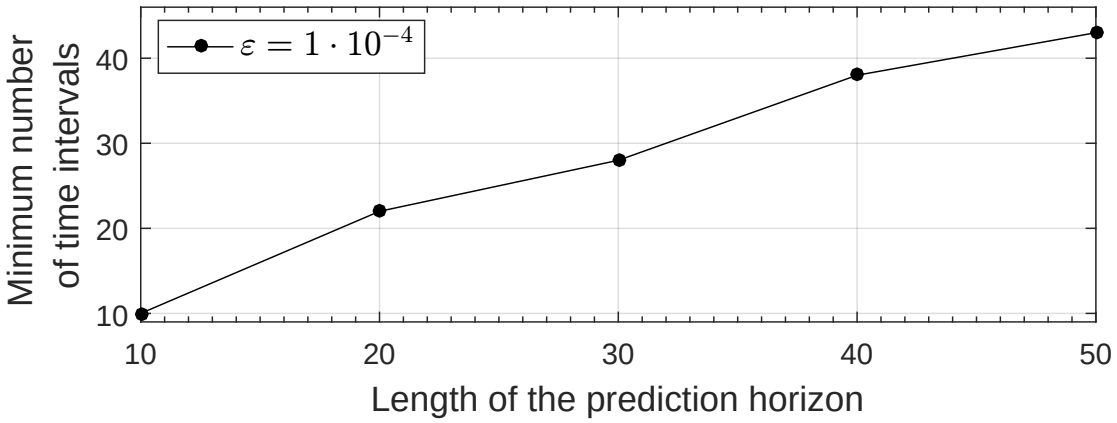

Figure 5.2: Impact of the prediction horizon

It can be seen that when using the bilevel-II approach with different lengths of the prediction horizon, a suitable number of time intervals which guarantees a predefined error tolerance can be found. For instance, if a very low error tolerance is specified,

i.e., if very accurate results are desired, then the problem should be discretized using a very large number of intervals, which will be computationally expensive for an NMPC application. As a result, the solutions here provide alternative options for designing of a model predictive controller, from which a compromise between the length of the prediction horizon and the numerical accuracy can be decided.

The second demonstration example problem is from Luus [71] and was also analyzed by Balsa-Canto et al. [9]. The purpose is to determine optimal control profiles for maximizing the economic benefits, with the problem (4.6) as defined in the previous section. The controls include the flow-rates of three feed streams and an electrical energy input for the photochemical reaction. The system dynamics is described by several simultaneous chemical reactions. This particular problem is challenging, because the system is highly nonlinear and the optimal control profiles are quite complicated [9]. For demonstration of our bi-level approach, the minimum number of intervals is determined relying on the maximum approximation error of the state variable $x_1(t)$. The results of bilevel-I problem are given in Table 5.2. It is shown that the number of time intervals required increases, when a higher accuracy of the state trajectory is specified. Using the determined number of intervals to solve the original problem, the resulting error is smaller than the corresponding error tolerance.

Table 5.2: Obtained results for the **NCSTR** problem

	Bilevel-I		Original NLP
ε	N	e_{max}	$\hat{e}_{max}$
$1 \cdot 10^{-3}$	8	$6.228 \cdot 10^{-4}$	$0.533 \cdot 10^{-4}$
$1.125 \cdot 10^{-4}$	12	$1.191 \cdot 10^{-4}$	$0.081 \cdot 10^{-4}$
$5 \cdot 10^{-6}$	22	$4.4 \cdot 10^{-6}$	$0.4 \cdot 10^{-6}$

For the formulation of the bilevel-II problem, the constraints of the initial state values for $x_1(t)$-$x_7(t)$ are treated in the same way as in the last case study, since they are mole fractions. Two different lengths for the prediction horizon are considered, i.e.,

Table 5.3: Results of the bilevel-II approach for the **NCSTR** problem

	$t_f = 0.1$		$t_f = 0.2$	
ε	N	e_{max}	N	e_{max}
$1 \cdot 10^{-3}$	5	$8.96181 \cdot 10^{-4}$	9	$6.00485 \cdot 10^{-4}$
$1.125 \cdot 10^{-4}$	7	$3.78346 \cdot 10^{-5}$	13	$5.03151 \cdot 10^{-5}$
$5 \cdot 10^{-6}$	12	$3.37374 \cdot 10^{-6}$	24	$4.79487 \cdot 10^{-6}$

$t_f = 0.1$ and $t_f = 0.2$. It can be seen in Table 5.3, that if a shorter prediction horizon is used (second column), the number of intervals is smaller than those obtained by solving the bilevel-I problem. However, a shorter length of the prediction horizon in a NMPC application may be insufficient in terms of, e.g., handling model uncertainties or disturbances. In the third column of Table 5.3, the bilevel-II approach is applied with $t_f = 0.2$. As expected, if the initial state values are taken as variables, the

solution of the bilevel-II problem leads to the approximation errors larger than those obtained by solving the bilevel-I problem. Consequently, the number of time intervals required will be higher by all investigated tolerances ε as indicated in the last column in Table 5.3.

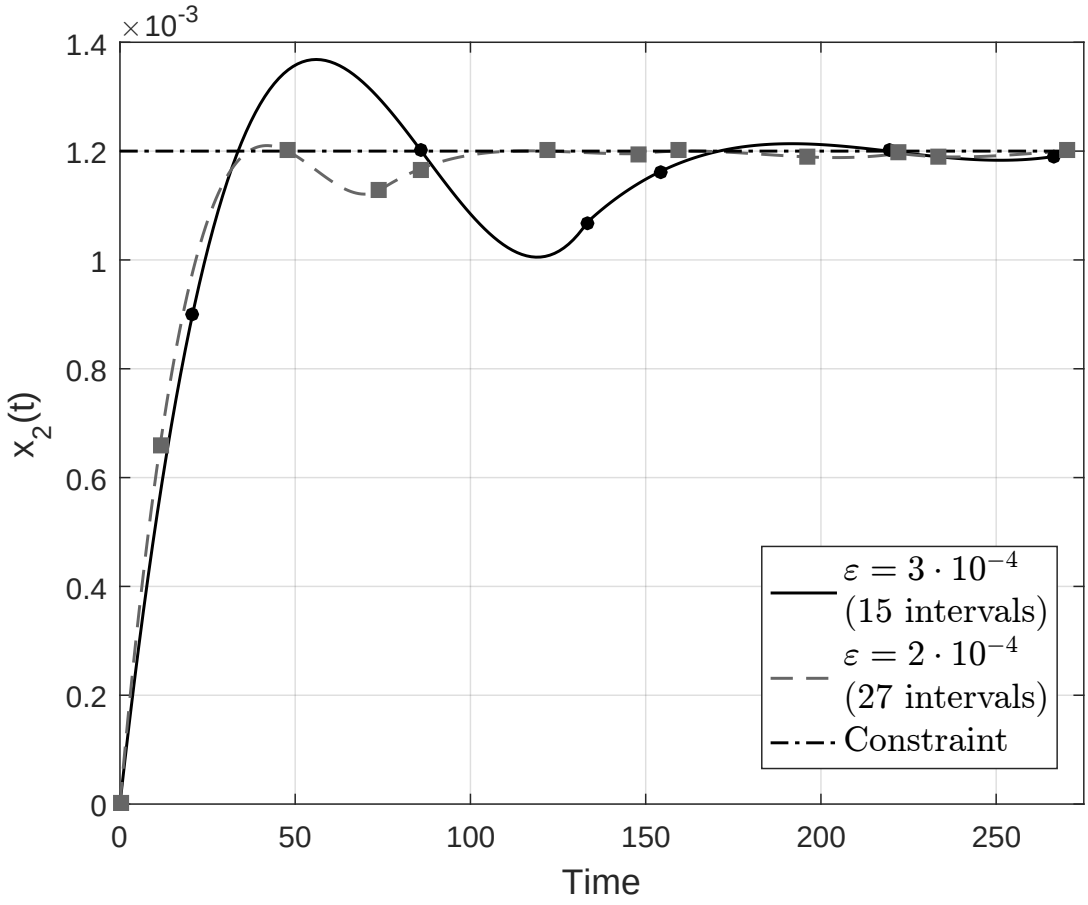

Figure 5.3: Impact of the approximation error at noncollocation points on the constraint violation

One of the important points during the solution of the dynamic optimization is to satisfy the inequality constraints in the process operation. If the number of intervals is fewer than required, the resulting numerical error of state variables may lead to violations of inequality constraints. To demonstrate this issue, the BCBTR problem has been selected, where $x_2(t)$ is chosen with a constraint $x_2(t) \leq 1.2 \cdot 10^{-3}$. The results are shown in Figure 5.3. By using $\varepsilon = 3 \cdot 10^{-4}$ and $\varepsilon = 2 \cdot 10^{-4}$, 15 and 27 time intervals are required, respectively. With these numbers of time intervals the original optimization problem (5.22) is discretized and then solved numerically. It can be seen that the violation of the inequality constraint can be neglected with 27 intervals and that the violation is significant with 15 intervals. These results were published in the work of Lazutkin et. al. [62].

Chapter 6

Numerical implementation issues

6.1 Component description

The major goal of designing a new CMSC framework is to eliminate several disadvantages of the first version of the previous framework suggested in [92]. At the same time to perform an efficient standalone implementation using open-source libraries and solvers of a toolchain for solving dynamic optimization problems. Therefore, in this work the developed toolchain aims to present multiple advantages for the end-users: extensibility, generality, problem independent design, user-friendly interface, integrability. The main software components of the framework [61, 63] are given in Figure 6.1. The major goal of this framework is to have the possibility to solve different dynamic optimization problems without providing any algorithmic implementations.

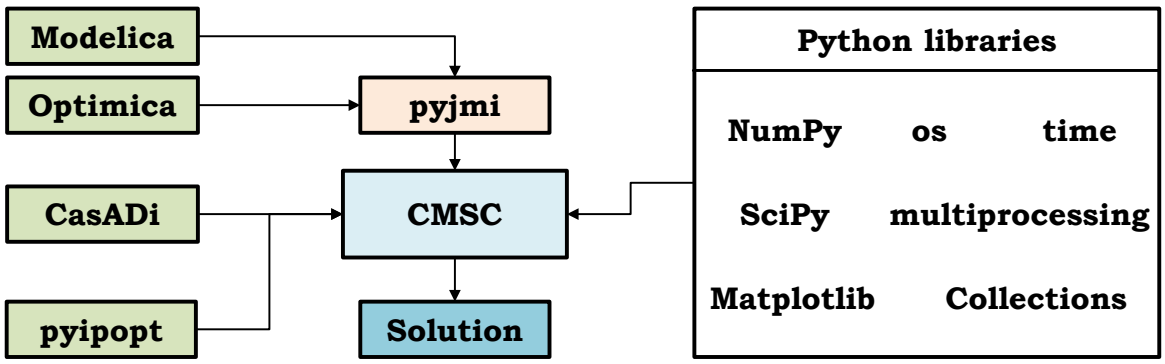

Figure 6.1: Main software components

Modelica [37] is an object-oriented modeling environment commonly used for complex industrial systems. It also possesses the typical characteristics of encapsulation, modularization, inheritance and polymorphism. Object-oriented modeling has become increasingly popular, especially since the turn of the millennium. One of the main reasons for this is the degree of abstraction, which makes it possible to use this technique in numerous areas with a consistent and standardized approach. The Optimica extension [2] allows the formulation of optimization problems. JModelica [3] provides an important platform for the user-generated implementation of efficient algorithms for the optimization of processes and technical systems, which is becoming increasingly important. In Figure 6.1 the JModelica is indicated by the module *pyjmi*, which contains the transfer function between Modelica and Optimica models and their symbolic

representation in Python [47]. The development of JModelica was aimed at creating an environment for detailed analysis and further development of powerful algorithms for the dynamic optimization and simulation of complex systems. Another essential tool used in this work is CasADi [4]. The main focus was on a flexible implementation of own algorithms for developers and users. General functions for the implementation are available from several pure Python libraries, i.e., parallel computing is implemented using the standard multiprocessing Python module.

6.2 Algorithm implementation

In summary, the transformation from the dynamic optimization problem to an NLP is done completely automatically, see Figure 6.2, where developed methods from Chapters 4 and 5 are also integrated within main CMSC framework as independent extensions.

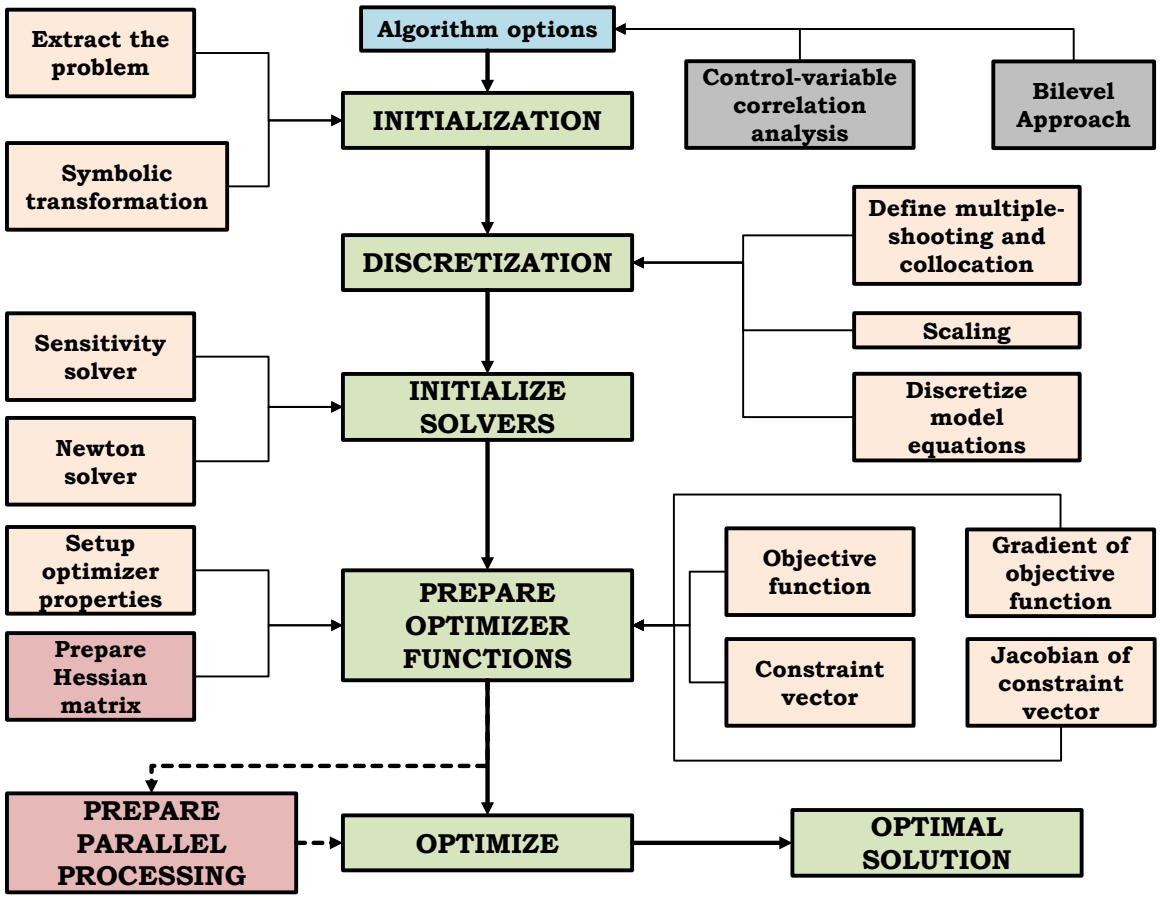

Figure 6.2: CMSC algorithm structure

The framework uses a set of parameters, such as number of time intervals and collocation points, defined in the algorithm options. The whole set of parameters is available

in the documentation. Based on these parameters, the first step is to transfer the formulated dynamic optimization problem from Modelica to its symbolic representation in Python. To guarantee the generality of the framework, the special preprocessing procedure was developed to analyze the optimization problem symbolically, which was not done in the previous implementation [92]. As a result, the important parameters (e.g., the number of state and control variables, type of the problem, model equations, etc.) can be extracted from the problem formulation automatically. Moreover, using the preprocessing procedure the dynamic optimization problem is available in Python language for further post-processing procedures.

The second step is to perform the discretization of the model equations. Using the defined number of time intervals and collocation points per time interval the framework automatically generates the vectors with symbolic variables. These vectors represent the state variables to be computed at collocation points, parameterized state and control variables. This step is done with the help of CasADi. The model equations are initialized as symbolic functions and then evaluating them using defined vectors. The required discretized nonlinear equations in the form of equation (3.3) is available. This system of equations is implemented in the appropriate symbolic function, which will be solved using the CasADi Newton solver (NS). Moreover, appropriate callback functions for sensitivity computations (SC) are constructed for the system of equations (3.4) and (3.5) by using the linear algebra solver CSparse.

Based on the used optimizer, i.e., IPOPT [100], the framework automatically prepares required symbolic functions for objective function and its gradient, constraint vector and its Jacobian. When using analytical second-order sensitivities (analytical Hessian), the appropriate function callback is activated. Otherwise, these sensitivities are approximated within the IPOPT solver. The construction of all functions is made completely automatically in a problem-independent manner. An example of the symbolic construction is given in the Appendix with the help of simple dynamic optimization problem. Note that the original version of the CMSC method in [92] suffers from the lack of automatic construction of such functions.

Based on the nature of the CMSC method, the parallel computations are available for computation of the state trajectories and corresponding sensitivities with respect to the optimization variables. The default option for the number of used processes is equal to one, which leads to the computation scheme given in Figure 6.3, where

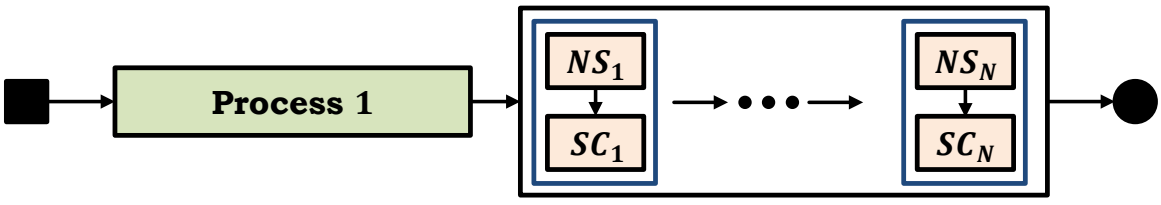

Figure 6.3: Sequential computations

a single process performs the sequential computation starting with the first interval.

This procedure is done in every optimizer iteration.

Providing the adaptation of the function callbacks, it is possible to provide parallel processing in order to reduce computation burdens. The modified scheme is given in Figure 6.4, where $p = \frac{N}{M}$, N and M denote the number of time intervals and the number of parallel processes, respectively .

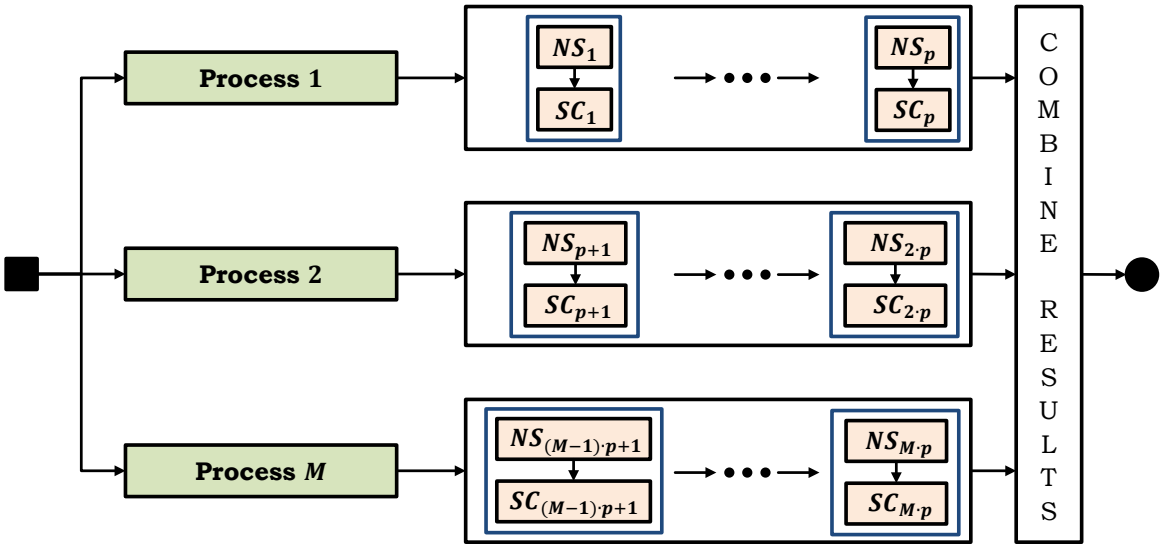

Figure 6.4: Parallel computations

After providing the problem decomposition, each process receives equal number of time intervals to be solved. However, other decomposition techniques may be used. It should be noted that the time taken for the communications between the master processor and the worker processors can be significant when too many worker processors are used to solve a small-scale problem. In such a case, the maximum speedup factor can be achieved when fewer worker processors are used. The parallelization is done using multiprocessing and not multithreading, since the CasADi package is not thread-safe. Using parallel processing the computation time can be significantly reduced [61].

As indicated in Figure 6.2, the proposed bilevel approach and control-variable correlation analysis are interfaced to the CMSC toolchain, where Modelica and Optimica models are used to formulate the dynamic optimization problems. Using JModelica, the Modelica and Optimica models are transferred to a symbolic representation and made accessible in Python. Then CasADi is used for performing the discretization of the model equations, automatic differentiation, the calculation of first-order sensitivities, and for generating the Jacobian matrix required by the optimizer as well as for solving the discretized DAEs. The simulation using PRBS, transformation to bilevel-I and bilevel-II problems and other important symbolic manipulations are made automatically. Finally, IPOPT [100] is used for solving the NLP problems, e.g., bilevel-I and bilevel-II problems.

In summary, the developed CMSC framework has the following important features: (i) user-friendly interface (ii) suitable for the rapid prototyping (iii) extensible (iv) general and (v) parallel processing.

6.3 Experiment: optimal control of the large-scale nonlinear system

For experimentation a large-scale dynamic optimization model is considered that is related with a distillation column and it is available under JModelica. The original model was developed by Diehl [33] and an extended version was coded in Modelica based on the work of Hedengren [46]. The distillation column has 40 trays for separating a mixture of methanol and n-propanol. This DAE model contains 125 differential states (molar vapor flux, temperature, liquid mole fractions for each tray, a reboiler, and a condenser), 1000 algebraic variables, and 2 control signals (volumetric reflux flow and heat input). The problem formulation can be found in the work of Cai [22]. Using an a priori simulation with PRBSs as the control profiles, the results show that the angle between the two control variables is 116.19°. Thus, the two controls are weakly correlated, and therefore, the BFGS method is expected to achieve efficient computations.

This problem is solved again by CMSC approach with the BFGS method and by the serial algorithm in the context of the collocation method available in JModelica by taking 60 time intervals. As the number of processors for parallel computing is increased, the total computation time decreases. As compared to pure collocation method, the CMSC approach takes less computation time when the number of processors is larger than two. A factor-of-4 speedup is achieved when solving this problem with 10 processors. The corresponding computation time is equal to ca. 227 seconds. Note that this speedup factor represents only the part that can be parallelized, that is, without including the time spent by IPOPT. However, when using a single processor, the time required for the solution by the proposed approach is much higher, since, unlike for the collocation method, model equations and sensitivities have to be solved in each NLP iteration. For instance, the CMSC approach takes almost 600 seconds, when the pure collocation method solves the resulted NLP problem in approximately 370 seconds. Therefore, it can be concluded that the proposed approach is suitable for solving large-scale problems using the parallel-computing strategy [61].

Chapter 7

Nonlinear model predictive control for autonomous driving

The presented CMSC method as well as the control-variable correlation analysis and the bilevel approach play significant role in designing and implementing the control framework for autonomous driving. However, autonomous vehicle should be capable to compute its orientation and position in global and local coordinate systems in order to supply this information to the underlying controller, i.e., to properly control speed and to compute steering strategy. The vehicle's position plays significant role while performing challenging driving tasks such as obstacle avoidance, driving in intersections, conducting overtake maneuver, parking, etc. However, the position of the vehicle cannot be computed exactly due to several reasons: (i) vehicle is not equipped with precise sensors because of their high costs (ii) restricted and uncertain sensor information (iii) computational demand. Fortunately, there is a method to provide data-filtering, i.e., state estimation in real-time.

7.1 Kalman filter

As it was already mentioned, the sensors have relative noisy data and therefore the data cannot be directly applied to the underlying controller. In 1960, R.E. Kalman published his famous paper describing a recursive solution to the discrete-data linear filtering problem. Since that time, due in large part to advances in digital computing, the Kalman filter has been the subject of extensive research and application, particularly in the area of autonomous or assisted navigation. The Kalman filter is a set of mathematical equations that provides an efficient state estimation of the process states in a way that minimizes the mean of the squared error. The filter is very powerful in several aspects: it supports estimations of past, present, and even future states, and it can do so even when the precise nature of the modeled system is unknown.

The importance of the Kalman filter may be highlighted from two sides. On the one hand, the unmeasurable or noisy signals (i.e., state variables) can provide valuable information about a physical process. Therefore, improved supervision of the process can be gained. On the other hand, the more information the controller has about the process it controls, the better (i.e., more accurate) it can control it. Moreover, state estimators can be practical or economical alternatives to real measurements, since some of the states cannot be even measured or the corresponding hardware is too expensive.

From the theoretical point of view, there is a necessary condition for the Kalman filter to work correctly. The system, for which the states are to be estimated, should be observable. Observability for the discrete-time systems can be defined as follows. The discrete-time linear system

$$x(k+1) = A \cdot x(k) + B \cdot u(k), \tag{7.1}$$
$$y(k) = C \cdot x(k) + D \cdot u(k), \tag{7.2}$$

is observable if there is a finite number of time steps k so that the knowledge about the input sequence $u(0), \cdots, u(k-1)$ and the output sequence $y(0), \cdots, y(k-1)$ is sufficient to determine the initial state of the system, $x(0)$. Formally, if the observability matrix M_o has the full rank (rank is equal to n, where n is the order of the system model), the system is observable. The rank can be checked by calculating the determinant of M_o. Non-observability has several consequences: (i) the transfer function from the input variable u to the output variable y has an order that is less than the number of state variables (ii) there are state variables or linear combinations of state variables that do not show any response (iii) the steady-state value of the Kalman filter gain cannot be computed; therefore, the state estimation cannot be computed. Naturally, the Kalman filter is a state estimator which produces an optimal estimate in the sense that the mean value $E[e_x(k) \cdot e_x^T(k)]$ of the estimation errors is minimized, where $e_x(k) = x_{est}(k) - x(k)$ is the estimation error vector. The Kalman filter estimate is sometimes denoted the least mean-square estimate. It is assumed that the system for which the states are to be estimated is excited by random disturbances, i.e., process noise, and that the measurements contain white measurement noise. The Kalman filter algorithm was originally developed for systems assumed to be represented with a linear state-space model equations. However, in many applications the system model is nonlinear. Nevertheless, the linear model is just a special case of a nonlinear model. The Kalman filter for nonlinear models is denoted as the extended Kalman filter (EKF).

The principal working scheme of the Kalman filter may be grouped into two parts: time and measurement update equations. The time update equations are responsible for projecting forward in time the current state and error covariance estimates to obtain the a priori estimates for the next time step. The measurement update equations are responsible for the feedback, i.e., for incorporating a new measurement into the a priori estimate to obtain an improved a posteriori estimate. The time update equations can also be thought of as predictor equations, while the measurement update equations can be thought of as corrector equations. The Kalman filter addresses the general problem of trying to estimate the state of a discrete-time controlled process that is governed by the linear stochastic difference equation

$$x(k+1) = A \cdot x(k) + B \cdot u(k) + w(k), \tag{7.3}$$

with the measurement

$$y(k+1) = H \cdot x(k) + v(k), \tag{7.4}$$

where H denotes measurement matrix. The random variables $w(k)$ and $v(k)$ represent the process and measurement noise, respectively. They are assumed to be independent of each other, white-noise-like, and with normal probability distributions. In practice, the process noise covariance and measurement noise covariance matrices might change with each time step or measurement, however in this work they assumed to be constant.

Without loss of generality, the vector $\hat{x}_k^- \in \Re^n$ denotes a priori state estimate at step k of the process prior to step k, and $\hat{x}_k \in \Re^n$ is a posteriori state estimate at step k given measurement y_k. Thus, a priori e_k^- and a posteriori e_k estimate errors are defined as

$$e_k^- = x_k - \hat{x}_k^-, \tag{7.5}$$
$$e_k = x_k - \hat{x}_k. \tag{7.6}$$

Defining a priori estimate error covariance P_k^- and a posteriori estimate error covariance P_k as

$$P_k^- = E[e_k^- \cdot e_k^{-T}], \tag{7.7}$$
$$P_k = E[e_k \cdot e_k^T], \tag{7.8}$$

the Kalman filter algorithm can be summarized as follows

$$\hat{x}_k^- = A \cdot \hat{x}_{k-1} + B \cdot u_{k-1}, \tag{7.9}$$
$$P_k^- = A \cdot P_{k-1} \cdot A^T + Q, \tag{7.10}$$
$$K_k = P_k^- \cdot H^T \cdot \left(H \cdot P_k^- \cdot H^T + R\right)^{-1}, \tag{7.11}$$
$$\hat{x}_k = \hat{x}_k^- + K_k \cdot \left(z_k - H \cdot \hat{x}_k^-\right), \tag{7.12}$$
$$P_k = (I - K_k \cdot H) \cdot P_k^-. \tag{7.13}$$

In the algorithm (7.9)-(7.13), the first task during the measurement update is to compute the Kalman gain matrix K_k. The next step is to actually measure the process to obtain z_k, and then to generate an a posteriori state estimate by incorporating the measurement. The final step is to obtain an a posteriori error covariance estimate. After each time and measurement update pair, the process is repeated with the previous a posteriori estimates used to project or predict the new a priori estimates.

In the actual implementation of the filter, the measurement noise covariance R is usually measured prior to operation of the filter. In general, obtaining the measurement covariance R is practically possible. However, the determination of the process noise covariance Q is more difficult since it is impossible to directly observe a system under estimation. Sometimes a relatively simple (poor) process model can produce acceptable results if one injects enough uncertainty into the process via the selection of Q. However, in this case one would hope that the process measurements are reliable. Anyway, the good performance of the estimation process my be obtained by tuning both R and Q matrices. For instance, the larger Q values the stronger measurement-based updating of the state estimates. In this work, both matrices are tuned and assumed

to be constant during the tests. Consequently, the estimation error covariance P_k and the Kalman gain K_k will stabilize quickly and then remain constant.

As described above, the Kalman filter addresses the general problem of trying to estimate the state of a discrete-time controlled process that is governed by a linear stochastic difference equation. However, the model for the autonomous vehicle is nonlinear, which is governed by the nonlinear stochastic difference equation in general form

$$x_k = f(x_{k-1}, u_{k-1}, w_{k-1}), \tag{7.14}$$
$$z_k = h(x_k, v_k), \tag{7.15}$$

where the random variables w_k and v_k again represent the process and measurement noise as previously. In practice the individual values of the noise w_k and v_k at each time step are unknown. However, one can approximate the state and measurement vector by assuming both noises to be zero

$$\tilde{x}_k = f(\tilde{x}_{k-1}, u_{k-1}, 0), \tag{7.16}$$
$$\tilde{z}_k = h(\tilde{x}_k, 0), \tag{7.17}$$

where $\tilde{x}_k$ is some a posteriori estimate of the state, from a previous time step k. To estimate a process with nonlinear character, equations (7.14) and (7.15) are linearized

$$x_k = \tilde{x}_k + A_k(x_{k-1} - \tilde{x}_{k-1}) + W_k \cdot w_{k-1}, \tag{7.18}$$
$$z_k = \tilde{z}_k + H_k(x_{k-1} - \tilde{x}_{k-1}) + V_k \cdot v_{k-1}, \tag{7.19}$$

where x_k and z_k are the actual state and measurement vectors, $\tilde{x}_k$ and $\tilde{z}_k$ are the approximate state and measurement vectors, with Jacobian matrices

$$A_k = \frac{\partial f}{\partial x}(\hat{x}_{k-1}, u_{k-1}, 0), \quad W_k = \tfrac{\partial f}{\partial w}(\hat{x}_{k-1}, u_{k-1}, 0), \tag{7.20}$$
$$H_k = \frac{\partial h}{\partial x}(\tilde{x}_k, 0), \qquad V_k = \tfrac{\partial h}{\partial v}(\tilde{x}_k, 0). \tag{7.21}$$

Consequently, the time and measurement update equations in the case of EKF can be summarized as follows:

$$\hat{x}_k^- = f(\hat{x}_{k-1}, u_{k-1}, 0), \tag{7.22}$$
$$P_k^- = A_k \cdot P_{k-1} \cdot A_k^T + W_k \cdot Q_{k-1} \cdot W_k^T, \tag{7.23}$$
$$K_k = P_k^- \cdot H_k^T \cdot \left(H_k \cdot P_k^- \cdot H_k^T + V_k \cdot R \cdot V_k^T\right)^{-1}, \tag{7.24}$$
$$\hat{x}_k = \hat{x}_k^- + K_k \cdot \left(z_k - h(\hat{x}_k^-, u_{k-1}, 0)\right), \tag{7.25}$$
$$P_k = (I - K_k \cdot H_k) \cdot P_k^-. \tag{7.26}$$

In this work, the EKF is used in both test studies, which are described in the next subsections, aiming to demonstrate the importance of the state estimation for

the autonomous driving.

7.2 Obstacle detection and avoidance

This experiment deals with an optimal control of an autonomous mobile robot which is used for achieving a desired final position (landing problem) and simultaneously following the desired trajectory (tracking problem) in the presence of the unknown obstacles [34].

7.2.1 Mobile robot and mathematical model

The mobile robot SUMMIT is considered as an autonomous vehicle with high mobility and off-road capability. The robot's high maneuverability and thus also its small turning circle is made possible by the fact that the front and rear axles can be steered. The high off-road capability is achieved by the all-wheel drive. It is equipped with a symmetric two-axles counter steering system. The symmetric property means that the distances between both the front and the rear axle to the mass center of the rigid body are equal. The planar positioning is achieved by steering angle and driving velocity control of the four wheels. The wheels are manipulated axle-wise. The angular velocity of the rear wheels is measured by an encoder. It is assumed that the front wheels behave like the rear wheels because the control input is the same. One brushless DC motor per axle drives the wheels with no differential. Two servo motors, one at each axle, serve for adjusting the steering angle. The mobile robot is not equipped with a braking system. It stops by blocking the wheels. A Hokuyo laser scanner is used as a sensor for the detection of obstacles. The communication between the computer and the robot is realized by a TCP/IP protocol using a Wi-Fi network and the software Player/Stage is used as an interface for sending control signals and receiving sensor data. Player/Stage is a free software used for research in the field of robotics and sensor systems. A detailed execution and documentation of this software can be found in the Internet.

The mathematical model of motion equations is formulated as follows

$$\dot{x}_1(t) = \frac{c_\alpha \cdot \cos(u_2) \cdot (-2 \cdot x_1 \cdot \frac{x_3}{u_1} \cdot (l_h - l_v))}{m \cdot u_1 \cdot \cos(x_1)} - x_3, \tag{7.27}$$

$$\dot{x}_2(t) = x_3, \tag{7.28}$$

$$\dot{x}_3(t) = \frac{c_\alpha \cdot \cos(u_2) \cdot u_2 \cdot (l_h + l_v) + x_1 \cdot (l_h - l_v) - \frac{x_3}{u_1} \cdot (l_h^2 + l_v^2)}{J_I}, \tag{7.29}$$

$$\dot{x}_4(t) = u_1 \cdot \cos(x_1 + x_2), \tag{7.30}$$

$$\dot{x}_5(t) = u_1 \cdot \sin(x_1 + x_2), \tag{7.31}$$

where c_α is a lateral tire stiffness, l_h and l_v are the distances between the wheel contact point of a rear wheel or front wheel and the fixed plane of the robot, m is the mass of the robot and J_I is the rotational moment of inertia of the robot. The control

variables u_1 and u_2 are the velocity and the steering angle of the robot. The model contains five state variables: x_1 - sideslip angle, x_2 - yaw angle, x_3 - yaw rate, x_4 - longitudinal coordinate, x_5 - lateral coordinate. The parameters $m = 14.695$ [kg], $l_h = 0.1924$ [m], $l_v = 0.1776$ [m] were measured directly. However, the parameters $c_\alpha = 250$ [N/rad] and $J_I = 0.5024$ [Nm] were determined experimentally by solving the parameter estimation problem.

7.2.2 Obstacle description

The laser scanner installed on the robot was used for obstacle detection. It is an infrared laser with a wavelength of 785 [nm] with following parameters: detection range - 5600 [mm], resolution - 1 [mm], scanning angle - 240°, angle resolution - 0.352°, scanning time - 100 [ms/scan]. The scanning angle of the laser is 240° For obstacle detection, however, it is not necessary to look at the entire area, since only the obstacles ahead are important. For this reason, the scan range to be analyzed is limited to 120° in this work. The entire scanning area contains 682 points. Using the chosen scan angle, only the area between point 170 up to point 511 is of interest. This corresponds to 60° for each side of the robot. The determination of the center obstacle is given in the Figure 7.1, where x_{abs}^R and y_{abs}^R are the absolute coordinates of the robot, x_{abs}^H and y_{abs}^H are the absolute coordinates, x_{rel} and y_{rel} are the relative coordinates of the obstacle. Figure 7.1 also shows the yaw angle ξ of the robot. When the robot is turned to the left relative to the axis x_{abs}, the yaw angle is negative, i.e., $\xi = -\psi$. At a turn to the right the yaw angle is therefore positive, i.e., $\xi = \psi$. The calculation of the absolute coordinates of the obstacle is described using following equations

$$x_{abs}^H = x_{abs}^R + x_{rel} \cdot \cos(\xi) - y_{rel} \cdot \sin(\xi), \tag{7.32}$$
$$y_{abs}^H = y_{abs}^R + y_{rel} \cdot \cos(\xi) + x_{rel} \cdot \sin(\xi), \tag{7.33}$$

Using the center of the obstacle, it can be mathematically described as ellipse (7.34), while the elliptical shape was stretched along the main axis, where

$$1 \leq \frac{(x_{pos} - x_{abs}^H)^2}{a^2} + \frac{(y_{pos} - y_{abs}^H)^2}{b^2}. \tag{7.34}$$

The parameters a and b describe main and minor ellipse axis, respectively. Physically, due to (7.34), the robot position must not be within the ellipse. Based on the boundary points of the obstacle obtained by the laser, b is calculated by

$$b = (y_{rel}^{begin} - y_{rel}^{end}) + 2 \cdot \theta, \tag{7.35}$$

where θ denotes the safety area between robot and obstacle. The ellipse equation requires that a should be always greater than b. In this work, $a = 1.2$ [m] was selected, if $b < 1.2$ [m]. Otherwise, $a = 1.15 \cdot b$, depending on the size of the obstacle.

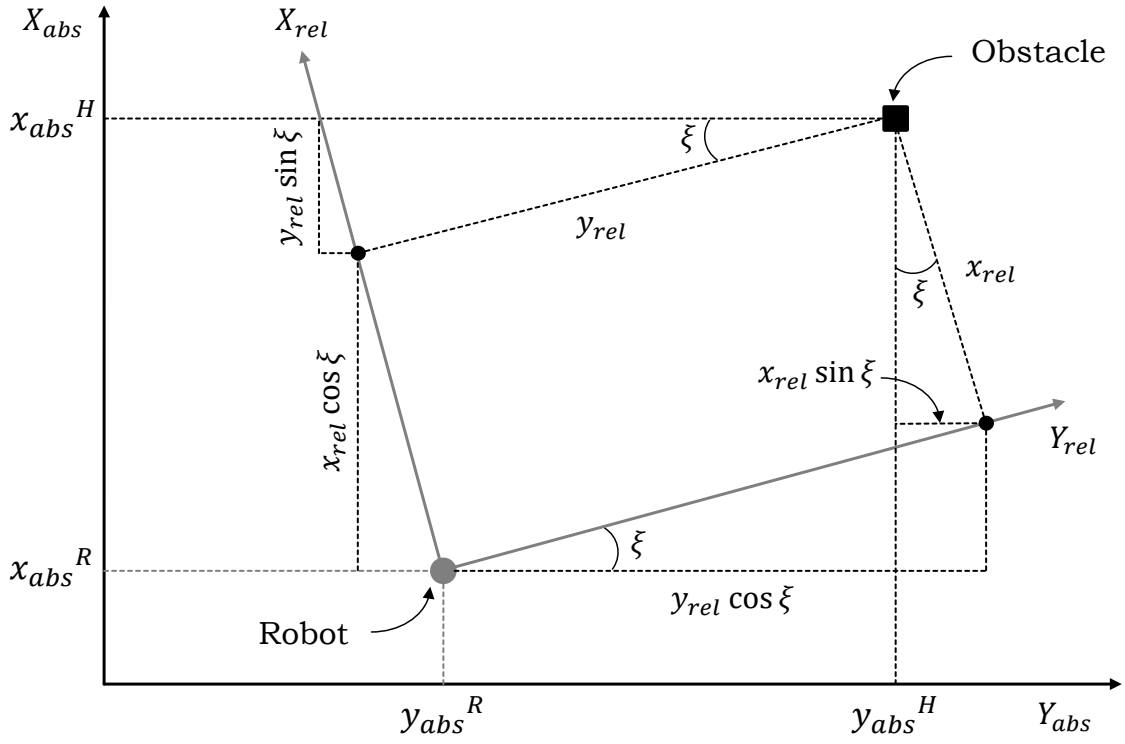

Figure 7.1: Determination of the absolute coordinates of the center of an obstacle

7.2.3 Pre-commissioning activities

During the test runs, it was found that the desired values of control variables transmitted to the robot (i.e., speed and steering angle) are not realized. Therefore, it is necessary to determine the interdependencies between desired and real values of steering angle as well as velocity.

According to the tests it was found that the real and desired velocities have linear relationships. The coefficient is equal to $k_v = 1.4$. For instance, if the desired velocity is equal to 1 [m/s], for internal controller, a signal which corresponds to the 1.4 [m/s] should be sent. This factor was determined by the time measurements between two fixed points of the coordinate plane, i.e., using the straight-forward movement of the mobile robot with fixed desired velocity.

However, the relationships between real and desired steering angle are nonlinear. In order to achieve the best results, the nonlinearity of the steering angle is approximated by a neural network with a single hidden layer, which contains six neurons. The number of neurons in the hidden layer is selected based on the accuracy of the approximate dependency between target and actual steering angles. The input data was collected based on the simple circle runs and the realized steering angle was computed using geometrical relations based on the length of the chord and the height of the circle segment above the chord.

Another important point for achieving the autonomy of the mobile robot is the

determination of its position. This problem was solved by employing the EKF. The diagonal entries of the covariance matrices Q and R characterize the yaw angle, longitudinal and lateral coordinates. The diagonal values in this matrices are selected as follows, $Q = [0.1, 0.1, 100]$ and $R = [1, 1, 10]$. The mutual influence of the state variables on each other during the measurements is not taken into account here. Therefore, the non-diagonal entries of the matrices Q and R are zeros. To evaluate the performance of the EKF the following test was conducted. The sine form change of the steering angle and constant velocity were transmitted to the low-level controller of the robot and the real and estimated trajectories were compared. However, there were relative large deviations between them.

This effect can be explained from the mechanical construction of the suspension system. The resulting steering angle is described by the following relationship: $\delta^{'} = u_2 \pm \nu$, where ν is the backlash (weakness of fastening between shaft and wheel) and $\delta^{'}$ represents the resulting steering angle of the robot. Therefore, in order to use the EKF, it is necessary that the corrected expectation value becomes zero. This is done by correcting the transmitted measured values, which is described below.

Assuming that the transmitted data from the mobile robot is correct, the following relationships can be formulated:

$$x_k = x_{k-1} + \Delta x_{k|k-1}, \tag{7.36}$$

$$y_k = y_{k-1} + \Delta y_{k|k-1}, \tag{7.37}$$

where x_k and y_k represent the robot position at discrete time k, x_{k-1} and y_{k-1} the robot position at discrete time $k-1$, $\Delta x_{k|k-1}$ and $\Delta y_{k|k-1}$ the distance covered by the robot during the interval $[k-1, k]$. However, the transmitted data is faulty, the coefficients of the measurement error along the longitudinal direction df and along the lateral direction ds are introduced. Both coefficients increase in time. During the test run, it was recognized that the increment in the longitudinal direction is negligible. However, the increment of the coefficient in the lateral direction cannot be neglected and can be computed as

$$ds_k = ds_{k-1} + dds \cdot \Delta t_{k|k-1}, \tag{7.38}$$

where ds_k is a value of the coefficient ds at time point k, ds_{k-1} is a value of the coefficient ds at the previous time point $k-1$, $\Delta t_{k|k-1}$ is the elapsed time between k and $k-1$ and dds is a gradient of ds. The coefficients $df = 0.01$ and $dds = 0.053$ were determined empirically by analyzing the recorded data from several test runs. Therefore, the following correction equations are added to the control framework.

$$\begin{aligned} x_k = {} & x_{k-1} + \Delta x_{k|k-1} + \Delta t_{k|k-1} \cdot ds_{k-1} \cdot v_{k|k-1} \cdot \sin(\psi_{k|k-1}) + \\ & + \Delta t_{k|k-1} \cdot df \cdot v_{k|k-1} \cdot \cos(\psi_{k|k-1}), \end{aligned} \tag{7.39}$$

$$\begin{aligned} y_k = {} & y_{k-1} + \Delta y_{k|k-1} + \Delta t_{k|k-1} \cdot ds_{k-1} \cdot v_{k|k-1} \cdot \cos(\psi_{k|k-1}) + \\ & + \Delta t_{k|k-1} \cdot df \cdot v_{k|k-1} \cdot \sin(\psi_{k|k-1}). \end{aligned} \tag{7.40}$$

After this modification, the maximum error between the measured and correct trajectory is about 0.08 [m]. Based on this value, it is concluded that the implemented EKF combined with adjustment equations (7.39) and (7.40) provides correct estimation of the mobile robot position, which plays a crucial role for avoiding the unknown obstacles.

The designed control framework is given in Figure 7.2.

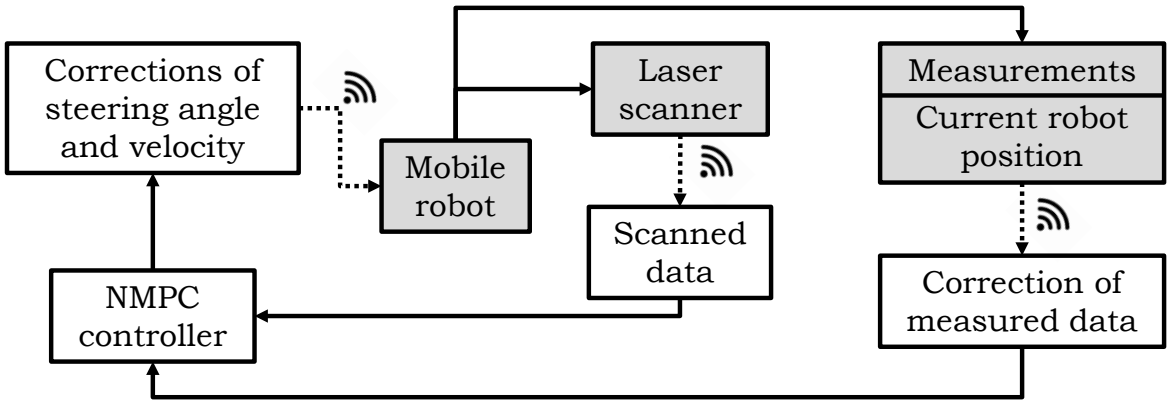

Figure 7.2: General control scheme of the robot SUMMIT

The description of the NMPC controller and real tests are given in the next subsection.

7.2.4 Experimental results using mobile robot

The NMPC controller involves the discretized mathematical model (7.27) - (7.31). Because of the robot mechanics, the control variables are constrained as

$$0.1 \leq u_1(t) \leq 1.0, \quad -0.06 \leq u_2(t) \leq 0.06, \tag{7.41}$$

where velocity $u_1(t)$ is given in [m/s] and steering angle $u_2(t)$ is measured in radians. It can be seen that the lower bound of the velocity is different from zero. This is done due to the model equations, since the velocity occurs as a divisor. However, if the robot achieves the desired position (or the estimated robot position is near several centimeters around the landing point), the NMPC controller will be shut down and the wheels will be blocked by a stop command. The state variables are also bounded as given below

$$-0.15 \leq x_1(t) \leq 0.15, \tag{7.42}$$
$$-\pi \leq x_2(t) \leq \pi, \tag{7.43}$$
$$-0.8 \leq x_3(t) \leq 0.8, \tag{7.44}$$
$$0 \leq x_4(t) \leq 22, \tag{7.45}$$
$$-1.5 \leq x_5(t) \leq 1.5. \tag{7.46}$$

The bounds imposed on the longitudinal $x_4(t)$ and lateral $x_5(t)$ coordinates are due to the testing scenario. Before the real test will be provided, the model equations are firstly analyzed using the proposed control-variable correlation analysis and bilevel approach. It was found that there are no correlations between control variables. The angle between columns in the sensitivity matrix is equal to 95.46°. Therefore, the BFGS approximation of the Hessian matrix can be used without any drawbacks in the computational performance. The bilevel approach was applied by taking into account the time horizon with fixed final time $t_f = 2.72$ [s]. The desired approximation tolerance ϵ is equal to $2.5 \cdot 10^{-4}$. On the one hand, by solving the bilevel-I problem only $N = 6$ time intervals are needed, and the maximum approximation error is equal to 0.00023308. On the other hand, the solution of the bilevel-II problem indicates that the desired tolerance can be achieved only if $N = 16$ time intervals are used for discretization. The corresponding maximum approximation error is equal to 0.00022176. Therefore, this obtained minimum number of time intervals is used for the practical test. The performance index incorporates several factors: (i) the exactness of reaching the final position (ii) the deviation of the longitudinal and lateral coordinates from the desired direction and (iii) the control effort. The objective function J is formulated as

$$J = P_x \cdot (x_{pos,N} - x_{end,N})^2 + P_y \cdot (y_{pos,N} - y_{end,N})^2 + \\ + \sum_{k=0}^{N-1} \left[q_x \cdot (x_{pos,k} - x_{end,k})^2 + q_y \cdot (y_{pos,k} - y_{end,k})^2 + r_v \cdot u_{1,k}^2 + r_\delta \cdot u_{2,k}^2 \right] \quad (7.47)$$

involving following weight coefficients $P_x = 1, P_y = 10, q_x = 1, q_y = 5, r_v = r_\delta = 0.5$.

The resulting NLP problem contains $N = 16$ time intervals (i.e., $\Delta t = 0.17$). The problem was solved using the IPOPT optimizer installed on a notebook with an Intel Core i3-2350CPU@2.3 GHz x 4, 64 bit Ubuntu 12.04 LTS Linux. Because of the mechanics, it was found to use additional velocity constraints in the first three time intervals, i.e., the maximum velocity is restricted to 0.4 [m/s], 0.7 [m/s], 0.9 [m/s]. The required sensitivities were computed with the help of Eigen library (see: *www.eigen.tuxfamily.org*). The state trajectories inside each time interval were computed using own programmed Newton method. The tolerance in IPOPT is equal to 10^{-4} due to the robot mechanics. The obtained results are given in Figure 7.3. It can be seen that the robot has no collision with unknown obstacles.

The computation time is given in Figure 7.4. It can be seen that using the CMSC approach the real-time control can be guaranteed using the NMPC scheme. The computation time slightly rises in front of each obstacle, since more time is needed to compute the optimal trajectory. The mean value of the computation time is equal to ca. 12 [ms], which means that BFGS approximation can be used in the NMPC framework, as was predicted by the control-variable correlation analysis.

As indicated in Figure 7.5 before each obstacle (red circles) the robot velocity reduces almost to the minimum value. This effect is expected due to the secured areas around the obstacles, mechanical capabilities of the mobile robot, and used objective function. Using the obtained optimal control profiles the maximum approximation

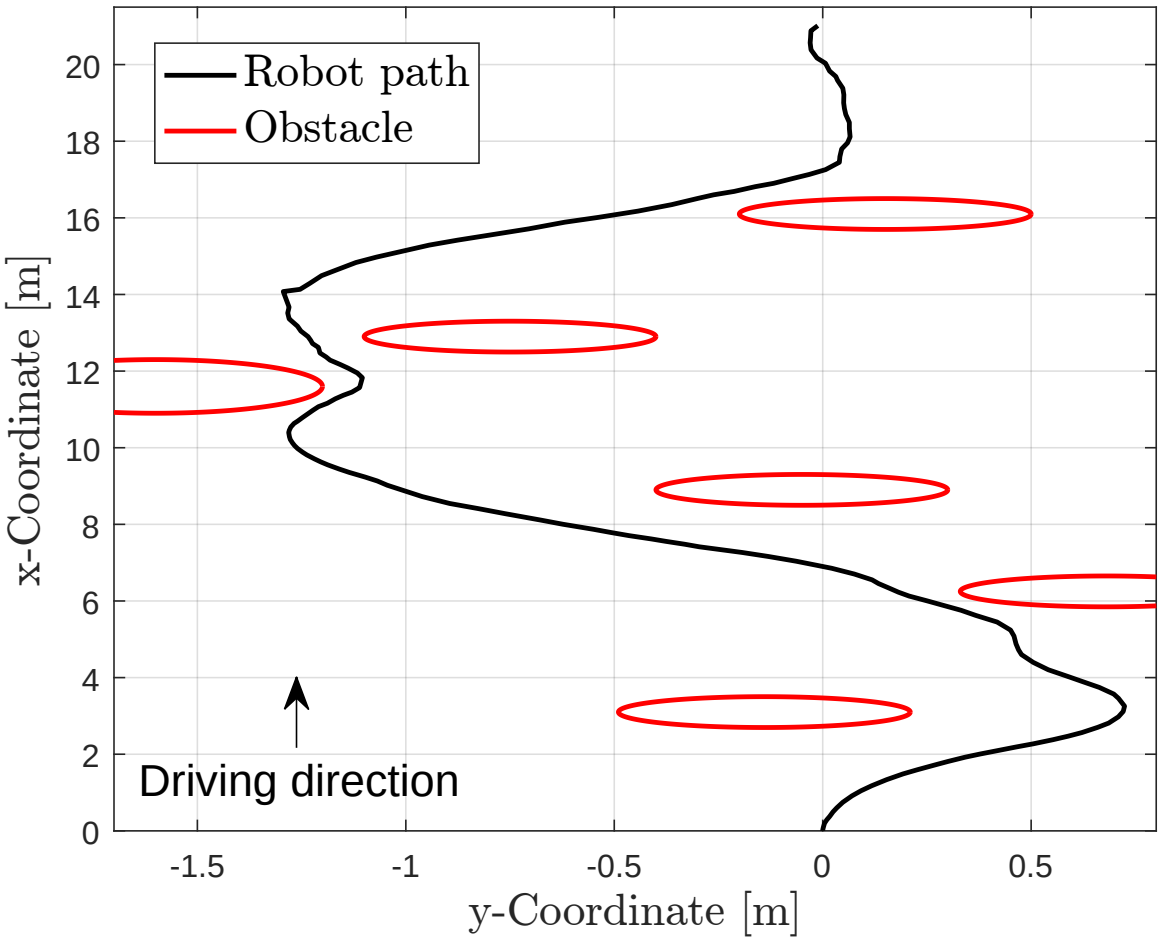

Figure 7.3: Obstacle avoidance - trajectory

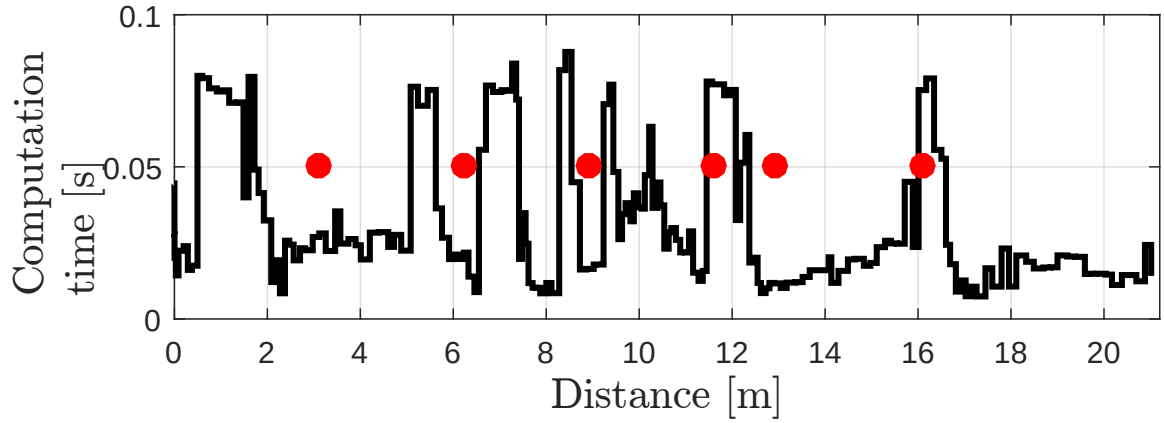

Figure 7.4: Obstacle avoidance - computation time

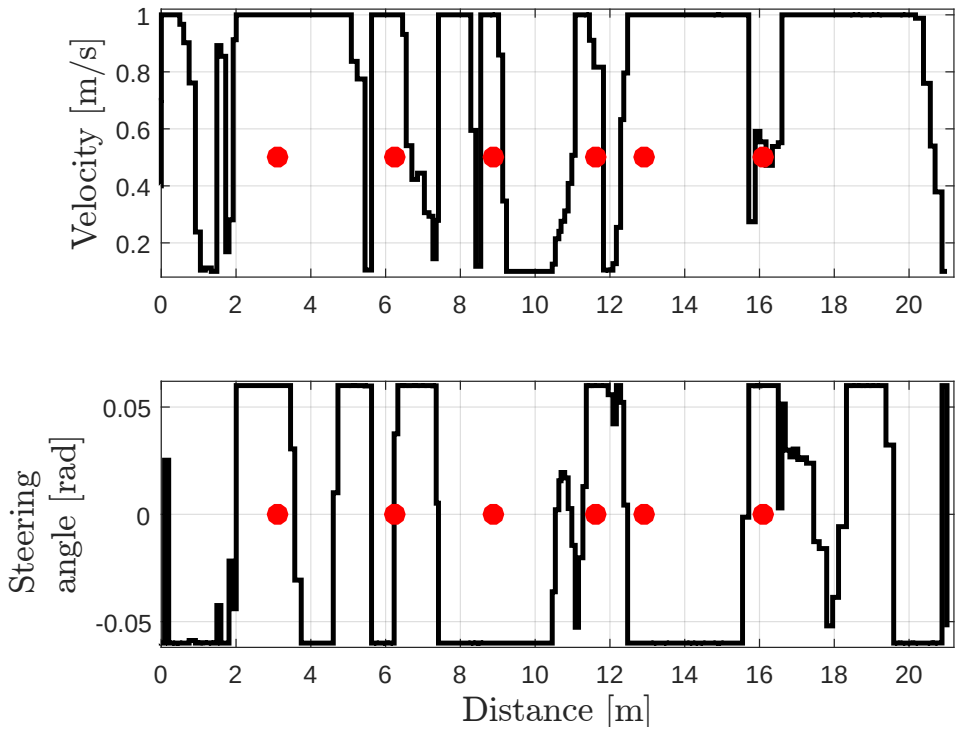

Figure 7.5: Obstacle avoidance - control trajectories

error was computed. It is equal to $1.425 \cdot 10^{-5}$, which is below the desired approximation tolerance. Thus, the proposed bilevel approach can correctly determine the minimum number of time intervals.

7.3 City driving scenario

7.3.1 Vehicle description and control framework design

The object under investigation is the Audi Q2 model vehicle (scale 1:8). In comparison to the mobile robot, the prescribed development framework is ADTF (**A**utomotive **D**ata and **T**ime-triggered **F**ramework). The Audi Q2 model allows more flexible manipulations than the mobile robot SUMMIT. These features are reflected in user-specified adjustment of low-level regulators, possibilities to get raw-data from sensors, and implementation of additional components for numerical computations. The vehicle is equipped with a set of ultrasonic sensors, camera, speed wheel sensor, and miniTX board. This includes an Intel Core i3 processor, an 8 GB RAM, a fast 128 GB M.2 SSD hard drive and an NIVIDIA GeForce GTX1050Ti graphics card. In addition to two gigabit Ethernet ports, the board also has several USB3.0 interfaces and a USB-C port. Furthermore, a Bluetooth and a WLAN module (IEEE 802.11ac) is available. In Figure 7.6 the developed framework for real-time control is presented.

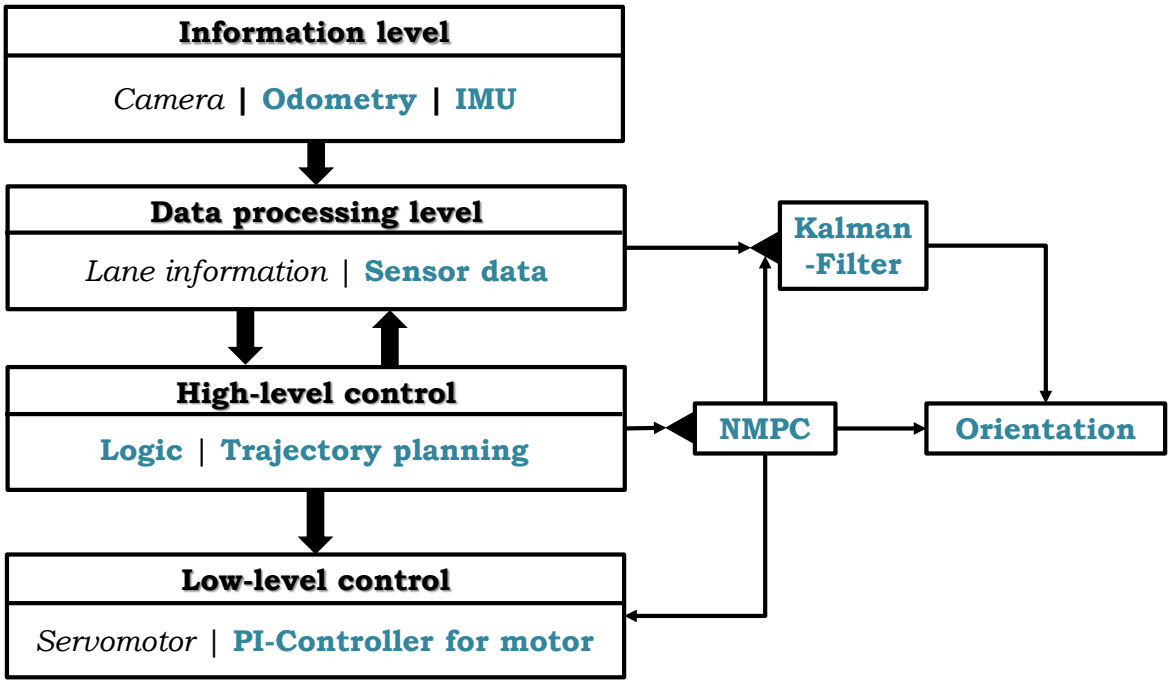

Figure 7.6: Developed framework for Audi Q2 model vehicle

The whole control framework is separated into four major levels. In the information level all required data from sensors and hardware is collected and supplied to the data processing level, where it will be processed. The lane information is obtained using image processing techniques. Note that the information about lane profile is supposed to be available and the low-level controller for the servomotor, which is used for steering, is adjusted. Therefore, the implementation aspects of these parts are out of the scope of this thesis. In addition, the Kalman-Filter is also implemented in this level. The implementation of Kalman-Filter allows to know the vehicle position on the coordinate plane in every sampling instance. The vehicle orientation includes longitudinal and lateral coordinates as well as heading angle. Based on the driving tests the covariance matrices for the process and observation noise are equal to $Q = 10$ and $R = 15$, respectively. The initial value for the estimated error covariance is equal to $P_0 = 0.5$. In the high-level control the logic unit and the trajectory planing are implemented. The trajectory planning is combined with NMPC controller. The controller receives the preplanned trajectories if required. However, in the lane following mode the NMPC controller computes new control sequences always in local vehicle coordinates. The logic unit is responsible for overall system monitoring, e.g., switching the mode of the NMPC controller. In the low level control a PI-controller is designed to control the vehicle velocity. To design the proper coefficients of the proportional G_P and integral G_I gain coefficients, several tests using unit step function were performed. Based on the obtained results the following coefficients were selected $G_P = 20$ and $G_I = 20$. The regulator for the steering angle is already available and

further modifications are not required.

7.3.2 Preliminary analysis

In this section the main focus of the experiments is to investigate the main feature of the autonomous driving, namely lane keeping assistant system. This model vehicle was employed for the Audi Autonomous Driving Cup 2017 competition.

The mathematical model for this vehicle is formulated as follows

$$\dot{x}(t) = v(t) \cdot \cos(\psi(t) + \beta(t)) \tag{7.48}$$

$$\dot{y}(t) = v(t) \cdot \sin(\psi(t) + \beta(t)) \tag{7.49}$$

$$\dot{\psi}(t) = \frac{v(t)}{l_f + l_r} \cdot \tan(\delta(t)) \tag{7.50}$$

$$\beta \approx \frac{l_r}{l_f + l_r} \cdot \delta(t) \tag{7.51}$$

where $\beta(t)$ denotes sideslip angle [rad], $\psi(t)$ is heading angle [rad], $v(t)$ - velocity [m/s], $\delta(t)$ - steering angle [rad], $x(t)$ and $y(t)$ are longitudinal and lateral coordinates in [m], l_f and l_r in [m] are distances between center of gravity and front / rear axes, respectively. This model mimics the main state and control variables, which are important to achieve the goal.

Before the adjustment of the lane-keeping assistant system, the vehicle model (7.48)-(7.51) will be firstly analyzed using control-variable correlation analysis, to check which Hessian matrix (analytical or approximated) should be used and how many time intervals should be involved for discretization.

According to the control-variable correlation analysis, it was found that control variables $v(t)$ and $\delta(t)$ are uncorrelated, i.e., the angle between columns in the sensitivity matrix is equal to 99.361°. It means that for this model the Hessian approximation using BFGS method will be enough to guarantee fast convergence rate.

For the analysis of the discretization a time horizon should be selected and fixed. Because of the constraints imposed on the vehicle velocity, i.e., $0.4 \leq v(t) \leq 1.0$ [m/s], the time horizon is equal to $0 \leq t \leq t_f$, where $t_f = 1.2$ [s]. This value can guarantee an appropriate prediction length and can balance between prediction, computational load, and spontaneous change of the lane shape. Another important motivation for such prediction length is that the vehicle model defined above is simple and cannot capture the important longitudinal and lateral vehicle dynamics. Because of the bilevel approach, described in Chapter 5, the tolerance of the approximation should be selected. Since the vehicle model is represented as point motion of the center of gravity, the approximation error should be low enough, e.g., $10^{-5} \leq \varepsilon \leq 10^{-7}$, to avoid the effects of, e.g., inertia, vehicle parameters, and unpredictable disturbances. The results of the bilevel approach are summarized in Table 7.1, where all three noncollocation points are investigated. In Chapter 5 the bilevel approach was provided only for the second noncollocation point $\hat{t}^2$, which lies between first and second collocation point. Here, the first $\hat{t}^1$ and third $\hat{t}^3$ noncollocation points are also investigated to analyze the

approximation error, whose selection is also based on the same principle as in Chapter 5. The analysis in Table 7.1 was conducted under following additional operation

Table 7.1: Obtained results using bilevel-II approach

	$\hat{t}^1$		$\hat{t}^2$		$\hat{t}^3$	
ε	N	e_{max}	N	e_{max}	N	e_{max}
$5 \cdot 10^{-5}$	4	$2.328 \cdot 10^{-5}$	3	$3.540 \cdot 10^{-6}$	3	$3.812 \cdot 10^{-5}$
$1 \cdot 10^{-6}$	9	$9.096 \cdot 10^{-7}$	4	$8.414 \cdot 10^{-7}$	8	$7.516 \cdot 10^{-7}$
$3 \cdot 10^{-7}$	12	$2.882 \cdot 10^{-7}$	5	$2.759 \cdot 10^{-7}$	11	$1.972 \cdot 10^{-7}$

conditions $x(t_0) = 4.5821$ and $y(t_0) = 3.6539$, as well as imposed constraints on these state variables, i.e., $1.108 \leq x(t) \leq 4.582$ and $2.994 \leq y(t) \leq 3.874$. The boundaries imposed on longitudinal and lateral coordinates describe the allowed movement of the vehicle inside this rectangle. Especially, the lateral constraints describe the road lanes, which must not be violated by the wheels. Because of the mechanical limitations, the steering angle is also restricted as $-0.46 \leq \delta(t) \leq 0.49$ radians. According to the results in Table 7.1 the number of required time intervals is dependent of the position of the noncollocation point. The largest impact is caused by the first noncollocation point. Therefore, for the practical implementation of the NMPC scheme, three experiments are conducted, i.e., using 4, 9, and 12 time intervals.

For demonstration purposes the slalom maneuver was selected. In the critical traffic situations during the city driving cycle the similar driving behavior can occur. To describe such kind of behavior the Bézier curves have been employed using Bernstein polynomials, which have many advantages. These curves are simple, differentiable, and dependent only from time and control points, which are used to describe the shape of the curve. Here, three connected cubic Bézier curves are used with the following four control points, defined in Table 7.2, where the control points are described pairwise for longitudinal and lateral coordinates.

Table 7.2: Bézier curves for slalom maneuver

Control point	1^{st} Bézier curve	2^{nd} Bézier curve	3^{d} Bézier curve
1	$(4.5821, 3.6539)$	$(3.3821, 3.2139)$	$(2.3821, 3.6539)$
2	$(3.9821, 3.6539)$	$(2.8821, 3.2139)$	$(1.7821, 3.6539)$
3	$(3.9821, 3.2139)$	$(2.8821, 3.6539)$	$(1.7821, 3.2139)$
3	$(3.3821, 3.2139)$	$(2.3821, 3.6539)$	$(1.2821, 3.2139)$

It can be seen that starting point of the second trajectory is the same as the last point of the first trajectory. The same relations hold true between third and second trajectories. For the practical experiment all three curves are combined together and discretized using 70 points applying equidistant distance between neighboring discrete points. The number of discrete points for trajectory tracking is chosen empirically.

Consequently, the NMPC problem is formulated as follows

$$\min_{\mathbf{x}_p,\mathbf{v},\delta} \quad \sum_{i=0}^{N} Q_x \cdot (x_{r,i} - x_{p,i}^{(1)})^2 + Q_y \cdot (y_{r,i} - x_{p,i}^{(2)})^2 + \sum_{j=0}^{N-1} R_d \cdot \delta_i^2 + \sum_{j=0}^{N-2} R_c \cdot \frac{(\delta_{j+1} - \delta_j)^2}{\Delta t} \tag{7.52}$$

$$\begin{aligned} \text{subject to:} \quad & \mathbf{x}_{p,0} - \mathbf{x}_0 = 0, \\ & \mathbf{x}_{p,i} = \hat{\mathbf{x}}_i \left(\mathbf{x}_{p,i-1}, v_{i-1}, \delta_{i-1}\right), \ i = 1, \ldots, N, \\ & t_0 \leq t \leq t_f, \ t_0 = 0, \ t_f = 1.2, \\ & 0.4 \leq v_i \leq 1.0, \ i = 1, \ldots, N, \\ & -0.46 \leq \delta_i \leq 0.49, \ i = 1, \ldots, N, \end{aligned} \tag{7.53}$$

where $Q_x = 10$ and $Q_y = 50$ denote weighting matrices for the longitudinal and lateral coordinates, $R_d = 1$ describes the weighting matrix for the energy minimization during steering activity, and $R_c = 0.1$ is responsible for controlling the change rate of the steering angle, $x_{p,i}^{(1)}$ and $x_{p,i}^{(2)}$ denote longitudinal and lateral coordinates, respectively. The large weighting matrix for lateral coordinates is necessary because the vehicle dimensions (width and length) are not included directly in the model equations.

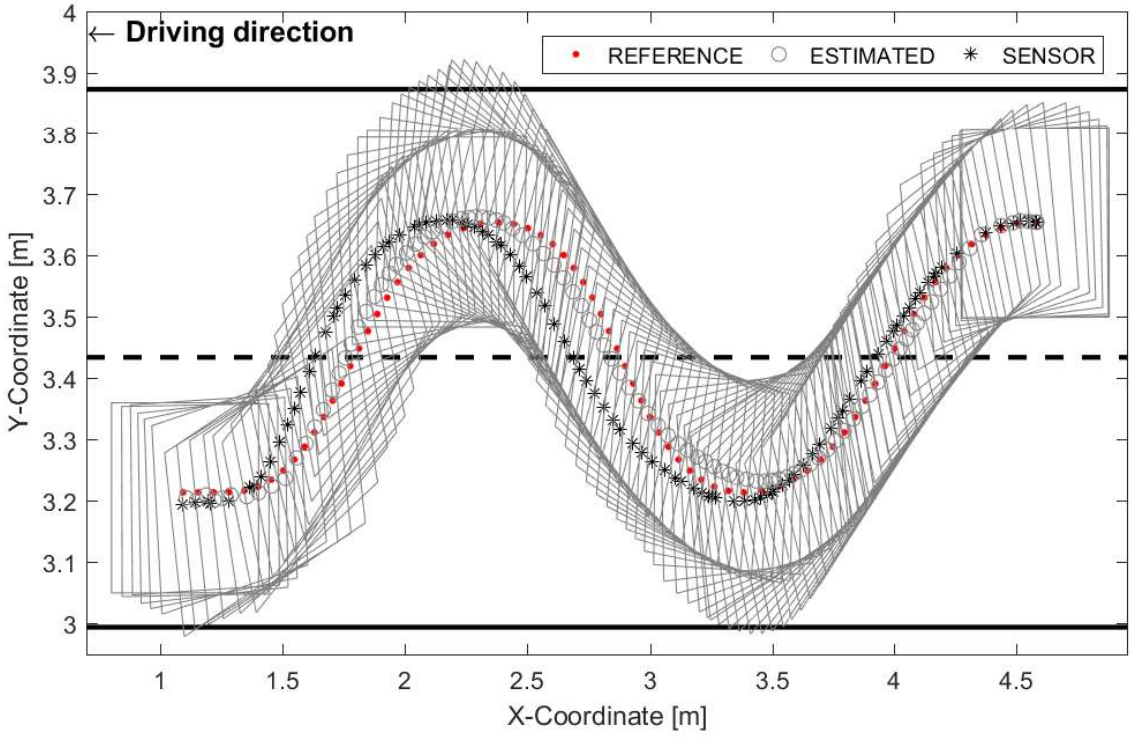

Figure 7.7: Slalom driving test using 12 time intervals

Using 4 and 9 time intervals for the discretization deliver unsatisfactory results of

the driving behavior. However, the maximum approximation error is guaranteed. In addition, the lane width is not also considered as a constraint by NMPC formulation. Therefore, the car body violates the lane restrictions more then 7 centimeters. This happens because the reference trajectory is complicated to be performed within the imposed physical constraints on the steering angle and the prediction horizon is short. On the other hand, using 12 time intervals the driving performance is good and the vehicle can perform safe and stable operations. The results are given in Figure 7.7.

The corresponding approximation error after the slalom experiment for all three noncollocation points are equal to $7.589 \cdot 10^{-9}$, $3.101 \cdot 10^{-11}$, $3.927 \cdot 10^{-9}$, which indicates the correctness of the bilevel approach. Moreover, the maximum violation of the lane width is equal to 1.8 centimeter and can be neglected. In addition, the state estimation using EKF indicates improved results in comparison with the data delivered by odometry.

7.3.3 Experimental results using vehicle model

Near the point $(X, Y) = (8, 4)$ in the coordinate plane (Figure 7.8) the vehicle performs a right turn maneuver. Since camera information is not available, the vehicle drives using an appropriate preplanned trajectory using the Bézier curve with objective function defined in problem (7.53) applying $Q_x = 10$ and $Q_y = 20$. Using the slightly modified objective function for the lane keeping, see equation (7.54), the experiment was conducted.

$$\begin{aligned} \min_{\mathbf{x}_p, \mathbf{v}, \delta} \quad & Q_{x_N} \cdot (x_{r,N} - x_{p,N}^{(1)})^2 + \sum_{i=0}^{N} Q_y \cdot (y_{r,i} - x_{p,i}^{(2)})^2 + \sum_{i=0}^{N-1} R_d \cdot \delta_i^2 \\ & + \sum_{j=0}^{N-2} R_c \cdot \frac{(\delta_{j+1} - \delta_j)^2}{\Delta t} + \sum_{j=0}^{N-2} R_v \cdot \frac{(v_{j+1} - v_j)^2}{\Delta t}, \end{aligned} \tag{7.54}$$

where $Q_{x_N} = 2$, $Q_y = 2$, $R_c = R_v = 0.1$. The results of the test are indicated in Figure 7.8, where the dotted line denotes the estimated state trajectory. The overall results demonstrates very stable performance of the implemented NMPC controller. The state estimation of the vehicle trajectory was computed with the help of EKF, where the measurement signal is the IMU sensor value of the heading angle. Because of the sensor capabilities it is hard to obtain better results. However, the obtained estimated vehicle trajectory violates the center line of the lane insignificantly. In Figure 7.9, the computation time for each MPC iteration is given. The mean value is 19 milliseconds. The computation time demonstrates the capability of the NMPC controller to operate in real-time. Since the NMPC controller involved in the complicated control framework, where all required computations are conducted using only CPU, the computation time is a little bit higher than the test using only simulations. In the simulative investigations the mean computation time is equal to 11 milliseconds. However, the real computation time may be further reduced using decomposition of the designed framework. This can be done if, e.g., the image processing algorithms

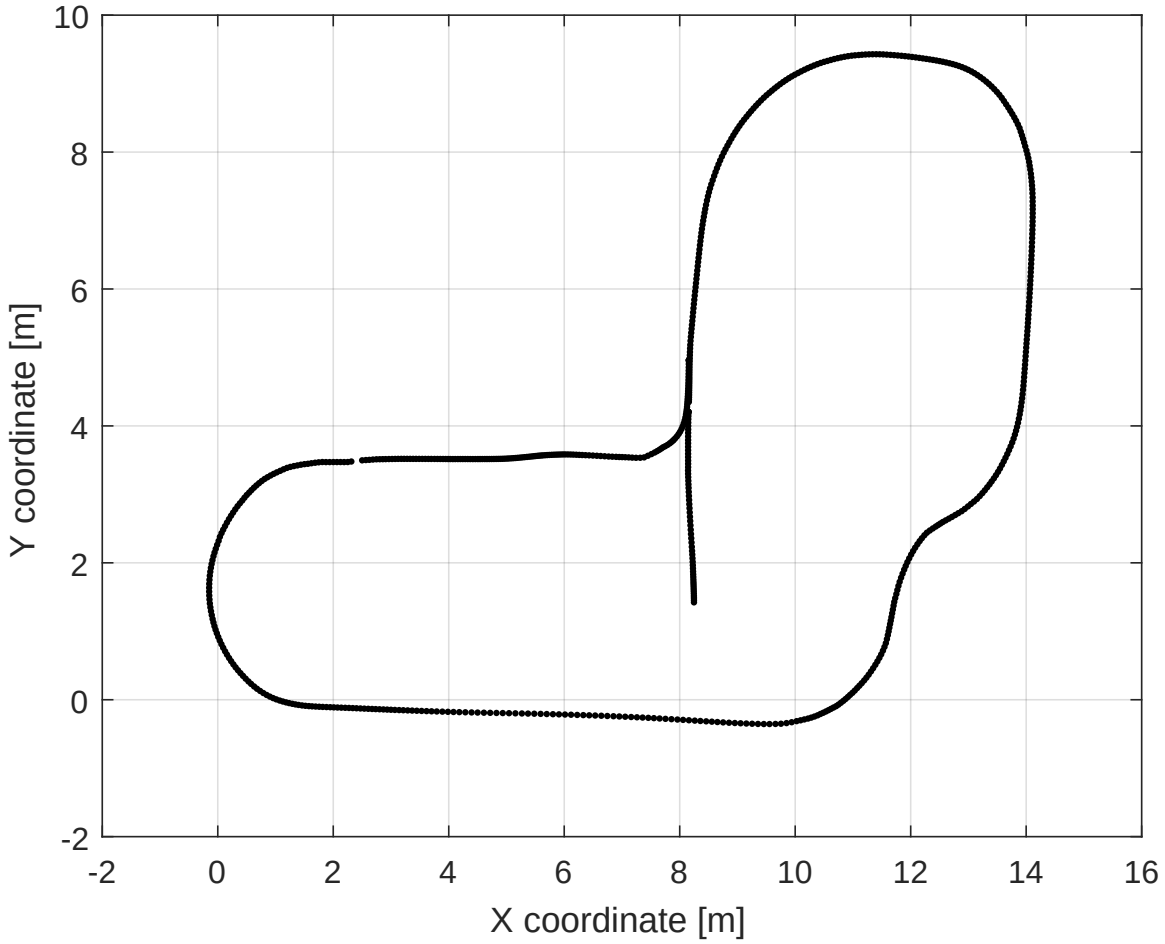

Figure 7.8: Lane keeping - driving route

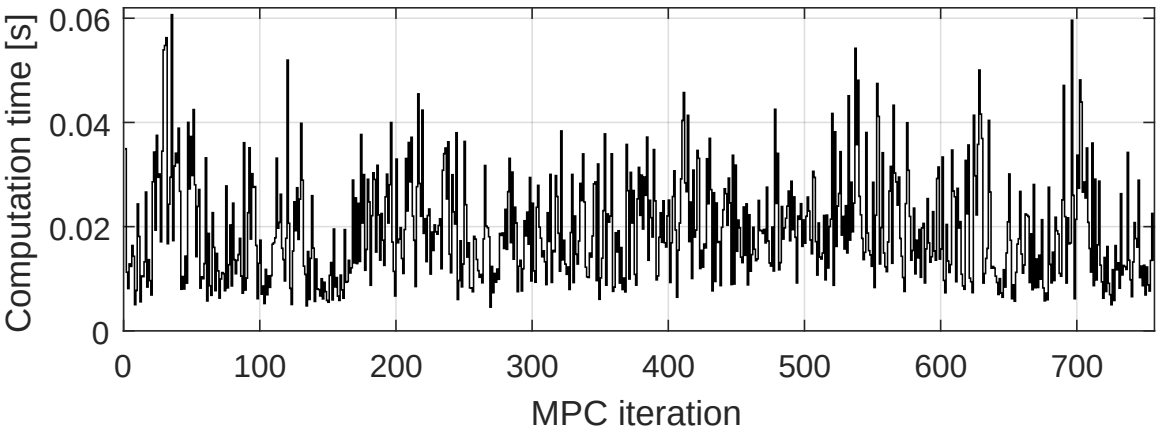

Figure 7.9: Lane keeping - computation time

will be implemented only in graphics processing unit and the NMPC framework with low-level controllers will be implemented in a CPU. However, this point is left pending further development.

The obtained control signals are given in Figures 7.10.

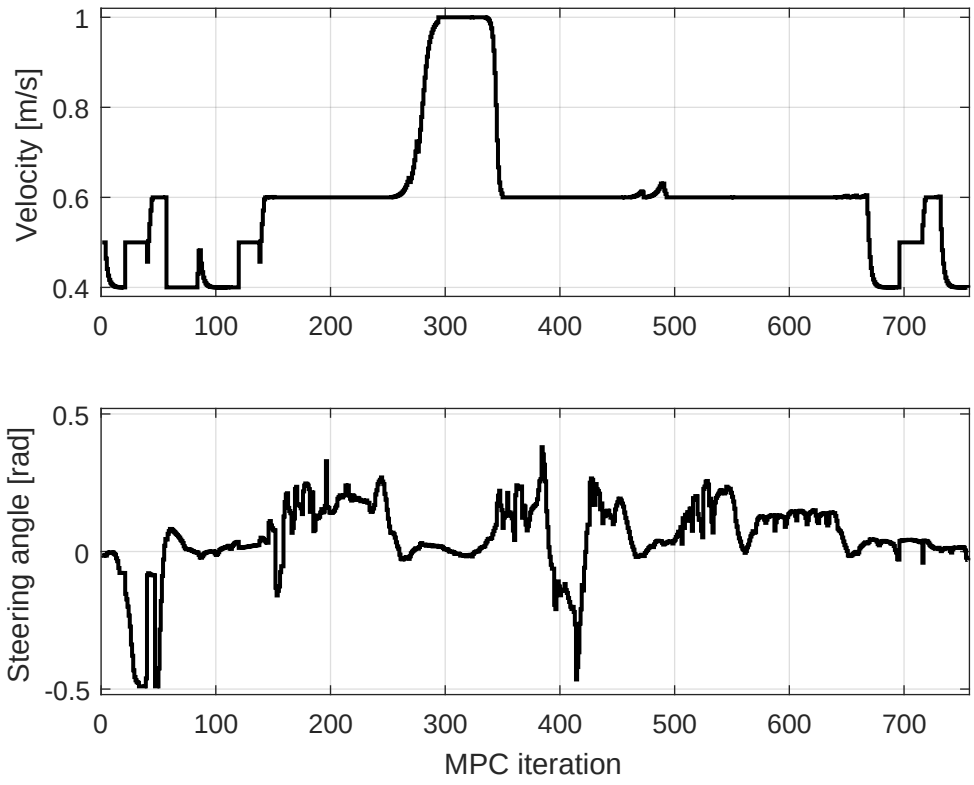

Figure 7.10: Lane keeping - optimal control profiles

The change of the velocity is caused by the lane shape. Nearly for 300 MPC iterations the vehicle drives along a straight line. Therefore, the vehicle can accelerate up to the upper limit of the allowed velocity. The fluctuations of the steering angle in the straight segments of the test track are due to the slightly unstable detection of the lane constraints. However, the fluctuations are insignificant and almost cannot be detected by the human eye. However, the NMPC controller accepts in every iteration the camera information and provides corresponding steering actions. Note that the vehicle drops off the initial values of the coordinates and heading angle to zero vector. It reduces the computational efforts. Nevertheless, in the background of the NMPC the state estimation for the global vehicle position is running. This is done to identify and plan the trajectory for the right turn properly. The approximation error obtained using bilevel-I approach and bilevel-II is guaranteed during the whole test run. The real approximation error of the state trajectories is below the predefined tolerance $\varepsilon = 3 \cdot 10^{-7}$. The insignificant violations of the lane lines was encountered during the test. The car body violates them around 1 centimeter as maximum. Therefore, the CMSC approach and developed theoretical methods for a priori analysis of the

dynamic optimization problems are suitable for autonomous driving.

Chapter 8

Conclusions and future research

8.1 Summary of contributions

An efficient solution approach with analytical second-order derivatives and parallel computing based on the CMSC method is proposed. To accelerate the solution process, an analytical Hessian for the CMSC method is derived. In addition, a novel approach for correlation analysis of control variables is proposed, that can be used to identify a priori the degree of difficulty for solving dynamic optimization problems. This analysis can easily be done by examining the angles of the columns in the sensitivity matrix through simulation for which the PRBS as control signal is proposed. A decision on whether to use a BFGS approximation or an analytical Hessian can be made based on the result of this analysis. Nevertheless, a more detailed theoretical investigation of this method is left for future research. Parallel computations in terms of time intervals are applied to reduce the computational costs. In comparison to previous algorithms, the parallel computation in this work is performed for the computations in individual time intervals on a stand-alone computer. The developed framework was tested on four dynamic optimization problems including a large-scale problem with more then 200000 variables after discretization. As a future work, the proposed framework will be made available as a web-based toolchain for wider public availability.

For solving the dynamic optimization problems the very first step is to discretize model equations over the fixed time horizon. The determination of the number of time intervals represents an important issue for efficient solution of dynamic optimization problems. However, there have been no comprehensive studies addressing this issue. This work presents an approach for a priori determination of a minimum number of time intervals based on error estimation of state trajectories. The proposed approach is a bilevel solution strategy, where the outer loop is responsible for finding a minimum number of time intervals and the inner loop solves an error maximization problem. The approximation errors are estimated at specific noncollocation points within the given time horizon. The found number of time intervals using the bilevel approach is sufficient to guarantee the accuracy of the state profiles of the original dynamic optimization problem. It is based on the properties of the inner problem, since the maximum approximation error is found using the optimization method. The simultaneous impact of control variables and initial conditions of state variables on the numerical error is investigated, so that the a priori determined number of time intervals are valid for varying operating conditions. In addition, the proposed bilevel strategy can facilitate efficient design of NMPC in the sense of balancing between accuracy, length of the prediction horizon, and computational burden. On the basis

of the resulting number of time intervals, one can identify in advance the numerical performance in solving the NMPC problem. The selection of control horizon based on the control performance is beyond the scope of this work. Without loss of generality, the number of time intervals for the control horizon can be the same as for the prediction horizon. To demonstrate the viability of the bilevel approach, two case studies are presented and it is shown that the a priori identified number of time intervals can guarantee the error tolerance when it is applied to solve the original problem.

In addition, the theoretically developed methods were integrated in the general purpose framework, which is also designed within this work. The major advantages of this framework are (i) problem-independent solution interface, which relies on the symbolic problem transformation (ii) simple extensibility of the framework (iii) parallel computing approach for reduction of the computational costs (iv) enhanced suitability for facilitating rapid-prototyping procedure.

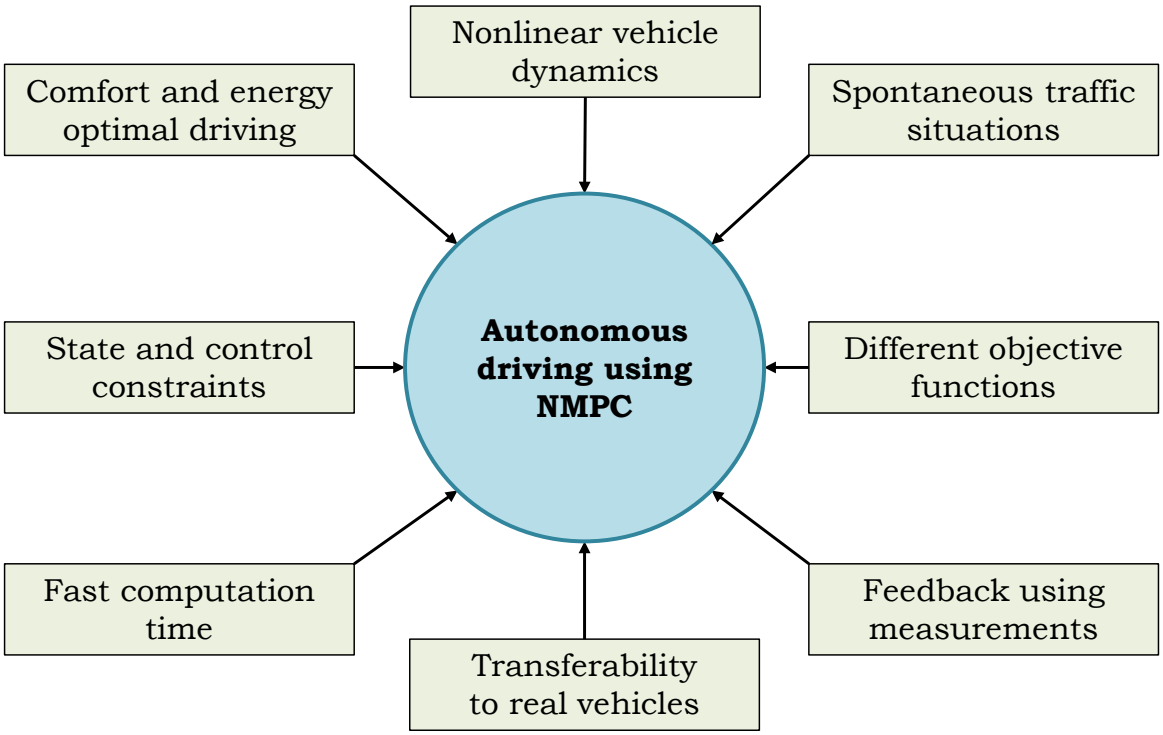

Figure 8.1: Advantages of the NMPC for autonomous driving

The main advantages of the NMPC strategy over, e.g., PID controller, are summarized in Figure 8.1. The obtained results are properly and throughly discussed. It was shown that proposed CMSC method is suitable for implementation of the real-time feedback controller using different scenarios for autonomous driving vehicles.

Both developed methods were tested on the real engineering dynamic optimization problems. The practical applicability of these methods was shown through realization of the NMPC scheme using a mobile robot (1:5) and scaled (1:8) Audi Q2 vehicle

model, aiming to demonstrate the suitability of the CMSC method for autonomous vehicles.

First, the mobile robot SUMMIT was used as a test vehicle to solve the trajectory tracking and landing problems in an unknown environment with obstacles. The obstacles were recognized using the installed laser scanner. The robot can successfully overwhelm all encountered difficulties during the test run. The practical application of these investigations is not only autonomous driving function for conventional vehicles to avoid obstacles. The designed NMPC controller can be implemented and used for special mission robots which investigate the planets, such as "Curiosity", which is a car-sized rover designed to explore Mars. Therefore, the obtained results are very important from practical point of view. The second part of practical experiments was conducted using the Audi Q2 vehicle model with special interest on designing control scheme for the lane keeping assistant system. The information of the lane was supplied to the objective function as a virtually constructed middle line of the lane. The main task of the NMPC controller was to follow this points. Practical experiments include not only the development of the predictive controller, but also in designing the Kalman filter for position estimation, developing the low-level PI controller and implementation of the control framework. The conducted experiments demonstrates outstanding results, approving the applicability of the CMSC method for designing the lane keeping assistant.

In summary, it can be concluded that the fourth level of autonomous driving can be achieved under laboratory conditions. Moreover, the theoretically developed methods for a priori analysis of the dynamic optimization problems show the correct results not only in simulative investigations, but also if they are applied on practical engineering applications. The designed NMPC controller using CMSC method indicates real-time capability and can be further developed for use in the serial production of autonomous vehicles. The capability of the controller was proven using different test scenarios.

8.2 Further research directions

This doctoral thesis can be extended in several ways.

The modeling of the visibility due to varying weather conditions is very important for autonomous driving but has not been principally investigated in previous studies. Bad weather conditions such as rain, snow, fog have significant effects on visibility. Intuitively, the driver will reduce the speed in the case of a low visibility. For autonomous driving, sensors (e.g., camera, laser scanner etc.) will be used to detect the environment but the delivered information may be uncertain due to bad weather conditions. Therefore, it is necessary to model the weather conditions and the related visibility, based on which the controller can provide proper control actions to navigate under bad weather conditions. Although it is inevitably important to remark that the design of a high-level control strategy for autonomous driving has not yet attracted much attention both in theoretical research and in industrial applications. Therefore, the next step is to carry out a systematic study aiming at developing a model-based

control strategy to enhance the robustness and adaptability of autonomous driving under uncertain and varying conditions.

Based on the literature review, there is no agreement, which vehicle model should be using in the controller design. On the one hand, for some traffic situations a simple vehicle model may be used. On the other hand, the controller requires more information about the real vehicle state. One extension leads to the development of a comprehensive vehicle model which mimics sufficiently enough the real behavior of the vehicle, balancing between complexity of the model equations and numerical performance during the solution. Consequently, the next extension leads to the application of the proposed approach within the switching (hybrid) model predictive control framework, which can automatically choose the most appropriate vehicle models (which can be obtained from a comprehensive vehicle model using simplifications). The main point of this extension is a mathematical analysis of the hybrid controller in the presence of switching model dynamics, objective functions, and constraints.

A novel vehicle model will be developed in the form of nonlinear differential equations, containing several sets of parameters: road (tire-road friction coefficient; road grade and bank angles), environment (lateral and longitudinal wind forces; visibility index), and vehicle (mass; tire cornering stiffness) parameters, respectively. Based on our literature review, the existing models (kinematic model, dynamic model, tire model, friction model etc.) treat these parameters individually, i.e., different models contain different sets of parameters. Note that such a vehicle model including all these parameters is not available. Therefore, this extension is to establish such a compact model for the underlying model-based control algorithm to enhance the performance of autonomous driving in terms of predictability, adaptability and robustness.

The considered autonomous driving vehicle is equipped with a number of sensors, such as inertial measurement unit, ABS, cameras, etc. Therefore, the information obtained from these sensors is uncertain, which actually calls for the design of so-called stochastic NMPC. However, the stochastic optimization problems are hard to be solved in real-time. Therefore, the theoretical investigation and practical implementation is considered as a major issue for designing autonomous driving vehicles.

Moreover, from the implementation point of view, several extensions are also possible. Since the NMPC controller relies on the optimizer and sensitivity computations using a linear algebra solver, one can highlight the custom implementation of them, focusing on the specific characteristics of the vehicle models.

The next extension is dedicated to the design of a supervisory control scheme based on the artificial intelligence techniques. With the help of artificial intelligence many problems may be solved much easier in comparison with classical methods. Moreover, these methods may provide a proper answer for designing the underlying NMPC controller. For instance, convolutional neural networks may be applied for (i) the automatic selection of the weighting matrices in the objective function (ii) the fast recognition of the environment situation through camera (iii) overall system monitoring.

Last but not least extension requires the purely theoretical investigation, leading to the mathematical analysis of the convergence rate of the CMSC method.

Chapter 9

Appendix

Illustration example for comparison between the Nyquist-Shannon theorem and the bilevel approach

To demonstrate the efficiency of the proposed bilevel approach over the Nyquist-Shannon theorem for determining sampling intervals, a simple linear control system is considered as shown in Figure 9.1, where a proportional controller is defined as $K_p = 1$ and the set point $W(s)$ is a step function.

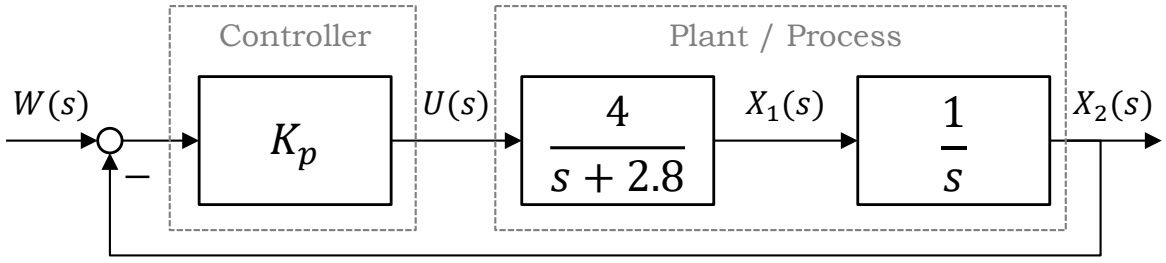

Figure 9.1: Closed loop control scheme.

The model equations in the state space form are as follows

$$\dot{x}_1(t) = -2.8 \cdot x_1(t) + 4 \cdot u(t), \tag{9.1}$$

$$\dot{x}_2(t) = x_1(t). \tag{9.2}$$

Our goal is to find the minimum number of time intervals that guarantees a maximum approximation error $\varepsilon = 0.002$ of the state $x_2(t)$ in the time horizon $t \in [0, 5]$. Without loss of generality, the control signal is considered to be limited as $-1 \leq u(t) \leq 1$ and the initial conditions of the state variables are $x_1(0) = 0$, $x_2(0) = 0$.

Applying the proposed bilevel approach, eight time intervals are needed to satisfy the given tolerance and the corresponding maximum approximation error at the non-collocation points is 0.001286. In contrast, using the Nyquist-Shannon theorem the sampling frequency f_s should determined by

$$f_s \geq 2 \cdot f_B, \tag{9.3}$$

where $f_B = 0.321$ Hz denotes the bandwidth frequency of the closed-loop system. Therefore, the lower limit of the sampling frequency f_s should be 0.642 Hz. Therefore, the minimum number of time intervals determined by (9.3) is 3.2 which is rounded to 4. Using 4 time intervals, the resulting maximum approximation error is 0.006751 which violates the tolerance. Hence, practically the sampling frequency is chosen from 20 up to 40 times larger than the bandwidth frequency [35]. For testing purposes, the factor of 20 is chosen, which leads to a much higher number of time intervals, i.e. $N = 33$, and also a much higher accuracy (i.e. the maximum approximation error is $1.4364 \cdot 10^{-6}$) than necessary. Consequently, the proposed bilevel approach is highly efficient and outperforms the results obtained by the Nyquist-Shannon theorem.

Demonstration of the CMSC method with the help of a kinematic vehicle model

For demonstration purpose the kinematic vehicle model from Chapter 7 is chosen. The problem is described as follows:

$$\dot{x_1}(t) = u_1(t) \cdot \cos(x_3(t) + \frac{l_r}{l_f + l_r} \cdot u_2(t)), \tag{9.4}$$

$$\dot{x_2}(t) = u_1(t) \cdot \sin(x_3(t) + \frac{l_r}{l_f + l_r} \cdot u_2(t)), \tag{9.5}$$

$$\dot{x_3}(t) = \frac{u_1(t)}{l_f + l_r} \cdot \tan(u_2(t)). \tag{9.6}$$

The model equations contains three state and two control variables, i.e., $n_x = 3$ and $n_u = 2$, respectively. For the sake of brevity, the number of time intervals N is equal to 3. Applying the three-point-collocation scheme, e.g. $N_c = 3$, the number of collocation points N_{NV} to be computed is equal to

$$N_{NV} = n_x \cdot N_c \cdot N. \tag{9.7}$$

Consequently, the number of optimization variables N_{OV} is equal to

$$N_{OV} = (n_x + n_u) \cdot N + n_x. \tag{9.8}$$

The vector of state variables at collocation points $X^{c,i}$ in i-th intervals, $i = \{1, \ldots, N\}$ can be constructed as follows:

$$X^{c,i} = \left[x_{1,1}^{c,i}, x_{2,1}^{c,i}, x_{3,1}^{c,i}, x_{1,2}^{c,i}, x_{2,2}^{c,i}, x_{3,2}^{c,i}, x_{1,3}^{c,i}, x_{2,3}^{c,i}, x_{3,3}^{c,i}\right]. \tag{9.9}$$

The corresponding optimization variables $X^{o,i}$ in the i-th intervals, $i = \{0, \ldots, N\}$, are defined as

$$X^{o,i} = \left[x_{0,1}^{p,i}, x_{0,2}^{p,i}, x_{0,3}^{p,i}, u_1^i, u_2^i\right]. \tag{9.10}$$

Therefore, the model equations (9.4) - (9.6) can be transformed into nonlinear algebraic equations:

$$G_1 = D_1 - \Delta t \cdot u_1^i \cdot \cos\left(x_{3,1}^{c,i} + \left(\frac{l_r}{l_f + l_r} \cdot u_2^i\right)\right), \tag{9.11}$$

$$G_2 = D_2 - \Delta t \cdot u_1^i \cdot \sin\left(x_{3,1}^{c,i} + \left(\frac{l_r}{l_f + l_r} \cdot u_2^i\right)\right), \tag{9.12}$$

$$G_3 = D_3 - \Delta t \cdot \frac{u_1^i}{l_f + l_r} \cdot \tan\left(u_2^i\right), \tag{9.13}$$

$$G_4 = D_4 - \Delta t \cdot u_1^i \cdot \cos\left(x_{3,2}^{c,i} + \left(\frac{l_r}{l_f + l_r} \cdot u_2^i\right)\right), \tag{9.14}$$

$$G_5 = D_5 - \Delta t \cdot u_1^i \cdot \sin\left(x_{3,2}^{c,i} + \left(\frac{l_r}{l_f + l_r} \cdot u_2^i\right)\right), \tag{9.15}$$

$$G_6 = D_6 - \Delta t \cdot \frac{u_1^i}{l_f + l_r} \cdot \tan\left(u_2^i\right), \tag{9.16}$$

$$G_7 = D_7 - \Delta t \cdot u_1^i \cdot \cos\left(x_{3,3}^{c,i} + \left(\frac{l_r}{l_f + l_r} \cdot u_2^i\right)\right), \tag{9.17}$$

$$G_8 = D_8 - \Delta t \cdot u_1^i \cdot \sin\left(x_{3,3}^{c,i} + \left(\frac{l_r}{l_f + l_r} \cdot u_2^i\right)\right), \tag{9.18}$$

$$G_9 = D_9 - \Delta t \cdot \frac{u_1^i}{l_f + l_r} \cdot \tan\left(u_2^i\right), \tag{9.19}$$

where

$$D_1 = \dot{L}_{0,0} \cdot x_{1,1}^{c,i} + \dot{L}_{0,1} \cdot x_{1,2}^{c,i} + \dot{L}_{0,2} \cdot x_{1,3}^{c,i} + \dot{L}_{0,3} \cdot x_{0,1}^{p,i}, \tag{9.20}$$

$$D_2 = \dot{L}_{0,0} \cdot x_{2,1}^{c,i} + \dot{L}_{0,1} \cdot x_{2,2}^{c,i} + \dot{L}_{0,2} \cdot x_{2,3}^{c,i} + \dot{L}_{0,3} \cdot x_{0,2}^{p,i}, \tag{9.21}$$

$$D_3 = \dot{L}_{0,0} \cdot x_{3,1}^{c,i} + \dot{L}_{0,1} \cdot x_{3,2}^{c,i} + \dot{L}_{0,2} \cdot x_{3,3}^{c,i} + \dot{L}_{0,3} \cdot x_{0,3}^{p,i}, \tag{9.22}$$

$$D_4 = \dot{L}_{1,0} \cdot x_{1,1}^{c,i} + \dot{L}_{1,1} \cdot x_{1,2}^{c,i} + \dot{L}_{1,2} \cdot x_{1,3}^{c,i} + \dot{L}_{1,3} \cdot x_{0,1}^{p,i}, \tag{9.23}$$

$$D_5 = \dot{L}_{1,0} \cdot x_{2,1}^{c,i} + \dot{L}_{1,1} \cdot x_{2,2}^{c,i} + \dot{L}_{1,2} \cdot x_{2,3}^{c,i} + \dot{L}_{1,3} \cdot x_{0,2}^{p,i}, \tag{9.24}$$

$$D_6 = \dot{L}_{1,0} \cdot x_{3,1}^{c,i} + \dot{L}_{1,1} \cdot x_{3,2}^{c,i} + \dot{L}_{1,2} \cdot x_{3,3}^{c,i} + \dot{L}_{1,3} \cdot x_{0,3}^{p,i}, \tag{9.25}$$

$$D_7 = \dot{L}_{2,0} \cdot x_{1,1}^{c,i} + \dot{L}_{2,1} \cdot x_{1,2}^{c,i} + \dot{L}_{2,2} \cdot x_{1,3}^{c,i} + \dot{L}_{2,3} \cdot x_{0,1}^{p,i}, \tag{9.26}$$

$$D_8 = \dot{L}_{2,0} \cdot x_{2,1}^{c,i} + \dot{L}_{2,1} \cdot x_{2,2}^{c,i} + \dot{L}_{2,2} \cdot x_{2,3}^{c,i} + \dot{L}_{2,3} \cdot x_{0,2}^{p,i}, \tag{9.27}$$

$$D_9 = \dot{L}_{2,0} \cdot x_{3,1}^{c,i} + \dot{L}_{2,1} \cdot x_{3,2}^{c,i} + \dot{L}_{2,2} \cdot x_{3,3}^{c,i} + \dot{L}_{2,3} \cdot x_{0,3}^{p,i}. \tag{9.28}$$

and

$$\dot{L} = \begin{bmatrix} 3.224744871392 & 1.16784008469 & -0.25319726474 & -4.1393876913 \\ -3.56784008469 & 0.77525512860 & 1.053197264742 & 1.73938769134 \\ 5.531972647422 & -7.5319726474 & 5.000000000000 & -3.0 \end{bmatrix}. \tag{9.29}$$

The discretized model equations $G_1, \ldots, G_9$ are combined together into one function G^i, as indicated in equation (3.3) in Chapter 3. The sensitivity equations (3.4) and (3.5) are merged together and the corresponding right-hand $\frac{\partial G^i}{\partial X^{c,i}}$ and left-hand side $\frac{\partial G^i}{\partial X^{o,i}}$ matrices are defined in sparsity format as given below.

$$\frac{\partial G^i}{\partial X^{c,i}} = \begin{bmatrix}
(0,0) & = & \dot{L}_{0,0} \\
(3,0) & = & \dot{L}_{1,0} \\
(6,0) & = & \dot{L}_{2,0} \\
(1,1) & = & \dot{L}_{0,0} \\
(4,1) & = & \dot{L}_{1,0} \\
(7,1) & = & \dot{L}_{2,0} \\
(0,2) & = & \Delta t \cdot (-(u_1^i \cdot (-\sin((x_{3,1}^{c,i} + (\frac{l_r}{l_f+l_r} \cdot u_2^i)))))) \\
(1,2) & = & \Delta t \cdot (-(u_1^i \cdot \cos((x_{3,1}^{c,i} + (\frac{l_r}{l_f+l_r} \cdot u_2^i))))) \\
(2,2) & = & \dot{L}_{0,0} \\
(5,2) & = & \dot{L}_{1,0} \\
(8,2) & = & \dot{L}_{2,0} \\
(0,3) & = & \dot{L}_{0,1} \\
(3,3) & = & \dot{L}_{1,1} \\
(6,3) & = & \dot{L}_{2,1} \\
(1,4) & = & \dot{L}_{0,1} \\
(4,4) & = & \dot{L}_{1,1} \\
(7,4) & = & \dot{L}_{2,1} \\
(2,5) & = & \dot{L}_{0,1} \\
(3,5) & = & \Delta t \cdot (-(u_1^i \cdot (-\sin((x_{3,2}^{c,i} + (\frac{l_r}{l_f+l_r} \cdot u_2^i)))))) \\
(4,5) & = & \Delta t \cdot (-(u_1^i \cdot \cos((x_{3,2}^{c,i} + (\frac{l_r}{l_f+l_r} \cdot u_2^i))))) \\
(5,5) & = & \dot{L}_{1,1} \\
(8,5) & = & \dot{L}_{2,1} \\
(0,6) & = & \dot{L}_{0,2} \\
(3,6) & = & \dot{L}_{1,2} \\
(6,6) & = & \dot{L}_{2,2} \\
(1,7) & = & \dot{L}_{0,2} \\
(4,7) & = & \dot{L}_{1,2} \\
(7,7) & = & \dot{L}_{2,2} \\
(2,8) & = & \dot{L}_{0,2} \\
(5,8) & = & \dot{L}_{1,2} \\
(6,8) & = & \Delta t \cdot (-(u_1^i \cdot (-\sin((x_{3,3}^{c,i} + (\frac{l_r}{l_f+l_r} \cdot u_2^i)))))) \\
(7,8) & = & \Delta t \cdot (-(u_1^i \cdot \cos((x_{3,3}^{c,i} + (\frac{l_r}{l_f+l_r} \cdot u_2^i))))) \\
(8,8) & = & \dot{L}_{2,2}
\end{bmatrix}, \tag{9.30}$$

where indices in round brackets mean the position of element in the matrix. All other elements in this matrix are essentially zeros.

$$\frac{\partial G^i}{\partial X^{o,i}} = \begin{bmatrix}
(0,0) & = & \dot{L}_{0,3} \\
(3,0) & = & \dot{L}_{1,3} \\
(6,0) & = & \dot{L}_{2,3} \\
(1,1) & = & \dot{L}_{0,3} \\
(4,1) & = & \dot{L}_{1,3} \\
(7,1) & = & \dot{L}_{2,3} \\
(2,2) & = & \dot{L}_{0,3} \\
(5,2) & = & \dot{L}_{1,3} \\
(8,2) & = & \dot{L}_{2,3} \\
(0,3) & = & \Delta t \cdot (-\cos((x_{3,1}^{c,i} + (\frac{l_r}{l_f+l_r} \cdot u_2^i)))) \\
(1,3) & = & \Delta t \cdot (-\sin((x_{3,1}^{c,i} + (\frac{l_r}{l_f+l_r} \cdot u_2^i)))) \\
(2,3) & = & \Delta t \cdot (-\frac{1}{l_f+l_r} \cdot \tan(u_2^i)) \\
(3,3) & = & \Delta t \cdot (-\cos((x_{3,2}^{c,i} + (\frac{l_r}{l_f+l_r} \cdot u_2^i)))) \\
(4,3) & = & \Delta t \cdot (-\sin((x_{3,2}^{c,i} + (\frac{l_r}{l_f+l_r} \cdot u_2^i)))) \\
(5,3) & = & \Delta t \cdot (-\frac{1}{l_f+l_r} \cdot \tan(u_2^i)) \\
(6,3) & = & \Delta t \cdot (-\cos((x_{3,3}^{c,i} + (\frac{l_r}{l_f+l_r} \cdot u_2^i)))) \\
(7,3) & = & \Delta t \cdot (-\sin((x_{3,3}^{c,i} + (\frac{l_r}{l_f+l_r} \cdot u_2^i)))) \\
(8,3) & = & \Delta t \cdot (-\frac{1}{l_f+l_r} \cdot \tan(u_2^i)) \\
(0,4) & = & \Delta t \cdot (-(u_1^i \cdot (\frac{1}{l_r+l_r} \cdot (-\sin((x_{3,1}^{c,i} + (\frac{l_r}{l_f+l_r} \cdot u_2^i))))))) \\
(1,4) & = & \Delta t \cdot (-(u_1^i \cdot (\frac{1}{l_r+l_r} \cdot \cos((x_{3,1}^{c,i} + (\frac{l_r}{l_f+l_r} \cdot u_2^i)))))) \\
(2,4) & = & \Delta t \cdot (-(\frac{u_1^i}{l_f+l_r} \cdot \frac{1}{\cos^2(u_2^i)})) \\
(3,4) & = & \Delta t \cdot (-(u_1^i \cdot (\frac{1}{l_f+l_r} \cdot (-\sin((x_{3,2}^{c,i} + (\frac{l_r}{l_f+l_r} \cdot u_2^i))))))) \\
(4,4) & = & \Delta t \cdot (-(u_1^i \cdot (\frac{1}{l_f+l_r} \cdot \cos((x_{3,2}^{c,i} + (\frac{l_r}{l_f+l_r} \cdot u_2^i)))))) \\
(5,4) & = & \Delta t \cdot (-(\frac{u_1^i}{l_f+l_r} \cdot \frac{1}{\cos^2(u_2^i)})) \\
(6,4) & = & \Delta t \cdot (-(u_1^i \cdot (\frac{1}{l_f+l_r} \cdot (-\sin((x_{3,3}^{c,i} + (\frac{l_r}{l_f+l_r} \cdot u_2^i))))))) \\
(7,4) & = & \Delta t \cdot (-(u_1^i \cdot (\frac{1}{l_f+l_r} \cdot \cos((x_{3,3}^{c,i} + (\frac{l_r}{l_f+l_r} \cdot u_2^i)))))) \\
(8,4) & = & \Delta t \cdot (-(\frac{u_1^i}{l_f+l_r} \cdot \frac{1}{\cos^2(u_2^i)}))
\end{bmatrix}. \tag{9.31}$$

The next step is to formulate the constraints vector. Since an NLP problem is solved in NMPC framework, the initial state values $[x_{0,1}, x_{0,2}, x_{0,3}]^T$ change from one to the next NLP iteration. Therefore, initial continuity conditions B_I should be formulated:

$$B_I = \begin{bmatrix} x_{0,1} - x_{0,1}^{p,0} \\ x_{0,2} - x_{0,2}^{p,0} \\ x_{0,3} - x_{0,3}^{p,0} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}. \tag{9.32}$$

Moreover, according to the CMSC method the continuity condition between intervals B_{CC} must be stated to guarantee the continuity of the state trajectories and the state

values at collocation points must satisfy the variable bounds, where lower and upper bounds of the constraints are defined as B_{CC}^{min} and B_{CC}^{max}, respectively.

$$B_{CC} = \begin{bmatrix} x_{1,1}^{c,1} \\ x_{2,1}^{c,1} \\ x_{3,1}^{c,1} \\ x_{1,2}^{c,1} \\ x_{2,2}^{c,1} \\ x_{3,2}^{c,1} \\ x_{1,3}^{c,1} - x_{0,1}^{p,1} \\ x_{2,3}^{c,1} - x_{0,2}^{p,1} \\ x_{3,3}^{c,1} - x_{0,3}^{p,i} \\ x_{1,1}^{c,2} \\ x_{2,1}^{c,2} \\ x_{3,1}^{c,2} \\ x_{1,2}^{c,2} \\ x_{2,2}^{c,2} \\ x_{3,2}^{c,2} \\ x_{1,3}^{c,2} - x_{0,1}^{p,2} \\ x_{2,3}^{c,2} - x_{0,2}^{p,2} \\ x_{3,3}^{c,2} - x_{0,3}^{p,2} \\ x_{1,1}^{c,3} \\ x_{2,1}^{c,3} \\ x_{3,1}^{c,3} \\ x_{1,2}^{c,3} \\ x_{2,2}^{c,3} \\ x_{3,2}^{c,3} \\ x_{1,3}^{c,3} - x_{0,1}^{p,3} \\ x_{2,3}^{c,3} - x_{0,2}^{p,3} \\ x_{3,3}^{c,3} - x_{0,3}^{p,3} \end{bmatrix}, B_{CC}^{min} = \begin{bmatrix} x_{1,min} \\ x_{2,min} \\ x_{3,min} \\ x_{1,min} \\ x_{2,min} \\ x_{3,min} \\ 0 \\ 0 \\ 0 \\ x_{1,min} \\ x_{2,min} \\ x_{3,min} \\ x_{1,min} \\ x_{2,min} \\ x_{3,min} \\ 0 \\ 0 \\ 0 \\ x_{1,min} \\ x_{2,min} \\ x_{3,min} \\ x_{1,min} \\ x_{2,min} \\ x_{3,min} \\ 0 \\ 0 \\ 0 \end{bmatrix}, B_{CC}^{max} = \begin{bmatrix} x_{1,max} \\ x_{2,max} \\ x_{3,max} \\ x_{1,max} \\ x_{2,max} \\ x_{3,max} \\ 0 \\ 0 \\ 0 \\ x_{1,max} \\ x_{2,max} \\ x_{3,max} \\ x_{1,max} \\ x_{2,max} \\ x_{3,max} \\ 0 \\ 0 \\ 0 \\ x_{1,max} \\ x_{2,max} \\ x_{3,max} \\ x_{1,max} \\ x_{2,max} \\ x_{3,max} \\ 0 \\ 0 \\ 0 \end{bmatrix}. \tag{9.33}$$

The Jacobian matrix ∇B_{CC} of the constraints vector is defined as follows:

$$\nabla B_{CC} = \begin{bmatrix} [-I]_{n_x \times n_x} & & & \\ \left[\frac{\partial X^{c,1}}{\partial X^{o,1}}\right]_{q \times w} & \left[-\tilde{I}\right]_{q \times n_x} & & \\ & \left[\frac{\partial X^{c,2}}{\partial X^{o,2}}\right]_{q \times w} & \left[-\tilde{I}\right]_{q \times n_x} & \\ & & \left[\frac{\partial X^{c,3}}{\partial X^{o,3}}\right]_{q \times w} & \left[-\tilde{I}\right]_{q \times n_x} \end{bmatrix}, \tag{9.34}$$

where $q = n_x \cdot N_c$, $w = n_x + n_u$; I and $\tilde{I}$ are appropriate matrices which comes from the derivatives with respect to parametrized initial and continuity conditions, respectively. It should be noted that only nonzero elements should be defined in the IPOPT.

The objective function and its corresponding gradient supplied by the user and, therefore, are not considered here. To facilitate the computational rate, it is worth to use such objective function, that depends explicitly only on the optimization variables. Thus, the computation of the gradients can be done in the straight-forward manner.

Bibliography

[1] A. H. Ahmed and K. Wright. Error estimation for collocation solution of linear ordinary differential equations. *Comp and Maths with Appls.*, 5/6(12B):1053–1059, 1986.

[2] J. Åkesson. Optimica - an extension of Modelica supporting dynamic optimization. *Proceeding of the* 6^{th} *international Modelica conference*, pages 57–66, 2008.

[3] J. Åkesson, K. E. Årzén, M. Gåfvert, T. Bergdahl, and H. Tummescheit. Modeling and optimization with optimica and jmodelica.org-languages and tools for solving large-scale dynamic optimization problems. *Comput. Chem. Eng.*, 11(34):1737–1749, 2010.

[4] J. Andersson, J. Åkesson, and M. Diehl. CasADi: A symbolic package for automatic differentiation and optimal control. *Lect. Notes Comput. Sci. Eng.*, (87):297–307, 2012.

[5] U. M. Ascher and L. R. Petzold. Projected implicit runge-kutta methods for differential-algebraic equations. *SIAM J. Numer. Anal.*, 4(28):1097–1120, 1990.

[6] W. Auzinger, O. Koch, and E. Weinmüller. Efficient collocation schemes for singular-boundary value problems. *Numerical Algorithms*, (31):5–25, 2002.

[7] W. Auzinger, O. Koch, and E. Weinmüller. Efficient mesh selection for collocation methods applied to singular bvps. *Journal of Computational and Applied Mathematics*, (180):213–227, 2005.

[8] B. Bachmann, L. Ochel, V. Ruge, M. Gebremedhin, P. Fritzson, V. Nezhadali, L. Eriksson, and M. Sivertsson. Sundials: Suite of nonlinear and differential/algebraic equation solvers. *Proceedings of the* 9^{th} *international Modelica conference*, pages 659–668, 2012.

[9] E. Balsa-Canto, J. R. Banga, A. A. Alonso, and V. S. Vassiliadis. Dynamic optimization of chemical and biochemical processes using restricted second-order information. *Comput. Chem. Eng.*, 4-6(25):539–546, 2001.

[10] E. Balsa-Canto, J. R. Banga, A. A. Alonso, and V. S. Vassiliadis. Dynamic optimization of distributed parameter systems using second-order directional derivatives. *Ind. Eng. Chem. Res.*, 21(43):6756–6765, 2004.

[11] M. Bartl, P. Li, and L. T. Biegler. Improvement of state profile accuracy in nonlinear dynamic optimization with the quasi-sequential approach. *AlChE J.*, 8(57):2185–2197, 2011.

[12] T. Barz, R. Klaus, L. Zhu, G. Wozny, and H. Arellano-Garcia. Generation of discrete first- and second-order sensitivities for single shooting. *AIChE J.*, 10(58):3110–3122, 2012.

[13] T. Barz, S. Kuntsche, G. Wozny, and H. Arellano-Garcia. An efficient sparse approach to sensitivity generation for large-scale dynamic optimization. *Comput. Chem. Eng.*, 10(35):2053–2065, 2010.

[14] C. E. Beal and J. C. Gerdes. Model predictive control for vehicle stabilization at the limits of handling. *IEEE transactions on control systems technlogy*, 4(21):1258–1269, 2013.

[15] D. P. Bertsekas. *Dynamic Programming and Optimal Control*. 2017.

[16] L. T. Biegler. Efficient solution of dynamic optimization and nmpc problems. *Progress in Systems and Control Theory*, (26):219–243, 2000.

[17] L. T. Biegler, A. M. Cervantes, and A. Wächter. Advances in simultaneous strategies for dynamic process optimization. *Chem. Eng. Sci.*, 4(57):575–593, 2002.

[18] T. Binder, A. Cruse, C. A. K. C. Villar, and W. Marquardt. Dynamic optimization using a wavelet based adaptive control vector parameterization strategy. *Comp Chem Eng*, 2-7(24):1201–1207, 2000.

[19] H. G. Bock and K. J. Plitt. A multiple shooting algorithm for direct solution of optimal control problems. *Proceedings of the* 9^{th} *IFAC world congress*, pages 242–246, 1984.

[20] F. Borrelli, P. Falcone, T. Keviczky, J. Asgari, and D. Hrovat. Mpc-based approach to active steering for autonomous vehicle systems. *International journal of vehicle autonomous systems*, 2(3):265–291, 2005.

[21] S. P. Boyd and L. Vandenberghe. *Berichte über verteilte Messysteme*. 2004.

[22] X. Cai. *On the efficient implementation of efficient numerical methods for dynamic optimization using GPU*. 2017.

[23] S. D. Cairano, H. E. Tseng, D. Bernardini, and A. Bemporad. Steering vehicle control by switched model predictive control. *IFAC proceedings volumes*, 7(43):1–6, 2010.

[24] M. Choi and S. B. Choi. Model predictive control for vehicle yaw stability with practical concerns. *IEEE Transactions on vehicular technology*, 8(63):3539–3548, 2014.

[25] Y. Chu and J. Hahn. Parameter set selection for estimation of nonlinear dynamic systems. *AIChE J.*, 11(53):2858–2870, 2007.

[26] J. E. Cuthrell and L. T. Biegler. On the optimization of differential-algebraic process systems. *AlChE J.*, (33):1257–1270, 1987.

[27] J. E. Cuthrell and L. T. Biegler. Simultaneous optimization and solution methods for batch reactor control profiles. *Comput. Chem. Eng.*, 1-2(13):49–62, 1989.

[28] R. D'Ambrosio, M. Ferro, Z. Jackiewicz, and B. Paternoster. Two-step almost collocation methods for ordinary differential equations. *Numerical Algorithms*, 2-3(53):195–217, 2010.

[29] C. L. Darby, W. W. Hager, and A. V. Rao. An *hp*-adaptive pseudospectral method for solving optimal control problems. *Optim. Control Appl. Meth.*, (32):476–502, 2011.

[30] C. DeBoor. Good approximation by splines with variable knots-ii. *Lect Notes Math*, (363):12–20, 1974.

[31] C. DeBoor. Practical guide to splines. *Applied Mathematical Sciences*, 1978.

[32] C. DeBoor and R. Swartz. Collocation at gaussian points. *SIAM J. Numer. Ana.*, 4(10):582–606, 1973.

[33] M. Diehl. *Real-time optimization for large scale nonlinear processes. Ph.D. thesis.* 2001.

[34] E. Drozdova, S. Hopfgarten, E. Lazutkin, and P. Li. Autonomous driving of a mobile robot using a combined multiple-shooting and collocation meethod. *9-th IFAC symposium on intelligent autonomous vehicles*, 15(49):193–198, 2016.

[35] G. F. Franklin, J. D. Powell, and M. L. Workman. *Digital control of dynamic systems, Third edition.* 1998.

[36] J. V. Frasch, A. Gray, M. Zanon, H. J. Ferreau, S. Sager, F. Borrelli, and M. Diehl. An auto-generated nonlinear mpc algorithm for real-time obstacle avoidance of ground vehicles. *European Control Conference*, pages 4136–4141, 2013.

[37] P. Fritzson. *Principles of object-oriented modeling and simulation with Modelica 3.3: A cyber-physical approach.* 2014.

[38] Y. Gao, A. Gray, H. E. Tseng, and F. Borelli. A tube-based robust nonlinear predictive control approach to semi-autonomous ground vehicles. *Vehicle system dynamics: international journal of vehicle mechanics and mobility*, 6(52):802–823, 2014.

[39] Y. Gao, T. Lin, F. Borrelli, E. Tseng, and D. Hrovat. Predictive control of autonomous ground vehicles with obstacle avoidance on slippery roads. *Dynamic Systems and Control Conference*, 2010.

[40] C. E. Garcia, D. M. Prett, and M. Morari. Model predictive control: theory and practice. *Automatica*, 3(25):335–348, 1989.

[41] A. Geletu, M. Klöppel, A. Hoffmann, and P. Li. A tractable approximation of non-convex chance constrained optimization with non-gaussian uncertainties. *Journal of Engineering Optimization*, 4(47):495–520, 2015.

[42] L. Grüne and J. Pannek. *Nonlinear model predictive control. Theory and algorithms.* 2011.

[43] B.-Y. Guo and Z.-Q. Wang. Legendre-gauss collocation methods for ordinary differential equations. *Adv Comput Math*, (30):249–280, 2009.

[44] R. Haber and H. Unbehauen. Structure identification of nonlinear dynamic systems - A survey on input/output approaches. *Automatica*, 4(26):651–677, 1990.

[45] A. Hartwich, C. Stockmann, C. Terboven, S. Feuerriegel, and W. Marquardt. Parallel sensitivity analysis for efficient large-scale dynamic optimization. *Optim. Eng.*, 4(12):489–508, 2011.

[46] J. D. Hedengren. A nonlinear model library for dynamics and control. *CACHE news*, 2008.

[47] D. Hellmann. *The Python Standard Library by Example.* 2011.

[48] A. C. Hindmarsh, P. N. Brown, K. E. Grant, S. L. Lee, R. Serban, D. E. Shumaker, and C. S. Woodward. Sundials: Suite of nonlinear and differential/algebraic equation solvers. *ACM. T. Math. Software*, 3(31):363–396, 2005.

[49] W. R. Hong, S. Q. Wang, P. Li, G. Wozny, and L. T. Biegler. A quasi-sequential approach to large-scale dynamic optimization problems. *AlChE J.*, 1(52):255–268, 2006.

[50] B. Houska, H. J. Ferreau, and M. Diehl. An auto-generated real-time iteration algorithm for nonlinear mpc in the microsecond range. *Automatica*, 10(47):2279–2285, 2011.

[51] G. T. Huntington and A. V. Rao. Comparison of global and local collocation methods for optimal control. *J Guid Control Dyn*, (31):432–436, 2008.

[52] J. L. Junkins and J. D. Turner. Optimal continuous torque attitude maneuvers. *J. Guid. Control*, 3(3):210–217, 1980.

[53] J. Kang, Y. Cao, D. P. Word, and C. D. Laird. An interior-point method for efficient solution of block-structured NLP problems using an implicit Schur-complement decomposition. *Comput. Chem. Eng.*, (71):563–573, 2014.

[54] N. Karmarkar. A new polynomial-time algorithm for linear programming. *Proceedings of the Sixteenth Annual ACM Symposium on Theory of Computing*, pages 302–311, 1984.

[55] A. Katriniok, J. P. Maschuw, F. Christen, L. Eckstein, and D. Abel. Optimal vehicle dynamics control for combined longitudinal and lateral autonomous vehicle guidance. *European Control Conference*, pages 974–979, 2013.

[56] T. Keviczky, P. Falcone, F. Borelli, J. Asgari, and D. Hrovat. Predictive control approach to autonomous vehicle steering. *Proceedings of the american control conference*, 2006.

[57] K. D. Kim and P. R. Kumar. An mpc-based approach to provable system-wide safey and liveness of autonomous ground traffic. *IEEE Transactions on automatic control*, 12(59):3341–3356, 2014.

[58] C. Kirches, L. Wirsching, H. G. Bock, and J. P. Schlöder. Efficient direct multiple shooting for nonlinear model predictive control on long horizons. *J. Process Control*, 3(22):540–551, 2012.

[59] O. Koch. Asymptotically correct error estimation for collocation methods applied to singular boundary value problems. *J Sci Comput*, (101):143–164, 2005.

[60] E. Lazutkin, A. Geletu, S. Hopfgarten, and P. Li. Modified multiple shooting combined with collocation method in JModelica.org with symbolic calculations. *Proceedings of the* 10^{th} *International Modelica Conference*, pages 999–1006, 2014.

[61] E. Lazutkin, A. Geletu, S. Hopfgarten, and P. Li. An analytical hessian and parallel-computing approach for efficient dynamic optimization based on control-variable correlation analysis. *Ind. Eng. Chem. Res.*, 48(54):12086–12095, 2015.

[62] E. Lazutkin, A. Geletu, and P. Li. An approach to determining the number of time intervals for solving dynamic optimization problems. *Ind. Eng. Chem. Res.*, 12(57):4340–4350, 2018.

[63] E. Lazutkin, S. Hopfgarten, A. Geletu, and P. Li. A toolchain for solving dynamic optimization problems using symbolic and parallel computing. *Proceedings of the* 11^{th} *International Modelica Conference*, pages 311–320, 2015.

[64] D. B. Leineweber, I. Bauer, H. G. Bock, and J. P. Schlöder. An efficient multiple shooting based reduced SQP strategy for large-scale dynamic process optimization-Part I: theoretical aspects. *Comput. Chem. Eng.*, 2(27):157–166, 2003.

[65] D. B. Leineweber, I. Bauer, H. G. Bock, and J. P. Schlöder. An efficient multiple shooting based reduced SQP strategy for large-scale dynamic process

optimization-Part II: software aspects and applications. *Comput. Chem. Eng.*, 2(27):167–174, 2003.

[66] P. Li and Q. D. Vu. Identification of parameter correlations for parameter estimation in dynamic biological models. *BMC Systems Biology*, 7(91), 2013.

[67] S. Li, L. Petzold, and W. Zhu. Sensitivity analysis of differential-algebraic equations: a comparison of methods on a special problem. *Appl. Numer. Math.*, (32):161–174, 2000.

[68] L. Ljung. *System identification - Theory for the user*. 1999.

[69] J. S. Logsdon and L. T. Biegler. Accurate solution of differential-algebraic optimization problems. *Ind. Eng. Chem. Res.*, (28):1628–1639, 1989.

[70] L. H. Luo, H. Liu, P. Li, and H. Wang. Model predictive control for adaptive cruise control with multi-objectives: comfort, fuel-economy, safety and car-following. *Journal of Zhejiang University-Science A*, 3(11):191–201, 2010.

[71] R. Luus. Application of dynamic programming to high-dimensional non-linear optimal control problems. *Inter. J. of Contr.*, 1(52):239–250, 1990.

[72] R. Luus, J. Dittrich, and F. J. Keil. Multiplicity of solutions in the optimization of a bifunctional catalyst blend in a tubular reactor. *The Canadian Journal of Chemical Engineering*, (70):780–785, 2006.

[73] F. J. MacWilliams and N. J. A. Sloane. Pseudo-random sequences and arrays. *Proceedings of the IEEE*, 12(64):1715–1729, 1976.

[74] K. A. P. McLean and K. B. McAuley. Mathematical modelling of chemical processesobtaining the best model predictions and parameter estimates using identifiability and estimability procedures. *Can. J. Chem. Eng.*, 2(90):351–366, 2012.

[75] S. Mehrotra. On the implementation of a primal-dual interior point method. *SIAM J. Optimization*, 2(4):575–601, 1992.

[76] M. A. Mehrpouya, M. Shamsi, and M. Razzaghi. A combined adaptive control parametrization and homotopy continuaton technique for the numerical solution of bang-bang optimal control problems. *ANZIAM J.*, 8(56):48–65, 2014.

[77] J. Nocedal and S. J. Wright. *Numerical Optimization*. 2006.

[78] D. B. Özyurt and P. I. Barton. Large-scale dynamic optimization with the directional second order adjoint method. *Ind. Eng. Chem. Res.*, 6(44):1804–1811, 2005.

[79] L. T. Paiva and F. A. C. C. Fontes. Adaptive time-mesh refinement in optimal control problems with state constraints. *Discrete and Continuous Dynamical Systems*, pages 4553–4572, 2015.

[80] A. Propoi. Use of linear programming methods for synthesizing sampled-data automatic systems. *Automation and remote control*, 7(24):837–844, 1963.

[81] J. K. Reid and I. S. Du. Ma27 - a set of fortran subroutines for solving sparse symmentric sets of linear equations. *Tech. rep. AERE R 10533*, 1982.

[82] J. Richalet, A. Rault, J. L. Testud, and J. Papon. Model predictive heuristic control: Applications to industrial processes. *Automatica*, 5(14):413–428, 1978.

[83] R. D. I. Robinett, D. G. Wilson, G. R. Eisler, and J. E. Hurtado. *Applied Dynamic Programming for Optimization of Dynamical Systems. Advances in Design and Control*. 2005.

[84] R. D. Russell and J. Christiansen. Adaptive mesh selection strategies for solving boundary value problems. *SIAM J Numer Anal.*, (15):59–80, 1978.

[85] D. V. Sarwate and M. B. Pursley. Crosscorrelation properties of pseudo-random and related sequences. *Proceedings of the IEEE*, 5(68):593–619, 1980.

[86] A. Schäfer, P. Kühl, M. Diehl, J. Schlöder, and H. G. Bock. Fast reduced multiple-shooting method for nonlinear model predictive control. *Chem. Eng. and Proc.*, 11(46):1200–1214, 2007.

[87] G. Schildbach and F. Borelli. Scenario model predictive control for lane change assistance on highways. *IEEE intelligent vehicle symposium*, pages 611–616, 2015.

[88] M. Schlegel, K. Stockmann, T. Binder, and W. Marquardt. Dynamic optimization using adaptive control vector parameterization. *Comp Chem Eng*, 8(29):1731–1751, 2005.

[89] P. Seferlis and A. N. Hrymak. Adaptive collocation on finite elements models for the optimization of multistage distillation units. *Chemical Engineering Science*, 9(49):1369–1382, 1994.

[90] S. W. Smith. *The scientist and engineer's guide to digital signal processing, Second edition*. 1999.

[91] H. J. Stetter. The defect correction principle and discretization methods. *Numer. Math.*, (29):425–443, 1978.

[92] J. Tamimi and P. Li. A combined approach to nonlinear model predictive control of fast systems. *J. Process Control*, 9(20):1092–1102, 2010.

[93] P. Tanartkit and L. T. Biegler. A nested, simultaneous approach for dynamic optimization problems-i. *Comput Chem Eng*, (21):735–741, 1996.

[94] P. Tanartkit and L. T. Biegler. A nested, simultaneous approach for dynamic optimization problems-ii: the outer problem. *Comput Chem Eng*, (21):1365–1388, 1997.

[95] A. L. Tits, A. Wächter, S. Bakhtiari, T. J. Urban, and C. T. Lawrence. A primal-dual interior-point method for nonlinear programming with strong global and local convergence properties. *SIAM J. Optim.*, 1(14):173–199, 2003.

[96] S. Vasantharajan and L. T. Biegler. Simultaneous strategies for optimization of differential-algebraic systems with enforcement of error criteria. *Comput Chem Eng*, (14):1083–1100, 1990.

[97] V. S. Vassiliadis, E. Balsa-Canto, and J. R. Banga. Second-order sensitivities of general dynamic systems with application to optimal control problems. *Chem. Eng. Sci.*, 17(54):3851–3860, 1999.

[98] V. S. Vassiliadis, R. W. H. Sargent, and C. C. Pantelides. Solution of a class of multistage dynamic optimization problems. 1. problems without path constraints. *Ind. Eng. Chem. Res.*, 9(33):2111–2122, 1994.

[99] V. S. Vassiliadis, R. W. H. Sargent, and C. C. Pantelides. Solution of a class of multistage dynamic optimization problems. 2. problems with path constraints. *Ind. Eng. Chem. Res.*, 9(33):2123–2133, 1994.

[100] A. Wächter and L. T. Biegler. On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming. *Math. Prog.*, 1(106):25–57, 2006.

[101] Y. Wang and S. Boyd. Fast model predictive control using online optimization. *IEEE Trans. Contr. Syst. Tech.*, 2(18):267–278, 2010.

[102] Z.-Q. Wang and B.-Y. Guo. Legendre-gauss-radau collocation method for solving initial value problems of first order ordinary differential equations. *J Sci Comput*, (52):226–255, 2012.

[103] I. D. Washington and C. L. E. Swartz. Design under uncertainty using parallel multiperiod dynamic optimization. *AlChE J.*, 9(60):3151–3168, 2014.

[104] D. P. Word, J. Kang, J. Åkesson, and C. D. Laird. Efficient parallel solution of large-scale nonlinear dynamic optimization problems. *Comput. Optim. Appl.*, 3(59):667–688, 2014.

[105] K. Wright. Adaptive methods for piecewise polynomial collocation for ordinary differential equations. *BIT Numerical Mathematics*, (47):197–212, 2007.

[106] K. Wright, A. H. Ahmed, and A. H. Seleman. Mesh selection in collocation for boundary value problems. *IMA Journal of Numerical Analysis*, (11):7–20, 1991.

[107] H. Yu, J. Duan, S. Taheri, H. Cheng, and Z. Qi. A model predictive control approach combined unscented kalman filter vehicle state estimation in intelligent vehicle trajectory tracking. *Advances in mechanical engineering*, 5(7):1–14, 2015.

[108] P. E. Zadunaisky. On the estimation of errors propagated in the numerical integration of odes. *Numer. Math.*, (27):21–39, 1976.

[109] V. M. Zavala, C. D. Laird, and L. T. Biegler. Interior-point decomposition approaches for parallel solution of large-scale nonlinear parameter estimation problems. *Chem. Eng. Sci.*, 19(63):4834–4845, 2008.

[110] Y. Zhao and P. Tsiotras. Density functions for mesh refinement in numerical optimal control. *J Guid Control Dyn*, 1(34):271–277, 2011.

Index