Inaugural-Dissertation

zur

Erlangung der Doktorwürde

der

Gesamtfakultät für Mathematik, Ingenieur- und Naturwissenschaften

der

Ruprecht-Karls-Universität

Heidelberg

vorgelegt von

Schrot, Ihno, M. Sc.

aus Gifhorn

Tag der mündlichen Prüfung:

.....

EFFICIENT NUMERICAL METHODS FOR NONLINEAR MODEL PREDICTIVE CONTROL WITH APPLICATIONS IN ADAPTIVE CRUISE CONTROL

Advisor:
Prof. Dr. Ekaterina A. Kostina

Zusammenfassung

Angesichts der steigenden Bedeutung von nachhaltiger Mobilität gewinnt die Entwicklung von fortschrittlichen Regelstrategien für Fahrzeuge zunehmend an Bedeutung. In dieser Arbeit leisten wir einen Beitrag zu diesen Bestrebungen, indem wir an effizienten numerischen Methoden arbeiten, die entscheidend für die Realisierung eines auf nichtlinearer modellprädiktiver Regelung (NMPC) basierenden ökologischen adaptiven Geschwindigkeitsregelungssystems (EACC) für Elektrofahrzeuge sind. EACC birgt als NMPC-Anwendung jedoch besondere Herausforderungen, die es vor der praktischen Anwendung dieser Methoden zu überwinden gilt. In dieser Arbeit entwickeln wir Lösungen für diese Herausforderungen.

NMPC ist eine fortschrittliche Regelungsstrategie, bei der in festen Abtastintervallen das Systemverhalten mithilfe eines mathematischen Modells vorhergesagt und optimiert wird. Zu jeder Abtastzeit wird dabei ein Optimalsteuerungsproblem (OCP) gelöst, welches durch den aktuellen Systemzustand parametrisiert ist. So können Störungen berücksichtigt und abgefangen werden. Eine etablierte Methode, um NMPC in Echtzeit zu ermöglichen, besteht darin, das OCP mithilfe der Mehrzielmethode (DMS) zu diskretisieren und anschließend mit dem Echtzeit-Iterationen (RTI) Schema oder dessen Erweiterung, den Multi-Level Iterationen (MLI), zu lösen. Das RTI Schema reduziert die Rechenzeit, indem es nur eine geringe Anzahl an Iterationen einer maßgeschneiderten Variante der Sequentiellen Quadratischen Programmierungsmethode (SQP) durchführt und dabei die Ähnlichkeit aufeinanderfolgender OCPs ausnutzt. Das MLI Schema steigert diese Effizienz weiter, indem es eine Hierarchie aus inexakten SQP-Iterationen aufbaut und bereits berechnete Ableitungen wiederverwendet. Dies ist für EACC besonders vorteilhaft, da die Steuerungen auf Hardware mit begrenzter Rechenkapazität schnell berechnet werden müssen.

Damit das Potential dieser Verfahren für die Realisierung eines NMPC-basierten EACC Systems voll ausgeschöpft werden kann, müssen wir uns zunächst mit der Herausforderung der Interpolation mehrdimensionaler Kennfelder (LUTs) beschäftigen. Kennfelder sind in realistischen Fahrzeugmodellen unverzichtbar. Bei deren Interpolation sollten relevante Datenformen wie Monotonie oder Konvexität erhalten werden. Gleichzeitig muss die Interpolation ausreichend glatt sein, um den Einsatz von ableitungsbasierten Optimierungsverfahren zu ermöglichen. In dieser Arbeit entwickeln wir eine Interpolationstechnik, die unseres Wissens nach die erste glatte, mehrdimensionale und formerhaltende Interpolationsmethode ist. Unsere Methode kann beliebige eindimensionale, glatte, formerhaltende Interpolationsverfahren auf mehrere Dimensionen erweitern.

Essentiell für eine reale Fahrzeugregelung ist es zudem, externe Einflüsse wie die Straßensteigung oder andere Fahrzeuge in den numerischen Methoden für NMPC zu berücksichtigen. Wir präsentieren neuartige Ansätze, die diese externen Einflüsse explizit in die DMS-Diskretisierung sowie in die RTI- und MLI-Schemata einbinden. Dies ermöglicht eine flexiblere Reaktion auf reale Fahrsituationen.

Aufbauend darauf entwickeln wir SensEIS Feedback, eine Strategie, die den beschränkten Rechenressourcen bei EACC Systemen Rechnung trägt. SensEIS Feedback reduziert den Online-Rechenaufwand, indem die Steuerungen während der Fahrt durch Ausnutzung von vorab bestimmten Lösungen für typische Fahrszenarien berechnet werden. Dabei wählt der Regler das am besten passende Szenario aus und berechnet die Steuerung durch wenige Matrix-Vektor-Multiplikationen oder durch das Lösen eines einzelnen Quadratischen Optimierungsproblems, wodurch die Feedbackverzögerung erheblich verringert wird.

Neben diesen algorithmischen Beiträgen erweitern wir zudem die Theorie zu inexakter NMPC, indem wir für eine Klasse semilinearer parabolischer partieller Differentialgleichungen (PDEs) asymptotische Stabilität von inexakten NMPC Methoden nachweisen. Damit schaffen wir die Grundlage für die Anwendung von RTI und MLI für NMPC bei Systemen, die mit PDEs modelliert werden. Im Kontext der Elektrofahrzeugregelung könnte dies unter anderem im Bereich des Thermomanagements Bedeutung erlangen.

Zur Bewertung des Potenzials unserer Methoden führen wir numerische Experimente mit realen Fahrdaten durch. Dabei verdeutlichen unsere Ergebnisse mit Einsparungen von über $3.4\,\%$ im Vergleich zu menschlichen Fahrern das beträchtliche Potenzial unserer Ansätze für eine nachhaltige Mobilität.

Abstract

The intensifying need for more energy-efficient transportation is driving the development of advanced control strategies for sustainable mobility. In this thesis, we contribute to these efforts by developing efficient numerical methods that are key to realizing an Ecological Adaptive Cruise Control (EACC) system for an electric vehicle based on Nonlinear Model Predictive Control (NMPC). As an NMPC application, EACC poses several challenges that must be addressed before these methods can be employed, and we tackle those challenges in this work.

NMPC is a closed-loop control strategy that uses a dynamical system's model to predict and optimize its behavior. The current system state parametrizes an Optimal Control Problem (OCP), which is solved at fixed sampling times to update the control, thereby reacting to disturbances as they occur. A well-established approach for real-time NMPC is to discretize the OCP via Direct Multiple Shooting (DMS), then solve it using the Real-Time Iterations (RTI) scheme or its extension, the Multi-Level Iterations (MLI) scheme. RTI cuts down on computational cost by carrying out a minimal number of tailored Sequential Quadratic Programming (SQP) iterations, exploiting similarities in consecutive OCPs. MLI extends this efficiency by creating a hierarchy of inexact SQP iterations, reusing previous derivative information – a design that is well-suited for EACC, where control updates must be computed quickly on hardware with limited resources.

To fully leverage these schemes in EACC, we must address the first major challenge: the interpolation of multivariate Lookup Tables (LUTs). LUTs are indispensable in realistic vehicle models and their interpolation must preserve vital data "shapes" like monotonicity or convexity. At the same time, the interpolation must be sufficiently smooth to enable derivative-based optimization. Addressing this, we propose what appears to be the first smooth multivariate shape-preserving interpolation method. Our method can extend any existing univariate smooth shape-preserving interpolation method to higher dimensions.

In addition to ensuring faithful interpolation of LUTs, we further address the treatment of external inputs, such as road elevation and the behavior of preceding vehicles – another essential aspect of realistic vehicle control. Our proposed approaches explicitly incorporate external inputs within the DMS discretization and the RTI and MLI schemes, granting more flexibility in reacting to real-world variations.

Building upon the external input incorporation, we introduce the Sensitivity and External Input Scenario based (SensEIS) feedback strategy, recognizing the limited computational resources in many automotive settings. SensEIS feedback reduces online computations by exploiting precomputed control responses for common driving scenarios. Online, the controller selects the best-matching scenario and updates the control using only a few matrix-vector multiplications or by solving a single Quadratic Program, thus significantly reducing feedback delay.

Alongside these algorithmic developments, we also extend the theory of inexact NMPC by proving asymptotic stability of inexact NMPC for problems modeled by a class of semilinear parabolic Partial Differential Equations (PDEs). This establishes a theoretical underpinning for extending the RTI and MLI schemes to PDE-governed problems. In electric vehicle control, NMPC of PDE-governed systems can be relevant for example in the context of thermal management.

Finally, to assess the potential of our methods, we conduct numerical experiments with real-world driving data. The results show energy savings of over $3.4\,\%$ compared to the human driver, indicating a significant potential of our methods for advancing sustainable mobility.

Acknowledgements

"This is ten percent luck, twenty percent skill
Fifteen percent concentrated power of will
Five percent pleasure, fifty percent pain
And a hundred percent reason to remember the name"

— Fort Minor and Styles of Beyond, *Remember the Name*, 2005*

... or rather "the names" of the people I would like to thank below, either for their support or for the great time I had with them during my PhD. I am very grateful to all of you!

First of all, I would like to thank my supervisor, Ekaterina Kostina, for giving me the opportunity to work on these exciting projects, for her constant support and her trust in me.

For the great cooperation I would like to thank Hans Georg Bock, Andreas Sommer, Julian Niederer, Andrea Flexeder, Christian Fleck, Matthias Bitzer, Christian Bertsch and Christoph Hansknecht in the context of the EACC project, Manuel Schaller and Karl Worthmann in the context of the stability analysis and Andreas Potschka in both projects. A special thanks goes to Manuel Schaller for patiently explaining the details of PDE-constrained optimal control to me, taking the time to give me thorough feedback on my work and supporting me even after the stability proof was done, to Hans Georg Bock for inspiring me to work in the field of optimal control in the first place, for teaching me a great deal about it, and for the initial spark for SensEIS feedback, and to Andreas Sommer for helping me with our MLI software.

I owe another great deal of thanks to Hridya Vinod Varma, Manuel Weiß, Marta Sauter, Manuel Schaller, Christian Alber, Julian Niederer, Andrea Flexeder, Dominik Cebulla and Niels Wächter for taking the time to read earlier drafts of my thesis and providing valuable feedback. This has greatly improved the quality of my thesis.

I would like to thank Herta Fitzer and the team at the doctoral office for their administrative support. I would also like to thank the teams at the counselling services that have helped me in the past years.

Furthermore, I gratefully acknowledge the support of the Bundesministerium für Bildung und Forschung (BMBF) in the research project *Modellierung, Optimierung und Regelung vernetzter Fahrzeuge und Fahrzeugflotten mit heterogenen Antriebstechnologien in Echtzeit* (MORFAE).

Thanks also to all the students I had the pleasure of teaching and supervising - which I really enjoyed - and whose positive feedback kept me motivated beyond the teaching itself.

Throughout my PhD I have been very fortunate to spend a lot of time with a lot of great people and I feel that I will never be able to do justice to all of you in the following lines. So please forgive me if I do not give you the credit you deserve, and know that I really appreciate the time we spent together!

Coming to the Mathematikon in the morning and working here – and having coffee and lunch breaks – has always been a pleasure for me, thanks to all the friends and colleagues, especially but not only from the *SimOpt*, *NumOpt*, *Scoop* and the *Numerical Analysis and UQ* groups.

A big thank you to my friends from tennis, climbing and the Wasseralm crew for helping me unwind after a hard day or period of work. Further, I want to express my gratitude to the "chemists" for sharing their kindness, humour and good times with me. This thanks particularly goes to Franka, whom I additionally wish to thank for numerous shared active and relaxed moments and meals. Speaking of meals, the Görtz breakfasts with Leo, Franka, Melli, Tobi and all the others of you who joined were and still are one of my highlights every week.

*This song was played at the circuit training sessions of the University sports that I attended almost every Thursday evening during my time as a PhD student. And even though the percentages don't really describe my PhD journey accurately, at some point I started joking that these lyrics would make a great and unique start to my acknowledgements when I finished this thesis. Especially as it is one of my all-time favourite songs. In fact, I joked about using this quote so many times that I finally decided to actually do it, so here we are. On this occasion: Thanks to Julius, Franka and Leo for the fun circuit training sessions!

Moreover, I wish to extend my heartfelt thanks to my long-time friends, Niels, Fenja, Marvin, Charlotte, Maike and Lorenzo for supporting me throughout my PhD journey. I owe you not only many great memories and moments of joy, but also the motivation to keep going when things got tough. Thank you for being there for me, successfully lifting my spirits and reminding me that there is more to life than a PhD. I think it's fair to say that this work would not have been possible without you. This is especially true for Jasper, whose support and encouragement I have always been able to count on, and for which I cannot thank him enough.

To all my friends: I look forward to all our future moments together, be it in the mountains, on the tennis court or wherever life takes us.

To my sister Gesa, thank you for just being a great sister, for making me laugh and for your words of advice and encouragement whenever I needed them (and maybe even more so when I didn't). To my parents, I am eternally grateful for your unconditional support and love throughout my life. I am so grateful for the values you have instilled in me and the opportunities you have given me.

Finally, I would like to thank my girlfriend Leonie. I am incredibly grateful for your patience, support and love, especially during the times when I was stressed or overwhelmed. Full of joy, I am looking forward to our next steps into the future!

A Quick Guide to Reading This Thesis

Inspired by the PhD theses of Théo Winterhalter and Anja Petković Komel, this thesis is written in a 1.5 column format. In this format, we have a wide margin. The wide margin accommodates accompanying comments, citations, reminders, etc.

This allows us to keep the main text clean and focused on the main arguments. In particular, we can provide additional information and remind the reader of previously explained concepts without disrupting the flow of the main text. Moreover, the reader does not have to frequently jump back and forth in the document, e.g., to look up an interesting reference. Finally, the wide margin allows the reader to jot down their own notes or calculations – something I personally do a lot when reading mathematical texts.

While the previous features are particularly useful for the paper version of the thesis, the electronic version has some additional features for convenient electronic reading. The references to definitions (like Definition 2.1), theorems (like Theorem 4.3), equations (like Equation (6.4)), and other objects are clickable so one can quickly travel to the relevant part of the document.

Citations like [208] appear in the margin (also clickable) in a short version listing just the first author, as well as in the Bibliography where the full citations can be found. Comments that accompany the main text are placed close to the text for which they provide additional information. If we wish to remind the reader of a concept, we use a reminder like the one to the right. Sometimes we repeat the definitions of commonly used mathematical concepts. In such cases, we place a box with a green header on the side. Finally, we place small illustrations in the margin to help the reader visualize the concepts we are discussing.

Similarly, as in most mathematical texts, we use different environments for theorems, definitions, lemmas, and examples that make them stand out. In this document, they look as follows.

Theorem 0.1 Theorems and main ideas stand out in these boxes with a red title, so they are easy to spot.

Corollary 0.1 Lemmas and corollaries, which accompany a theorem, appear similar to theorems, but with a black title.

Definition 0.1 Definitions and assumptions appear with a red bar next to them. The *defining notions* will appear in *red*.

▪ **Example 0.1** This is an insightful example. ▪

[208]: Zanelli et al. (2021), "A Lyapunov function for the combined system-optimizer dynamics in inexact model predictive control"

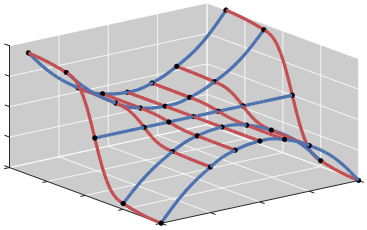
This is an example of an accompanying comment that provides helpful, but not essential, additional information for the main text.

Reminder: a concept

This is a reminder of a concept that we have encountered previously.

Definition: side definition

This is a *side definition*.



An illustrative figure, which we will encounter as Figure 5.4.

Contents

1. Introduction

On the very day we started writing the introduction to this thesis, the Copernicus Climate Change Service published its annual climate summary report for the year 2024 [48]. Its opening words are:

> *"2024 saw unprecedented global temperatures, following on from the remarkable warmth of 2023. It also became the first year with an average temperature clearly exceeding $1.5\,°C$ above the pre-industrial level - a threshold set by the Paris Agreement to significantly reduce the risks and impacts of climate change. Multiple global records were broken, for greenhouse gas levels, and for both air temperature and sea surface temperature, contributing to extreme events, including floods, heatwaves and wildfires. These data highlight the accelerating impacts of human-caused climate change."*
>
> — Copernicus Climate Change Service [48]

Moreover, the report states that

> *"One or two years that exceed $1.5\,°C$ above the pre-industrial level does not imply that the Paris Agreement has been breached. However, with the current rate of warming at more than $0.2\,°C$ per decade, the probability of breaching the $1.5\,°C$ target of the Paris Agreement within the 2030s is highly likely."*
>
> — Copernicus Climate Change Service [48]

The message of these words could not be clearer. The situation is urgent. And we, as scientists, have a special responsibility in the efforts to mitigate human-caused climate change and to adapt to the adverse impacts of climate change. This responsibility is also reflected in Article 4 of the Paris Agreement [192].

With this in mind, the German Bundesministerium für Bildung und Forschung (BMBF) (engl.: Federal Ministry of Education and Research) has developed a strategy for the research for sustainability [34], the first goal of which is to achieve the climate goals of the Paris Agreement. One of the actions outlined in the BMBFs strategy is to ensure sustainable mobility in urban and rural areas. A look at Germany's energy consumption underscores the importance of this action. The German Umweltbundesamt (engl.: Federal Environmental Agency) reported an energy consumption for Germany for the year 2023 of about $2368\,\mathrm{TW\,h}$ [191]. Approximately $29.4\,\%$ of this consumption is attributed to the transportation sector. The development of novel techniques to achieve energy savings in this sector is therefore especially urgent and relevant.

A key enabler in this effort whose great potential the BMBF wants to tap is Mathematical Modeling, Simulation and Optimization (MSO).

[48]: Copernicus Climate Change Service (2024), *Global Climate Highlights 2024*

[192]: United Nations Framework Convention on Climate Change (UNFCCC) (2015), *Paris Agreement*

[34]: Bundesministerium für Bildung und Forschung (BMBF) (2020), *Forschung für Nachhaltigkeit*

[191]: Umweltbundesamt (2024), *Energieverbrauch nach Energieträgern und Sektoren*

MSO offers a powerful set of tools for improving efficiency and reducing energy consumption and greenhouse gas emissions across various sectors.

One possibility to leverage MSO to contribute to ensure sustainable mobility is the development of an Ecological Adaptive Cruise Control (EACC) system. EACC is an Advanced Driver-Assistance System (ADAS) for vehicles that enables more efficient driving by harnessing knowledge about the road and traffic ahead and the ideal operating points of a vehicle. For a recent review on the development of EACC in general, see [148]. In particular, improvements for EACC can be put into practice in the near future and still have a lot of potential.

[148]: Pan et al. (2022), "A review of the development trend of adaptive cruise control for ecological driving"

To successfully drive a car in reality, we need to be able to react to unforeseen disturbances while we are driving. Therefore, it is not enough to compute an energy-efficient control profile, such as speed or acceleration, in advance and then sticking to it without adjustments while driving. This approach would be called an offline or open-loop control. Instead, we need to compute our control in an online or closed-loop fashion which means that we continuously recompute our control taking into account the current situation. Developing suitable methods to achieve such a control is what process control is about. A large number of different methods have been developed in process control. An advanced method that is particularly powerful is Model Predictive Control (MPC) or Nonlinear Model Predictive Control (NMPC) when we consider nonlinear problems. The main idea of NMPC, which is an adaptation of the summary given in [101, Section 3.1], is as follows. At each time point of a given time grid, we optimize the predicted future behaviour of the system under consideration over a finite time horizon and apply the first element of the resulting control sequence until the next time point. The particular strength of the NMPC approach is that it allows to

[101]: Grüne et al. (2017), *Nonlinear Model Predictive Control*

- minimize an objective

by

- using a process model

while

- obeying constraints
- and being a closed-loop control.

We will explain the basic principle and algorithm of NMPC in detail in Section 2.1.

The large number of successful applications of NMPC in the literature testifies to its potential. We will provide more references to applications of MPC and NMPC in general in Chapter 2 and to the application of NMPC to ADASs and EACC-like systems in Chapter 8 and for now only mention the survey [175, Section 6] for an impressive and recent list of MPC applications.

[175]: Schwenzer et al. (2021), "Review on model predictive control: an engineering perspective"

The main challenge we have to overcome if we want to put NMPC-based EACC into practice is to solve the challenging Optimal Control Problems (OCPs) fast enough. Fortunately, the development of efficient numerical methods for real-time feasible NMPC in the last two decades has brought the use of NMPC in real-time within reach. In particular, the development of the Real-Time Iterations (RTI) scheme

by DIEHL and coworkers [57, 59, 61, 64, 67] and its extension the Multi-Level Iterations (MLI) scheme by WIRSCHING and coworkers [28, 200], both based on the Direct Multiple Shooting (DMS) method presented in [29], have led to drastic reductions in computing time.

The overall goal of this thesis is to find solutions to the challenges that still need to be overcome if we want to leverage the MLI scheme to realize an NMPC-based EACC in practice. In the following, we first present the detailed objectives of this thesis that result from this overall objective. Then, we summarize the main contributions that we develop in this thesis and outline the structure of this thesis.

1.1. Objectives

In order to achieve the overall goal of this thesis, we must develop a problem formulation that represents an EACC system and then solve it using our efficient numerical methods for NMPC. It is important that our problem formulation involves the main challenges such that our problem forms a representative example. Only then we can argue that our numerical methods are indeed suited for a real-world deployment if they perform well for this example. To this end, the first objective of this thesis is to develop a vehicle model and an OCP formulation that together are suitable for a real-life EACC system. In particular, this means incorporating multivariate Lookup Tables (LUTs) and external inputs such as road elevation and the speed of a preceding vehicle (PPO) into both the vehicle model and the OCP formulation, as these are key components in real-life problems.

This leads directly to the next two objectives. We need to develop a smooth shape-preserving interpolation method for multivariate LUTs so that they can suitably evaluated in the optimization process. We also need to add new, more sophisticated approaches to handle external inputs in the DMS method and the RTI and MLI schemes.

The next objective is to develop a new feedback method that is even faster than the established levels of the MLI scheme by exploiting the fact that many driving situations are recurrent and can be solved in an offline fashion. It should be possible to couple this feedback method with the MLI scheme.

In addition to the availability of fast numerical methods, the reliability of these numerical methods is of paramount importance for the control of real vehicles. For the control of dynamical systems that can be modeled with Ordinary Differential Equations (ODEs), the corresponding theory is already well established. As a first step towards laying the foundation for the extension of our numerical methods to dynamical systems modeled by Partial Differential Equations (PDEs), the final objective of this thesis is to prove stability of inexact NMPC for a class of semilinear parabolic PDEs.

[57]: Diehl (2001), "Real-time optimization for large scale nonlinear processes"
[67]: Diehl et al. (2001), "Real-time optimization for large scale processes: Nonlinear model predictive control of a high purity distillation column"
[61]: Diehl et al. (2002), "Real-time optimization and nonlinear model predictive control of processes governed by differential-algebraic equations"
[59]: Diehl et al. (2003), "Newton-type methods for the approximate solution of nonlinear programming problems in real-time"
[64]: Diehl et al. (2005), "Nominal stability of real-time iteration scheme for nonlinear model predictive control"
[200]: Wirsching (2018), "Multi-level iteration schemes with adaptive level choice for nonlinear model predictive control"
[28]: Bock et al. (2007), "Constrained Optimal Feedback Control of Systems Governed by Large Differential Algebraic Equations"
[29]: Bock et al. (1984), "A Multiple Shooting Algorithm for Direct Solution of Optimal Control Problems"

In summary, the objectives of this thesis are

(i) to develop a realistic EACC problem formulation consisting of a vehicle model and an OCP formulation including multivariate Lookup Tables (LUTs) and external inputs, and to solve it using our numerical methods for NMPC,
(ii) to develop a smooth shape-preserving interpolation method for multivariate LUTs,
(iii) to explicitly incorporate external inputs into the DMS method and the RTI and MLI schemes,
(iv) to develop a novel feedback method that makes use of offline solved scenarios and is faster than the established MLI levels but can be coupled with the MLI scheme,
(v) to extend the existing stability results for inexact NMPC to a class of semilinear parabolic PDEs.

1.2. Contributions

The main contributions of this thesis that we develop in the process of working towards the above goals are the following.

A stability proof for inexact NMPC for a class of semilinear parabolic PDEs

When we use schemes such as RTI or MLI to compute the controls in an NMPC setting, we are performing inexact NMPC. The approach is called inexact NMPC because we are not solving the resulting OCP to optimality and are using approximate solutions. A particularly important question is whether the resulting controller is capable of steering the dynamical system of interest to a desired state. This question is not only relevant from a control theoretical point of view, but also for practitioners who need to know whether a controller is reliable. For dynamical systems that can be modeled using ODEs, this question is answered for a wide range of NMPC formulations. For inexact NMPC of systems described by PDEs, however, we are not aware of any existing work. With the stability proof that we present in Chapter 4, we thus take the first steps in this area of research.

We study the interplay between the system dynamics and the dynamics of the optimizer, i.e., the system-optimizer dynamics. The system dynamics in our case are given by a class of semilinear parabolic PDEs similar to those considered in seminal works, such as [44, 160], on the control of semilinear parabolic equations. The formulation of the OCPs under consideration is also similar to those in these works. We eventually construct a forward invariant set on which we prove that the origin is an asymptotically stable point of the system-optimizer dynamics. We also give an appropriate Lyapunov function.

[44]: Casas (1997), "Pontryagin's Principle for State-Constrained Boundary Control Problems of Semilinear Parabolic Equations"
[160]: Raymond et al. (1999), "Hamiltonian Pontryagin's Principles for Control Problems Governed by Semilinear Parabolic Equations"

A smooth multivariate shape-preserving interpolation method

For both vehicle dynamics modeling and OCP formulation for a realistic EACC problem, it is common practice in engineering to use LUTs. In the vehicle model, LUTs from measurements are used for dependencies where no functional expression is available. In the OCP formulation, LUTs are used to fine-tune the behavior of the vehicle to ensure a comfortable ride for the driver. For the optimization, we need to interpolate the LUTs to be able to evaluate the dependency at points that are not included as data points in the LUTs. Often the data in the LUTs have certain characteristics, such as monotonicity or convexity. Such features or shapes should be preserved by the interpolation. In addition, the interpolating function must be smooth enough for our derivative-based optimization methods. While many smooth shape-preserving interpolation methods are available for uni- and bivariate LUTs, such methods are not yet available for multivariate LUTs. In Chapter 5 of this thesis we develop such a smooth multivariate shape-preserving interpolation method. In particular, we contribute to the state of the art by

- proposing a classification scheme for shape-preservation in the multivariate case,
- developing a method to extend any smooth shape-preserving univariate interpolation schemes to the multivariate case using a multivariate blending scheme that is inspired by Coons' patches,
- proving the interpolation and shape-preservation property of our proposed method for the general multivariate case,
- proving the required smoothness property exemplarily for the trivariate case,
- and providing numerical results that in particular numerically validate the smoothness property also in the quadrivariate case.

Explicit incorporation of external inputs into the DMS method and the RTI and MLI schemes

When we want to control a vehicle in a real-world scenario, we have to take several external inputs into account. For example, the slope of the road affects how much power we need to maintain a certain speed, or the behavior of a vehicle in front of us limits the speed at which our vehicle can safely travel. What is special about these external inputs is that they are functions of the free variable, i.e., time or position. The common approach to incorporate external inputs into the DMS method, and thus the RTI and MLI schemes, is to add an artificial differential state equal to the free variable, thereby reformulating the external inputs as functions of the differential states. However, other approaches have not been thoroughly explored. In Chapter 6 we contribute a novel approach to incorporate external inputs into DMS, RTI and MLI. In particular, we

- propose a novel discretization strategy for external inputs within the framework of the DMS method,
- adapt the state, constraint, and objective function discretizations and the resulting Nonlinear Program (NLP),
- present two affiliated new strategies for incorporating external inputs into both the RTI and MLI schemes,
- compare their interpretations,
- and develop an adapted condensing strategy for the Quadratic Program (QP) that arises in the second strategy.

A Sensitivity and External Input Scenario based feedback strategy

To realize an effective NMPC-based EACC system, it is paramount to reduce the computation time required to update the controller as far as possible. While the MLI scheme has already enabled drastic reductions in the computation times, one potential for even shorter computation times has not yet been exploited. This potential arises from the fact that many driving situations recur and can be precomputed. From a more general point of view, the idea is to solve the OCPs that occur during NMPC for certain choices of external inputs and constant parameters, so-called scenarios, in an offline phase and then to use this information to compute feedback online in a very fast way. From this idea, we develop a feedback strategy in Chapter 7. Our feedback strategy is based on the sensitivity theorem for families of NLPs. We develop two variants of this feedback strategy, which we call Sensitivity and External Input Scenario based (SensEIS) feedback. In the first variant, we set up a feedback matrix for each scenario, which allows us to compute feedback with essentially a single matrix-vector product. In the second, we set up a feedback-generating QP. In the second variant, we thus have to solve a QP, but are able to treat active set changes. Both strategies build on our novel strategies for handling external inputs. Our contributions here include

- the development of a new Sensitivity and External Input Scenario based (SensEIS) feedback strategy,

which includes

- the construction of a feedback matrix and a feedback-generating Quadratic Program (QP) for each scenario, mapping changes in external inputs and constant parameters to a new control, and examining their respective structure resulting from the discretization using the DMS method with external inputs,
- the development of an adapted condensing strategy for the feedback generating QP,
- the introduction of a scenario selection technique that reduces chattering caused by repeatedly switching between scenarios,
- the presentation of a step size strategy that helps to avoid inequality violations when using the feedback matrix,
- a first approach on how the novel SensEIS feedback can be combined with the MLI scheme.

Problem formulation of and numerical results for a realistic EACC system

Our final major contribution is to set up a vehicle model and OCP formulation that serves as a challenging and representative example that, when solved, provides a strong argument that our numerical methods are suitable for implementing a real-world EACC system. We then put together the pieces from our previous contributions to successfully solve the resulting NMPC problem. More specifically, our contributions here are to

- set up a vehicle model that makes use of realistic LUTs,
- set up an OCP formulation that makes use of realistic LUTs and minimizes the energy consumption of our vehicle, leads to a comfortable ride for the driver, and considers safety constraints,
- use our novel interpolation method from Chapter 5 to interpolate the LUTs,
- perform numerical experiments for a selection of the numerical methods discussed and developed throughout this thesis, in which we use data from a real measurement drive for the external inputs,
- and demonstrating with our numerical experiments the potential to save more than $3.4\,\%$ of energy compared to the preceding vehicle from the measurement drive with only negligible constraint violations [1].

1: Only the speed limit is violated - with a maximum violation of about $0.000\,255\,\mathrm{m\,s^{-1}}$.

1.3. Thesis outline

This thesis consists of two main parts. In the first part, we provide the mathematical background necessary to present our contributions in this thesis. The contributions themselves are presented in the second part of the thesis. The main part of this thesis ends with a conclusion and outlook in Chapter 9. The Appendix includes auxiliary material.

Part 1: Mathematical Background

The control approach of interest throughout this thesis is NMPC. Therefore, we begin this thesis by explaining the basic principle and algorithm of NMPC in Chapter 2. In addition to that, we present stability results related to NMPC, which we later need for our own stability proof in Chapter 4. The OCPs that arise during NMPC are infinite-dimensional optimization problems. To implement NMPC in practice, we need methods to solve these efficiently. The subsequent Chapter 3 therefore revolves around efficient numerical methods for NMPC. In particular, we first give an overview of solution approaches for OCPs. Afterwards, we explain the efficient numerical methods on which our own contributions are based. This specific sequence of methods consists of the DMS as a parameterization method, a Sequential Quadratic Programming (SQP) method tailored to the resulting NLPs, and then the RTI and MLI schemes as highly efficient further developments tailored to the use in NMPC.

Part 2: Contributions

In Chapter 4 we first establish stability of inexact NMPC for a class of semilinear parabolic PDEs which is a result of a rather foundational nature. The subsequent chapters then contain the building blocks that we develop and need to solve our desired application problem. The first building block is the smooth multivariate shape-preserving interpolation method that we need to interpolate the LUTs. This interpolation method is introduced in Chapter 5. Next, in Chapter 6 we turn our attention to the handling of external inputs in the DMS method and the RTI and MLI schemes. Closely related to the treatment of external inputs is SensEIS feedback, which we develop in Chapter 7. Finally, we set up the EACC application and report numerical results of our numerical methods for it in Chapter 8.

Appendix

In Appendix A we prove a statement given in Example 3.2. Appendix B contains the proof of the claimed smoothness property for our novel multivariate shape-preserving interpolation method from Chapter 5 for the trivariate case. Auxiliary material for Chapter 6 in the form of the structure of the relevant Jacobians and Hessians occurring in the QPs (6.10) and (6.9) and the condensing procedure for QP (6.10) is contained in Appendix C and Appendix D. Finally, Appendix E complements Chapter 7 by presenting the adapted condensing procedure for the QP (7.23).

Mathematical Background

2. Nonlinear Model Predictive Control

When developing an Ecological Adaptive Cruise Control (EACC) system, we must consider several requirements. Firstly, we want to ensure that the resulting driving behavior is ecological, meaning we aim to minimize objectives such as total energy consumption or emissions. Additionally, vehicles differ from each other, so a speed or control profile that is efficient for one vehicle may be inefficient for another. Therefore, we would like to take into account a *model* of the vehicle's dynamics, including both the powertrain and motion dynamics. Of course, we must also comply with legal and safety *constraints*, such as speed limits and maintaining a safe distance.

If the list of requirements ended here, this scenario would be a prime example of optimal control. However, we have not yet mentioned one of the most important requirements. A vehicle on the road is constantly influenced by surrounding traffic, so we need to react quickly to disturbances. For example, we must adjust our control if our vehicle needs to slow down unexpectedly due to a vehicle in front of us. In other words, we need *closed-loop control* instead of open-loop control. In short, we need a control strategy that allows us to:

- minimize an objective by
- utilizing a process model while
- obeying constraints
- and is closed-loop.

Optimal control is a broad subject with a large body of both theory and applications. In addition, MPC has its origins in optimal control [101, p. 4], [159, p. 1]. We refer the interested reader to [6, 21, 88, 131, 157, 184, 196].

While other popular controllers such as Proportional-Integral Derivative (PID) [9] or fuzzy controllers [185] are closed-loop, they either cannot handle constraints or do not leverage process models. A control strategy that satisfies all four main requirements is *Model Predictive Control (MPC)*. When considering nonlinear systems, we refer to it as *Nonlinear Model Predictive Control (NMPC)*.

[9]: Åström et al. (1995), *PID Controllers: Theory, Design, and Tuning*
[185]: Tsoukalas et al. (1997), *Fuzzy and neural approaches in engineering*

Both MPC and NMPC have been successfully applied to a wide range of problems. In the early 2000s, a survey by Qin and Badgwell [156] reported more than 4600 MPC applications by various vendors, indicating the potential of MPC for industrial applications. A more recent and impressive list of MPC applications is presented in [175, Section 6]. Specifically, MPC techniques have also been applied to vehicle control, as demonstrated by [30, 31, 112, 118, 136, 137, 162, 199, 206, 209]. We will reference works where MPC has been applied to Ecological Adaptive Cruise Control (EACC) in Section 8.1.

[156]: Qin et al. (2003), "A survey of industrial model predictive control technology"

[175]: Schwenzer et al. (2021), "Review on model predictive control: an engineering perspective"

Given the extent of the research on MPC, or even just on NMPC, a comprehensive presentation of NMPC is beyond the scope of this work. Instead, we provide a concise presentation of NMPC tailored to offer the background knowledge needed to present our contributions. For more comprehensive presentations of MPC, we refer to [40, 101, 124, 159, 164, 197].

For an overview of the history of MPC development, we recommend the paper [126]. In this chapter, we limit ourselves to explaining the basic principle and algorithm in Section 2.1 and presenting stability results in Section 2.2 relevant to our contribution in Chapter 4.

[126]: Lee (2011), "Model predictive control: Review of the three decades of development"

2.1. Basic principle and algorithm

We start our presentation of NMPC by stressing that NMPC is not a specific method. Instead, NMPC is an overarching idea that can be developed into a multitude of methods. The main source of variation is how the control sequence is computed. The different methods can often be grouped into subcategories of NMPC, e.g., Economic MPC, Robust MPC, Output MPC, Distributed MPC, or inexact MPC. For details on these subcategories, we refer to [159].

We use only the term NMPC from now on, even though large parts of the upcoming explanations are not specific to NMPC but also apply to MPC. Where a clear distinction is necessary, we will highlight this.

[159]: Rawlings et al. (2022), *Model predictive control: Theory, computation, and design*

Our presentation of NMPC is already tailored in some details such that our NMPC method fits well into our NMPC framework as we use it throughout this thesis. For a more general presentation of NMPC, we recommend the excellent textbook [101], which we will refer to several times. To summarize the main idea of NMPC, we adapt the summary given in [101, Section 3.1].

Our NMPC methods that we develop throughout this thesis fall into the category of inexact Economic NMPC.

[101]: Grüne et al. (2017), *Nonlinear Model Predictive Control*

Main idea — Nonlinear Model Predictive Control (NMPC). At each time point of a given time grid, we optimize the predicted future behavior of the system of consideration over a finite time horizon and apply the first element of the resulting control sequence until the next time point.

In other words, we repeat in NMPC the following main steps at each sampling time t^j:

(i) Obtain the current state x^j.
(ii) Compute an optimal control sequence $\left(u_0^j, \dots, u_N^j\right)$ that optimizes the predicted future behavior of the system over a finite prediction horizon $I_{\text{hor}}(t^j)$.
(iii) Apply the first element of the control sequence as the feedback value u^j until the next sampling time t^{j+1}.

For us, optimizing the predicted future behavior means setting up and solving an Optimal Control Problem (OCP), cf. Subsection 2.1.3.

This main idea of NMPC is also illustrated by Figure 2.1.

In the following subsections, we properly define the used expressions and fill the main idea with more mathematical details. First, we describe the general setup, including, for example, the aforementioned time grid, and most of the relevant terminology in Subsection 2.1.1. Then, we turn our attention in Subsection 2.1.2 to how we can utilize a given Ordinary Differential Equation (ODE) model of our system in the NMPC framework to predict the future. Afterwards, we shed light on the optimization process in NMPC in Subsection 2.1.3. Finally, we present Algorithm 2.1 in Subsection 2.1.4 that formalizes the above main loop and summarizes our view on the NMPC framework as we utilize it in later chapters.

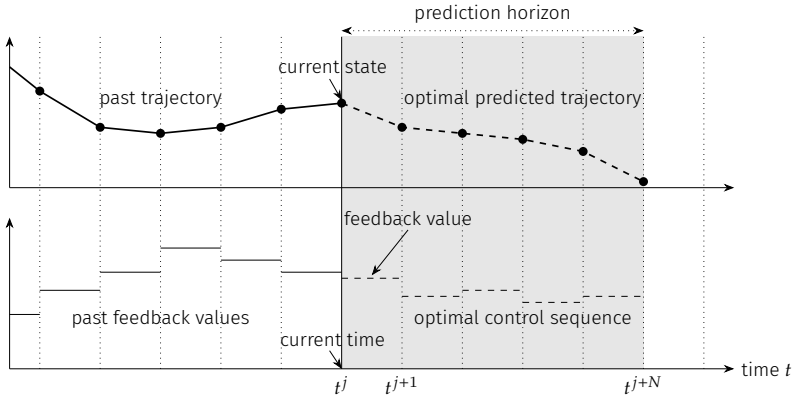


Figure 2.1.: Illustration of the NMPC scheme adapted from [101, Fig. 1.1]. At the current sampling time, we obtain the current state, compute an optimal control sequence that leads to an optimal predicted trajectory, and use the first element of the control sequence as the feedback value.

2.1.1. General setup and terminology

To do NMPC, we update our current control for the system at each time point of a time grid $0 = t_0 < t_1 < \ldots < t^j < \ldots$. Depending on the application, the time grid can either have a finite endpoint, or we can control the system indefinitely. Moreover, heterogeneous time grids are also possible. In some cases, the time grid is not even predetermined. In most cases, however, a predetermined homogeneous time grid is used. We will also focus on this setting and consider infinite time grids, i.e., we consider $j \in \mathbb{N}_0$. We describe homogeneous time grids using the following terminology.

Definition 2.1 In a homogeneous time grid, the *sampling times* t^j are given as

$$t^j = j \cdot T, \ j \in \mathbb{N}_0,$$

with a *sampling period* $T > 0$. We call $[t^j, t^{j+1})$ the *j-th sampling interval*.

For simplicity, we also focus on the case where the prediction horizon ends on a sampling time, i.e., we define:

Definition 2.2 With a slight overload of notation, we refer by *horizon length* to both $2 \leq N \in \mathbb{N}$ and $T_{\text{hor}} := N \cdot T$. The *prediction horizon at sampling time t^j* is defined as the interval

$$I_{\text{hor}}(t^j) := [t^j, t^{j+N}).$$

With the definition of the prediction horizon in place, we can define what we mean by control sequence in the main idea of NMPC.

Definition 2.3 Let the *control space* U and the *control sequence space* U^N be arbitrary metric spaces. Optimizing the system behavior over the prediction horizon $I_{\text{hor}}(t^j)$ yields a *control sequence* $\left(u_0^j, \ldots, u_N^j\right) \in U^N$, where each element u_k^j, $k = 0, \ldots, N$, is associated with the sampling interval $[t^{j+k}, t^{j+k+1})$. We set the *feedback value* $u^j \in U$ at sampling time t^j to the first element of the control sequence, i.e.,

$$u^j := u_0^j. \tag{2.1}$$

We do not require $U^N = U \times U \times \ldots \times U$.

As mentioned at the beginning of this section, NMPC methods vary in the way that the control sequence is computed. Therefore, we avoid a more formal description of how the optimization procedure yields a control sequence.

We continue by specifying the evolution of the system.

In this thesis, except for Chapter 4, we have $X = \mathbb{R}^{n_x}$ and $U = \mathbb{R}^{n_u}$.

In simple terms, the transition map θ is our model of how the system evolves from one state to the next. θ can be rather general and does not even have to be continuous [101, p. 13].

Definition 2.4 We denote the *system state* at sampling time t^j by $x^j \in X$. Again, the *state space* X can be an arbitrary metric space. The current state x^j, its successor x^{j+1}, and the feedback value u^j are related through the *transition map* $\theta\colon X \times U \to X$ by

$$x^{j+1} = \theta(x^j, u^j). \tag{2.2}$$

So far, we have described how the current states and controls are related to each other in a general way. In this general formulation, there are two main components that we have to specify in order to perform NMPC. The first component is the transition map θ. The second one is the optimization procedure that yields the control sequence. We discuss these components in the following subsections.

2.1.2. Choice of the transition map for continuous control systems

With this general formulation, in particular with respect to the transition map, we are so far describing NMPC for general discrete-time systems, cf. [101, Section 2.1]. In many applications, including our EACC system that we present in Chapter 8, however, we are considering dynamical systems, i.e., systems where we model the evolution of the system using an ODE. In this case, we assume that the evolution of the state x can be modeled using an Initial Value Problem (IVP) of the form

$$\begin{aligned} \dot{x}(t) &= f(x(t), u(t)), \quad t \in [t_{\text{start}}, t_{\text{final}}] \subset \mathbb{R}, \\ x(t_{\text{start}}) &= x_{\text{init}}, \end{aligned} \tag{2.3}$$

with x_{init}, $x(t) \in \mathbb{R}^{n_x}$, $u(t) \in \mathbb{R}^{n_u}$ for all $t \in [t_{\text{start}}, t_{\text{final}}]$, where n_x, $n_u \in \mathbb{N}$, and with a vector field $f\colon \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_x}$.

[101]: Grüne et al. (2017), *Nonlinear Model Predictive Control*

In Chapter 4 we use a semilinear parabolic partial differential equation instead of an ODE. Also, Differential Algebraic Equations (DAEs) are possible.

We refer to this setting as the sampled data case, cf. [101, Section 2.1]. The idea of the so-called sampling is now to identify the transition map θ with the solution of the IVP (2.3). To be able to reasonably do so, we have to make sure that a solution of the IVP (2.3) exists and that it is unique. This can be guaranteed by means of Carathéodory's Theorem, see e.g. [178, Theorem 54, p. 476]. To that end, we will from now on assume that the prerequisites of Carathéodory's Theorem are satisfied. In particular, we make the slightly stronger Assumption 2.1 for f as it is formulated in [101, Assumption 2.4]. Assumption 2.1 allows applying Carathéodory's Theorem to the IVP (2.3), cf. [101, p. 16].

[178]: Sontag (1998), *Mathematical Control Theory*

Assumption 2.1 The vector field $f\colon \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_x}$ is continuous and Lipschitz in its first argument in the following sense: for each $r > 0$ there exists a constant $L(r) > 0$ such that the inequality

$$\left\| f(x^1, u) - f(x^2, u) \right\| \leq L(r) \left\| x^1 - x^2 \right\|$$

holds for all x^1, $x^2 \in \mathbb{R}^{n_x}$ and all $u \in \mathbb{R}^{n_u}$ with $\left\|x^1\right\| < r$, $\left\|x^2\right\| < r$ and $\|u\| < r$.

In this thesis $\|\cdot\|$ denotes the Euclidean norm.

Furthermore, CARATHEODORY's Theorem requires that the continuous control $u(\cdot)$ is a locally LEBESGUE integrable function, i.e., we from here on assume that $u(\cdot) \in L^\infty([t_{\text{start}}, t_{\text{final}}], \mathbb{R}^{n_u})$. In NMPC, a common choice is to use piecewise constant controls. As piecewise constant functions are a subset of $L^\infty([t_{\text{start}}, t_{\text{final}}], \mathbb{R}^{n_u})$, CARATHEODORY's Theorem guarantees the existence of a unique solution also for this type of control.

Definition: $L^\infty([t_{\text{start}}, t_{\text{final}}], \mathbb{R}^{n_u})$

$L^\infty([t_{\text{start}}, t_{\text{final}}], \mathbb{R}^{n_u})$ is the *space of essentially bounded measurable maps* from $[t_{\text{start}}, t_{\text{final}}] \subset \mathbb{R}$ to $\mathbb{R}^{n_u}$. For details, see e.g. [178, Section C.1] or [1, Paragraph 2.10].

Now that we have made sure that a solution of IVP (2.3) exists, we can specify the choice of the transition map in a formal way.

Definition 2.5 We denote the *solution of the IVP (2.3)* by

$$x(\cdot; x_{\text{init}}, u) \colon [t_{\text{start}}, t_{\text{final}}] \to \mathbb{R}^{n_x}$$

and its evaluation at $t \in [t_{\text{start}}, t_{\text{final}}]$ by

$$x(t; x_{\text{init}}, u) \in \mathbb{R}^{n_x},$$

where we highlight the dependence on the initial state $x_{\text{init}} \in \mathbb{R}^{n_x}$ and the control $u(\cdot) \in L^\infty([t_{\text{start}}, t_{\text{final}}], \mathbb{R}^{n_u})$.

To define the transition map for the sampled data case, we need $x(\cdot; x^j, u^j)$ which, in accordance with Definition 2.5, is the solution of the IVP

$$\begin{aligned} \dot{x}(t) &= f\big(x(t), u^j(t)\big), \quad t \in [t^j, t^{j+1}], \\ x\big(t^j\big) &= x^j. \end{aligned} \tag{2.4}$$

In contrast to Definition 2.5, the initial value has changed from x_{init} to x^j in the expression $x(\cdot; x^j, u^j)$.

In other words, $x(\cdot; x^j, u^j)$ is the trajectory of the controlled system originating from the current state x^j and controlled using the control $u^j(\cdot) \in L^\infty([t^j, t^{j+1}], \mathbb{R}^{n_u})$ over the j-th sampling interval according to the model represented by the IVP (2.4). Finally, we set

$$\theta(x^j, u^j) = x\big(t^{j+1}; x^j, u^j\big) \; \forall j \in \mathbb{N}_0. \tag{2.5}$$

The control space U is accordingly restricted to be a subspace of $L^\infty([t^j, t^{j+1}], \mathbb{R}^{n_u})$ and the control sequence space U^N to be a subspace of $L^\infty([t^j, t^j + T], \mathbb{R}^{n_u})$.

With the choice (2.5), the states x^{j+1} of the discrete-time system (2.2) coincide with $x(t^{j+1}; x^j, u^j)$, i.e., the values of the continuous state at the sampling time t^{j+1} for all $j \in \mathbb{N}_0$, cf. [101, Section 2.2, Thm. 2.7].

[101]: Grüne et al. (2017), *Nonlinear Model Predictive Control*

2.1.3. Computation of the control sequence

We have now discussed the general setting of NMPC and elaborated on how we can model the system evolution using an ODE if we are given a control. But, we still need to clarify how we come up with a control sequence, and with it the feedback value. As mentioned at the beginning of this chapter, NMPC methods can vary strongly in this aspect. Hence, we do not attempt to provide a comprehensive overview here. Instead, we focus on the case where the control sequence is computed by solving an OCP at each sampling time. Moreover, we limit our presentation to an OCP formulation for which our numerical methods are designed. Along the way, we occasionally point to other common variations, but refer to [101, 159] for comprehensive overviews and more details.

[159]: Rawlings et al. (2022), *Model predictive control: Theory, computation, and design*

With that said, we focus on the case where we first compute a continuous control $u(\cdot)$ by solving an OCP and then extract the control sequence $\left(u_0^j, \dots, u_N^j\right)$ from it. The OCP has the form

$$\begin{aligned} \min_{x(\cdot),\, u(\cdot)} \quad & \int_{t^j}^{t^j+T_{\mathrm{hor}}} \Psi(x(t), u(t))\mathrm{d}t + \Phi\left(x\left(t^j + T_{\mathrm{hor}}\right)\right) \\ \text{s.t.} \quad & \dot{x}(t) = f\left(x(t), u^j(t)\right), \quad t \in I_{\mathrm{hor}}\left(t^j\right), \\ & 0 \leq h(x(t), u(t)), \quad t \in I_{\mathrm{hor}}\left(t^j\right), \\ & 0 = r^{\mathrm{e}}\left(x\left(t^j\right), x\left(t^j + T_{\mathrm{hor}}\right)\right), \\ & 0 \leq r^{\mathrm{i}}\left(x\left(t^j\right), x\left(t^j + T_{\mathrm{hor}}\right)\right), \\ & x\left(t^j\right) = x^j, \end{aligned} \tag{2.6}$$

with $I_{\mathrm{hor}}\left(t^j\right) \coloneqq [t^j, t^j + T_{\mathrm{hor}}]$ and

$$\begin{aligned} \Psi &\colon \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}, \\ \Phi &\colon \mathbb{R}^{n_x} \to \mathbb{R}, \\ h &\colon \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_h}, \\ r^{\mathrm{e}} &\colon \mathbb{R}^{n_x} \times \mathbb{R}^{n_x} \to \mathbb{R}^{n_{r^{\mathrm{e}}}}, \\ r^{\mathrm{i}} &\colon \mathbb{R}^{n_x} \times \mathbb{R}^{n_x} \to \mathbb{R}^{n_{r^{\mathrm{i}}}}, \end{aligned} \tag{2.7}$$

where $n_h, n_{r^{\mathrm{e}}}, n_{r^{\mathrm{i}}} \in \mathbb{N}$. An appropriate choice for the spaces of $x(\cdot)$ and $u(\cdot)$ are

$$\begin{aligned} x(\cdot) &\in W^{1,\infty}\left(I_{\mathrm{hor}}\left(t^j\right), \mathbb{R}^{n_x}\right), \\ u(\cdot) &\in L^{\infty}\left(I_{\mathrm{hor}}\left(t^j\right), \mathbb{R}^{n_u}\right). \end{aligned} \tag{2.8}$$

Definition: $W^{1,\infty}\left(I_{\mathrm{hor}}\left(t^j\right), \mathbb{R}^{n_x}\right)$

$W^{1,\infty}\left(I_{\mathrm{hor}}\left(t^j\right), \mathbb{R}^{n_x}\right)$ is, as usual, the *Sobolev space* that contains all functions which themselves and whose first derivatives are measurable and essentially bounded functions. For details see e.g. [1, Def. 3.2].

The OCP (2.6) covers a wide range of problem formulations as many other relevant OCP formulations can be transformed into the form (2.6). For a good overview of these transformations we refer to [138, Section 5.1]. In particular, OCPs with non-autonomous ODEs, as we will encounter them in Chapter 6, can be transformed to OCPs of the form (2.6) by introducing an additional state.

[138]: Meyer (2020), "Numerical solution of optimal control problems with explicit and implicit switches"

From OCP (2.6) we can also clearly see one of the main advantages of NMPC, namely the possibility to treat a wide range of constraints. The formulation (2.6) covers constraint types that reach from simple box constraints over pure state or control constraints all the way to mixed state-control constraints, boundary constraints and even periodicity constraints – and all of that also in nonlinear form.

Again, we would like to formulate conditions that assert the existence of solutions, this time of OCP (2.6). However, since the OCP (2.6) is an infinite-dimensional optimization problem, answering this question needs further theory regarding optimization in BANACH spaces. As the main focus of this thesis lays on numerical methods, we do not cover this theory here in detail. We refer to [138, Chapter 5] for a concise overview of the required theory. For a detailed discussion of the theory we further refer to [88, Chapter 2]. We instead only state the common smoothness assumption for the appearing functions that are needed to assert existence of solutions, cf. [138, Assumption 5.2], [88, Assumption 2.2.5].

[88]: Gerdts (2024), *Optimal Control of ODEs and DAEs*

Assumption 2.2 Let

$$(\hat{x}, \hat{u}) \in W^{1,\infty}\left(I_{\text{hor}}\left(t^j\right), \mathbb{R}^{n_x}\right) \times L^{\infty}\left(I_{\text{hor}}\left(t^j\right), \mathbb{R}^{n_u}\right)$$

be given and let M be a sufficiently large convex compact neighbourhood of

$$\left\{(\hat{x}(t), \hat{u}(t)) \in \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \;\middle|\; t \in I_{\text{hor}}\left(t^j\right)\right\}$$

The functions Ψ, Φ, f, h, r^{e}, r^{i} that appear in the OCP (2.6) satisfy the following conditions:

(i) Φ is continuously differentiable.
(ii) r^{e}, r^{i} are continuously differentiable with respect to both arguments.
(iii) The mappings

$$\begin{aligned}(x, u) &\mapsto \Psi(x, u),\\ (x, u) &\mapsto f(x, u),\\ (x, u) &\mapsto h(x, u)\end{aligned}$$

are continuously differentiable in M uniformly for $t \in I_{\text{hor}}\left(t^j\right)$.
(iv) The partial derivatives $\partial_x \Psi, \partial_u \Psi, \partial_x f, \partial_u f, \partial_x h, \partial_u h$ are bounded in $I_{\text{hor}}\left(t^j\right) \times M$.

If we have found a solution $u(\cdot)$ of the OCP (2.6), we obtain the control sequence by setting

$$u_k^j = u(\cdot)|_{t \in [t^j + kT, t^j + (k+1)T]}. \tag{2.9}$$

That means, we divide the OCP solution over the sampling intervals.

As already stated in Definition 2.3, see Equation (2.1), the feedback value u^j is given by the first element of the control sequence, i.e. we have

$$u^j = u_0^j = u(\cdot)|_{t \in [t^j, t^j + T]}. \tag{2.10}$$

The feedback value u^j thus depends directly on the current state. In that sense we make the following definition.

Definition 2.6 The map

$$\mu \colon X \to U$$

that maps

$$x^j \mapsto u^j,$$

with u^j given by Equation (2.10), where $u(\cdot)$ is an (approximate) solution of the OCP (2.6), is called the *(NMPC) feedback law*. If $u(\cdot)$ is indeed a solution of the OCP (2.6), we call μ the *nominal (NMPC) feedback law* and set it apart by denoting it with μ^*.

With Definition 2.6 in place, we can define what nominal NMPC is.

Definition 2.7 If we apply the nominal feedback law μ^* at all sampling times, the resulting closed-loop system

$$x^{j+1} = \theta\left(x^j, \mu^*\left(x^j\right)\right)$$

is called the *nominal closed-loop system*. If further the nominal closed-loop system describes the true behaviour of the system states, i.e. no disturbances occur, we perform *nominal NMPC*.

If we use a control $u(\cdot)$ that is not necessarily a solution of OCP (2.6), we are considering an *inexact NMPC* or *nonoptimal NMPC* method.

Reminder: Transition map

θ is the transition map as defined in Definition 2.4.

Our methods as we present them in this thesis fall into the category of inexact NMPC.

2.1.4. NMPC algorithm

Now that we have filled the main idea of NMPC with details, we conclude our description of NMPC by stating the main algorithm as we also employ it throughout this thesis.

Algorithm 2.1: Main NMPC algorithm.

Input: Time grid $t_0 < t_1 < \ldots < t^j < \ldots$

At each sampling time t^j, $j \in \mathbb{N}_0$ do:

1. $x^j \leftarrow$ Get current state
2. $u(\cdot) \leftarrow$ (Approximately) solve the OCP (2.6) to obtain a control
3. $\left(u_0^j, \ldots, u_N^j\right) \leftarrow$ Extract control sequence using Equation (2.9)
4. $u^j \leftarrow$ Set feedback value using Equation (2.1)
5. Apply u^j to the system

In practice, the current state has usually to be measured.

An important observation is that only the current state x^j changes in the OCP (2.6) between two sampling times. In that sense, we deal with a sequence of parametric OCPs during our NMPC method. We will exploit this parametric property in our numerical method.

2.2. Stability results

One of our contributions presented in Chapter 4 of this thesis is a proof that an inexact NMPC scheme like ours can still lead to stability. In the current section, we introduce terminology and stability results that we will later need in Chapter 4. In Subsection 2.2.1, we present stability results for general nonlinear systems. How these general stability results can be applied to NMPC is described in Subsection 2.2.2.

2.2.1. Stability results for nonlinear systems

Let Y be a metric space. We denote its metric by $d_Y : Y \times Y \to \mathbb{R}_0^+$. We consider a nonlinear system

$$\begin{aligned} y^{j+1} &= \xi(y^j),\ j \in \mathbb{N}_0 \\ y^0 &\in Y \end{aligned} \tag{2.11}$$

When we turn back to NMPC, the system state x^j will take the role of y^j and $\theta(x^j, \mu(x^j))$ the role of $\xi(y^j)$.

with a not necessarily continuous map $\xi : Y \to Y$. If we want to highlight the dependence of y^j on the initial state y^0, we also write $y^j(y^0)$. In the following, we define what it means for such a system to be stable and how stability can be verified using LYAPUNOV functions.

We start by making two definitions in connection with the nonlinear system (2.11).

Definition 2.8 A state $y^* \in Y$ is called an *equilibrium* of the nonlinear system (2.11) if

$$\xi(y^*) = y^*.$$

Definition 2.9 A set $\Gamma \subseteq Y$ is called *forward invariant* for the nonlinear system (2.11) if

$$\xi(y) \in \Gamma \ \forall y \in \Gamma.$$

Next, we follow [101, Definition 2.13] and use comparison functions to define asymptotic stability. We define the following four classes of comparison functions.

[101]: Grüne et al. (2017), *Nonlinear Model Predictive Control*

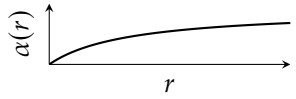
Definition 2.10 We define the following four classes of *comparison functions*.

$$\mathcal{K} := \left\{ \alpha : \mathbb{R}_0^+ \to \mathbb{R}_0^+ \;\middle|\; \alpha \text{ is continuous \& strictly increasing with } \alpha(0) = 0 \right\},$$

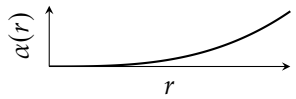
$$\mathcal{K}_\infty := \left\{ \alpha \in \mathcal{K} \;\middle|\; \alpha \text{ is unbounded} \right\},$$

$$\mathcal{L} := \left\{ \delta : \mathbb{R}_0^+ \to \mathbb{R}_0^+ \;\middle|\; \delta \text{ is continuous \& strictly decreasing with } \lim_{s\to\infty} \delta(s) = 0 \right\},$$

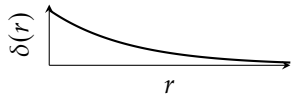
$$\mathcal{KL} := \left\{ \beta : \mathbb{R}_0^+ \times \mathbb{R}_0^+ \to \mathbb{R}_0^+ \;\middle|\; \beta \text{ is continuous, } \beta(\cdot, s) \in \mathcal{K},\ \beta(r, \cdot) \in \mathcal{L}\ \forall s, r \in \mathbb{R}_0^+ \right\}.$$



(a) Typical $\mathcal{K}$ function



(b) Typical $\mathcal{K}_\infty$ function



(c) Typical $\mathcal{L}$ function

Figure 2.2.: Illustrations of typical functions of the comparison function classes.

HAHN was the first to use comparison functions in [106]. About two decades later, comparison functions also gained popularity in nonlinear control theory, especially due to SONTAG, see e.g. [177].

[106]: Hahn (1967), *Stability of Motion*
[177]: Sontag (1989), "Smooth stabilization implies coprime factorization"

As a last step before we define asymptotic stability, we introduce for a given radius $r > 0$ the ball

$$\mathcal{B}_r(y^*) := \left\{ y \in Y \;\middle|\; d_Y(y, y^*) < r \right\}.$$

We follow the presentation in [101, Definition 2.14] and define stability in the following way.

In this thesis, we frequently say that the nonlinear system as such is asymptotically stable. With that we mean that an asymptotically stable equilibrium y^* exists.

Definition 2.11 Let y^* be an equilibrium and $\Gamma \subset Y$ be a forward invariant set of the nonlinear system (2.11). We say that y^* is

(i) *locally asymptotically stable* if there exists a function $\beta \in \mathcal{KL}$ and $r > 0$ such that

$$d_Y\left(y^j\left(y^0\right), y^*\right) \leq \beta\left(d_Y\left(y^0, y^*\right), j\right) \tag{2.12}$$

holds for all $y^0 \in \mathcal{B}_r\left(y^*\right)$ and all $j \in \mathbb{N}_0$,

(ii) *asymptotically stable on a forward invariant set* $\Gamma \ni y^*$ if there exists $\beta \in \mathcal{KL}$ such that Equation (2.12) is satisfied for all $y^0 \in \Gamma$ and all $j \in \mathbb{N}_0$,

(iii) *globally asymptotically stable* if there exists $\beta \in \mathcal{KL}$ such that Equation (2.12) is satisfied for all $y^0 \in Y$ and all $j \in \mathbb{N}_0$.

For other variants of stability, e.g. P-practically asymptotic stability, see e.g. [101, Section 2.3].

Inequality (2.12) means that the distance of the current state $y^j\left(y^0\right)$ to the equilibrium y^* generally decreases with an increasing iteration number j and is bounded by the initial distance, i.e., the distance of the initial state y^0 to y^*.

The distance does not need to decrease in every iteration however.

For NMPC, however, verifying asymptotic stability by checking Definition 2.11 directly is often difficult. Instead, it is easier to establish asymptotic stability by finding a LYAPUNOV function. We define LYAPUNOV functions as in [101, Definition 2.18].

For an interesting survey on how LYAPUNOV's stability theorem has influenced the field of feedback control, we refer to [139].

[101]: Grüne et al. (2017), *Nonlinear Model Predictive Control*

Definition 2.12 Consider the nonlinear system (2.11), a point $y^* \in Y$ and let $S \subseteq Y$ be a subset of the state space. A function $V : S \to \mathbb{R}_0^+$ is called a *LYAPUNOV function* on S, if the following conditions are satisfied:

(i) There exist functions $\alpha_1, \alpha_2 \in \mathcal{K}_\infty$ such that

$$\alpha_1\left(d_Y\left(y, y^*\right)\right) \leq V\left(y\right) \leq \alpha_2\left(d_Y\left(y, y^*\right)\right)$$

holds for all $y \in S$.

(ii) There exists a function $\alpha_3 \in \mathcal{K}$ such that

$$V\left(\xi\left(y\right)\right) \leq V\left(y\right) - \alpha_3\left(d_Y\left(y, y^*\right)\right)$$

holds for all $y \in S$ with $\xi\left(y\right) \in S$.

The following Theorem 2.1 justifies looking for a LYAPUNOV function instead of checking Definition 2.11 for asymptotic stability directly and is taken from [101, Theorem 2.19].

Theorem 2.1 — Asymptotic stability via Lyapunov functions.
Let y^* be an equilibrium of the nonlinear system (2.11). If there exists a LYAPUNOV function V on S, then y^* is

(i) locally asymptotically stable if S contains a ball $\mathcal{B}_r\left(y^*\right)$ with $\xi\left(y\right) \in S$ for all $y \in \mathcal{B}_r\left(y^*\right)$,

(ii) asymptotically stable on S if S is forward invariant and $y^* \in S$,

(iii) globally asymptotically stable if $S = Y$.

To prove Theorem 2.1, we construct the functions α_1, α_2, α_3 from the LYAPUNOV function V. For the full proof, we refer to [101, p. 33-35].

2.2.2. Applicability to NMPC

In the NMPC case, the nonlinear system (2.11) is replaced by the closed-loop system

$$x^{j+1} = \theta\big(x^j, \mu\big(x^j\big)\big), \tag{2.13}$$

cf. Definition 2.7. Moreover, the transition map θ is given by the solution of the IVP (2.4), cf. Equation (2.5). So in our NMPC setting, a continuous process underlies the discrete-time system (2.11). This case is referred to as a sampled data system. The stability results that we have presented in Subsection 2.2.1 only apply to the states y^j at the sampling times t^j and not over the entire sampling intervals $[t^j, t^{j+1})$. In fact, it is also possible to extend the notion of stability to sampled data systems and ensure that the system is stable over the entire sampling intervals. For a detailed discussion on this topic, we refer to [101, Section 2.4].

We have not considered sampled data systems here, as we focus in our stability proof in Chapter 4 on the interplay of the system and the optimizer, which forms a discrete-time system.

3. Efficient Numerical Methods for NMPC

As explained in Chapter 2, the main computational task in NMPC is solving the OCPs in the second step of the NMPC Algorithm 2.1. For clarity, we restate the OCP as it appears at each NMPC sampling time.

$$
\begin{aligned}
\min_{x(\cdot),\,u(\cdot)} \quad & \int_{t^j}^{t^j+T_{\mathrm{hor}}} \Psi(x(t),u(t))\mathrm{d}t + \Phi\big(x\big(t^j+T_{\mathrm{hor}}\big)\big) \\
\text{s.t.} \quad & \dot{x}(t) = f(x(t),u(t)), \quad t \in I_{\mathrm{hor}}\big(t^j\big), \\
& 0 \le h(x(t),u(t)), \quad t \in I_{\mathrm{hor}}\big(t^j\big), \\
& 0 = r^{\mathrm{e}}\big(x\big(t^j\big), x\big(t^j+T_{\mathrm{hor}}\big)\big), \\
& 0 \le r^{\mathrm{i}}\big(x\big(t^j\big), x\big(t^j+T_{\mathrm{hor}}\big)\big), \\
& x\big(t^j\big) = x^j.
\end{aligned}
\tag{3.1}
$$

OCP (3.1) is an infinite-dimensional optimization problem. We provide a brief overview of different classes of solution methods available to address OCP (3.1) in Section 3.1. Our numerical method employs the Direct Multiple Shooting (DMS) method to discretize OCP (3.1) and the Sequential Quadratic Programming (SQP) method to solve the resulting Nonlinear Program (NLP). We explain DMS in Section 3.2 and the SQP method in Section 3.3.

Providing feedback to the system as quickly as possible is crucial for many NMPC applications. Consequently, highly efficient numerical methods are required to compute new feedback rapidly. One way to achieve this is by exploiting the fact that subsequent OCPs are only parametrized by the current state x^j. The Real-Time Iterations (RTI) scheme, based on DMS and the SQP method, is a successful approach to leverage this structure. An enhancement of RTI is the Multi-Level Iterations (MLI) scheme. We present the RTI scheme in Section 3.4 and the MLI scheme in Section 3.5.

Reminder: OCP (3.1)

- OCP (3.1) appeared already as OCP (2.6) on page 16.
- $x(\cdot) \in W^{1,\infty}\big(I_{\mathrm{hor}}\big(t^j\big), \mathbb{R}^{n_x}\big)$ is the state trajectory.
- $u(\cdot) \in L^{\infty}\big(I_{\mathrm{hor}}\big(t^j\big), \mathbb{R}^{n_u}\big)$ is the control trajectory.
- t^j is the j-th sampling time, see Definition 2.1.
- T_{hor} is the length of the prediction horizon, see Definition 2.2.
- $x^j \in \mathbb{R}^{n_x}$ is the current system state, see Definition 2.4.
- $I_{\mathrm{hor}}(t^j) := [t^j, t^j + T_{\mathrm{hor}}]$.
- Ψ, Φ, f, h, r^{e}, and r^{i} are functions satisfying Assumptions 2.1 and 2.2 and, from Subsection 3.3.2 on, Assumption 3.2. See Equation (2.7) for their dimensions.

3.1. Solution approaches for Optimal Control Problems

In the following, our primary goal is to position our solution approach, which uses DMS for discretization and the SQP method for solving the NLP, within the broad spectrum of existing approaches. A comprehensive survey of OCP solution methods is beyond the scope of this thesis. Our categorization of solution approaches is common, though not the only possible one, and is certainly incomplete with respect to all existing methods. For detailed surveys, we refer to [26, 157, 158]. The textbook [181] contains numerous references to specific methods.

[26]: Biral et al. (2016), "Notes on Numerical Methods for Solving Optimal Control Problems"
[157]: Rao (2010), "A Survey of Numerical Methods for Optimal Control"
[158]: Rao (2014), "Trajectory Optimization: A Survey"
[181]: Teo et al. (2021), *Applied and Computational Optimal Control*

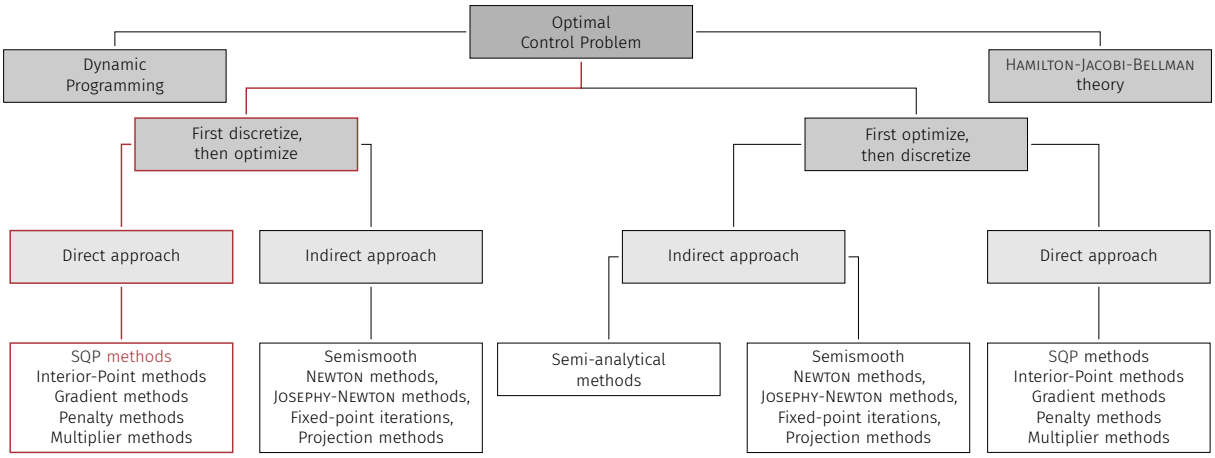


Figure 3.1.: Classification of solution approaches for OCPs. Adapted from [88, Figure 1.9, p. 44]. Highlighted in red is where our methods are located.

The classification of solution approaches we follow is illustrated in Figure 3.1. The entire tree of methods, as shown in Figure 3.1, is divideded into the "first discretize, then optimize" and the "first optimize, then discretize" approaches. The former is adopted in this thesis, while the latter is also known as the *function space approach*.

As the names suggest, in the "first discretize, then optimize" approach, we first discretize the infinite-dimensional OCP (3.1) to obtain a finite-dimensional NLP. Subsequently, the resulting NLP is optimized using either *direct* or *indirect* methods. In direct methods, optimization techniques such as the SQP method or interior-point methods are applied directly to the NLP. In indirect methods, the focus is on solving the optimality conditions of the NLP.

The distinction between direct and indirect methods is not always clear-cut, as many direct methods are developed by considering the optimality conditions. In some cases, direct and indirect methods are even equivalent. For instance, applying the basic SQP method for equality-constrained NLPs is equivalent to applying Newton's method to its optimality conditions, see [143, Section 18.1].

In the "first optimize, then discretize" approach, the order of these two steps is reversed. This means that methods operating in suitable function spaces are applied first, followed by discretization to numerically implement these methods. In the optimization step, direct or indirect methods can again be used, but unlike the previous case, these methods operate in infinite-dimensional function spaces instead of finite-dimensional ones. A selection of optimization methods for Banach spaces can be found in [189]. A concise presentation of necessary optimality conditions for infinite-dimensional problems is provided in [88, Section 2.3].

[189]: Ulbrich (2009), "Optimization Methods in Banach Spaces"

[88]: Gerdts (2024), *Optimal Control of ODEs and DAEs*

In the "first optimize, then discretize" approach, researchers often use the term indirect methods to specifically refer to methods based on Pontryagin's Maximum Principle (PMP). For a collection of the original works by Pontryagin, see [151]. The PMP provides first-order necessary optimality conditions. From these conditions, the user must manually derive a nonlinear Multipoint Boundary Value Problem (MPBVP). This step requires a thorough understanding of both general control theory and the specific problem at hand. Moreover, this step is typically not automatable. The MPBVP can then be solved using numerical methods such as multiple shooting, cf. [27, 33]. In this sense, these methods can be considered semi-analytical.

[151]: Pontryagin et al. (1986), *The mathematical theory of optimal processes*
[27]: Bock (1978), "Numerische Berechnung zustandsbeschränkter optimaler Steuerungen mit der Mehrzielmethode"
[33]: Bulirsch (1971), "Die Mehrzielmethode zur numerischen Lösung von nichtlinearen Randwertproblemen und Aufgaben der optimalen Steuerung"

Two additional notable classes of approaches are *Dynamic Programming (DP)* and *Hamilton-Jacobi-Bellman (HJB) theory*.

The cornerstone of DP is Bellman's principle of optimality:

> *"An optimal policy has the property that, whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision."*
>
> — Bellman [18–20]

[18]: Bellman (1954), "The theory of dynamic programming"
[19]: Bellman (1957), *Dynamic Programming*
[20]: Bellman (1966), "Dynamic Programming"

From this principle, the Bellman equation is derived for discrete-time systems to determine the value function. DP originates from the study of multi-stage decision processes, which encompass a wide range of optimization problems. In particular, DP can be applied to OCPs, see, e.g., [22]. HJB theory extends DP but primarily focuses on continuous-time optimal control problems. The value function remains central to the approach but is determined by the HJB equation.

[22]: Bertsekas (2020), *Dynamic programming and optimal control*

To conclude this overview, we locate our method within this categorization. In the first step of our method, we discretize OCP (3.1) using DMS. In the second step, we apply a tailored variant of the SQP method to solve the resulting NLP. Thus, our approach falls into the category of "first discretize, then optimize" methods. Furthermore, it is a direct approach, as we work with the NLPs rather than their optimality conditions. The main advantage of this approach is its numerical robustness, which is generally considered superior to function space approaches. Compared to the PMP, this approach frees the user from particularly challenging and error-prone analytical tasks.

3.2. Direct Multiple Shooting discretization

As discussed at the end of Section 3.1, we adopt a "first discretize, then optimize" approach. The task of this section is to discretize 3.1, i.e., to transform 3.1 into a finite-dimensional NLP. Our chosen method for this task is the *Direct Multiple Shooting (DMS)* method.

[4]: Albersmeyer (2010), "Adjoint based algorithms and numerical methods for sensitivity generation and optimization of large scale dynamic systems"
[27]: Bock (1978), "Numerische Berechnung zustandsbeschränkter optimaler Steuerungen mit der Mehrzielmethode"
[29]: Bock et al. (1984), "A Multiple Shooting Algorithm for Direct Solution of Optimal Control Problems"
[33]: Bulirsch (1971), "Die Mehrzielmethode zur numerischen Lösung von nichtlinearen Randwertproblemen und Aufgaben der optimalen Steuerung"
[128]: Leineweber (1999), *Efficient reduced SQP methods for the optimization of chemical processes described by large sparse DAE models*
[150]: Plitt (1981), "Ein superlinear konvergentes Mehrzielverfahren zur direkten Berechnung beschränkter optimaler Steuerungen"
[153]: Potschka (2011), "A direct method for the numerical solution of optimization problems with time-periodic PDE constraints"

Originally, multiple shooting was developed for MPBVPs arising from the application of PMP by Bulirsch and Bock [27, 33]. Later, Plitt introduced it as a discretization technique for OCPs in his diploma thesis [150], supervised by Bock. Together, they published the seminal paper on DMS for OCPs [29]. Initially, it was developed for ODE-constrained OCPs. However, Leineweber and coworkers extended it to DAEs in [128–130]. Potschka further extended it to PDEs [153]. Efficient structure exploitation strategies and sensitivity generation techniques, enabling the use of DMS for large-scale systems, were presented in [4, 165, 166]. Moreover, DMS has been implemented in several software packages. The most prominent of these are `MUSCOD-II` [122], `ACADO` [109], and its successor `acados` [194].

We describe DMS in the following subsections in four steps. First, in Subsection 3.2.1, we explain how to discretize the control $u(\cdot)$. Next, we introduce the state discretization in Subsection 3.2.2. Subsequently, we explain how to discretize the constraints and the objective function in Subsection 3.2.3 and Subsection 3.2.4, respectively. Finally, we present the resulting NLP in Subsection 3.2.5.

3.2.1. Control discretization

The idea behind control discretization in DMS is to define the control piecewise over a suitable partition of the control horizon of the OCP. This partition is referred to as the shooting grid.

In NMPC, we have $\mathcal{T} = [t^j, t^j + T_{\mathrm{hor}}]$.

Definition 3.1 The partition $t_{\mathrm{s}} = \tau_0 < \tau_1 < \ldots < \tau_m < \tau_M = t_{\mathrm{f}}$ of the control horizon $\mathcal{T} = [t_{\mathrm{s}}, t_{\mathrm{f}}] \subset \mathbb{R}$ is called the *shooting grid*. The points τ_m, $m = 0, \ldots, M$ are called *shooting nodes*, and the intervals $[\tau_m, \tau_{m+1})$, $m = 0, \ldots, M-1$ are called *shooting intervals*.

The shooting grid is not necessarily equidistant, although this is common.

For each shooting interval, we approximate the control u. Specifically, for each shooting interval $[\tau_m, \tau_{m+1})$, $m = 0, \ldots, M-1$ and each component u_i, $i = 0, \ldots, n_u - 1$ of the control, we use a basis function

$$\xi^u_{m,i} \colon [\tau_m, \tau_{m+1}) \times \mathbb{R}^{n_{q_{m,i}}} \to \mathbb{R}, \quad (\tau, q_{m,i}) \mapsto \xi^u_{m,i}(\tau; q_{m,i}),$$

While the basis functions can differ for each component and interval, it is common to use the same basis function for all shooting intervals.

where $q_{m,i} \in \mathbb{R}^{n_{q_{m,i}}}$ are coefficients characterizing the basis function $\xi^u_{m,i}$. To ensure that the resulting controls belong to the control space $L^\infty(\mathcal{T}, \mathbb{R}^{n_u})$, as specified on page 16, we require that

$$\begin{pmatrix} \xi^u_{m,0} \\ \vdots \\ \xi^u_{m,n_u-1} \end{pmatrix} \in L^\infty([\tau_m, \tau_{m+1}), \mathbb{R}^{n_u}).$$

We collect the coefficients corresponding to the same shooting interval in a vector

It holds $n_{q_m} = \sum_{i=0}^{n_u-1} n_{q_{m,i}}$ and $n_q = \sum_{m=0}^{M-1} n_{q_m}$.

$$q_m := (q_{m,0}^T, \ldots, q_{m,n_u-1}^T)^T \in \mathbb{R}^{n_{q_m}}. \tag{3.2}$$

Similarly,

$$q := (q_0^T, \ldots, q_{M-1}^T)^T \in \mathbb{R}^{n_q} \tag{3.3}$$

denotes the collection of all coefficients. The control profile characterized by the coefficients q for a given choice of basis functions is denoted by $u(q)$. In other words, $u(q)$ is defined by

$$u(\tau; q) := \begin{pmatrix} \xi^u_{m,0}(\tau; q_{m,0}) \\ \vdots \\ \xi^u_{m,n_u-1}(\tau; q_{m,n_u-1}) \end{pmatrix}, \tag{3.4}$$

for $\tau \in [\tau_m, \tau_{m+1})$, $m = 0, \ldots, M-1$.

▪ **Example 3.1** The two most common choices for basis functions are:

(i) Constant functions, i.e.,

$$\xi^u_{m,i}(\tau, q_{m,i}) = q_{m,i} \in \mathbb{R}, \ \forall \tau \in [\tau_m, \tau_{m+1}),$$

(ii) Linear functions, i.e., with $q_{m,i} = (q^{\mathrm{l}}_{m,i}, q^{\mathrm{r}}_{m,i})^T \in \mathbb{R}^2$, we set

$$\xi^u_{m,i}(\tau, q_{m,i}) = \frac{\tau_{m+1} - \tau}{\tau_{m+1} - \tau_m} q^{\mathrm{l}}_{m,i} + \frac{\tau - \tau_m}{\tau_{m+1} - \tau_m} q^{\mathrm{r}}_{m,i}.$$

▪

To ensure that the control is continuous, we can use continuous basis functions combined with control continuity conditions.

Definition 3.2 Let basis functions $\xi^u_{m,i}$ be given. Continuity of the i-th control component at the m-th shooting node, with $0 < m < M$, is ensured by adding the *(control) continuity condition*

$$0 = \xi^u_{m,i}(\tau_m, q_{m,i}) - \xi^u_{m-1,i}(\tau_m, q_{m-1,i}) \tag{3.5}$$

to the DMS NLP.

3.2.2. State discretization

One of the main features that distinguishes DMS from other discretization techniques, such as Direct Single Shooting or Direct Collocation, is that it introduces a state variable for each shooting node.

In single shooting, we introduce a state variable only for the initial state, while in collocation methods, several collocation points are used on each interval.

Definition 3.3 For each shooting node τ_m, we introduce a variable $s_m \in \mathbb{R}^{n_x}$ that represents the state at the shooting node and is called a *node value*. We collect the node values in a single vector

$$s := \begin{pmatrix} s_0 \\ \vdots \\ s_M \end{pmatrix} \in \mathbb{R}^{n_s}.$$

$n_s := (M+1)n_x$

These state variables serve as initial values for M IVPs. The IVP with the initial state s_m determines the state trajectory x_m on the m-th shooting interval and is given by

$$\begin{aligned} \dot{x}_m(\tau) &= f\big(x_m(\tau), u(\tau; q)\big), \quad \tau \in [\tau_m, \tau_{m+1}), \\ x_m(\tau_m) &= s_m. \end{aligned} \tag{3.6}$$

Reminder: M

M denotes the number of shooting intervals, see Definition 3.1.

The independence of the IVPs for the shooting intervals allows efficient parallelization of the computationally expensive ODE integration step.

We denote the solution of IVP (3.6) by $x_m(\tau; s_m, q_m)$ or by $x(\tau; s_m, q_m)$ whenever the respective shooting interval is clear from the context.

The existence of a unique solution for IVP (3.6) is ensured by Assumption 2.1, as discussed on page 14.

In most real-world applications, the state trajectory must be continuous over the entire control horizon. However, so far, the state variables and IVPs for the individual shooting intervals are independent of each other, which can lead to discontinuities in the state trajectory at the shooting nodes. To address this, we enforce continuity at the shooting nodes by introducing the so-called matching conditions.

Definition 3.4 To ensure continuity of the state trajectory x over the entire control horizon $\mathcal{T}$, we impose the *matching conditions*:

$$x(\tau_{m+1}; s_m, q_m) - s_{m+1} = 0, \ m = 0, \dots, M-1. \tag{3.7}$$

The matching conditions ensure that the endpoint of the state trajectory on the m-th shooting interval coincides with the initial state s_{m+1} on the next shooting interval.

While introducing additional optimization variables s_m generally increases the overall problem dimensions of the final NLP, the problem structure induced by the matching conditions allows the application of a tailored SQP method.

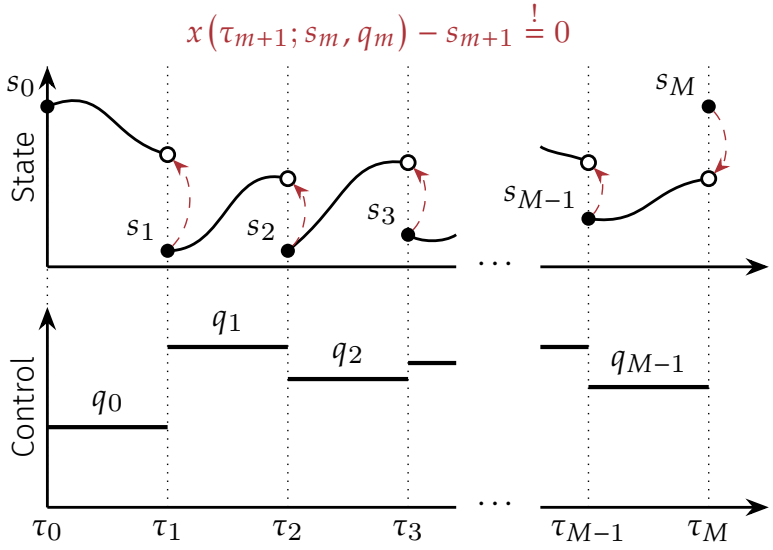


Figure 3.2.: Illustration of the control and state discretization in DMS. The control horizon $\mathcal{T} = [t_s, t_f]$ is divided over the shooting grid $t_s = \tau_0 < \tau_1 < \ldots < \tau_m < \tau_M = t_f$. For each shooting node, a node value s_M is introduced. Over each shooting interval, coefficients q_m determine the control. The colored arrows indicate how the matching conditions (3.7) enforce continuity of the state trajectory.

In this tailored SQP method, the state variables s_m, $m = 1, \ldots, M$ can be eliminated from the subproblems. This elimination technique is called *condensing*. Thus, the additional computational workload required to handle more optimization variables is reduced, and the advantages of DMS outweigh this overhead in most applications. We describe the SQP method and the condensing technique in Section 3.3.

A major advantage of DMS, especially compared to single shooting, is that it is significantly easier to find initial guesses for the optimization variables for which the IVPs are solvable. For example, consider an OCP for chemical processes. In the single shooting setting, we only guess the initial controls q and the concentrations at τ_0. A forward integration with these guesses over the entire control horizon $\mathcal{T}$ can easily lead to negative concentrations. However, the process model might not be defined for negative concentrations, making it impossible to fully evaluate the discretized OCP. In DMS, we have the advantage of providing an initial guess for the controls at all shooting nodes, which can be chosen to be positive. Moreover, since the integration horizons are shorter, it is much more likely that all IVPs can be solved. The matching conditions ensure that the final solution is continuous, so there is no penalty for the discontinuous initial trajectory.

The concept of control and state discretization in DMS is illustrated in Figure 3.2.

3.2.3. Constraint discretization

The formulation of the mixed state-control constraints h in OCP (3.1) implies that they must be satisfied at all points $\tau \in \mathcal{T}$. In theory, we could ensure that the constraints are satisfied at almost all $\tau \in \mathcal{T}$ by imposing the constraints

$$0 = \int_{\tau_m}^{\tau_{m+1}} \min\{0, h(x_m(\tau; s_m, q_m), u(\tau; q_m))\} \mathrm{d}\tau, \quad m = 0, \ldots, M-1. \tag{3.8}$$

The minimum is applied componentwise.

Unfortunately, evaluating Equation (3.8) requires additional integrations, which are computationally more expensive than simple evaluations of h. An alternative approach to track constraint violations within the shooting intervals is based on semi-infinite programming. This method was introduced by Potschka in his diploma thesis [152] and later elaborated in [154].

[152]: Potschka (2006), "Handling path constraints in a direct multiple shooting method for optimal control problems"
[154]: Potschka et al. (2009), "A minima tracking variant of semi-infinite programming for the treatment of path constraints within direct solution of optimal control problems"

In DMS, however, the mixed state-control constraints are enforced only at the shooting nodes. Formally, we discretize the mixed state-control constraints by replacing them with

$$0 \leq h\big(s_m, u\big(\tau; q_m\big)\big), \quad m = 0, \dots, M-1. \tag{3.9}$$

The mixed state-control constraints (3.9) are not enforced at the final shooting node τ_M, as only the terminal constraints r^{e}, r^{i}, which do not depend on the control, are meaningful at the final node.

For simplicity, we slightly overload our notation and write (3.9) as

$$0 \leq h\big(s_m, q_m\big), \quad m = 0, \dots, M-1. \tag{3.10}$$

With this approach, violations of the mixed state-control constraints may occur at points between the shooting nodes. However, in practice, choosing a sufficiently fine shooting grid is usually sufficient to avoid significant violations. Another option is to introduce additional points between the shooting nodes where the constraints are enforced. The rationale behind both options is that for smooth mixed state-control constraints and dynamics, the violations can be bounded by the distance between the points where the constraints are enforced. This rationale is illustrated with the following example.

Reminder: Assumption 2.1

We assume that f and h are continuously differentiable and therefore Lipschitz continuous.

▪ **Example 3.2** Assume we use piecewise constant controls and that f and h are Lipschitz continuous with respect to both arguments, and that $f(0,0) = 0$. If $h\big(s_m, q_m\big) = 0$, then positive constants L_x, L_u exist such that

$$\big\|h\big(x_m\big(\tau; s_m, q_m\big), q_m\big)\big\| \leq (\hat{\tau} - \tau_m)\big(L_x\|s_m\| + L_u\big\|q_m\big\|\big)$$

for all $\hat{\tau} \in [\tau_m, \tau_{m+1})$. The proof is given in Appendix A. ▪

For the boundary constraints r^{e} and r^{i} the discretization is straightforward. We only need to replace $x\big(t^j\big)$ by s_0 and $x\big(t^j + T_{\mathrm{hor}}\big)$ by s_M, i.e. we replace

$$\begin{aligned} r^{\mathrm{e}}\big(x\big(t^j\big), x\big(t^j + T_{\mathrm{hor}}\big)\big) &= 0, \\ r^{\mathrm{i}}\big(x\big(t^j\big), x\big(t^j + T_{\mathrm{hor}}\big)\big) &= 0 \end{aligned}$$

with

$$r^{\mathrm{e}}(s_0, s_M) = 0, \tag{3.11}$$
$$r^{\mathrm{i}}(s_0, s_M) = 0. \tag{3.12}$$

3.2.4. Objective function discretization

The objective function

$$\int_{t^j}^{t^j+T_{\mathrm{hor}}} \Psi(x(t),u(t))\mathrm{d}t + \Phi\big(x\big(t^j+T_{\mathrm{hor}}\big)\big)$$

is replaced in the NLP by a sum

Similarly to the boundary constraints, we replace $x(t^j+T_{\mathrm{hor}})$ with s_M for the MAYER objective term Φ.

$$\sum_{m=0}^{M-1} \Psi_m\big(s_m,q_m\big) + \Phi(s_M) \tag{3.13}$$

Here, $\Psi_m\big(s_m,q_m\big)$ is either an exact representation of the LAGRANGE objective function, i.e.,

The integration can be performed simultaneously with the state integration.

$$\Psi_m\big(s_m,q_m\big) = \int_{\tau_m}^{\tau_{m+1}} \Psi\big(x_m\big(\tau;s_m,q_m\big),u\big(\tau;q_m\big)\big)\mathrm{d}\tau,$$

or an approximation thereof. A simple approximation that does not require integration is:

$$\Psi_m\big(s_m,q_m\big) = (\tau_{m+1}-\tau_m)\Psi\big(s_m,u\big(\tau_m;q_m\big)\big).$$

3.2.5. Resulting Nonlinear Program

We conclude our description of the DMS method by stating the resulting NLP, whose components we have set up in the preceding subsections. The optimization variables in the DMS discretization of 3.1 are the control coefficients $q \in \mathbb{R}^{n_q}$ (see Equation (3.2), Equation (3.3), Equation (3.4)) and the node values $s \in \mathbb{R}^{n_s}$ (see Definition 3.3). The objective function discretization is given by Equation (3.13).

The first set of constraints in the DMS NLP are the matching conditions (3.7). Since we will use only piecewise constant controls throughout this thesis, we do not include the control continuity conditions (3.5). For the discretization of the mixed state-control constraints, we adopt the most common approach and enforce them only at the shooting nodes, i.e., we add the constraints (3.9), abbreviated in the form of (3.10), to the DMS NLP. Finally, the DMS NLP also incorporates the boundary constraints (3.11) and (3.12). The DMS NLP is then given by

$$\min_{\substack{s\in\mathbb{R}^{n_s}\\ q\in\mathbb{R}^{n_q}}} \quad \sum_{m=0}^{M-1} \Psi_m\big(s_m,q_m\big) + \Phi(s_M) \tag{3.14a}$$

$$\text{s.t.} \quad 0 = x\big(\tau_{m+1};s_m,q_m\big) - s_{m+1}, \quad m=0,\dots,M-1, \tag{3.14b}$$

$$0 \le h\big(s_m,q_m\big), \quad m=0,\dots,M-1, \tag{3.14c}$$

$$0 = r^{\mathrm{e}}(s_0,s_M), \tag{3.14d}$$

$$0 \le r^{\mathrm{i}}(s_0,s_M), \tag{3.14e}$$

$$0 = x^j - s_0. \tag{3.14f}$$

The introduction of node values for each shooting node, together with the matching conditions (3.14b), induces a structure in the NLP (3.14) that can be efficiently exploited when applying the SQP method to solve it. We will explain this structure exploitation in detail in Subsection 3.3.2. Furthermore, the structure of the NLP (3.14) allows its components to be evaluated in parallel on multi-core CPUs.

3.3. Sequential Quadratic Programming method

The Sequential Quadratic Programming (SQP) method, about which POWELL famously remarked (see, for example, [155, p. 155]),

> *"It can be programmed in an afternoon if one has a quadratic programming subroutine available [...]."*
>
> — POWELL [155]

is one of the most popular and successful methods for solving nonlinear programs. In particular, our efficient numerical methods for NMPC, namely the real-time iterations (see Section 3.4) and the multi-level iterations (see Section 3.5), are based on the SQP method. Below, we provide a concise presentation of the SQP method in general and then discuss its application to NMPC with DMS. More detailed presentations of the SQP method, including its history, convergence properties, globalization strategies, and practical implementations, can be found in [143, Chapter 18], [190, Chapter 19], or [87, Section 5.5].

[155]: Powell (1977), "A fast algorithm for nonlinearly constrained optimization calculations"

In fact, the SQP method comes in a wide variety of forms. Therefore, the SQP method should be understood more as a general framework that now encompasses a class of designs. In [143, p. 529], it is suggested to refer to these as "active-set methods for nonlinear programming".

[143]: Nocedal et al. (2006), *Numerical Optimization*
[190]: Ulbrich et al. (2012), *Nichtlineare Optimierung*
[87]: Geiger et al. (2002), *Theorie und Numerik restringierter Optimierungsaufgaben*

3.3.1. General SQP framework

An insightful and common way to present the SQP method is to interpret it as a NEWTON's method applied to the optimality system of an equality-constrained NLP. Thus, we first consider the NLP

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & J(x) \\ \text{s.t.} \quad & 0 = c(x) \end{aligned} \tag{3.15}$$

where $J\colon \mathbb{R}^n \to \mathbb{R}$ and $c\colon \mathbb{R}^n \to \mathbb{R}^{n_c}$ are twice continuously differentiable functions. We define the Lagrangian of (3.15) as follows.

Definition 3.5 $\mathscr{L}\colon \mathbb{R}^n \times \mathbb{R}^{n_c} \to \mathbb{R}$ is the *Lagrangian* of the NLP (3.15), defined as

$$\mathscr{L}(x, \lambda) := J(x) - \lambda^T c(x),$$

where $\lambda \in \mathbb{R}^{n_c}$ are the *LAGRANGE multipliers* or *dual variables* corresponding to the equality constraints $c(x) = 0$.

The KARUSH-KUHN-TUCKER (KKT) optimality conditions of (3.15)

$$F(x, \lambda) := \begin{pmatrix} \nabla_x \mathscr{L}(x, \lambda) \\ c(x) \end{pmatrix} = 0 \tag{3.16}$$

form a nonlinear system of equations. Applying NEWTON's method to solve Equation (3.16), we solve at each step with current iterates (x^k, λ^k) the linear system

$$\begin{pmatrix} \nabla^2_{xx}\mathcal{L}(x^k,\lambda^k) & \left(\frac{\partial}{\partial x}c(x^k)\right)^T \\ \frac{\partial}{\partial x}c(x^k) & 0 \end{pmatrix}\begin{pmatrix} \Delta x^k \\ -\Delta\lambda^k \end{pmatrix} = -\begin{pmatrix} \nabla_x\mathcal{L}(x^k,\lambda^k) \\ c(x^k) \end{pmatrix}. \tag{3.17}$$

For a proof of this statement and remarks on when Assumption 3.1 is satisfied, see, for example, [143, Lemma 16.1, Section 18.1].

It is well known that the KKT matrix on the right-hand side is nonsingular if, at $(x, \lambda) = (x^k, \lambda^k)$, the following assumption is satisfied.

LICQ stands for linear independence constraint qualification. PD stands for positive definiteness.

Assumption 3.1

(i) **LICQ:** The Jacobian of the constraints $\frac{\partial}{\partial x}c(x)$ has full row rank.
(ii) **PD:** The Hessian of the Lagrangian with respect to the primal variables is positive definite on the tangent space of the equality constraints, i.e.

$$p^T\nabla^2_{xx}\mathcal{L}(x,\lambda)p > 0 \text{ for all } p \neq 0 \text{ with} \frac{\partial}{\partial x}c(x)p = 0.$$

We then iterate

$$\begin{aligned} x^{k+1} &= x^k + \Delta x^k, \\ \lambda^{k+1} &= \lambda^k + \Delta\lambda^k. \end{aligned}$$

Utilizing that

$$\nabla_x\mathcal{L}\left(x^k,\lambda^k\right) = \nabla_x J\left(x^k\right) - \left(\frac{\partial}{\partial x}c\left(x^k\right)\right)^T\lambda^k,$$

we see that we can equivalently solve the linear system

$$\begin{pmatrix} \nabla^2_{xx}\mathcal{L}(x^k,\lambda^k) & \left(\frac{\partial}{\partial x}c(x^k)\right)^T \\ \frac{\partial}{\partial x}c(x^k) & 0 \end{pmatrix}\begin{pmatrix} \Delta x^k \\ -\lambda_{\mathrm{QP}} \end{pmatrix} = -\begin{pmatrix} \nabla_x J(x^k) \\ c(x^k) \end{pmatrix} \tag{3.18}$$

and iterate

$$\begin{aligned} x^{k+1} &= x^k + \Delta x^k, \\ \lambda^{k+1} &= \lambda_{\mathrm{QP}}. \end{aligned} \tag{3.19}$$

The key insight is that the linear system (3.18) constitutes the KKT optimality conditions of the Quadratic Program (QP)

$$\begin{aligned} \min_{\Delta x \in \mathbb{R}^n} \quad & \frac{1}{2}\Delta x^T\nabla^2_{xx}\mathcal{L}\left(x^k,\lambda^k\right)\Delta x + \nabla_x J\left(x^k\right)^T\Delta x \\ \text{s.t.} \quad & 0 = c\left(x^k\right) + \frac{\partial}{\partial x}c\left(x^k\right)\Delta x. \end{aligned} \tag{3.20}$$

The main loop of the SQP method thus consists of solving the QP (3.20) and updating the iterates according to Equation (3.19).

A significant advantage of the SQP method is that it can be extended to NLPs with inequality constraints as follows. Consider the NLP

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & J(x) \\ \text{s.t.} \quad & 0 = c(x), \\ & 0 \leq d(x) \end{aligned} \tag{3.21}$$

where $d\colon \mathbb{R}^n \to \mathbb{R}^{n_d}$ is a twice continuously differentiable function. Let $\mu \in \mathbb{R}^{n_d}$ be the LAGRANGE multipliers for the inequality constraints $d(x) \geq 0$. In each iteration of the SQP method, we now solve the QP

$$\begin{aligned} \min_{\Delta x \in \mathbb{R}^n} \quad & \frac{1}{2}\Delta x^T \nabla^2_{xx}\mathcal{L}\left(x^k, \lambda^k, \mu^k\right)\Delta x + \nabla_x J\left(x^k\right)^T \Delta x \\ \text{s.t.} \quad & 0 = c\left(x^k\right) + \frac{\partial}{\partial x} c\left(x^k\right)\Delta x, \\ & 0 \leq d\left(x^k\right) + \frac{\partial}{\partial x} d\left(x^k\right)\Delta x. \end{aligned} \tag{3.22}$$

We denote the optimal solution of (3.22) by Δx^k and the corresponding LAGRANGE multipliers by λ_{QP} and μ_{QP}, and iterate

$$\begin{aligned} x^{k+1} &= x^k + \Delta x^k, \\ \lambda^{k+1} &= \lambda_{\mathrm{QP}}, \\ \mu^{k+1} &= \mu_{\mathrm{QP}}. \end{aligned} \tag{3.23}$$

We summarize the main SQP method in Algorithm 3.1.

Algorithm 3.1: Local full-step SQP method with exact derivatives.

Input: Initial guesses $x^0 \in \mathbb{R}^n, \lambda^0 \in \mathbb{R}^{n_c}, \mu^0 \in \mathbb{R}^{n_d}$

while *convergence criterion not satisfied* **do**

1 $c\left(x^k\right), d\left(x^k\right) \leftarrow$ Evaluate constraint residuals,
$\nabla_x J\left(x^k\right) \leftarrow$ Evaluate objective function gradient,
$\frac{\partial}{\partial x} c\left(x^k\right), \frac{\partial}{\partial x} d\left(x^k\right) \leftarrow$ Evaluate constraint Jacobians,
$\nabla^2_{xx}\mathcal{L}\left(x^k, \lambda^k, \mu^k\right) \leftarrow$ Evaluate Hessian of Lagrangian

2 $\Delta x^k, \lambda_{\mathrm{QP}}, \mu_{\mathrm{QP}} \leftarrow$ Solve QP (3.22)

3 $x^{k+1}, \lambda^{k+1}, \mu^{k+1} \leftarrow$ Update iterates using Equation (3.23)

As the title of Algorithm 3.1 indicates, there are particularities to Algorithm 3.1. First, we are not employing any globalization strategies, as reflected in the term "local". Second, we always take a full step. Third, all derivatives are exact. All of these features can be varied in practical SQP methods, for which we refer to [143, Chapter 18]. Moreover, solving the QP (3.22) is not a trivial task, and we refer to [143, Chapter 16] for solution methods.

[143]: Nocedal et al. (2006), *Numerical Optimization*

3.3.2. Tailored SQP method for the DMS NLP

The numerical methods for NMPC that we present and develop in this thesis are based on an SQP method that is tailored to the DMS NLP (3.14). As stated in the previous subsection Subsection 3.3.1, all functions in the NLP which we want to solve have to be twice continuously differentiable for the SQP method to be applicable. Therefore, we make the following assumption regarding the DMS NLP (3.14).

Assumption 3.2 All functions appearing in the DMS NLP (3.14) are twice continuously differentiable with respect to the primal variables s and q. In particular, we tighten Assumption 2.1 to require that the vector field $f\colon \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_x}$ is twice continuously differentiable with respect to x and u.

[200]: Wirsching (2018), "Multi-level iteration schemes with adaptive level choice for nonlinear model predictive control"

In this subsection, we take a closer look at the QPs that arise when applying the SQP method to solve the NLP (3.14), which results from the DMS discretization of the OCP (3.1). We closely follow [200, Section 2.5.1]. Let $s \in \mathbb{R}^{n_s}$ and $q \in \mathbb{R}^{n_q}$ be the current primal iterates, and $\lambda \in \mathbb{R}^{n_c}$ and $\mu \in \mathbb{R}^{n_d}$ the current dual iterates of the SQP method. The QP that must be solved in each iteration of the SQP method is given by

For simplicity, we omit the SQP iteration index.

λ and μ denote the LAGRANGE multipliers corresponding to all equality and inequality constraints, respectively, of (3.14).

$$\min_{\substack{\Delta s=\left(\Delta s_0^T,\dots,\Delta s_M^T\right)^T\in\mathbb{R}^{n_s}\\ \Delta q=\left(\Delta q_0^T,\dots,\Delta q_{M-1}^T\right)^T\in\mathbb{R}^{n_q}}} \quad \frac{1}{2}\begin{pmatrix}\Delta s\\ \Delta q\end{pmatrix}^T\begin{pmatrix}B^{ss} & B^{sq}\\ B^{qs} & B^{qq}\end{pmatrix}\begin{pmatrix}\Delta s\\ \Delta q\end{pmatrix}+\begin{pmatrix}b^s\\ b^q\end{pmatrix}^T\begin{pmatrix}\Delta s\\ \Delta q\end{pmatrix} \tag{3.24a}$$

$$\text{s.t.}\quad 0 = S_m^s\Delta s_m + S_m^q\Delta q_m - \Delta s_{m+1} + \delta_m, \quad m=0,\dots,M-1, \tag{3.24b}$$

$$0 \le H_m^s\Delta s_m + H_m^q\Delta q_m + h_m, \quad m=0,\dots,M-1, \tag{3.24c}$$

$$0 = R_{s_0}^{\mathrm{e}}\Delta s_0 + R_{s_M}^{\mathrm{e}}\Delta s_M + r^{\mathrm{e}}, \tag{3.24d}$$

$$0 = R_{s_0}^{\mathrm{i}}\Delta s_0 + R_{s_M}^{\mathrm{i}}\Delta s_M + r^{\mathrm{i}}, \tag{3.24e}$$

$$0 = x^j - s_0 - \Delta s_0. \tag{3.24f}$$

The convention in the following is that capital letters represent matrices, such as Jacobians and Hessians, while lowercase letters represent residuals and gradients.

To present the QP (3.24) concisely, we have introduced several abbreviations, which we define below before proceeding to investigate QP (3.24) further.

Interlude: Notation In the objective function, we have introduced

All quantities defined here can alternatively be defined as approximations of the exact Hessians and gradients.

$$B^{ss} := \nabla_{ss}^2\mathscr{L}(s,q,\lambda,\mu), \tag{3.25a}$$

$$B^{sq} := \nabla_{sq}^2\mathscr{L}(s,q,\lambda,\mu), \tag{3.25b}$$

$$B^{qs} := \nabla_{qs}^2\mathscr{L}(s,q,\lambda,\mu), \tag{3.25c}$$

$$B^{qq} := \nabla_{qq}^2\mathscr{L}(s,q,\lambda,\mu), \tag{3.25d}$$

and

$$b^s := \nabla_s \left(\sum_{m=0}^{M-1} \Psi_m\left(s_m, q_m\right) + \Phi(s_M) \right), \tag{3.26a}$$

$$b^q := \nabla_q \left(\sum_{m=0}^{M-1} \Psi_m\left(s_m, q_m\right) + \Phi(s_M) \right). \tag{3.26b}$$

One notable feature of the DMS discretization is that the Hessian of the Lagrangian $\mathscr{L}$ of the NLP (3.14) is sparse, as

$$\begin{aligned}
0 &= \nabla^2_{s_m s_l} \mathscr{L}\left(s, q, \lambda, \mu\right), \quad \text{if} \quad (m,l) \neq (0,M), (M,0) \text{ and } m \neq l, \\
0 &= \nabla^2_{s_m q_l} \mathscr{L}\left(s, q, \lambda, \mu\right) = \nabla^2_{q_l s_m} \mathscr{L}\left(s, q, \lambda, \mu\right)^T, \quad \text{if} \quad m \neq l, \\
0 &= \nabla^2_{q_m q_l} \mathscr{L}\left(s, q, \lambda, \mu\right), \quad \text{if} \quad m \neq l.
\end{aligned}$$

If the boundary constraints r^{e} and r^{i} do not depend on both arguments, i.e., s_0 and s_M, the Lagrangian is even separable.

Therefore, the Hessian blocks $B^{ss}, B^{sq}, B^{qs}, B^{qq}$ have the structure

$$B^{ss} = \begin{pmatrix}
B^{s_0 s_0} & 0 & \cdots & 0 & B^{s_0 s_M} \\
0 & B^{s_1 s_1} & 0 & \cdots & 0 \\
\vdots & 0 & \ddots & 0 & \vdots \\
0 & 0 & \cdots & B^{s_{M-1} s_{M-1}} & 0 \\
B^{s_M s_0} & 0 & \cdots & 0 & B^{s_M s_M}
\end{pmatrix},$$

$$B^{sq} = \begin{pmatrix}
B^{s_0 q_0} & \cdots & 0 \\
0 & \ddots & 0 \\
0 & \cdots & B^{s_{M-1} q_{M-1}} \\
0 & \cdots & 0
\end{pmatrix} = \left(B^{qs}\right)^T,$$

$$B^{qq} = \begin{pmatrix}
B^{q_0 q_0} & \cdots & 0 \\
0 & \ddots & 0 \\
0 & \cdots & B^{q_{M-1} q_{M-1}}
\end{pmatrix}.$$

If we are working with an approximation of the Hessian, the approximation should preserve this structure.

In the constraints, we have introduced the sensitivity matrices

$$S^s_m := \frac{\partial}{\partial s_m} x\left(\tau_{m+1}; s_m, q_m\right), \qquad m = 0, \ldots, M-1, \tag{3.27a}$$

$$S^q_m := \frac{\partial}{\partial q_m} x\left(\tau_{m+1}; s_m, q_m\right), \qquad m = 0, \ldots, M-1 \tag{3.27b}$$

alongside the Jacobians of the constraints

$$H^s_m := \frac{\partial}{\partial s_m} h\left(s_m, q_m\right), \qquad m = 0, \ldots, M-1, \tag{3.28a}$$

$$H^q_m := \frac{\partial}{\partial q_m} h\left(s_m, q_m\right), \qquad m = 0, \ldots, M-1, \tag{3.28b}$$

$$R^{\mathrm{e}}_{s_m} := \frac{\partial}{\partial s_m} r^{\mathrm{e}}(s_0, s_M), \qquad m \in \{0, M\}, \tag{3.28c}$$

$$R^{\mathrm{i}}_{s_m} := \frac{\partial}{\partial s_m} r^{\mathrm{i}}(s_0, s_M), \qquad m \in \{0, M\}. \tag{3.28d}$$

To denote the residuals, we have introduced

$$\delta_m := x(\tau_{m+1}; s_m, q_m) - s_{m+1}, \qquad m = 0, \dots, M-1, \tag{3.29a}$$
$$h_m := h(s_m, q_m), \qquad m = 0, \dots, M-1, \tag{3.29b}$$
$$r^{\mathrm{e}} := r^{\mathrm{e}}(s_0, s_M), \tag{3.29c}$$
$$r^{\mathrm{i}} := r^{\mathrm{i}}(s_0, s_M). \tag{3.29d}$$

For the boundary constraints $r^{\mathrm{e}}, r^{\mathrm{i}}$, we overload our notation as it is clear from the context whether we are referring to the mappings or their evaluations.

Now, we turn our attention back to a closer investigation of the structure of QP (3.24).

Condensing of (3.24) In QP (3.24), the steps in all node values

$$\Delta s = \left(\Delta s_0^T, \dots, \Delta s_M^T\right)^T \in \mathbb{R}^{n_s}$$

and the steps in all controls

$$\Delta q = \left(\Delta q_0^T, \dots, \Delta q_{M-1}^T\right)^T \in \mathbb{R}^{n_q}$$

are considered as optimization variables. But, the linearized matching conditions (3.24b) eliminate the degrees of freedom associated with $\Delta s_1, \dots, \Delta s_M$. This feature can be exploited to derive a QP with fewer variables and constraints, while still yielding the same steps Δs and Δq. The process of transforming QP (3.24) into this smaller QP (3.41) is referred to as condensing. To compute Δs_{m+1} for $m = 0, \dots, M-1$, we can recursively apply the linearized matching condition (3.24b) to obtain

$$\begin{aligned}
\Delta s_{m+1} &= S_m^s \Delta s_m + S_m^q \Delta q_m + \delta_m \\
&= S_m^s \left(S_{m-1}^s \Delta s_{m-1} + S_{m-1}^q \Delta q_{m-1} + \delta_{m-1}\right) + S_m^q \Delta q_m + \delta_m \\
&= S_m^s S_{m-1}^s \Delta s_{m-1} + S_m^q \Delta q_m + S_m^s S_{m-1}^q \Delta q_{m-1} + \delta_m + S_m^s \delta_{m-1} \\
&\vdots \\
&=: E_{m+1}^{s_0} \Delta s_0 + \sum_{l=0}^{m} E_{m+1}^{q_l} \Delta q_l + \hat{\delta}_m.
\end{aligned} \tag{3.30}$$

The condensing procedure was already introduced in the seminal paper on DMS [29].

The condensing matrices $E_m^{s_0}$ applied to Δs_0 are given by the recursion

$$E_m^{s_0} = S_{m-1}^s E_{m-1}^{s_0}, \ m = 2, \dots, M, \qquad \text{with} \qquad E_1^{s_0} = S_0^s. \tag{3.31}$$

The condensing matrices $E_m^{q_l}$ applied to Δq_l in the computation of Δs_m are defined for each $l = 0, \dots, m-1$ by the recursion

$$E_m^{q_l} = S_{m-1}^s E_{m-1}^{q_l}, \ m = l+2, \dots, M, \qquad \text{with} \qquad E_{l+1}^{q_l} = S_l^q. \tag{3.32}$$

The condensed matching condition residuals $\hat{\delta}_m$ are also defined by a recursion, which reads

$$\hat{\delta}_m = S_{m-1}^s \hat{\delta}_{m-1} + \delta_m, \ m = 1, \dots, M-1 \qquad \text{with} \qquad \hat{\delta}_0 = \delta_0. \tag{3.33}$$

In matrix form, we thus have

$$\begin{pmatrix} \Delta s_1 \\ \Delta s_2 \\ \vdots \\ \Delta s_M \end{pmatrix} = \begin{pmatrix} E_1^{s_0} & E_1^{q_0} & & & \\ E_2^{s_0} & E_2^{q_0} & E_2^{q_1} & & \\ \vdots & & & \ddots & \\ E_M^{s_0} & E_M^{q_0} & \cdots & & E_M^{q_{M-1}} \end{pmatrix} \begin{pmatrix} \Delta s_0 \\ \Delta q_0 \\ \vdots \\ \Delta q_{M-1} \end{pmatrix} + \begin{pmatrix} \hat{\delta}_0 \\ \hat{\delta}_1 \\ \vdots \\ \hat{\delta}_{M-1} \end{pmatrix}. \tag{3.34}$$

If we replace $\Delta s_1, \ldots, \Delta s_M$ using Equation (3.34), we also need to reformulate the objective function accordingly. To that end, we introduce the condensed Hessian

$$\left(\begin{array}{c|c} \hat{B}^{s_0 s_0} & \hat{B}^{s_0 q} \\ \hline \hat{B}^{q s_0} & \hat{B}^{q q} \end{array}\right) := \left(\begin{array}{c|ccc} \hat{B}^{s_0 s_0} & \hat{B}^{s_0 q_0} & \ldots & \hat{B}^{s_0 q_{M-1}} \\ \hline \hat{B}^{q_0 s_0} & \hat{B}^{q_0 q_0} & \ldots & \hat{B}^{q_0 q_{M-1}} \\ \vdots & \vdots & & \vdots \\ \hat{B}^{q_{M-1} s_0} & \hat{B}^{q_{M-1} q_0} & \ldots & \hat{B}^{q_{M-1} q_{M-1}} \end{array}\right) \tag{3.35}$$

with

$$\hat{B}^{s_0 s_0} := B^{s_0 s_0} + \sum_{m=1}^{M} \left(E_m^{s_0}\right)^T B^{s_m s_m} E_m^{s_0} + B^{s_0 s_M} E_M^{s_0} + \left(E_M^{s_0}\right)^T B^{s_M s_0}, \tag{3.36a}$$

$$\left(\hat{B}^{q_0 s_0}\right)^T = \hat{B}^{s_0 q_0} := B^{s_0 q_0} + B^{s_0 s_M} E_M^{q_0} + \sum_{l=1}^{M} \left(E_l^{s_0}\right)^T B^{s_l s_l} E_l^{q_0}, \tag{3.36b}$$

$$\left(\hat{B}^{q_m s_0}\right)^T = \hat{B}^{s_0 q_m} := \left(E_m^{s_0}\right)^T B^{s_m q_m} + B^{s_0 s_M} E_M^{q_m} + \sum_{l=m+1}^{M} \left(E_l^{s_0}\right)^T B^{s_l s_l} E_l^{q_m}, \; m = 1, \ldots, M-1, \tag{3.36c}$$

$$\hat{B}^{q_m q_m} := B^{q_m q_m} + \sum_{l=m+1}^{M} \left(E_l^{q_m}\right)^T B^{s_l s_l} E_l^{q_m}, \; m = 0, \ldots, M-1, \tag{3.36d}$$

$$\left(\hat{B}^{q_p q_m}\right)^T = \hat{B}^{q_m q_p} := \left(E_p^{q_m}\right)^T B^{s_p q_p} + \sum_{l=p+1}^{M} \left(E_l^{q_m}\right)^T B^{s_l s_l} E_l^{q_p}, \; m = 0, \ldots, M-2, \; p = m+1, \ldots, M-1. \tag{3.36e}$$

Moreover, we introduce the condensed gradients

$$\left(\begin{array}{c} \hat{b}^{s_0} \\ \hline \hat{b}^{q} \end{array}\right) := \left(\begin{array}{c} \hat{b}^{s_0} \\ \hline \hat{b}^{q_0} \\ \vdots \\ \hat{b}^{q_{M-1}} \end{array}\right) \tag{3.37}$$

with

$$\hat{b}^{s_0} := b^{s_0} + \sum_{m=1}^{M} \left(E_m^{s_0}\right)^T \left(B^{s_m s_m} \hat{\delta}_{m-1} + b^{s_m}\right) + B^{s_0 s_M} \hat{\delta}_{M-1}, \tag{3.38a}$$

$$\hat{b}^{q_0} := b^{q_0} + \sum_{l=1}^{M} \left(E_l^{q_0}\right)^T \left(b^{s_l} + B^{s_l s_l} \hat{\delta}_{l-1}\right), \tag{3.38b}$$

$$\hat{b}^{q_m} := b^{q_m} + B^{q_m s_m} \hat{\delta}_{m-1} + \sum_{l=m+1}^{M} \left(E_l^{q_m}\right)^T \left(b^{s_l} + B^{s_l s_l} \hat{\delta}_{l-1}\right), \tag{3.38c}$$

for $m = 0, \ldots, M-1$.

The formulas (3.36a), (3.36b), (3.36c) and (3.38a) differ from their counterparts presented in [200, p.35]. This is because the boundary constraints r^e and r^i are forgotten in [200] in the QP formulation, which has the label (2.30) in their work, that corresponds to the DMS NLP, which has the label (2.19).

Definition: b^{s_m} and b^{q_m}

b^{s_m} is the part of b^s that corresponds to s_m, i.e.

$$b^{s_m} = \begin{cases} \nabla_{s_M} \Phi(s_M), & \text{if } m = M, \\ \nabla_{s_m} \Psi_m(s_m, q_m), & \text{else.} \end{cases}$$

And similarly for b^{q_m}:

$$b^{q_m} = \nabla_{q_m} \Psi_m(s_m, q_m), \; 0 \leq m < M.$$

To reformulate the mixed state-control constraints (3.24c) in terms of Δs_0 and Δq, we introduce the condensed mixed state-control constraints matrices and residuals

$$\hat{H}_m^s := H_m^s E_m^{s_0}, \qquad m = 1, \dots, M-1, \tag{3.39a}$$

$$\hat{H}_m^{q_l} := \begin{cases} H_m^s E_m^{q_l}, & \text{if } l = 0, \dots, m-1, \\ H_m^q, & \text{if } l = m \end{cases} \qquad m = 1, \dots, M-1, \tag{3.39b}$$

$$\hat{h}_m := h_m + H_m^s \hat{\delta}_{m-1}, \qquad m = 1, \dots, M-1. \tag{3.39c}$$

To reformulate the boundary constraints (3.24d) and (3.24e), we introduce the condensed boundary equality and inequality constraints matrices and residuals

$$\hat{R}_0^{\mathrm{e}} := R_{s_0}^{\mathrm{e}} + R_{s_M}^{\mathrm{e}} E_M^{s_0}, \quad \hat{R}_0^{\mathrm{i}} := R_{s_0}^{\mathrm{i}} + R_{s_M}^{\mathrm{i}} E_M^{s_0}, \tag{3.40a}$$

$$\hat{R}_{q_m}^{\mathrm{e}} := R_{s_M}^{\mathrm{e}} E_M^{q_m}, \quad \hat{R}_{q_m}^{\mathrm{i}} := R_{s_M}^{\mathrm{i}} E_M^{q_m}, \quad m = 0, \dots, M-1, \tag{3.40b}$$

$$\hat{r}^{\mathrm{e}} := r^{\mathrm{e}} + R_{s_M}^{\mathrm{e}} \hat{\delta}_{M-1}, \quad \hat{r}^{\mathrm{i}} := r^{\mathrm{i}} + R_{s_M}^{\mathrm{i}} \hat{\delta}_{M-1}. \tag{3.40c}$$

We can now state the condensed QP

$$\min_{\substack{\Delta s_0 \in \mathbb{R}^{n_x} \\ \Delta q = \begin{pmatrix} \Delta q_0 \\ \vdots \\ \Delta q_{M-1} \end{pmatrix} \in \mathbb{R}^{n_q}}} \quad \frac{1}{2} \begin{pmatrix} \Delta s_0 \\ \Delta q \end{pmatrix}^T \begin{pmatrix} \hat{B}^{s_0 s_0} & \hat{B}^{s_0 q} \\ \hat{B}^{q s_0} & \hat{B}^{q q} \end{pmatrix} \begin{pmatrix} \Delta s_0 \\ \Delta q \end{pmatrix} + \begin{pmatrix} \hat{b}^{s_0} \\ \hat{b}^{q} \end{pmatrix}^T \begin{pmatrix} \Delta s_0 \\ \Delta q \end{pmatrix} \tag{3.41a}$$

$$\text{s.t.} \quad 0 \leq H_0^s \Delta s_0 + H_0^q \Delta q_0 + h_m, \tag{3.41b}$$

$$0 \leq \hat{H}_m^s \Delta s_0 + \sum_{l=0}^{m} \hat{H}_m^{q_l} \Delta q_m + \hat{h}_m, \quad m = 1, \dots, M-1, \tag{3.41c}$$

$$0 = \hat{R}_0^{\mathrm{e}} \Delta s_0 + \sum_{l=0}^{M-1} \hat{R}_{q_l}^{\mathrm{e}} \Delta q_l + \hat{r}^{\mathrm{e}}, \tag{3.41d}$$

$$0 \leq \hat{R}_0^{\mathrm{i}} \Delta s_0 + \sum_{l=0}^{M-1} \hat{R}_{q_l}^{\mathrm{i}} \Delta q_l + \hat{r}^{\mathrm{i}}, \tag{3.41e}$$

$$0 = x^j - s_0 - \Delta s_0. \tag{3.41f}$$

The constraint (3.41b) is equal to (3.24c) with $m = 0$. For $m > 0$ (3.24c) is reformulated in terms of Δs_0 and Δq leading to (3.41c). Even though the QP (3.41) appears to have more constraints, it indeed has Mn_x constraints less than the uncondensed QP (3.24) as the linearized matching conditions (3.24b) vanish.

Blow up of the solution of the condensed QP (3.41) If we solve the condensed QP (3.41), we only obtain steps Δs_0 and Δq in the first place. We can however compute the remaining steps Δs_m, $m = 1, \dots, M$ by means of Equation (3.34). The LAGRANGE multipliers that we obtain for the condensed constraints (3.41b) - (3.41f) are equal to the ones that correspond to the constraints (3.24c) - (3.24f). The LAGRANGE multipliers for the linearized matching conditions (3.24b) can be reconstructed as described in [127, p. 92][128].

SQP method for NLP (3.14) with condensing We summarize the tailored SQP method for the NLP (3.14) with condensing, as described in this section, in Algorithm 3.2. Depending on the use case, it may be beneficial to use one of the variations of the condensing procedure, as presented in [8, 117, 170].

[8]: Andersson (2013), "A general-purpose software framework for dynamic optimization"
[117]: Kirches et al. (2011), "Block-structured quadratic programming for the direct multiple shooting method for optimal control"
[127]: Leineweber (1995), "Analyse und Restrukturierung eines Verfahrens zur direkten Lösung von Optimal-Steuerungsproblemen"
[128]: Leineweber (1999), *Efficient reduced SQP methods for the optimization of chemical processes described by large sparse DAE models*
[170]: Scholz (2016), "Stabiles Condensing für Optimale Steuerung"

Algorithm 3.2: Tailored SQP method for NLP (3.14) with condensing.

Input: Current state x^j, initial guesses $s^0, q^0, \lambda^0, \mu^0$

while *convergence criterion not satisfied* **do**

1 Evaluate constraint residuals (3.33), Evaluate obj. fct. gradients (3.26), Evaluate sensitivity matrices (3.27), Evaluate Jacobians (3.28), Evaluate Hessian (3.25) of Lagrangian

$$\begin{aligned} \delta_m, h_m, r^{\mathrm{e}}, r^{\mathrm{i}} &\leftarrow \\ b^s, b^q &\leftarrow \\ S_m^s, S_m^q &\leftarrow \\ H_m^s, H_m^q, R_s^{\mathrm{e}}, R_s^{\mathrm{i}} &\leftarrow \\ B^{ss}, B^{sq}, B^{qs}, B^{qq} &\leftarrow \end{aligned}$$

2 Condense matching condition residuals (3.33), Compute condensing matrices (3.31), (3.32)

$$\begin{aligned} \hat{\delta}_m &\leftarrow \\ E_m^{s_0}, E_m^{q_l} &\leftarrow \end{aligned}$$

3 Compute condensed gradients (3.37), (3.38), Condense mixed state-control constraints (3.24c) using (3.39), Condense boundary constraints (3.24d) and (3.24e) using (3.40), Condense Hessian using (3.35), (3.36)

$$\begin{aligned} \hat{b}^{s_0}, \hat{b}^q &\leftarrow \\ \hat{h}_m, \hat{H}_m^{q_l}, \hat{H}_m^s &\leftarrow \\ \hat{r}^{\mathrm{e}}, \hat{R}_{q_m}^{\mathrm{e}}, \hat{R}_0^{\mathrm{e}}, \hat{r}^{\mathrm{i}}, \hat{R}_{q_m}^{\mathrm{i}}, \hat{R}_0^{\mathrm{i}} &\leftarrow \\ \hat{B}^{s_0 s_0}, \hat{B}^{s_0 q}, \hat{B}^{q s_0}, \hat{B}^{qq} &\leftarrow \end{aligned}$$

4 Evaluate initial value constraint (3.41f)

5 $\Delta s_0, \Delta q, \hat{\lambda}_{\mathrm{QP}}, \hat{\mu}_{\mathrm{QP}} \leftarrow$ Solve condensed QP (3.41)

6 Blow up node value steps using (3.34), Compute LAGRANGE multipliers, cf. p. 38

$$\begin{aligned} \Delta s_1, \ldots, \Delta s_M &\leftarrow \\ \lambda_{\mathrm{QP}}, \mu_{\mathrm{QP}} &\leftarrow \end{aligned}$$

7 Update iterates by

$$\begin{aligned} s^{k+1} &= s^k + \Delta s, \\ q^{k+1} &= q^k + \Delta q, \\ \lambda^{k+1} &= \lambda_{\mathrm{QP}}, \\ \mu^{k+1} &= \mu_{\mathrm{QP}}. \end{aligned}$$

3.4. Real-Time Iterations

Using DMS as a tool to discretize 3.1 and the tailored SQP method, as described in Algorithm 3.2, to solve the resulting DMS NLP (3.14), we can, in principle, perform nominal NMPC as outlined in Algorithm 3.3.

Algorithm 3.3: Nominal NMPC with DMS and tailored SQP method.

Input: Time grid $t_0 < t_1 < \ldots < t^j < \ldots$

At each sampling time t^j, $j \in \mathbb{N}_0$ do:

1 $x^j \leftarrow$ Get current state
2 NLP (3.14) $\leftarrow$ Apply DMS to discretize 3.1
3 $q_0 \leftarrow$ Apply the tailored SQP method Algorithm 3.2 to (3.14)
4 $u_0^j \leftarrow$ Set feedback value using (2.1) and (3.4)
5 Apply u_0^j to the system

In most cases, the DMS setup, such as the choice of the shooting grid or the control basis functions, remains unchanged throughout the entire NMPC procedure. In these cases, we only need to update the DMS NLP with the current variables in step 2 in Algorithm 3.3.

However, Algorithm 3.3 has one major drawback: iterating the tailored SQP method in step 3 until convergence can be time-consuming. While iterating until convergence provides an accurate feedback value for the system at time t^j, the runtime required forces us to apply the feedback value only at time $t^j + \Delta t_\mathrm{d}$, introducing a feedback delay $\Delta t_\mathrm{d} > 0$. Although u_0^j is an accurate feedback value for the system at time t^j, it may be suboptimal at time $t^j + \Delta t_\mathrm{d}$.

One way to address this issue is to predict the state x^{j+1} and compute an accurate feedback for this predicted state. This is the idea behind the advanced-step NMPC controller presented in [211].

[211]: Zavala et al. (2009), "The advanced-step NMPC controller: Optimality, stability and robustness"

Alternatively, we can compute only an approximate feedback for the system at time t^j, minimizing the feedback delay. This approach forms the basis of the Real-Time Iterations (RTI), primarily developed by Diehl in his PhD thesis [57] and in [59, 61, 64, 67]. The RTI achieves a significant reduction in feedback delay through two main ideas:

Combinations of the advanced-step NMPC controller and RTI have also been developed in [145, 146].

[57]: Diehl (2001), "Real-time optimization for large scale nonlinear processes"
[59]: Diehl et al. (2003), "Newton-type methods for the approximate solution of nonlinear programming problems in real-time"
[61]: Diehl et al. (2002), "Real-time optimization and nonlinear model predictive control of processes governed by differential-algebraic equations"
[64]: Diehl et al. (2005), "Nominal stability of real-time iteration scheme for nonlinear model predictive control"
[67]: Diehl et al. (2001), "Real-time optimization for large scale processes: Nonlinear model predictive control of a high purity distillation column"

Main idea — Real-Time Iterations (RTI) – I. The SQP method is not iterated until convergence. Instead, only a single SQP iteration, i.e., a single (condensed) QP solve, is performed per sampling time.

Main idea — Real-Time Iterations (RTI) – II. The computations are divided into three phases: preparation, feedback, and transition. The most computationally expensive tasks are moved to the preparation phase, which occurs before the sampling time t^j. Once the current state x^j is known at sampling time t^j, the feedback phase is executed. This phase involves only a single (condensed) QP solve, making it very fast, and the feedback value u_0^j is applied to the system. In the transition phase, which occurs after the feedback phase and before the next preparation phase, the optimization variables are updated. If idle time remains, additional auxiliary computations can be performed during the transition phase. The timing of these phases and the communication between the system and the optimizer are illustrated in Figure 3.3.

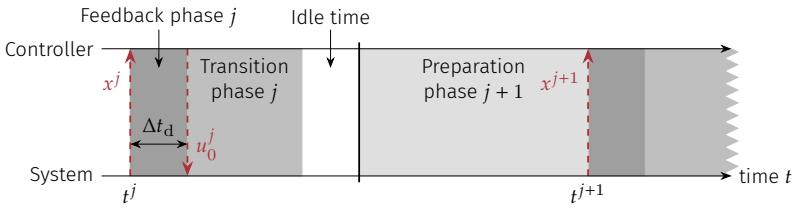


Figure 3.3.: Timing of the RTI phases and temporal communication (highlighted in red) between the system and the optimizer. Adapted from [171, Figure 4.4].

Key to these two ideas is the Initial Value Embedding (IVE), which we describe in Subsection 3.4.1. The main steps of NMPC with RTI for the sampling time t^j are outlined in Algorithm 3.4. Details on the three referenced phases are provided in Subsection 3.4.2.

Algorithm 3.4: RTI scheme at sampling time t^j.

Input: Primal variables s^{j-1}, q^{j-1}, dual variables λ^{j-1}, μ^{j-1} from previous sampling time t^{j-1}

Output: Primal variables s^j, q^j, dual variables λ^j, μ^j for current sampling time t^j

Preparation phase (between t^{j-1} and t^j):

1 Prepare condensed QP (3.41) except for (3.41f)

At sampling time t^j:

2 Current state x^j becomes known

Feedback phase (from t^j to $t^j + \Delta t_\mathrm{d}$):

3 Evaluate (3.41f) to complete QP (3.41)

4 $q_0 \leftarrow$ Solve QP (3.41)

5 $u_0^j \leftarrow$ Set feedback value using (2.1) and (3.4)

At time $t^j + \Delta t_\mathrm{d}$:

6 Apply u_0^j to the system

Transition phase (after $t^j + \Delta t_\mathrm{d}$):

7 Blow up solution of QP (3.41)

Since its development, the RTI scheme has been successfully applied to numerous real-world problems, ranging from the control of high-purity distillation columns [67] to optimal robot control [58], the control of power-generating kites and wind turbines [66, 73, 97–99, 115, 210], autonomous vehicles [209], and embedded control of cranes [110] or ground vehicles [80].

The RTI scheme is occasionally regarded in Time-Distributed Optimization (TDO) as a specific variant of Time-Distributed Sequential Quadratic Programming (TD-SQP), for example in [132].

[58]: Diehl et al. (2006), "Fast Direct Multiple Shooting Algorithms for Optimal Robot Control"
[66]: Diehl et al. (2004), "Efficient NMPC of unstable periodic systems using approximate infinite horizon closed loop costing"
[67]: Diehl et al. (2001), "Real-time optimization for large scale processes: Nonlinear model predictive control of a high purity distillation column"
[73]: Ferreau et al. (2011), "Real-time control of a kite-model using an auto-generated nonlinear MPC algorithm"
[80]: Frasch et al. (2013), "An auto-generated nonlinear MPC algorithm for real-time obstacle avoidance of ground vehicles"
[97]: Gros et al. (2012), "Aircraft control based on fast non-linear MPC & multiple-shooting"
[115]: Ilzhöfer et al. (2007), "Nonlinear MPC of kites under varying wind conditions for a new class of large-scale wind power generators"
[209]: Zanon et al. (2014), "Model Predictive Control of Autonomous Vehicles"
[210]: Zanon et al. (2014), "Control of Dual-Airfoil Airborne Wind Energy systems based on nonlinear MPC and MHE"

[132]: Liao-McPherson et al. (2020), "Time-distributed optimization for real-time model predictive control: Stability, robustness, and constraint satisfaction"

3.4.1. Initial Value Embedding

When transitioning from one sampling time to the next, and consequently from one NLP solution process to the next, an important question arises: what initial guesses should be used to restart the SQP iterations? The conventional approach is as follows.

Conventional approach Once the current state x^j becomes known, we set $s_0 = x^j$ and perform a forward integration using the control sequence $\left(u_0^{j-1}, \dots, u_N^{j-1}\right)$ computed at the previous sampling time t^{j-1} to determine $s_1, \dots, s_M$. We then start Algorithm 3.2 with this initial guess for the node values and controls. While the initial value constraint (3.14f) and the matching conditions (3.14b) are already satisfied with this initial guess, this approach has a significant drawback: three computationally expensive tasks must be performed after the sampling time t^j. Consequently, the feedback delay is substantial. These three tasks are the integration of the IVPs (3.6), the derivative computations for QP (3.24) at the specified initial guess, and the condensing of QP (3.24) as described in Subsection 3.3.2. Additionally, the integration can fail, similar to single shooting. This contradicts the paradigm of providing feedback as quickly as possible.

Initial Value Embedding Fortunately, all three expensive computations can be moved to the preparation phase. This is achieved by using the states and controls computed at the previous sampling time as initial guesses, along with the derivatives from the previous sampling time. Specifically, the derivatives are evaluated at these states and controls, ensuring consistency between the derivatives and variables. The initial guess for s_0 will not initially match x^j. However, since the current state, or initial value, is embedded linearly into the NLP (3.14) via (3.14f), s_0 will equal x^j after the first SQP iteration. In addition to enabling the relocation of expensive computations to the preparation phase, the IVE initialization is reported to exhibit faster convergence of the SQP method, as noted in [65].

The IVE initialization technique is not unique to the RTI. Instead, the IVE can be more generally understood as a technique to augment parameterized NLPs by adding a trivial constraint that fixes an optimization variable to the parameterization parameter. More details can be found in [200, Section 4.1].

[65]: Diehl et al. (2002), "An Efficient Algorithm for Nonlinear Model Predictive Control of Large-Scale Systems Part I: Description of the Method (Ein effizienter Algorithmus für die nichtlineare prädiktive Regelung großer Systeme Teil I: Methodenbeschreibung)"

3.4.2. RTI phases

As described in the main idea of RTI - II, we divide the computations required for the single SQP iteration into three phases. The temporal sequence of these phases was illustrated already in Figure 3.3 and Algorithm 3.4. Below, we discuss these phases in more detail.

Preparation phase The IVE initialization strategy allows us to perform steps 1 – 3 of Algorithm 3.2 during the preparation phase. According to the IVE, the current guess for the optimization variables is provided by the primal variables s^{j-1}, q^{j-1} and the dual variables λ^{j-1}, μ^{j-1} from the previous sampling time t^{j-1}. All evaluations and derivatives in step 1 of Algorithm 3.2 are based on these variables.

Feedback phase Once the current state x^j becomes known, we can finalize the setup of the condensed QP (3.41) (step 3 in Algorithm 3.4). This marks the first step of the feedback phase. In the second step, we solve the QP (3.41) to obtain s_0^j, q^j (step 4 in Algorithm 3.4). We then immediately apply the corresponding control u_0^j derived from q_0^j (steps 5 – 6 in Algorithm 3.4).

Thus, the feedback phase primarily consists of solving the QP. Consequently, the feedback delay is determined by the duration of the QP solution process. A fast QP solution method is therefore crucial for the successful implementation of the RTI scheme. As noted above, we must solve a sequence of QPs parameterized by the current state x^j, with only small changes in the matrices between subsequent feedback phases. A QP solution method specifically tailored for such sequences is the online active set method for quadratic programming. Originally, the foundations of this method were developed independently of the MPC context under the name primal-dual parametric quadratic programming method by Best [23]. Later, Ferreau, Bock, and Diehl revisited this method in the MPC context, leading to the online active set method presented in [72], which was subsequently implemented in the open-source software package qpOASES [74]. It has been successfully combined with RTI, for example in [80, 195], and its enhancement MLI, as demonstrated in [171–173, 200, 203].

[23]: Best (1996), "An Algorithm for the Solution of the Parametric Quadratic Programming Problem"
[72]: Ferreau et al. (2008), "An online active set strategy to overcome the limitations of explicit MPC"
[74]: Ferreau et al. (2014), "qpOASES: a parametric active-set algorithm for quadratic programming"
[80]: Frasch et al. (2013), "An auto-generated nonlinear MPC algorithm for real-time obstacle avoidance of ground vehicles"
[172]: Scholz et al. (2020), "Model-based optimal feedback control for microgrids with multi-level iterations"
[173]: Scholz et al. (2021), "Multi-level iterations for microgrid control with automatic level choice"
[171]: Scholz (2022), "Model-based optimal feedback control For microgrids"
[195]: Verschueren et al. (2016), "Time-optimal race car driving using an online exact hessian based nonlinear MPC algorithm"
[203]: Wirsching et al. (2007), "An online active set strategy for fast adjoint based nonlinear model predictive control"
[200]: Wirsching (2018), "Multi-level iteration schemes with adaptive level choice for nonlinear model predictive control"

Transition phase In the transition phase, we compute the remaining variables $s_1^j \ldots, s_M^j$, λ^j, μ^j by expanding the result of QP (3.41), as described on p. 38. These computations correspond to step 6 and step 7 of Algorithm 3.2.

3.4.3. Theoretical aspects

So far, we have focused on explaining how the RTI scheme enables a fast numerical method for NMPC, but we have not yet provided insights into the theoretical aspects of the RTI. Therefore, we briefly present the interpretation of the QP step as a Tangential Predictor (TP) and mention relevant stability results for NMPC with RTI.

The QP solution as TP The NLP (3.14) is parameterized by the current state x^j. From parametric optimization, as discussed in [103], we know that the local minimizer of (3.14) depends piecewise differentiably on x^j under suitable assumptions, which are detailed in [103]. If the variables at sampling time t^{j-1} are indeed a local minimizer of (3.14), it has been shown in [57, Theorem 3.6] that the solution step of the QP (3.24) or (3.41), respectively, serves as a generalized TP for solving (3.14) with the updated system state x^j, provided the IVE initialization strategy is used. In simple terms, a TP is a first-order approximation at smooth parts of the solution manifold. A generalized TP can even "jump" over non-differentiable points of the solution manifold by accounting for active set changes. For further explanations, see [57, Section 3.4], [62, Section 5], or [200, Section 4.1]. The concept is illustrated in Figure 3.4. As noted in [62, p. 408], the QP solution step is only approximately a generalized TP if

[103]: Guddat et al. (1990), *Parametric Optimization: Singularities, Pathfollowing and Jumps*

[57]: Diehl (2001), "Real-time optimization for large scale nonlinear processes"

[62]: Diehl et al. (2009), "Efficient Numerical Methods for Nonlinear MPC and Moving Horizon Estimation"
[200]: Wirsching (2018), "Multi-level iteration schemes with adaptive level choice for nonlinear model predictive control"

(i) the variables from the previous sampling time t^{j-1} are only an approximate local minimizer, or
(ii) any of the Jacobians, the Hessian, the Lagrange gradient, or the constraint residuals are not evaluated exactly.

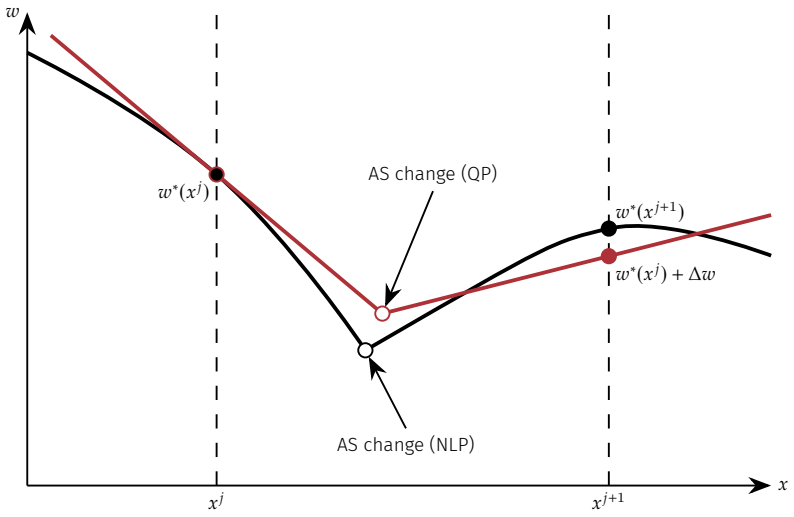


Figure 3.4.: Illustration of the generalized TP adapted from [200, Figure 4.2]. The solution manifold of the NLP (3.14) (black) exhibits a non-differentiability where an active set change occurs between x^j and x^{j+1}. If we set up the QP (3.24) in the optimal primal solution $w^*(x^{j+1})$, where $w = (s^T, q^T)^T$, the solution manifold of the QP (3.24) (red) also exhibits an active set change near the active set change in the NLP. The step $\Delta\Delta w$ computed by solving the QP (3.24) remains a good approximation to the true solution $w^*(x^{j+1})$ as it "jumps" over the active set changes.

Reminder: Stability of NMPC

We provide background information about the stability of NMPC schemes in Section 2.2.In the following, we reference works that address the conditions under which NMPC with RTI can still yield a stable NMPC scheme. Nominal stability refers to the stability of nominal NMPC, i.e., NMPC where the system behaves exactly as predicted, see Definition 2.7.

[63]: Diehl et al. (2007), "A Stabilizing Real-Time Implementation of Nonlinear Model Predictive Control"
[64]: Diehl et al. (2005), "Nominal stability of real-time iteration scheme for nonlinear model predictive control"
[208]: Zanelli et al. (2021), "A Lyapunov function for the combined system-optimizer dynamics in inexact model predictive control"

Stability results In [57, 60], contractivity of the optimization variables was demonstrated, meaning that the SQP iterations, disturbed by the varying current state x^j, still converge. However, this result does not directly imply the stability of the resulting NMPC scheme. Subsequent works have shown nominal stability of NMPC with RTI without mixed state-control constraints, using terminal constraints (e.g., [64]) or terminal costs (e.g., [63]). A more general approach is presented in [208], where asymptotic stability of the combined system-optimizer dynamics is shown for inexact NMPC with Q-linearly convergent optimization algorithms. Additionally, [208] references several related results on the stability of NMPC schemes similar to the RTI. In Chapter 4 of this thesis, we contribute a stability proof for RTI-like schemes for a class of semilinear parabolic PDEs, using [208] as a starting point.

3.5. Multi-Level Iterations

Reminder: Sampling period

The sampling period is the time interval between two subsequent sampling times, see Definition 2.1.

[200]: Wirsching (2018), "Multi-level iteration schemes with adaptive level choice for nonlinear model predictive control"
[171]: Scholz (2022), "Model-based optimal feedback control For microgrids"

The RTI scheme presented in Section 3.4 enables us to drastically reduce the feedback delay by moving the expensive integration and derivative computations to the preparation phase, which takes place before the actual sampling time t^j. However, these expensive computations still need to be carried out. Consequently, the sampling period must be long enough to accommodate the preparation phase. In other words, while the RTI scheme leads to shorter feedback delays, the expensive computations in the preparation phase still limit how quickly we can sample. This limitation is particularly problematic for systems with very fast dynamics, such as a chain of masses connected by springs, as considered in [200, Section 10.1], or the control of microgrids, as discussed in [171–173]. In such cases, the resulting NMPC scheme may be too slow to achieve good performance.

The next stage in the development of the RTI scheme, which addresses the challenge of accelerating the preparation phase, is the Multi-Level Iterations (MLI) scheme. The MLI scheme is built around the idea of employing different variations, known as levels, of inexact SQP instead of the exact SQP used in the RTI scheme. The MLI scheme was first introduced in [28, Section 4] and has since been successfully applied to mechanical and chemical processes, as documented in [5, 133, 201–203]. Extensions of the MLI scheme have been developed, including EULER steps for model updates and a local feedback law based on the fastest MLI level in [120], mixed-level and fractional-level MLI schemes in [81], and its application to NMPC on long horizons in [119]. Moreover, the PhD thesis [200] by WIRSCHING provides a comprehensive overview of the MLI scheme. In recent years, the MLI scheme has been applied to economic NMPC in [144], extended to Moving Horizon Estimation (MHE), and subsequently applied to microgrid control in [104, 171–173]. It has also been combined with the advanced-step controller in [82, 145]. Finally, initial ideas for extending the MLI scheme to systems with switches have been proposed in [138].

In the following, we first present the different MLI levels in Subsection 3.5.1 and then discuss additional aspects, such as communication and scheduling of the levels, in Subsection 3.5.2.

[5]: Albersmeyer et al. (2009), "Fast Nonlinear Model Predictive Control with an Application in Automotive Engineering"
[28]: Bock et al. (2007), "Constrained Optimal Feedback Control of Systems Governed by Large Differential Algebraic Equations"
[81]: Frasch et al. (2012), "Mixed-Level Iteration Schemes for Nonlinear Model Predictive Control"
[82]: Frey et al. (2024), "Advanced-Step Real-Time Iterations With Four Levels – New Error Bounds and Fast Implementation in acados"
[104]: Gutekunst et al. (2020), "Fast moving horizon estimation using multi-level iterations for microgrid control"
[119]: Kirches et al. (2012), "Efficient direct multiple shooting for nonlinear model predictive control on long horizons"
[138]: Meyer (2020), "Numerical solution of optimal control problems with explicit and implicit switches"
[144]: Nurkanović et al. (2021), "Multi-level Iterations for Economic Nonlinear Model Predictive Control"
[172]: Scholz et al. (2020), "Model-based optimal feedback control for microgrids with multi-level iterations"
[201]: Wirsching et al. (2008), "An Adjoint-based Numerical Method for Fast Nonlinear Model Predictive Control"

3.5.1. MLI levels

The MLI scheme is an extension of the RTI scheme. As such, we still divide the relevant computations into a preparation phase, a feedback phase, and a transition phase, as described in Subsection 3.4.2. The timing of these phases remains as presented in Algorithm 3.4 and Figure 3.3. Essentially, the MLI scheme comprises different strategies for setting up the QP (3.41) in step 1 of Algorithm 3.4. The computations performed in the preparation phase thus vary depending on the strategy. These strategies are referred to as levels in MLI. The different levels can communicate, i.e., exchange data, with each other. This communication occurs during the transition phase. The feedback phase, as described in steps 3 – 6 of Algorithm 3.4, remains unchanged. For NMPC with MLI, the feedback phase still involves solving QP (3.41).

To present the MLI scheme concisely, and since the specific structure of the QP (3.41) is less significant at this point, we consolidate all optimization variables, equality constraints, and inequality constraints into single expressions. We write QP (3.41) for level $\mathrm{X} \in \{\mathrm{A}, \mathrm{B}, \mathrm{C}, \mathrm{D}\}$ at sampling time t^j in the form

$$\min_{\Delta w} \quad \frac{1}{2}\Delta w^T Q_{\mathrm{X}}^j \Delta w + \left(p_{\mathrm{X}}^j\right)^T \Delta w \tag{3.42a}$$

$$\text{s.t.} \quad 0 = c_{\mathrm{X}}^j + C_{\mathrm{X}}^j \Delta w, \tag{3.42b}$$

$$0 \leq d_{\mathrm{X}}^j + D_{\mathrm{X}}^j \Delta w. \tag{3.42c}$$

Attention:

In our description of the tailored SQP method and the RTI in Subsection 3.3.2 and Section 3.4, we have generally assumed that all derivatives are computed exactly, even though many implementations of the RTI use approximations. In Levels A–C of the MLI scheme, only a subset of the residuals and derivatives is computed with the current variable set. Therefore, the components of QP (3.42) should be understood as approximations in most cases in this section.

[200]: Wirsching (2018), "Multi-level iteration schemes with adaptive level choice for nonlinear model predictive control"

The main idea of the MLI scheme is to update the components of QP (3.42) differently, depending on the current level. Typically, as described in [200], the MLI scheme comprises four levels, which are:

- Level D, or full linearization iterations,
- Level C, or optimality iterations,
- Level B, or feasibility iterations,
- Level A, or feedback iterations.

The computational complexity decreases from Level D to Level A.

In addition to the components of QP (3.42), each level $\mathrm{X} \in \{\mathrm{B}, \mathrm{C}, \mathrm{D}\}$ maintains its own set of primal and dual variables w_{X}^j, λ_{X}^j, μ_{X}^j. Level A does not maintain its own variable set. Instead, Level A holds reference primal and dual variables. Similarly, Level B and C maintain reference data, as described in the respective paragraphs for each level.

Level D – Full linearization iterations Level D corresponds to the RTI scheme. Thus, we perform step 1 of Algorithm 3.4 in the preparation phase using the current Level D variables w_{D}^j, λ_{D}^j, μ_{D}^j. These variables are typically the results from the previous sampling time when Level D was executed. Alternatively, though less commonly, these variables may have been communicated from one of the lower levels. This is a viable option, for instance, when Level D is scheduled infrequently, and its previous set of variables might therefore be outdated.

Let Δw and λ_{QP}, μ_{QP} be the solution obtained by solving QP (3.42) with its components provided by Level D. During the transition phase of Level D, its variables are updated according to

$$w_{\mathrm{D}}^{j+1} = w_{\mathrm{D}}^j + \Delta w, \qquad \lambda_{\mathrm{D}}^{j+1} = \lambda_{\mathrm{QP}}, \qquad \mu_{\mathrm{D}}^{j+1} = \mu_{\mathrm{QP}}.$$

The alternative name "full linearization iterations" arises from the fact that the constraint Jacobians C_{D}^j, D_{D}^j and the objective function gradient p_{D}^j are computed. The lion's share of the computational load in Level D stems from these first-order derivative computations and the calculation of the new Hessian Q_{D}^j, unless cheaper Hessian approximations are used. Since Level D aligns with the RTI scheme, it also inherits the convergence and stability properties of the RTI discussed in Subsection 3.4.3.

Level C – Optimality iterations For many systems, the linearizations and the Hessian do not change significantly from one sampling time to the next. This is particularly true when very short sampling periods are achieved, which is the primary goal of the MLI scheme. At the same time, computing these linearizations and the Hessian is the most computationally expensive task in Level D. Therefore, the idea of Level C is to reuse the linearizations and Hessian from previous Level D iterations. To achieve this, Level C maintains reference values $\bar{Q}_{\mathrm{C}}^j, \bar{C}_{\mathrm{C}}^j, \bar{D}_{\mathrm{C}}^j$ for the Hessian and constraint Jacobians. These reference values are communicated from Level D. If Level D is not scheduled at all, the reference values can be fixed values provided at the start of

the NMPC procedure. A typical choice in this case, especially for tracking or stabilizing NMPC, is to evaluate the linearizations and Hessian around a target state that the system is being steered toward. These reference values are then used for the linearizations and Hessian, i.e., we set:

$$Q_{\mathrm{C}}^{j} = \bar{Q}_{\mathrm{C}}^{j}, \qquad C_{\mathrm{C}}^{j} = \bar{C}_{\mathrm{C}}^{j}, \qquad D_{\mathrm{C}}^{j} = \bar{D}_{\mathrm{C}}^{j}.$$

The residuals c_{C}^{j} and d_{C}^{j} are newly evaluated at w_{C}^{j}.

As a result of reusing the matrices, we need to replace the gradient of the NLP objective function in the QP objective with a modified gradient. This modified gradient, and thus p_{C}^{j}, is given by

$$\begin{aligned} p_{\mathrm{C}}^{j} &= \begin{pmatrix} b^{s} \\ b^{q} \end{pmatrix} + \left(C_{\mathrm{C}}^{j} - \frac{\partial}{\partial w} c_{\mathrm{C}}^{j} \right)^{T} \lambda_{\mathrm{C}}^{j} + \left(D_{\mathrm{C}}^{j} - \frac{\partial}{\partial w} d_{\mathrm{C}}^{j} \right)^{T} \mu_{\mathrm{C}}^{j} \\ &= \nabla_{w} \mathcal{L}\left(w_{\mathrm{C}}^{j}, \lambda_{\mathrm{C}}^{j}, \mu_{\mathrm{C}}^{j} \right) + \left(C_{\mathrm{C}}^{j} \right)^{T} \lambda_{\mathrm{C}}^{j} + \left(D_{\mathrm{C}}^{j} \right)^{T} \mu_{\mathrm{C}}^{j}. \end{aligned} \tag{3.43}$$

To understand where Equation (3.43) originates, we revisit our derivation of the SQP method as NEWTON's method applied to the KKT system in Subsection 3.3.1. Let us replace the constraint Jacobian in the linear system (3.17), which determines the NEWTON step for the KKT optimality system (3.16), with an approximation M. In this case, the resulting linear system is no longer equivalent to the linear system (3.18) that constitutes the KKT system of the SQP QP (3.20). Instead, we need to replace $\nabla_x J\left(x^k\right)$ in (3.18) with $\nabla_x \mathcal{L}\left(x^k, \lambda^k\right) + M$. Replacing M with C_{C}^{j} and extending this concept to inequality-constrained problems leads us to the modified gradient Equation (3.43).

What we describe here in a condensed manner is an inexact SQP method. Inexact SQP methods can be interpreted as inexact NEWTON methods – or NEWTON-type methods depending on the naming convention you follow – applied to the KKT optimality systems (3.16) for equality-constrained problems.

The Level C variables are updated analogously to Level D, i.e.,

$$w_{\mathrm{C}}^{j+1} = w_{\mathrm{C}}^{j} + \Delta w, \qquad \lambda_{\mathrm{C}}^{j+1} = \lambda_{\mathrm{QP}}, \qquad \mu_{\mathrm{C}}^{j+1} = \mu_{\mathrm{QP}}.$$

Δw and λ_{QP}, μ_{QP} denote the solution obtained by solving QP (3.42) with its components provided by Level C.

At first glance, it seems that we need to compute the constraint Jacobians $\frac{\partial}{\partial w} c_{\mathrm{C}}^{j}$ and $\frac{\partial}{\partial w} d_{\mathrm{C}}^{j}$ to compute the modified gradient. However, they only appear as matrix-vector products with the LAGRANGE multipliers λ_{C}^{j} and μ_{C}^{j}. Fortunately, these matrix-vector products can be computed without explicitly calculating the full Jacobians by using the adjoint Internal Numerical Differentiation (IND), see [4], or the reverse mode of Automatic Differentiation (AD), see [95]. Consequently, the dominant computational expense in the preparation phase of Level C becomes the computation of the gradient $\nabla_x J\left(x^k\right)$ of the Lagrangian of NLP (3.14). To compute $\nabla_x J\left(x^k\right)$, we not only need to compute the aforementioned matrix-vector products but also the gradient of the objective function of NLP (3.14). Since the constraint Jacobians and Hessian are not recomputed in Level C, we can also reuse the condensed Hessian and condensed mixed state-control constraint matrices, which reduces the cost of condensing. Additionally, the online active set method mentioned on p. 43 is accelerated if the constraint matrices remain unchanged [200].

[4]: Albersmeyer (2010), "Adjoint based algorithms and numerical methods for sensitivity generation and optimization of large scale dynamic systems"

[95]: Griewank et al. (2008), *Evaluating Derivatives*

[200]: Wirsching (2018), "Multi-level iteration schemes with adaptive level choice for nonlinear model predictive control"

If we fix a current state x^j and then perform an SQP method where the QPs are defined by Level C QPs, we are guaranteed local Q-linear convergence of the iterates toward a KKT point, see [200, Theorem 5.3]. This fact explains the alternative name of Level C as optimality iterations.

[200]: Wirsching (2018), "Multi-level iteration schemes with adaptive level choice for nonlinear model predictive control"

Level B – Feasibility iterations Transitioning down to Level B, we aim to avoid the most expensive computations of Level C. Similar to Level C, Level B maintains its own primal-dual variables w_{B}^j, λ_{B}^j, μ_{B}^j and reference matrices $\bar{Q}_{\mathrm{B}}^j$, $\bar{C}_{\mathrm{B}}^j$, $\bar{D}_{\mathrm{B}}^j$, which are obtained from Level D or offline computations. For the same reasons as in Level C, Level B also uses a modified gradient p_{B}^j. Since the most expensive computation in Level C is the evaluation of the LAGRANGE gradient $\nabla_x J\left(x^k\right)$, which is required to construct the modified gradient p_{C}^j, Level B eliminates derivative computations entirely by approximating the modified gradient p_{B}^j using a first-order TAYLOR expansion

$$p_{\mathrm{B}}^j = \bar{p}_{\mathrm{B}}^j + \bar{Q}_{\mathrm{B}}^j\left(w_{\mathrm{B}}^j - \bar{w}_{\mathrm{B}}^j\right), \tag{3.44}$$

where a reference modified gradient $\bar{p}_{\mathrm{B}}^j$, reference variables $\bar{w}_{\mathrm{B}}^j$, and reference Hessian $\bar{Q}_{\mathrm{B}}^j$ are either communicated from Level D or precomputed offline.

Attention:
If Level C provides the reference gradient $\bar{p}_{\mathrm{B}}^j$, it is crucial to emphasize that $\bar{p}_{\mathrm{B}}^j = p_{\mathrm{C}}^j$ and not $\nabla_w \mathcal{L}\left(w_{\mathrm{C}}^j, \lambda_{\mathrm{C}}^j, \mu_{\mathrm{C}}^j\right)$, as $\bar{p}_{\mathrm{B}}^j$ must also be a modified gradient.

Accordingly, only the residuals c_{B}^j and d_{B}^j are newly evaluated at w_{B}^j in Level B. Again, the Level B variables are updated by

$$w_{\mathrm{B}}^{j+1} = w_{\mathrm{B}}^j + \Delta w, \qquad \lambda_{\mathrm{B}}^{j+1} = \lambda_{\mathrm{QP}}, \qquad \mu_{\mathrm{B}}^{j+1} = \mu_{\mathrm{QP}}.$$

Δw and $\lambda_{\mathrm{QP}}, \mu_{\mathrm{QP}}$ denote the solution obtained by solving QP (3.42) with its components provided by Level B.

For Level B, the integration of the IVPs (3.6) required for the constraints evaluations becomes the most expensive computation. As for Level C, we only need to redo parts of the condensing and parametric QP solvers benefit from the unchanged QP matrices. For SQP iterations with only Level B QPs, it has been shown that the iterates converge towards a feasible point for the NLP (3.14) that, however, is suboptimal. This earns Level B the alternative name of feasibility iterations. If at all, the limit point is a KKT point of a disturbed NLP as shown in [28]. See also [171, Theorem 5.2].

[28]: Bock et al. (2007), "Constrained Optimal Feedback Control of Systems Governed by Large Differential Algebraic Equations"
[171]: Scholz (2022), "Model-based optimal feedback control For microgrids"

Level A – Feedback iterations On the fastest Level A, we do not update any components of QP (3.42) except for the initial value constraint (3.41f). While Level A does not hold its own set of variables as indicated on p. 46, it thus holds reference values for the Hessian $\bar{Q}_{\mathrm{A}}^j$, the Jacobians $\bar{C}_{\mathrm{A}}^j, \bar{D}_{\mathrm{A}}^j$, the modified gradient $\bar{p}_{\mathrm{A}}^j$, the inequality residuals $\bar{d}_{\mathrm{A}}^j$ and the remaining components of the equality residuals. Moreover, it holds a reference value for the control variable q_0 which we denote by $\bar{q}_0$. As the resulting Level A QP is hence equal to the QP solved at the previous sampling time except for the initial value constraint, no condensing is required and a parametric QP solver has the least computations to do. Once we have obtained the solution

The residual of the initial value constraint (3.41f) is part of c^j.

Δw that includes Δq_0 from the QP solver, we only compute $\bar{q}_0 + \Delta q_0$ and apply the resulting control to the system. No further updates of optimization variables are computed. Using Level A comes down to applying a Linear Model Predictive Controller (LMPC) [200]. The price we pay for the short sampling period is that Level A iterates are not guaranteed to converge to a feasible point. We can also derive an explicit feedback law from Level A. For details see, e.g., [200, p. 69].

[200]: Wirsching (2018), "Multi-level iteration schemes with adaptive level choice for nonlinear model predictive control"

The computations and update formulas are summarized in Table 3.1.

Level	IVE $x^j - s_0$	Residuals $c(w),\ d(w)$	Gradient $p(w)$	Jacobians $C(w),\ D(w)$	Hessian $Q(w, \lambda, \mu)$
D	✓	✓	✓	✓	✓
C	✓	✓	✓	(✓)[1]	
B	✓	✓			
A	✓				

[1] Only the products $C(w)^T\lambda,\ D(w)^T\mu$ need to be computed in an adjoint fashion.

(a) Necessary computations.

Level	Residuals $c,\ d$	Gradient p	Jacobians $C,\ D$	Hessian Q
D	$c(w)$, $d(w)$	$p(w)$	$C(w)$, $D(w)$	$Q(w, \lambda, \mu)$
C	$c(w)$, $d(w)$	$\begin{pmatrix} b^s \\ b^q \end{pmatrix} + \left(\bar{C} - C(w)\right)^T \lambda + \left(\bar{D} - D(w)\right)^T \mu$	$\bar{C}$, $\bar{D}$	$\bar{Q}$
B	$c(w)$, $d(w)$	$\bar{p} + \bar{Q}(w - \bar{w})$	$\bar{C}$, $\bar{D}$	$\bar{Q}$
A	$\bar{c}$, $\bar{d}$	$\bar{p}$	$\bar{C}$, $\bar{D}$	$\bar{Q}$

(b) Update formulas.

Table 3.1.: Overview about the necessary computations and the respective update formulas for each MLI level. The farther to the right a computation is, the more expensive it is. Adapted from [171, Table 5.1].

3.5.2. Further aspects

In Subsection 3.5.1, we described how each MLI level operates. While we have already made some remarks about the communication between levels, we have yet to explain in more detail the interplay of the different levels. This interplay has two main aspects: the scheduling of the levels, i.e., determining when each level is applied, and the communication, i.e., the data exchange between the levels.

Scheduling of MLI levels For each sampling time t^j, we must decide which MLI level to apply. A straightforward option is to prescribe a schedule in advance and apply the levels according to this schedule. Such a schedule is typically written in the form $\mathrm{A}^{n_\mathrm{A}}\mathrm{B}^{n_\mathrm{B}}\mathrm{C}^{n_\mathrm{C}}\mathrm{D}^{n_\mathrm{D}}$ with $n_\mathrm{A}, n_\mathrm{B}, n_\mathrm{C}, n_\mathrm{D} \in \mathbb{N}$ which is interpreted as follows.

The scheduling periods $n_\mathrm{A}, n_\mathrm{B}, n_\mathrm{C}, n_\mathrm{D}$ must be chosen such that a level is scheduled for every sampling time. The simplest and most common way to ensure this is to set $n_\mathrm{X} = 1$ for the leftmost level X.

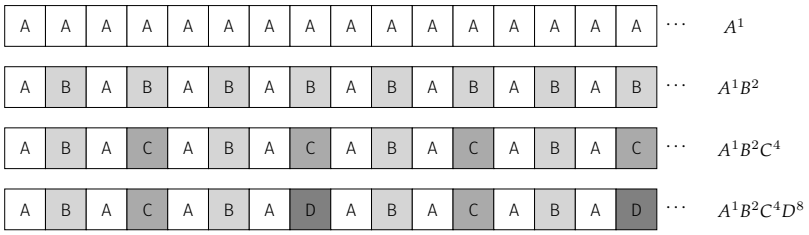


Figure 3.5.: Incremental construction of the MLI schedule $\mathrm{A}^1\mathrm{B}^2\mathrm{C}^4\mathrm{D}^8$ via the schedules A^1, $\mathrm{A}^1\mathrm{B}^2$, and $\mathrm{A}^1\mathrm{B}^2\mathrm{C}^4$. Adapted from [200, Figure 5.1].

We start with the leftmost level. Let this level be denoted as X. For now, we schedule this level X for every n_{X}-th sampling time. Then, we move one position to the right, where Level B is usually located in the general schedule. Let this level be denoted as Y. This level is then scheduled for every n_{Y}-th sampling time. If a sampling time is assigned to both level X and Y, the level scheduled later, i.e., level Y, takes precedence. We proceed analogously until all levels in the schedule have been processed. The resulting level choice at sampling time t^j is shown in Algorithm 3.5. An example of the incremental construction of the MLI schedule $\mathrm{A}^1\mathrm{B}^2\mathrm{C}^4\mathrm{D}^8$ is provided in Figure 3.5.

Algorithm 3.5: MLI level choice at t^j with a prescribed schedule.

Input: MLI schedule $\mathrm{X}_1^{n_{\mathrm{X}_1}} \cdots \mathrm{X}_N^{n_{\mathrm{X}_N}}$ with N levels and index j of current sampling time t^j

Output: Level Y to be applied around t^j

```
for i = N to 1 do
    if j mod n_{X_i} = 0 then
        Y ← X_i
        return
```

From an implementation perspective, Algorithm 3.5 is typically realized differently. Instead of looping over the levels, we create an array that holds the sequence in which the levels are applied and simply read its j-th entry to get Y. This array is constructed based on the description above.

In addition to this prescribed scheduling, adaptive level choice methods form a second group of scheduling strategies. In [200, Section 6.4], an adaptive level choice method based on contraction estimates is proposed. If the MLI scheme is executed on a multi-core CPU, the different levels can run in parallel. For this scenario, Scholz et. al. developed a level choice based on computation time in [172].

[200]: Wirsching (2018), "Multi-level iteration schemes with adaptive level choice for nonlinear model predictive control"
[172]: Scholz et al. (2020), "Model-based optimal feedback control for microgrids with multi-level iterations"

Communication between MLI levels The communication between MLI levels has two main aspects:

- the provision of the components of QP (3.42), and
- the exchange of optimization variables.

These two communication aspects differ in terms of the hierarchy of the levels involved and the "age" of the data being exchanged. For the communication of QP components, the hierarchy of the levels is typically top-down. In a top-down hierarchy, data is passed from higher levels to lower levels. This communication style matches the one described in [200, Algorithm 1]. This is because higher levels compute more components of the QP (3.42) than lower levels. For components updated in both communicating levels, such as the constraint residuals in Levels B, C, and D, it may also make sense to exchange these components in the opposite direction.

For the communication of optimization variables, four different hierarchies (and hybrid forms of these four) are reasonable. The most suitable hierarchy depends on the problem and the MLI schedule in use. The four communication forms are:

- Top-down variable communication: Optimization variables are passed only from higher levels to lower levels, cf. [200, Alg. 2].
- Bottom-up variable communication: Optimization variables are passed only from lower levels to higher levels, cf. [200, Alg. 3].
- Maximum variable communication: All levels exchange their optimization variables with all other levels, cf. [200, Alg. 4].
- Minimum variable communication: No optimization variables are exchanged between levels, cf. [200, Alg. 5].

For interpretations of these four forms and recommendations on when to use each, refer to [200, Section 5.3.2].

Let us clarify what we mean by the "age" of the data.The QP components are computed during the preparation phase of the levels and are evaluated at the variables w^j, λ^j, μ^j, i.e., as they are *before* the sampling time t^j. However, if the optimization variables themselves are exchanged between levels, it does not make sense to exchange them in their pre-sampling state. Instead, we first update the optimization variables to obtain w^{j+1}, λ^{j+1}, μ^{j+1} and then communicate these updated variables to the other levels. In this sense, the optimization variable data is "younger" than the QP component data, as it reflects the state *after* the sampling time t^j. Otherwise, redundant computations may occur, as illustrated in Example 3.3. The only exception is the reference control $\bar{q}_0$ obtained by Level A. This reference control must necessarily come from w^j, i.e., from before the sampling time. Otherwise, Level A may exhibit undesired behavior.

Level A is exempt from this as it operates without its own set of variables, as described in the previous subsection.

Attention:

In the description of MLI in [171], this difference in the data's age between the two communication channels is incorrectly ignored. There, equations (5.10) and (5.13), in combination with Algorithm 5.2 and Figure 5.2, suggest that optimization variables are also exchanged before being updated by the QP solution.

▪ **Example 3.3** Consider the case where Level D holds its set of primal optimization variables w_{D}^j during the preparation phase for sampling time t^j. In this phase, Level D evaluates the equality constraint residuals $c_{\mathrm{D}}^j = c\left(w_{\mathrm{D}}^j\right)$. Now, assume that Level C is scheduled for the next sampling time t^{j+1}. If we were to communicate w_{D}^j down to Level C instead of the updated w_{D}^{j+1}, the following redundancy would occur. During its preparation phase for t^{j+1}, Level C would also evaluate the constraints c. However, since $w_{\mathrm{C}}^{j+1} = w_{\mathrm{D}}^j$ due to this flawed communication scheme, we would have

$$c_{\mathrm{C}}^{j+1} = c\left(w_{\mathrm{C}}^{j+1}\right) = c\left(w_{\mathrm{D}}^j\right) = c_{\mathrm{D}}^j.$$

This example demonstrates that the optimization variables must be updated by the QP solution before being broadcast to other levels. ▪

In Figure 3.6, we illustrate the described communication. In Figure 3.6 top-down variable communication is employed.

Finally, deviations from the standard scheduling and communication schemes, such as mixed-level or fractional-level iterations, are also possible. For more details, refer to [200, Section 5.5]. A comprehensive study of different MLI schemes is provided in [107].

[200]: Wirsching (2018), "Multi-level iteration schemes with adaptive level choice for nonlinear model predictive control"
[107]: Haßkerl et al. (2016), "Study of the performance of the multi-level iteration scheme for dynamic online optimization for a fed-batch reactor example"

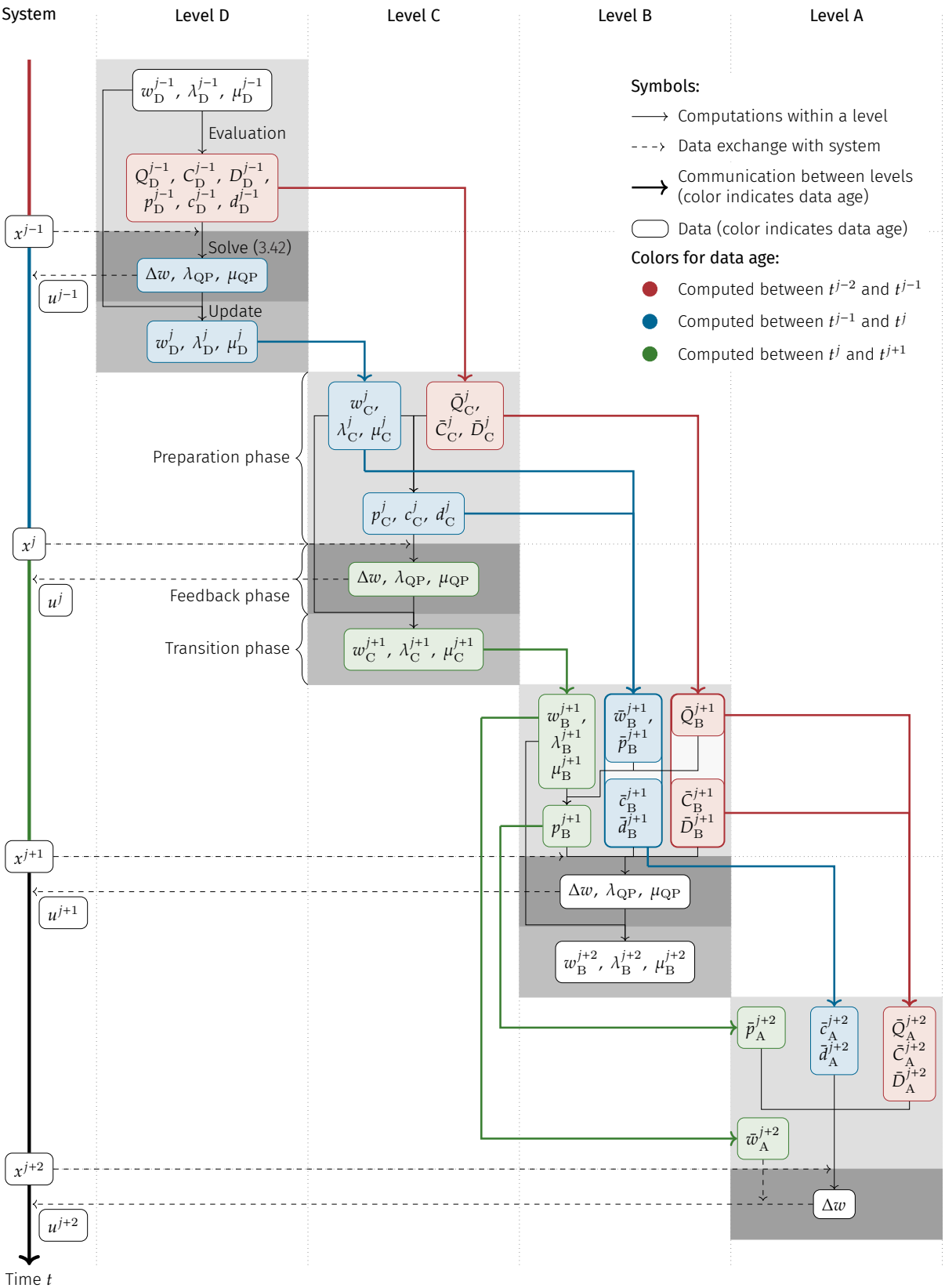


Figure 3.6.: Example of communication between the different MLI levels with top-down variable communication. Note that the sizes of the different phases do not represent their computational complexity.

Contributions

Stability of Inexact NMPC for a Class of Semilinear Parabolic PDEs

4.

While the main focus of this thesis is on NMPC based on an ODE model for the vehicle and engine dynamics, we also want to consider the internal dynamics of the vehicle, which may be modeled using PDEs, in future work. In particular, we want to investigate how the numerical methods for ODE-based NMPC presented and developed in this thesis can be transferred to PDE-based NMPC. From an engineering point of view, it is crucial to establish that the numerical methods are reliable. As a consequence, we have focused our attention first on establishing stability results. For the ODE-based NMPC, we have mentioned relevant results in Section 2.2.

For inexact NMPC schemes for PDEs, however, we are not aware of any existing work. In this sense, the results on the stability of inexact NMPC for a class of semilinear parabolic PDEs that we develop in this chapter are first steps in this direction. As such, we hope that our results will serve as a starting point for further results for other classes of PDEs and provide a justification for the development and use of corresponding numerical methods for real-world applications.

Our approach builds strongly on the work [208] that considers inexact NMPC for a class of ODEs. We begin with a description of the problem setting and the relevant assumptions in Section 4.1. Section 4.2 contains the proof of our main stability result, Theorem 4.3. The proof is structured into several steps.

[208]: Zanelli et al. (2021), "A Lyapunov function for the combined system-optimizer dynamics in inexact model predictive control"

4.1. Problem setting

We consider an NMPC scheme where we compute a new control at each node of an equidistant time grid $t_0, t_1, \ldots, t_j, \ldots$ with $t_j = (j-1)T$ for a $T > 0$. The prediction horizon at sampling time t_j is given by $[t_j, t_j + T_{\mathrm{hor}})$, where the horizon length is defined as $T_{\mathrm{hor}} := NT$ for an $N \in \mathbb{N}$. The NMPC scheme of interest is an inexact one, i.e., the control applied is not necessarily equal to the solution of the OCPs.

Reminder: NMPC

In a nutshell, we use a nonlinear model to predict and optimize the behavior of a dynamical system over a prediction horizon in NMPC. In particular, we recompute the control at each sampling time and apply it only until the next sampling time. See Chapter 2 or [101, Chapter 1] for details.

In the following, we first specify the class of PDEs that is used to describe the system dynamics. We then formulate the OCP arising in the NMPC setting of interest. In this chapter, we study the interplay between the system dynamics and the optimizer dynamics. By optimizer dynamics, we mean how the control applied to the system is modified by the optimization method, or optimizer for short, based on the current state and the previous control. We describe the optimizer dynamics and the combined system-optimizer dynamics in more detail in Subsection 4.1.3. Finally, we give some comments on the assumptions made in this section.

4.1.1. System dynamics

Definition: LIPSCHITZ domain

In simple terms, a *LIPSCHITZ domain* is a domain with a boundary that can locally be seen as the graph of a LIPSCHITZ continuous function. For a formal definition, see e.g. [7, A8.2, p. 259].

Definition: $C(\Omega)$ and $L^\infty(D)$

For a domain $D \subset \mathbb{R}^d$, the *space of continuous functions* is denoted by $C(D)$ and the space of *essentially bounded measurable functions* by $L^\infty(D)$. For details, see standard textbooks on the subject, e.g. [1, Paragraphs 1.26, 2.10].

The conditions stated here correspond mainly to the assumptions 5.1, 5.4 and 5.6 from [184].

Monotonically increasing in y means $\frac{\partial}{\partial y} f(x,y), \frac{\partial}{\partial y} g(x,y) \geq 0$.

$\frac{\partial^0}{\partial y^0}$ is the original function.

[184]: Tröltzsch (2009), *Optimale Steuerung partieller Differentialgleichungen: Theorie, Verfahren und Anwendungen*

[44]: Casas (1997), "Pontryagin's Principle for State-Constrained Boundary Control Problems of Semilinear Parabolic Equations"

[160]: Raymond et al. (1999), "Hamiltonian Pontryagin's Principles for Control Problems Governed by Semilinear Parabolic Equations"

Definition: L^p and $W(0,T)$

The *LEBESGUE space* with $1 \leq p < \infty$ is denoted by L^p. For details, see standard textbooks on the subject, e.g. [1, Paragraph 2.1]. The space $W(0,T)$ is a *HILBERT space* given as

$$W(0,T) := \left\{ y \in L^2\big(0,T,H^1(\Omega)\big) \;\middle|\; \frac{\partial}{\partial t} y \in L^2\big(0,T,H^1(\Omega)^*\big) \right\}$$

see, e.g. [184, p. 118], with the SOBOLEV space $H^1(\Omega)$ and its dual space $H^1(\Omega)^*$. For a definition of a *weak solution in* $W(0,T) \cap L^\infty(Q)$, see [184, p. 212].

Let $\Omega \subset \mathbb{R}^d$ be a bounded LIPSCHITZ domain. With $Q_{\mathrm{hor}} := \Omega \times (0, T_{\mathrm{hor}})$ we denote the space-time cylinder and with $\Sigma_{\mathrm{hor}} := \partial\Omega \times (0, T_{\mathrm{hor}})$ its lateral surface. We consider semilinear parabolic PDEs of the form

$$\begin{aligned} \dot{y} - \Delta y + f(x,y) &= 0 \quad \text{in } Q_{\mathrm{hor}}, \\ \partial_\nu y + g(x,y) &= u \quad \text{in } \Sigma_{\mathrm{hor}}, \\ y(0) &= y_0 \quad \text{in } \Omega, \end{aligned} \tag{4.1}$$

where $\dot{y} := \frac{\partial}{\partial t} y$, and $\partial_\nu y$ is the normal derivative, $y_0 \in C(\bar{\Omega})$ and

$$u \in U_{\mathrm{ad}} = \{ u \in L^\infty(\Sigma_{\mathrm{hor}}) \mid u_{\mathrm{l}}(x) \leq u(x,t) \leq u_{\mathrm{u}}(x) \text{ a.e. in } \Sigma_{\mathrm{hor}} \}, \tag{4.2}$$

and where f, g, u_{l} and u_{u} satisfy the following conditions: The functions $f : \Omega \times \mathbb{R} \to \mathbb{R}$ and $g : \partial\Omega \times \mathbb{R} \to \mathbb{R}$

(i) are measurable on Ω and $\partial\Omega$, respectively, for fixed $y \in \mathbb{R}$,
(ii) are twice differentiable with respect to y for almost all $x \in \Omega$ and $x \in \partial\Omega$, respectively,
(iii) are monotonically increasing in y for almost all $x \in \Omega$, $x \in \partial\Omega$, respectively,
(iv) satisfy $\left|\frac{\partial^l}{\partial y^l} f(x,0)\right| \leq M_1$ and $\left|\frac{\partial^l}{\partial y^l} g(x,0)\right| \leq M_2$ with constants $M_1, M_2 > 0$ for all $l = 0,1,2$ and for almost all $x \in \Omega$ and $x \in \partial\Omega$,

and their second derivative with respect to y

(v) is locally LIPSCHITZ-continuous in y for all $x \in \Omega$ and $x \in \partial\Omega$, respectively.

Moreover, we assume that

(vi) $u_{\mathrm{l}}, u_{\mathrm{u}} \in L^\infty(\partial\Omega)$ and that they satisfy $u_{\mathrm{l}}(x) \leq u_{\mathrm{u}}(x)$ for almost all $x \in \partial\Omega$.

Under these assumptions, the following Theorem 4.1 holds. It is taken from [184, Satz 5.5, p. 213] and summarizes results from [44, 160].

Theorem 4.1 — Existence of continuous weak solution of IVP (4.1).
Requirements: conditions (i), (iii) and (v)
The semilinear parabolic IVP (4.1) has for all $u \in L^s(\Sigma_{\mathrm{hor}})$, $y_0 \in C(\bar{\Omega})$, $s > d+1$ a unique weak solution $y \in W(0,T_{\mathrm{hor}}) \cap L^\infty(Q_{\mathrm{hor}})$. Moreover, y is continuous on $\overline{Q_{\mathrm{hor}}}$.

The IVP (4.1) describes the evolution of the state y with initial state y_0 and control u over the entire prediction horizon. For NMPC, however, the evolution from one sampling time t_j to the next sampling time t_{j+1} plays an important role, too. Let y_j be the current state at the sampling point t_j and u_j the control that is applied for the time $[t_j, t_{j+1})$. The respective IVP is obtained by replacing Q_{hor} by $Q := \Omega \times (0,T)$

and Σ_{hor} by $\Sigma := \partial\Omega \times (0,T)$ and y_0 by y_j and u by u_j, i.e. the IVP has the form

$$\begin{aligned} \dot{y} - \Delta y + f(x,y) &= 0 \quad \text{in } Q, \\ \partial_\nu y + g(x,y) &= u_j \quad \text{in } \Sigma, \\ y(0) &= y_j \quad \text{in } \Omega. \end{aligned} \tag{4.3}$$

We don't need to shift the time arguments by t_j as we chose all involved functions to not have an explicit time dependency.

Since we have only changed the time horizon, we still know from [184, Satz 5.5, p. 213] that there is a solution to IVP (4.3). For completeness, we formulate the following corollary.

Corollary 4.1 — Existence of continuous weak solution of IVP (4.3).
Requirements: conditions (i), (iii) and (v)
The semilinear parabolic IVP (4.3) has for all $u_j \in L^s(\Sigma)$, $y_j \in C(\bar{\Omega})$, $s > d+1$ a unique weak solution $y \in W(0,T) \cap L^\infty(Q)$. Moreover, y is continuous on $\bar{Q}$.

We denote this weak solution of the IVP (4.3) by $y(x,t;y_j,u_j) \in \mathbb{R}$. The dependencies highlight that y is evaluated at $(x,t) \in Q$ and that the initial value problem is equipped with the initial state $y_j \in C(\bar{\Omega})$ and the control $u_j \in L^s(\Sigma)$. We abbreviate

$$\begin{aligned} y(T;y_j,u_j) &:= y(\cdot,T;y_j,u_j) \in C(\bar{\Omega}), \\ y(y_j,u_j) &:= y(\cdot,\cdot;y_j,u_j) \in C(\bar{Q}). \end{aligned}$$

We omit the space and time dependencies of the control u for brevity.

Corollary 4.1 guarantees us that the system state remains continuous throughout the NMPC procedure, i.e. $y_j \in C(\bar{\Omega})$ for all $j \in \mathbb{N}$, if the system starts with an initial state $y_0 \in C(\bar{\Omega})$ and if we apply controls $u_j \in L^s(\Sigma)$ on all intervals $[t_j, t_{j+1})$.

Moreover, we can establish the following inequality by extending [184, Satz 5.8, p. 217]. Inequality (4.4) is important for us because it bounds the deviation in the state y caused by choosing different controls u.

Lemma 4.1 — Lipschitz continuity of y w.r.t. u.
Requirements: conditions (i) - (vi)
The inequality

$$\left\| y(T;y_0,u_1) - y(T;y_0,u_2) \right\|_{L^p(\Omega)} \leq L \|u_1 - u_2\|_{L^s(\Sigma)} \tag{4.4}$$

is satisfied for all $u_1, u_2 \in L^s(\Sigma), s > d+1$ and all $1 \leq p \leq \infty$ with a constant $L > 0$ that is independent of u_1 and u_2.

Proof. From [184, Satz 5.8, p. 217] we get the existence of $\widetilde{L} > 0$ such that

$$\begin{aligned}&\left\|y(y_0,u_1)-y(y_0,u_2)\right\|_{W(0,T)}+\left\|y(y_0,u_1)-y(y_0,u_2)\right\|_{C(\bar{Q})}\\&\quad\le \widetilde{L}\|u_1-u_2\|_{L^s(\Sigma)}.\end{aligned}$$

We exploit embedding results for LEBESGUE spaces and the definition of the space $L^\infty(\Omega)$, see e.g. [1, Chapter 2], to estimate

[1]: Adams et al. (2003), *Sobolev spaces*

$$\begin{aligned}&\left\|y(T;y_0,u_1)-y(T;y_0,u_2)\right\|_{L^p(\Omega)}\\&\quad\le \operatorname{vol}(\Omega)^{\frac{1}{p}}\left\|y(T;y_0,u_1)-y(T;y_0,u_2)\right\|_{L^\infty(\Omega)}\\&\quad= \operatorname{vol}(\Omega)^{\frac{1}{p}}\operatorname*{ess\,sup}_{x\in\Omega}\left|y(T;y_0,u_1)-y(T;y_0,u_2)\right|\\&\quad\le \operatorname{vol}(\Omega)^{\frac{1}{p}}\sup_{x\in\Omega}\left|y(T;y_0,u_1)-y(T;y_0,u_2)\right|\\&\quad\le \operatorname{vol}(\Omega)^{\frac{1}{p}}\max_{x\in\bar{\Omega}}\left|y(T;y_0,u_1)-y(T;y_0,u_2)\right|\\&\quad\le \operatorname{vol}(\Omega)^{\frac{1}{p}}\max_{(x,t)\in\bar{Q}}\left|y(y_0,u_1)-y(y_0,u_2)\right|\\&\quad= \operatorname{vol}(\Omega)^{\frac{1}{p}}\left\|y(y_0,u_1)-y(y_0,u_2)\right\|_{C(\bar{Q})}\\&\quad\le \operatorname{vol}(\Omega)^{\frac{1}{p}}\Big(\left\|y(y_0,u_1)-y(y_0,u_2)\right\|_{C(\bar{Q})}\\&\qquad+\left\|y(y_0,u_1)-y(y_0,u_2)\right\|_{W(0,T)}\Big).\end{aligned}$$

The assertion follows with $L := \operatorname{vol}(\Omega)^{\frac{1}{p}}\widetilde{L}$. ■

Finally, we assume that

(vii) f and g are such that $y(\cdot;0,0) = 0$.

We will formulate in Assumption 4.2 that the optimal control to the initial state $y_0 = 0$ will be $u^* = 0$.

We conclude this section with an example of a semilinear parabolic PDE that satisfies the assumptions made so far.

▪ **Example 4.1** The semilinear parabolic initial value problem

$$\begin{aligned}\dot{y}-\Delta y+y^3+y&=0 \quad \text{in } Q,\\ \partial_\nu y&=u \quad \text{in } \Sigma,\\ y(0)&=y_0 \quad \text{in } \Omega,\end{aligned}$$

satisfies the assumptions made in the present subsection. As domain we can, for example, use $\Omega := (0,1)\times(0,1) \subset \mathbb{R}^2$. As bounds we can choose $u_{\mathrm{l}}(x,t) = 0$ and $u_{\mathrm{u}}(x,t) = 1$. ▪

4.1.2. OCP formulation

We consider the case, where the arising OCP at sampling time point t_j with current state $y_j \in C(\bar{\Omega})$ has the form

$$\begin{aligned}
\min \quad & \int_\Omega \phi\big(x, y(x, T_{\text{hor}})\big)\mathrm{d}x + \iint_{Q_{\text{hor}}} \varphi\big(x, y(x,t)\big)\mathrm{d}x\mathrm{d}t \\
& + \iint_{\Sigma_{\text{hor}}} \Psi\big(x, y(x,t), u(x,t)\big)\mathrm{d}x\mathrm{d}t \\
\text{over} \quad & y \in C(\bar{Q}_{\text{hor}}),\ u \in L^s(\Sigma_{\text{hor}}), \\
\text{s.t.} \quad & \dot{y} - \Delta y + f(x,y) = 0 \quad \text{in } Q_{\text{hor}}, \\
& \partial_\nu y + g(x,y) = u \quad \text{in } \Sigma_{\text{hor}}, \\
& y(0) = y_j \quad \text{in } \Omega, \\
& u_1(x) \le u(x,t) \le u_{\text{u}}(x) \text{ a.e. in } \Sigma_{\text{hor}}
\end{aligned} \tag{4.5}$$

where the functions ϕ, φ, Ψ satisfy the following conditions. The functions $\phi : \Omega \times \mathbb{R} \to \mathbb{R}$, $\varphi : \Omega \times \mathbb{R} \to \mathbb{R}$ and $\Psi : \partial\Omega \times \mathbb{R}^2 \to \mathbb{R}$

The functions f, g, u_1, and u_{u} are as stated in Subsection 4.1.1.

(i) are measurable on Ω and $\partial\Omega$, respectively, for fixed $(y,u) \in \mathbb{R}^2$
(ii) are twice differentiable with respect to y and u for almost all $x \in \Omega$ and $x \in \partial\Omega$, respectively,
(iii) satisfy

$$\begin{aligned}
\left|\frac{\partial^l}{\partial y^l}\phi(x,0)\right| &\le M_3, \\
\left|\frac{\partial^l}{\partial y^l}\varphi(x,t,0)\right| &\le M_4, \\
\left|\frac{\partial^l}{\partial (y,u)^l}\Psi(x,t,0,0)\right| &\le M_5
\end{aligned}$$

with constants $M_3, M_4, M_5 > 0$ for all $l = 0, 1, 2$ and for almost all $x \in \Omega$ and $x \in \partial\Omega$,

$\frac{\partial^0}{\partial y^0}$ is the original function.

and their second derivatives with respect to y and u

(iv) are locally LIPSCHITZ-continuous in y and u for all $x \in \Omega$ and $x \in \partial\Omega$, respectively.

Existence of optimal solutions of the OCP (4.5) is proved by the following Theorem 4.2, which is taken from [184, Satz 5.7, p.215].

[184]: Tröltzsch (2009), *Optimale Steuerung partieller Differentialgleichungen: Theorie, Verfahren und Anwendungen*

Theorem 4.2 — Existence of optimal solution of OCP (4.5).
Requirements: conditions (i) - (vi) from Subsection 4.1.1
The OCP (4.5) has at least one optimal solution $\bar{u}(y_j)$ with associated optimal state $\bar{y}$ if the functions Ψ and φ are convex with respect to y and u.

By virtue of Theorem 4.2, we can define the mapping that maps the current state y_j to optimal control $\bar{u}$ restricted to the next sampling interval. Formally, u^* is defined as

$$u^* : C(\bar{\Omega}) \to \Sigma_{\text{hor}},\ y_j \mapsto u^*(y_j) := \bar{u}(y_j)|_{t\in[0,T)}.$$

In other words, u^* is the state-to-feedback map.

Reminder: Nominal NMPC

In nominal NMPC we always apply $u^*\left(y_j\right)$. See Chapter 2 for details.

We assume that the nominal NMPC satisfies Assumption 4.1 which will play an important role throughout this chapter. To formulate Assumption 4.1, we introduce the following notation.

For $r_y, r_u > 0, y_j \in L^p(\Omega)$ and $u_j \in L^s(\Sigma)$ we abbreviate

$$\mathcal{B}_{\Omega,p}\left(y_j, r_y\right) := \left\{\tilde{y} \in L^p(\Omega) \;\middle|\; \left\|\tilde{y} - y_j\right\|_{L^p(\Omega)} \leq r_y\right\},$$
$$\mathcal{B}_{\Sigma,s}\left(u_j, r_u\right) := \left\{\tilde{u} \in L^s(\Sigma) \;\middle|\; \left\|\tilde{u} - u_j\right\|_{L^s(\Sigma)} \leq r_u\right\}.$$

Moreover, for sets A, B, we use the notation $A \oplus B$ to denote the MINKOWSKI sum of A and B defined as $A \oplus B := \{a + b \mid a \in A, b \in B\}$.

V is basically a LYAPUNOV function with additional regularity for the nominal NMPC and $Y_{\bar{V}}$ is a level set of it.

Assumption 4.1 Let $V : C(\bar{\Omega}) \to \mathbb{R}$ be a continuous function and $\bar{V} > 0$ be a constant. Let $Y_{\bar{V}}$ be defined as

$$Y_{\bar{V}} := \left\{y \in C(\bar{\Omega}) \;\middle|\; V(y) \leq \bar{V}\right\}.$$

There exist constants $a_1, a_2, a_2, T_V > 0$ and $q \in \mathbb{N}$ and $1 \leq p \leq \infty$ such that for $T \leq T_V$ and for any $y_j \in Y_{\bar{V}}$ the conditions

$$a_1\left\|y_j\right\|^q_{L^p(\Omega)} \leq V\left(y_j\right) \leq a_2\left\|y_j\right\|^q_{L^p(\Omega)}, \tag{4.6}$$
$$V\left(y\left(T; y_j, u^*(y_j)\right)\right) \leq V\left(y_j\right) - T a_3\left\|y_j\right\|^q_{L^p(\Omega)} \tag{4.7}$$

hold. Moreover, there exist constants $r_y, \mu > 0$ such that for all $y^1, y^2 \in Y_{\bar{V}} \oplus \mathcal{B}_{\Omega,p}\left(0, r_y\right)$ the condition

$$\left|V\left(y^1\right)^{\frac{1}{q}} - V\left(y^2\right)^{\frac{1}{q}}\right| \leq \mu\left\|y^1 - y^2\right\|_{L^p(\Omega)} \tag{4.8}$$

holds.

Furthermore, we assume that u^* satisfies the following assumption.

r_u is determined by Assumption 4.3.

Assumption 4.2 Let r_y be as in Assumption 4.1 and $r_u > 0$ fixed. There exist positive constants $\alpha \in (0,1)$ and $T_\alpha, \gamma > 0$ such that for all $y_j \in Y_{\bar{V}}$, $0 \leq T \leq T_\alpha$ the condition

$$\left\|u^*\left(y\left(T; y_j, u^*\left(y_j\right)\right) - u^*\left(y_j\right)\right)\right\|_{L^s(\Sigma)} \leq \alpha r_u \tag{4.9}$$

holds. Moreover,

$$\left\|u^*\left(y^1\right) - u^*\left(y^2\right)\right\|_{L^s(\Sigma)} \leq \gamma\left\|y^1 - y^2\right\|_{L^p(\Omega)} \tag{4.10}$$

holds for all $y^1 \in Y_{\bar{V}}$ and $y^2 \in \mathcal{B}_{\Omega,p}\left(y^1, r_y\right)$. Finally, we define

$$y^* := 0, \tag{4.11}$$

and assume

$$u^*\left(y^*\right) = 0.$$

Attention:

The condition $y^* \in Y_{\bar{V}}$ is not guaranteed, and in our proof in Section 4.2 we need to pay close attention that it is satisfied when we wish to utilize Equation (4.12).

Assumption 4.2 implies for $y^* \in Y_{\bar{V}}$ that

$$\left\|u^*(y)\right\|_{L^s(\Sigma)} \leq \gamma\left\|y - y^*\right\|_{L^p(\Omega)} \overset{(4.11)}{=} \gamma\left\|y\right\|_{L^p(\Omega)} \quad \forall y \in \mathcal{B}_{\Omega,p}\left(0, r_y\right). \tag{4.12}$$

In combination with Example 4.1, we think of the following example.

▪ **Example 4.2** Let $\omega > 0$. At time t_j with current state $y_j \in C(\bar{\Omega})$ we solve the OCP given by

$$\begin{aligned} \min \ & \frac{1}{2} \iint_{Q_{\mathrm{hor}}} |y(x,t)|^2 \mathrm{d}x\mathrm{d}t + \frac{\omega}{2} \iint_{\Sigma_{\mathrm{hor}}} |u(x,t)|^2 \mathrm{d}x\mathrm{d}t \\ \text{over} \quad & y \in C(\bar{Q}_{\mathrm{hor}}),\ u \in L^s(\Sigma_{\mathrm{hor}}), \\ \text{s.t.} \quad & \dot{y} - \Delta y + y^3 + y = 0 \quad \text{in } Q_{\mathrm{hor}}, \\ & \partial_\nu y = u \quad \text{in } \Sigma_{\mathrm{hor}}, \\ & y(0) = y_j \quad \text{in } \Omega, \\ & 0 \le u(x,t) \le 1 \text{ a.e. in } \Sigma_{\mathrm{hor}}. \end{aligned} \tag{4.13}$$

▪

Attention:

So far, we have not been able to verify that Example 4.2 satisfies Assumptions 4.1 and 4.2. We will discuss this issue, among others, in Subsection 4.1.4.

4.1.3. System-optimizer dynamics

We are interested in the case of inexact NMPC. This means that we consider the case where a control u_j, which may differ from $u^*(y_j)$, is applied during the interval $[t_j, t_{j+1})$. In our setting the control u_j is computed by an optimization method as an (approximate) solution of the OCP (4.5). The current state y_j and the previous control u_{j-1} serve as input to the optimization method. In the case $j = 0$, the previous control u_{j-1} is replaced by an initial guess $u_{\mathrm{init}} \in L^s(\Sigma)$ for the control. To avoid case distinctions, we define $u_{-1} := u_{\mathrm{init}}$. In this sense, the optimizer defines a map $\xi : L^p(\Omega) \times L^s(\Sigma) \to L^s(\Sigma)$, which we call the optimizer dynamics, which determines

$$u_j := \xi(y_j, u_{j-1}) \quad \text{for all } j \in \mathbb{N}_0. \tag{4.14}$$

Definition: Optimizer

We will refer to this optimization method simply as the *optimizer*.

For the optimizer, we make the assumption that it exhibits Q-linear convergence if the initial guess u_{j-1} is sufficiently close to the optimal control $u^*(y_j)$. We formalize this assumption as

Assumption 4.3 Let r_y be as in Assumption 4.1. There exist positive constants $r_u > 0$ and $0 < \kappa < 1$ such that the optimizer computes u_j in $k \in \mathbb{N}$ iterations such that

$$\|u_j - u^*(y_j)\|_{L^s(\Sigma)} \le \kappa^k \|u_{j-1} - u^*(y_j)\|_{L^s(\Sigma)} \tag{4.15}$$

for all $y_j \in Y_{\bar{V}} \oplus \mathcal{B}_{\Omega,p}(0, r_y)$ and $u_{j-1} \in \mathcal{B}_{r_u,s}(u^*(y_j), r_u)$.

The interplay between the system dynamics (4.3) and the optimizer dynamics (4.14) is summarized in the system-optimizer dynamics which is given as

$$\begin{aligned} u_j &= \xi(y_j, u_{j-1}), \quad \text{for all } j \in \mathbb{N}_0, \\ y_j &= y(T; y_{j-1}, u_{j-1}), \quad \text{for all } j \in \mathbb{N}, \end{aligned} \tag{4.16}$$

with given $y_0 \in C(\bar{\Omega})$ and $u_{-1} = u_{\mathrm{init}} \in \mathcal{B}_{r_u,s}(u^*(y_0), r_u)$.

We could shift the index j in order to have $j \in \mathbb{N}_0$ in both parts of the dynamics. However, such a shift makes the notation less intuitive for the other parts of this work.

4.1.4. Discussion of the assumptions

Discussion of Assumption 4.1 In the finite-dimensional case and with a fixed T, Assumption 4.1 implies that the origin is exponentially stable for the nominal NMPC, cf. e.g. [159, Theorem 2.21]. In the present infinite-dimensional case however, this assumption requires further discussion. Equation (4.8) implies that the DINI derivative V'_+ of V along the optimal trajectory satisfies

[159]: Rawlings et al. (2022), *Model predictive control: Theory, computation, and design*

$$V'_+(y_j) = \limsup_{T \searrow 0} \frac{1}{T}\left(V\left(y\left(T; y_j, u^*(y_j)\right)\right) - V(y_j)\right) \overset{(4.7)}{\leq} -a_3 \|y_j\|^q_{L^p(\Omega)}.$$

Therefore, Assumption 4.1 implies that V is a LYAPUNOV function for the nominal NMPC, which is equivalent to asymptotic stability at zero on $Y_{\bar{V}}$ [140, Definition 11, Proposition 14]. As discussed in [208, Remark 4],
Equation (4.8) is satisfied if V is LIPSCHITZ continuous over $Y_{\bar{V}}$ and if $V^{\frac{1}{q}}$ is LIPSCHITZ continuous at $y = 0$. An example where this is the case is as follows. If $q = 2$, it suffices that V is twice continuously differentiable at $y = 0$.

[140]: Mironchenko et al. (2018), "Characterizations of Input-to-State Stability for Infinite-Dimensional Systems"

[208]: Zanelli et al. (2021), "A Lyapunov function for the combined system-optimizer dynamics in inexact model predictive control"

Discussion of Assumption 4.2 For our stability proof, it is essential that we can estimate

(i) how changes in the optimizer state affect the system state,
(ii) vice versa, how changes in the system state, more precisely the current state, affect the optimizer state.

While Lemma 4.1 gives us an estimate for (i), research on estimates for (ii) is still in its beginning. In particular, the only work we are aware of that addresses the question of stability with respect to initial data is [45]. But, the authors also state in [45]:

[45]: Casas et al. (2022), "Stability for Semilinear Parabolic Optimal Control Problems with Respect to Initial Data"

> *"We do not know associated works, where perturbations of the initial data were addressed in the context of PDE control."*
>
> — CASAS and TRÖLTZSCH [45]

While Theorem 3.4 in [45] already resembles Equation (4.10) of our Assumption 4.2, we need the stronger formulation Equation (4.10) for our proof. Unfortunately, checking Assumption 4.2 is difficult in general. In particular, we have not yet been able to verify Assumption 4.2 for our guiding Example 4.2. On the one hand, the results in [45] give us hope that Assumption 4.2 is satisfied by some problem classes.

Equation (4.9) also requires some discussion. In the finite-dimensional case, Assumption 9 from [208] can be used where we have to use Equation (4.9). At first sight it seems that also in the present infinite-dimensional case Equation (4.9) could be derived from Equation (4.10). To do this, however, we need estimates for $\left\|y\left(T; y_j, u^*(y_j)\right) - y_j\right\|_{L^p(\Omega)}$ for $T \to 0$. But, the existing theory for semilinear parabolic PDEs does not provide suitable estimates.

[208]: Zanelli et al. (2021), "A Lyapunov function for the combined system-optimizer dynamics in inexact model predictive control"

Informally, Equation (4.9) means that u^* does not exhibit jumps larger than r_u along the optimal trajectory starting from y_j. We can expect this behaviour if the system dynamics and the optimal control are sufficiently smooth. But, Equation (4.9) prevents us from considering control systems with bang-bang solutions. Then again, this is also true for Assumption 9 in [208].

Discussion of Assumption 4.3 Assumption 4.3 is simply the definition of Q-linear convergence. The availability of optimization methods that have a Q-linear convergence rate is not an issue. For examples of optimization methods with sometimes even stronger convergence rates, see e.g. [189].

[189]: Ulbrich (2009), "Optimization Methods in Banach Spaces"

Finally, we remark that we are not restricted to IVPs of the form (4.3), nor to OCPs of the form (4.5). We could consider any other PDE or OCP formulation, as long as all the assumptions made in this section are satisfied.

This includes that Lemma 4.1 can be proved.

4.2. Stability proof

The goal of this section is to show asymptotic stability of the origin $(y^*, u^*) = (0, 0)$ for the system-optimizer dynamics (4.16). Our proof consists of five main steps. To leverage our assumptions, we must ensure that y_{j+1} and u_j remain in $Y_{\bar{V}}$ and in $\mathcal{B}_{\Sigma,s}(u^*(y_{j+1}), r_u)$, respectively, if $y_j \in Y_{\bar{V}}$ and $u_{j-1} \in \mathcal{B}_{\Sigma,s}(u^*(y_j), r_u)$. For this purpose, in step 1 (Subsection 4.2.1) we establish the auxiliary results Lemma 4.3 and Corollary 4.2. In step 2 (Subsection 4.2.2) we use these two results to prove in Lemmas 4.4 and 4.5 that $u_j \in \mathcal{B}_{\Sigma,s}(u^*(y_{j+1}), r_u)$ and $y_{j+1} \in Y_{\bar{V}}$ if $y_j \in Y_{\bar{V}}$ and $u_{j-1} \in \mathcal{B}_{\Sigma,s}(u^*(y_j), r_u)$. With Lemmas 4.4 and 4.5 established, we conclude in the form of Lemma 4.6 that
$u_j \in \mathcal{B}_{\Sigma,s}(u^*(y_{j+1}), r_u)$ and $y_j \in Y_{\bar{V}}$ for all $j \in \mathbb{N}_0$ if $y_0 \in Y_{\bar{V}}$ and $u_{\text{init}} \in \mathcal{B}_{\Sigma,s}(u^*(y_0), r_u)$. In step 3 (Subsection 4.2.3), we derive estimates for the evolution of

$$E_j := \left\| u_j - u^*(y_j) \right\|_{L^s(\Sigma)}, \quad \text{and} \quad V_j := V(y_j)^{\frac{1}{q}}$$

in Lemmas 4.7 and 4.8 which require that $u_{j-1} \in \mathcal{B}_{\Sigma,s}(u^*(y_j), r_u)$ and $y_j \in Y_{\bar{V}}$. Therefore, it is imperative to establish Lemma 4.6 first. Lemmas 4.7 and 4.8 leave us with a positive linear system of the form

$$\begin{pmatrix} V_{j+1} \\ E_{j+1} \end{pmatrix} \leq A \begin{pmatrix} V_j \\ E_j \end{pmatrix} \quad \text{with} \quad A \in \mathbb{R}^{2\times 2}_{\geq 0}. \tag{4.17}$$

As $V_j \geq 0$ due to Assumption 4.1 and Lemma 4.6 and as $E_j \geq 0$ by definition and $A \in \mathbb{R}^{2\times 2}_{\geq 0}$ for all $j \in \mathbb{N}_0$, we have that

$$0 \leq \begin{pmatrix} V_j \\ E_j \end{pmatrix} \leq \begin{pmatrix} V_j^{\mathrm{u}} \\ E_j^{\mathrm{u}} \end{pmatrix},$$

Reminder: Asymptotic stability

In simplified terms, a point x^* is defined as asymptotically stable if the distance of the system state to x^* becomes arbitrarily small as the system evolves. To establish asymptotic stability, we primarily need to find a LYAPUNOV function. See Definition 2.11 on p. 20 for details.

where V_j^{u} and E_j^{u} are given by

We will encounter this equation later again as Equation (4.25).

$$\begin{pmatrix} V_{j+1}^{\mathrm{u}} \\ E_{j-1}^{\mathrm{u}} \end{pmatrix} = A \begin{pmatrix} V_j^{\mathrm{u}} \\ E_j^{\mathrm{u}} \end{pmatrix} \quad \text{with} \quad \begin{pmatrix} V_0^{\mathrm{u}} \\ E_0^{\mathrm{u}} \end{pmatrix} \coloneqq \begin{pmatrix} V_0 \\ E_0 \end{pmatrix}.$$

In step 4 (Subsection 4.2.4), we establish asymptotic stability of the positive linear system (4.25) in Lemma 4.9. Finally, in step 5 (Subsection 4.2.5), we prove asymptotic stability of the system-optimizer dynamics (4.16) in Theorem 4.3.

From now on we assume that

$$0 < T \leq \left\{ T_V, T_\alpha, \frac{a_2}{a_3} \right\}. \tag{4.18}$$

4.2.1. Step 1: auxiliary results

First, we establish with Lemma 4.2 that the state stays in the level set $Y_{\bar{V}}$ if we apply the optimal feedback u^*.

Lemma 4.2 — Optimal state remains in $Y_{\bar{V}}$.
Requirements: Assumption 4.1
We have with $\bar{a} \coloneqq \frac{a_3}{a_2}$ that

$$V\big(y\big(T; y_j, u^*(y_j)\big)\big) \overset{(4.6)}{\leq} (1 - T\bar{a}) V\big(y_j\big) \leq \bar{V}, \tag{4.19}$$

i.e. $y\big(T; y_j, u^*(y_j)\big) \in Y_{\bar{V}}$ for all $y_j \in Y_{\bar{V}}$, $u_{j-1} \in \mathcal{B}_{\Sigma,s}\big(u^*(y_j), r_u\big)$.

Proof. We leverage Assumption 4.1 to estimate

$$\begin{aligned} V\big(y\big(T; y_j, u^*(y_j)\big)\big) &\overset{(4.7)}{\leq} V\big(y_j\big) - T a_3 \big\|y_j\big\|_{L^p(\Omega)}^q \\ &\overset{(4.6)}{\leq} (1 - T\bar{a}) V\big(y_j\big) \leq \bar{V}, \end{aligned}$$

Equation (4.18) implies $T \leq \frac{1}{\bar{a}}$.

where we have exploited $T \leq \frac{1}{\bar{a}}$ and $y_j \in Y_{\bar{V}}$, i.e. $V\big(y_j\big) \leq \bar{V}$. ∎

Next, we bound the deviation of the trajectory with the suboptimal control u_j from the optimal trajectory.

Reminder: Iteration number k
With k we denote the number of iterations the optimizer does to compute u_j, see Assumption 4.3.

Lemma 4.3 — Bound on deviation away from optimal trajectory.
Requirements: Assumption 4.3
There exists a $K_1 \in \mathbb{N}$, which is independent of y_j and u_{j-1}, such that

$$\big\|y\big(T; y_j, u_j\big) - y\big(T; y_j, u^*(y_j)\big)\big\|_{L^p(\Omega)} \leq r_y \tag{4.20}$$

holds for all $y_j \in Y_{\bar{V}}, u_{j-1} \in \mathcal{B}_{\Sigma,s}\big(u^*(y_j), r_u\big)$ and all $k \geq K_1$.

Proof. We have

$$\big\|y\big(T; y_j, u_j\big) - y\big(T; y_j, u^*(y_j)\big)\big\|_{L^p(\Omega)} \overset{(4.4)}{\leq} L \big\|u_j - u^*(y_j)\big\|_{L^s(\Sigma)}.$$

As $y_j \in Y_{\bar{V}} \subset Y_{\bar{V}} \oplus \mathcal{B}_{\Omega,p}(0, r_y)$ and $u_{j-1} \in \mathcal{B}_{\Sigma,s}(u^*(y_j), r_u)$, we can exploit Assumption 4.3 to continue with

$$\begin{aligned}\left\|y(T; y_j, u_j) - y(T; y_j, u^*(y_j))\right\|_{L^p(\Omega)} &\overset{(4.15)}{\leq} \kappa^k L \left\|u_{j-1} - u^*(y_j)\right\|_{L^s(\Sigma)} \\ &\leq \kappa^k L r_u.\end{aligned}$$

$u_{j-1} \in \mathcal{B}_{\Sigma,s}\left(u^*\left(y_j\right), r_u\right)$ means that $\left\|u_{j-1} - u^*\left(y_j\right)\right\|_{L^s(\Sigma)} \leq r_u$.

As $\kappa \in (0,1)$, we find $K_1 \in \mathbb{N}$ such that

$$\left\|y(T; y_j, u_j) - y(T; y_j, u^*(y_j))\right\|_{L^p(\Omega)} \leq \kappa^k L r_u \leq r_y \quad \forall k \geq K_1.$$

K_1 depends on κ, L, r_u and r_y.

■

As a corollary, we obtain that the next state remains "close" to the level set $Y_{\bar{V}}$. The exact formulation of this statement is Corollary 4.2.

Corollary 4.2 — Successor state remains close to $Y_{\bar{V}}$.
Requirements: Assumption 4.1, Assumption 4.3, K_1 as in Lemma 4.3
We have

$$y(T; y_j, u_j) \in Y_{\bar{V}} \oplus \mathcal{B}_{\Omega,p}(0, r_y)$$

for all $y_j \in Y_{\bar{V}}$, $u_{j-1} \in \mathcal{B}_{\Sigma,s}(u^*(y_j), r_u)$ and all $k \geq K_1$.

We will improve this statement in the form of Lemma 4.5 that states $y\left(T; y_j, u_j\right) \in Y_{\bar{V}}$. But, we need Corollary 4.2 to arrive at Lemma 4.5.

Proof. We need to find $\hat{y} \in Y_{\bar{V}}$ with $\left\|y(T; y_j, u_j) - \hat{y}\right\|_{L^p(\Omega)} \leq r_y$. We show that $\hat{y} = y(T; y_j, u^*(y_j))$ does the trick. To that end, we need to check

(i) $\left\|y(T; y_j, u_j) - y(T; y_j, u^*(y_j))\right\|_{L^p(\Omega)} \leq r_y$,
(ii) $y(T; y_j, u^*(y_j)) \in Y_{\bar{V}}$.

Fortunately, we have already shown with Lemma 4.3 that (i) is satisfied and with Lemma 4.2 that (ii) is satisfied. ■

4.2.2. Step 2: forward invariant set for the system-optimizer dynamics

First, we establish that the controls u_j remain close enough to the optimal control $u^*(y_j)$ such that the optimizer continuously exhibits Q-linear convergence to $u^*(y_j)$. More formally:

Lemma 4.4 — Controls u_j remain in region of Q-linear convergence.
Requirements: Assumptions 4.2 and 4.3
There exists $K_2 \in \mathbb{N}$, which is independent of y_j and u_{j-1}, such that

$$\left\|u_j - u^*(y(T; y_j, u_j))\right\|_{L^s(\Sigma)} \leq r_u,$$

i.e. we have

$$u_j \in \mathcal{B}_{r_u,s}(u^*(y_{j+1}), r_u),$$

for all $y_j \in Y_{\bar{V}}$, $u_{j-1} \in \mathcal{B}_{\Sigma,s}(u^*(y_j), r_u)$ and $k \geq K_2$.

We have formulated in Assumption 4.3 that u_j has to be in the ball $\mathcal{B}_{r_u,s}\left(u^*\left(y_{j+1}\right), r_u\right)$ to guarantee Q-linear convergence of the optimizer.

Reminder:

$y_{j+1} = y\left(T; y_j, u_j\right)$, see (4.16).

Proof. We start off by estimating

$$\begin{aligned}\left\|u_j - u^*\left(y(T;y_j,u_j)\right)\right\|_{L^s(\Sigma)} \leq{}& \underbrace{\left\|u_j - u^*(y_j)\right\|_{L^s(\Sigma)}}_{\text{(I)}} \\ &+ \underbrace{\left\|u^*(y_j) - u^*\left(y\left(T;y_j,u^*(y_j)\right)\right)\right\|_{L^s(\Sigma)}}_{\text{(II)}} \\ &+ \underbrace{\left\|u^*\left(y\left(T;y_j,u^*(y_j)\right)\right) - u^*\left(y(T;y_j,u_j)\right)\right\|_{L^s(\Sigma)}}_{\text{(III)}}.\end{aligned}$$

We continue by estimating the summands (I), (II), and (III) separately:

(I): As $y_j \in Y_{\bar{V}}$ and $u_{j-1} \in \mathcal{B}_{\Sigma,s}\left(u^*(y_j), r_u\right)$, we can use Assumption 4.3 to estimate

$$\left\|u_j - u^*(y_j)\right\|_{L^s(\Sigma)} \overset{(4.15)}{\leq} \kappa^k \left\|u_{j-1} - u^*(y_j)\right\|_{L^s(\Sigma)} \leq \kappa^k r_u.$$

(II): As $y_j \in Y_{\bar{V}}$ and $0 < T \leq T_\alpha$, we can use Assumption 4.2 to bound

$$\left\|u^*(y_j) - u^*\left(y\left(T;y_j,u^*(y_j)\right)\right)\right\|_{L^s(\Sigma)} \overset{(4.9)}{\leq} \alpha r_u.$$

(III): By assumption we have $y_j \in Y_{\bar{V}}$ and $u_{j-1} \in \mathcal{B}_{\Sigma,s}\left(u^*(y_j), r_u\right)$. In addition to that, we have shown with Lemma 4.2, that

$$y\left(T;y_j,u^*(y_j)\right) \in Y_{\bar{V}}.$$

Moreover, Corollary 4.2 yields that

$$y_j \in \mathcal{B}_{\Omega,p}\left(y\left(T;y_j,u^*(y_j)\right), r_y\right).$$

Hence, we can use Lemma 4.1 and Assumptions 4.2 and 4.3 to estimate

$$\begin{aligned}&\left\|u^*\left(y\left(T;y_j,u^*(y_j)\right)\right) - u^*\left(y(T;y_j,u_j)\right)\right\|_{L^s(\Sigma)} \\ &\qquad \overset{(4.10)}{\leq} \gamma \left\|y\left(T;y_j,u^*(y_j)\right) - y(T;y_j,u_j)\right\|_{L^p(\Omega)} \\ &\qquad \overset{(4.4)}{\leq} \gamma L \left\|u_j - u^*(y_j)\right\|_{L^s(\Sigma)} \\ &\qquad \overset{(4.15)}{\leq} \kappa^k \gamma L \left\|u_{j-1} - u^*(y_j)\right\|_{L^s(\Sigma)} \leq \kappa^k \gamma L r_u.\end{aligned}$$

We put the estimates for (I), (II), (III) back together to obtain

$$\left\|u_j - u^*\left(y(T;y_j,u_j)\right)\right\|_{L^s(\Sigma)} \leq \left(\kappa^k(1+\gamma L) + \alpha\right) r_u.$$

As $\kappa, \alpha \in (0,1)$, we find $K_2 \in \mathbb{N}$ such that $\kappa^k(1+\gamma L) \leq 1-\alpha$ and accordingly

$$\left\|u_j - u^*\left(y(T;y_j,u_j)\right)\right\|_{L^s(\Sigma)} \leq r_u$$

for all $k \geq K_2$.

K_2 depends on κ, γ, L and α.

■

Next, we improve the statement $y(T; y_j, u_j) \in Y_{\bar{V}} \oplus \mathcal{B}_{\Omega,p}(0, r_y)$ from Corollary 4.2 to $y(T; y_j, u_j) \in Y_{\bar{V}}$.

Lemma 4.5 — Current state remains in $Y_{\bar{V}}$.
Requirements: Assumptions 4.1 and 4.3
There exists $K_3 \in \mathbb{N}$, which is independent of y_j and u_{j-1} but depends on T, such that $y(T; y_j, u_j) \in Y_{\bar{V}}$ for all $y_j \in Y_{\bar{V}}, u_{j-1} \in \mathcal{B}_{\Sigma,s}(u^*(y_j), r_u)$ and $k \geq K_3$.

Proof. Let $k \geq K_1$, where K_1 is as in Corollary 4.2. In this case, it follows from Corollary 4.2 that $y(T; y_j, u_j) \in Y_{\bar{V}} \oplus \mathcal{B}_{\Omega,p}(0, r_y)$, which allows us to use Assumption 4.1 to get

$$\begin{aligned} V(y(T; y_j, u_j))^{\frac{1}{q}} \leq &\left| V(y(T; y_j, u_j))^{\frac{1}{q}} - V(y(T; y_j, u^*(y_j)))^{\frac{1}{q}} \right| \\ &+ V(y(T; y_j, u^*(y_j)))^{\frac{1}{q}}. \end{aligned}$$

For the first summand we estimate

$$\begin{aligned} &\left| V(y(T; y_j, u_j))^{\frac{1}{q}} - V(y(T; y_j, u^*(y_j)))^{\frac{1}{q}} \right| \\ &\overset{(4.8)}{\leq} \mu \left\| y(T; y_j, u_j) - y(T; y_j, u^*(y_j)) \right\|_{L^p(\Omega)} \\ &\overset{(4.4)}{\leq} \mu L \left\| u_j - u^*(y_j) \right\|_{L^s(\Sigma)} \\ &\overset{(4.15)}{\leq} \kappa^k \mu L \left\| u_{j-1} - u^*(y_j) \right\|_{L^s(\Sigma)} \leq \kappa^k \mu L r_u. \end{aligned} \tag{4.21}$$

For the second summand we use Lemma 4.2 to obtain

$$V(y(T; y_j, u^*(y_j)))^{\frac{1}{q}} \leq (1 - T\bar{a})^{\frac{1}{q}} \bar{V}^{\frac{1}{q}}.$$

Together, we thus have

$$V(y(T; y_j, u_j))^{\frac{1}{q}} \leq \kappa^k \mu L r_u + (1 - T\bar{a})^{\frac{1}{q}} \bar{V}^{\frac{1}{q}}.$$

As $\kappa \in (0, 1)$ and $0 < T \leq \frac{1}{\bar{a}}$, we find $\bar{K}$ such that

$\bar{K}$ depends on $\kappa, \mu, L, r_u, T, a_2, a_3, q$ and $\bar{V}$.

$$\kappa^k \mu L r_u \leq \left(1 - (1 - T\bar{a})^{\frac{1}{q}}\right) \bar{V}^{\frac{1}{q}}$$

and accordingly

$$V(y(T; y_j, u_j))^{\frac{1}{q}} \leq \bar{V}^{\frac{1}{q}},$$

i.e. $y(T; y_j, u_j) \in Y_{\bar{V}}$ for all $k \geq K_3 := \max\{K_1, \bar{K}\}$. ∎

K_3 depends on r_y, κ, μ, L, r_u, T, a_2, a_3, q and $\bar{V}$.

Now, we can show that y and u_j remain in $Y_{\bar{V}}$ and $\mathcal{B}_{\Sigma,s}(u^*(y_{j+1}), r_u)$ throughout the entire NMPC procedure.

Lemma 4.6 — Lemmas 4.4 and 4.5 apply at all times.
Requirements: Assumption 4.1, Assumption 4.2, Assumption 4.3
There exists $K_4 \in \mathbb{N}$, which is independent of y_j and u_{j-1}, such that $y_j \in Y_{\bar{V}}$ and $u_{j-1} \in \mathcal{B}_{\Sigma,s}(u^*(y_j), r_u)$ for all $j \in \mathbb{N}_0$ if $y_0 \in Y_{\bar{V}}, u_{-1} = u_{\text{init}} \in \mathcal{B}_{\Sigma,s}(u^*(y_0), r_u)$ and $k \geq K_4$.

K_4 depends on κ, γ, L, μ, r_y, r_u, T, a_2, a_3, q, α and $\bar{V}$.

Proof. We set $K_4 := \max\{K_2, K_3\}$ and then prove the assertion by induction. For $j = 0$ the assertion is satisfied by the assumptions for y_0 and u_{init}. If the assertion holds for a fixed $j \in \mathbb{N}_0$, Lemma 4.4 yields $u_j \in \mathcal{B}_{\Sigma,s}\left(u^*\left(y_{j+1}\right), r_u\right)$ and Lemma 4.5 yields $y_{j+1} \in Y_{\bar{V}}$. ■

In other words, Lemma 4.6 states that the set

$$\Lambda := \left\{(y,u) \in C(\bar{\Omega}) \times L^s(\Sigma) \;\middle|\; V(y) \leq \bar{V}, \left\|u - u^*(y)\right\|_{L^s(\Sigma)} \leq r_u\right\}$$

That is if $k \geq K_4$.

is forward invariant under the system-optimizer dynamics (4.16).

4.2.3. Step 3: error contraction and LYAPUNOV decrease perturbation

With $y_j \in Y_{\bar{V}}$ and $u_{j-1} \in \mathcal{B}_{\Sigma,s}\left(u^*\left(y_j\right), r_u\right)$ for all $j \in \mathbb{N}_0$ established, we can now develop estimates for the perturbations of the error contraction and the LYAPUNOV decrease. To do so, we repeat the definitions

$$E_j := \left\|u_j - u^*\left(y_j\right)\right\|_{L^s(\Sigma)}$$

for the error contraction and

$$V_j \quad := V\left(y_j\right)^{\frac{1}{q}}$$

for the LYAPUNOV decrease. First, we study the contraction of E_j.

Lemma 4.7 — Error contraction estimate.
Requirements: Assumption 4.1, Assumption 4.2, Assumption 4.3, K_4 as in Lemma 4.6
Let $y_0 \in Y_{\bar{V}}, u_{-1} = u_{\text{init}} \in \mathcal{B}_{\Sigma,s}\left(u^*\left(y_0\right), r_u\right)$ and $k \geq K_4$. Then,

$$E_{j+1} \leq \kappa^k C_1 E_j + \kappa^k C_2 V_j,$$

holds for all $j \in \mathbb{N}_0$, where

$$C_1 := 1 + L\gamma,$$
$$C_2 := \gamma a_1^{-\frac{1}{q}} \left(1 + (1 - T\bar{a})^{\frac{1}{q}}\right).$$

Proof. From Lemma 4.6 we get $y_j \in Y_{\bar{V}}, u_{j-1} \in \mathcal{B}_{\Sigma,s}\left(u^*\left(y_j\right), r_u\right)$ for all $j \in \mathbb{N}_0$. Therefore, we can use Assumption 4.3 to estimate

$$\begin{aligned} E_{j+1} &\overset{(4.15)}{\leq} \kappa^k \left\|u_j - u^*\left(y\left(T; y_j, u_j\right)\right)\right\|_{L^s(\Sigma)} \\ &\leq \kappa^k \left\|u_j - u^*\left(y_j\right)\right\|_{L^s(\Sigma)} + \kappa^k \left\|u^*\left(y\left(T; y_j, u_j\right)\right) - u^*\left(y_j\right)\right\|_{L^s(\Sigma)} \\ &= \kappa^k E_j + \kappa^k \left\|u^*\left(y\left(T; y_j, u_j\right)\right) - u^*\left(y_j\right)\right\|_{L^s(\Sigma)}. \end{aligned} \tag{4.22}$$

To estimate the second summand, we recall that Lemma 4.3 yields that $y(T; y_j, u_j) \in \mathcal{B}_{\Omega,p}(y_j, r_y)$. This allows us to exploit Assumption 4.2 to obtain

> **Reminder:**
> We can use Lemma 4.3 as, in particular, $k \geq K_4 \geq \max\{K_2, K_3\} \geq K_3 := \max\{K_1, \bar{K}\} \geq K_1$.

$$\begin{aligned}\left\|u^*(y(T; y_j, u_j)) - u^*(y_j)\right\|_{L^s(\Sigma)} &\overset{(4.10)}{\leq} \gamma \left\|y(T; y_j, u_j) - y_j\right\|_{L^p(\Omega)} \\ &\leq \gamma \Big(\underbrace{\left\|y(T; y_j, u_j) - y(T; y_j, u^*(y_j))\right\|_{L^p(\Omega)}}_{\text{(I)}} \\ &\quad + \underbrace{\left\|y(T; y_j, u^*(y_j)) - y_j\right\|_{L^p(\Omega)}}_{\text{(II)}} \Big).\end{aligned}$$

We estimate the summands (I) and (II) separately.

(I): From Lemma 4.3 we get

$$\begin{aligned}\left\|y(T; y_j, u_j) - y(T; y_j, u^*(y_j))\right\|_{L^p(\Omega)} &\overset{(4.4)}{\leq} L \left\|u_j - u^*(y_j)\right\|_{L^s(\Sigma)} \\ &= L E_j.\end{aligned}$$

(II): As we have seen in the proof of Corollary 4.2, cf. Equation (4.19), we have $y(T; y_j, u^*(y_j)) \in Y_{\bar{V}}$. Therefore, we can use Assumption 4.1 to obtain

$$\begin{aligned}\left\|y(T; y_j, u^*(y_j)) - y_j\right\|_{L^p(\Omega)} &\leq \left\|y(T; y_j, u^*(y_j))\right\|_{L^p(\Omega)} + \left\|y_j\right\|_{L^p(\Omega)} \\ &\overset{(4.6)}{\leq} a_1^{-\frac{1}{q}} \left(V(y(T; y_j, u^*(y_j)))^{\frac{1}{q}} + V(y_j)^{\frac{1}{q}} \right) \\ &\overset{(4.19)}{\leq} a_1^{-\frac{1}{q}} \left(1 + (1 - T\bar{a})^{\frac{1}{q}} \right) V(y_j)^{\frac{1}{q}}.\end{aligned}$$

We combine (I) and (II) again to obtain

$$\begin{aligned}\left\|u^*(y(T; y_j, u_j)) - u^*(y_j)\right\|_{L^s(\Sigma)} &\leq \gamma L E_j + \gamma a_1^{-\frac{1}{q}} \left(1 + (1 - T\bar{a})^{\frac{1}{q}} \right) V_j \\ &= \gamma L E_j + C_2 V_j.\end{aligned} \tag{4.23}$$

We finish the proof by plugging in Equation (4.23) into Equation (4.22) and obtaining

$$\begin{aligned}E_{j+1} &\overset{(4.22)}{\leq} \kappa^k E_j + \kappa^k \left\|u^*(y(T; y_j, u_j)) - u^*(y_j)\right\|_{L^s(\Sigma)} \\ &\overset{(4.23)}{\leq} \kappa^k E_j + \kappa^k (\gamma L E_j + C_2 V_j) = \kappa^k C_1 E_j + \kappa^k C_2 V_j.\end{aligned}$$

■

Now, we examine the perturbation of the LYAPUNOV decrease (4.7) caused by using a suboptimal control u_j instead of the optimal u^*.

Lemma 4.8 — Lyapunov decrease perturbation.
Requirements: Assumption 4.1, Assumption 4.2, Assumption 4.3, and K_4 as in Lemma 4.6
Let $y_0 \in Y_{\bar{V}}, u_{-1} = u_{\text{init}} \in \mathcal{B}_{\Sigma,s}(u^*(y_0), r_u)$ and $k \geq K_4$. Then,

$$V_{j+1} \leq C_3 E_j + C_4 V_j$$

holds for all $j \in \mathbb{N}_0$, where

$$C_3 := \mu L,$$
$$C_4 := (1 - T\bar{a})^{\frac{1}{q}} \in [0, 1).$$

Proof. From Lemma 4.6 we get $y_j \in Y_{\bar{V}}, u_{j-1} \in \mathcal{B}_{\Sigma,s}(u^*(y_j), r_u)$ for all $j \in \mathbb{N}_0$. Therefore, we can employ Equation (4.19) of Lemma 4.2 and proceed as in the proof of Lemma 4.5, cf. Equation (4.21), to get

$$\begin{aligned} V_{j+1} &\leq \left| V\left(y\left(T; y_j, u_j\right)\right)^{\frac{1}{q}} - V\left(y\left(T; y_j, u^*(y_j)\right)\right)^{\frac{1}{q}} \right| \\ &\quad + V\left(y\left(T; y_j, u^*(y_j)\right)\right)^{\frac{1}{q}} \\ &\overset{(4.19),(4.21)}{\leq} \mu L \left\| u_j - u^*(y_j) \right\|_{L^s(\Sigma)} + (1 - T\bar{a})^{\frac{1}{q}} V(y_j)^{\frac{1}{q}} = C_3 E_j + C_4 V_j. \end{aligned}$$

Note that $C_4 \in [0, 1)$ as $0 < T \leq \frac{1}{\bar{a}}$, cf. Equation (4.18). ∎

4.2.4. Step 4: stability of the positive linear system

Lemmas 4.7 and 4.8 yield that the development of V_j and E_j can be written as

$$\begin{pmatrix} V_{j+1} \\ E_{j+1} \end{pmatrix} \leq \begin{pmatrix} C_4 & C_3 \\ \kappa^k C_2 & \kappa^k C_1 \end{pmatrix} \begin{pmatrix} V_j \\ E_j \end{pmatrix} =: A \begin{pmatrix} V_j \\ E_j \end{pmatrix}, \tag{4.24}$$

with

$$V_0 := V(y_0)^{\frac{1}{q}}, \quad \text{and} \quad E_0 := \left\| u_0 - u^* y_0 \right\|_{L^s(\Sigma)}.$$

We recall that u_0 is computed through k iterations of the optimizer started at u_{init}. As $\kappa^k, C_1, C_2, C_3, C_4 \geq 0$, we have $A \in \mathbb{R}^{2\times 2}_{\geq 0}$. As additionally, $E_j \geq 0$ for all $j \in \mathbb{N}_0$ by definition and

$$V_j = V(y_j)^{\frac{1}{q}} \overset{(4.6)}{\geq} a_1^{\frac{1}{q}} \left\| y_j \right\|_{L^p(\Omega)} \geq 0$$

for all $j \in \mathbb{N}_0$ by virtue of Lemma 4.6 and Assumption 4.1, we have that

$$V_j^{\mathrm{u}} \geq V_j \quad \text{and} \quad E_j^{\mathrm{u}} \geq E_j \quad \forall j \in \mathbb{N}_0$$

if we define V_j^{u} and E_j^{u} through

$$\begin{pmatrix} V_{j+1}^{\mathrm{u}} \\ E_{j-1}^{\mathrm{u}} \end{pmatrix} = A \begin{pmatrix} V_j^{\mathrm{u}} \\ E_j^{\mathrm{u}} \end{pmatrix} \quad \text{with} \quad \begin{pmatrix} V_0^{\mathrm{u}} \\ E_0^{\mathrm{u}} \end{pmatrix} := \begin{pmatrix} V_0 \\ E_0 \end{pmatrix}. \tag{4.25}$$

As $A \in \mathbb{R}^{2\times 2}_{\geq 0}$ and $V_j^{\mathrm{u}}, E_j^{\mathrm{u}} \geq 0$ for all $j \in \mathbb{N}_0$, the system (4.25) is a positive linear system. Thus, we can use [208, Theorem 23], which is an adapted version of the results presented in [116], to prove asymptotic stability of the origin $(V^{\mathrm{u}}, E^{\mathrm{u}}) = (0,0)$ and to state an associated LYAPUNOV function for the system (4.25).

[208]: Zanelli et al. (2021), "A Lyapunov function for the combined system-optimizer dynamics in inexact model predictive control"

[116]: Kaczorek (2008), "The Choice of the Forms of Lyapunov Functions for a Positive 2D Roesser Model"

Lemma 4.9 — Lyapunov function for system (4.25)**.**
There exists $K_5 \in \mathbb{N}$ such that for all $k \geq K_5$ the origin $(0,0) \in \mathbb{R}^2$ is asymptotically stable for the system (4.25) for all $(y_0, u_{\mathrm{init}}) \in \Lambda$ with a LYAPUNOV function $\tilde{V}$ in $\mathbb{R}^2_{\geq 0}$ which is given by

$$\tilde{V}\left((V_j^{\mathrm{u}}, E_j^{\mathrm{u}})\right) = V_j^{\mathrm{u}} + \beta E_j^{\mathrm{u}},$$

where $\beta := 2\mu L$.

Proof. By virtue of [208, Theorem 23], we need to find $\widehat{w} \in \mathbb{R}^2_{\geq 0}$ and $0 < \widehat{d} \in \mathbb{R}$ such that

$$\max_{i=1,2}\left[\left(A^T - \mathbb{I}\right)\widehat{w}\right]_i \leq -\widehat{d}.$$

We set $\widehat{w} = \begin{pmatrix} 1 \\ \beta \end{pmatrix}$. With that choice, we have $\left(A^T - \mathbb{I}\right)\widehat{w} < 0$ (componentwise) if

$$\begin{aligned} C_4 - 1 + \kappa^k C_2 \beta &< 0, \\ C_3 + \left(\kappa^k C_1 - 1\right)\beta &< 0, \\ \beta &> 0. \end{aligned} \tag{4.26}$$

As $\kappa \in (0,1), C_1 > 0$, there exists $\bar{K} \in \mathbb{N}$ such that $\kappa^k C_1 < 1$ for all $k \geq \bar{K}$. With that (4.26) is satisfied if and only if

$$\frac{C_3}{1-\kappa^k C_1} < \beta < \frac{1-C_4}{\kappa^k C_2} \quad \text{and} \quad \beta > 0.$$

As

$$\frac{C_3}{1-\kappa^k C_1} \overset{k\to\infty}{\longrightarrow} C_3 \quad \text{and} \quad \frac{1-C_4}{\kappa^k C_2} \overset{k\to\infty}{\longrightarrow} \infty,$$

there exists $\tilde{K} \in \mathbb{N}$ such that

$$\frac{C_3}{1-\kappa^k C_1} < 2C_3 < \frac{1-C_4}{\kappa^k C_2} \quad \forall k \geq \max\{\bar{K}, \tilde{K}\}.$$

Choosing $\beta = 2C_3 = 2\mu L$ and $K_5 := \max\{\bar{K}, \tilde{K}\}$ concludes the proof. ■

In fact, for all fixed $\delta \in (1, \infty)$ we find a $K_{5,\delta} \in \mathbb{N}$ such that $\beta = \delta\mu L$ can be chosen in the definition of $\tilde{V}$. The smaller δ, the smaller $K_{5,\delta}$ can be chosen.

Lemma 4.9 allows us to bound the current state y_j and the associated optimal control $u^*(y_j)$ in the form of Corollary 4.3.

Corollary 4.3 — Bounds for y_j and $u^*(y_j)$.
Requirements: Assumption 4.1, Assumption 4.2, Assumption 4.3 and K_5 as in Lemma 4.9

Let $k \geq K_5$. For all $(y_0, u_{\text{init}}) \in \Lambda$ exists a $J(y_0, u_{\text{init}}) \in \mathbb{N}$, which depends on y_0, u_{init}, such that

$$\|y_j\|_{L^p(\Omega)} \leq r_y, \quad \text{i.e.} \quad y_j \in \mathcal{B}_{\Omega,p}(0, r_y),$$

and

$$\|u^*(y_j)\|_{L^s(\Sigma)} \leq \gamma \|y_j\|_{L^p(\Omega)} \tag{4.27}$$

for all $j \geq J(y_0, u_{\text{init}})$.

Proof. From Lemma 4.9 we know that

$$0 \leq a_1^{\frac{1}{q}} \|y_j\|_{L^p(\Omega)} \leq V_j \leq V_j^{\text{u}} \overset{j\to\infty}{\longrightarrow} 0. \tag{4.28}$$

Therefore, there exists a $J(y_0, u_{\text{init}}) \in \mathbb{N}$ such that $\|y_j\|_{L^p(\Omega)} \leq r_y$ for all $j \geq J(y_0, u_{\text{init}})$. Moreover, Equation (4.28) implies that $y_j \overset{j\to\infty}{\longrightarrow} y^* = 0$. Now, we need the continuity of V from Assumption 4.1 to get

$$V(y^*) = V\left(\lim_{j\to\infty} y_j\right) = \lim_{j\to\infty} V(y_j) = 0,$$

which implies that $y^* \in Y_{\tilde{V}}$. This allows us to use Equation (4.12) to obtain

$$\|u^*(y_j)\|_{L^s(\Sigma)} \leq \gamma \|y_j\|_{L^p(\Omega)}$$

for all $j \geq J(y_0, u_{\text{init}})$ as we have now secured that $y^* = 0 \in Y_{\tilde{V}}$ and $y_j \in \mathcal{B}_{\Omega,p}(y^* = 0, r_y)$. ∎

4.2.5. Main result: asymptotic stability of the system-optimizer dynamics

The following Theorem 4.3 is the main result of this chapter. Theorem 4.3 states that the origin $(y^*, u^*) = (0, 0)$ is asymptotically stable for the system-optimizer dynamics (4.16). The forward invariant set S on which this stability holds is of a rather technical nature. However, we can illustrate its role by the following interpretation of Lemma 4.9 and Theorem 4.3 in conjunction with Corollary 4.3. For all $y_0 \in Y_{\tilde{V}}, u_{\text{init}} \in \mathcal{B}_{\Sigma,s}(u^*(y_0), r_u)$ the states y_j converge towards $y^* = 0$ and the controls u_j converge towards $u^*(y_j)$. As u^* is not necessarily continuous, this does not yet imply that $u_j \overset{j\to\infty}{\longrightarrow} u^* = 0$. But, for all $y_0 \in Y_{\tilde{V}}, u_{\text{init}} \in \mathcal{B}_{\Sigma,s}(u^*(y_0), r_u)$ exists a finite number of NMPC steps $J(y_0, u_{\text{init}})$ after which (y_j, u_j) have entered the forward invariant set S from where on asymptotic stability is guaranteed.

Theorem 4.3 — Asymptotic stability of the system-optimizer dynamics. Requirements: Assumption 4.1, Assumption 4.2, Assumption 4.3 and K_5 as in Lemma 4.9

Let $k \geq K_5$. Then, the equilibrium $(y^*, u^*) = (0,0) \in L^p(\Omega) \times L^s(\Sigma)$ is asymptotically stable on the forward invariant set

$$S := \bigcup_{(y_0, u_{\text{init}}) \in \Lambda} \left\{ \left(y_j(y_0, u_{\text{init}}), u_j(y_0, u_{\text{init}}) \right) \,\middle|\, j \geq J(y_0, u_{\text{init}}) \right\},$$

where $J(y_0, u_{\text{init}})$ is as in Corollary 4.3, with

$$V_{\text{so}}((y,u)) := V(y)^{\frac{1}{q}} + 2\mu L \left\| u - u^*(y) \right\|_{L^s(\Sigma)}$$

as LYAPUNOV function for the system-optimizer dynamics (4.16).

Proof. First we show that S is forward invariant. To that end, let $(y,u) \in S$. By construction of S there exist $(y_0, u_{\text{init}}) \in \Lambda$ and $J(y_0, u_{\text{init}}) \in \mathbb{N}$ such that

$$y = y_j(y_0, u_{\text{init}}), \quad \text{and} \quad u = u_j(y_0, u_{\text{init}})$$

for a $j \geq J(y_0, u_{\text{init}})$. Again by the construction of S, we immediately obtain

$$y(T; y, u) = y_{j+1}(y_0, u_{\text{init}}) \quad \text{and} \quad \xi(y(T; y, u), u) = u_{j+1}(y_0, u_{\text{init}})$$

and accordingly

$$\left(y(T; y, u), \xi(y(T; y, u), u) \right) \in S.$$

Next we show that V_{so} is indeed a LYAPUNOV function for the system-optimizer dynamics (4.16) on S. To that end, we need to show, cf. [101, Definition 2.18], that functions $\alpha_1, \alpha_2 \in \mathcal{K}_\infty$ and $\alpha_3 \in \mathcal{K}$ exist such that

(i) $\alpha_1\left(\left\| y_j - y^* \right\|_{L^p(\Omega)} + \left\| u_j - u^* \right\|_{L^s(\Sigma)} \right) \leq V_{\text{so}}((y_j, u_j))$,

(ii) $V_{\text{so}}((y_j, u_j)) \leq \alpha_2\left(\left\| y_j - y^* \right\|_{L^p(\Omega)} + \left\| u_j - u^* \right\|_{L^s(\Sigma)} \right)$,

(iii) $V_{\text{so}}((y_{j+1}, u_{j+1})) \leq V_{\text{so}}((y_j, u_j)) - \alpha_3\left(\left\| y_j - y^* \right\|_{L^p(\Omega)} + \left\| u_j - u^* \right\|_{L^s(\Sigma)} \right)$

hold for all $(y_j, u_j) \in S$.

As $j \geq J(y_0, u_{\text{init}})$ by construction for $(y_j, u_j) \in S$, we can use Equation (4.27) from Corollary 4.3.

[101]: Grüne et al. (2017), *Nonlinear Model Predictive Control*

Reminder: Comparison functions

$\mathcal{K} := \{\alpha \in C(\mathbb{R}_0^+, \mathbb{R}_0^+) \mid \alpha \text{ is strictly increasing with } \alpha(0) = 0.\}$

$\mathcal{K}_\infty := \{\alpha \in \mathcal{K} \mid \alpha \text{ is unbounded}\}$

Cf. Definition 2.10 on p. 19.

We can use our results from Subsection 4.2.4 because $(y_j, u_j) \in \Lambda$ as $S \subset \Lambda$.

(i) We make a case distinction for $a_1^{\frac{1}{q}} - \beta\gamma$:

Case: $a_1^{\frac{1}{q}} - \beta\gamma > 0$: Here, we have

$$\begin{aligned} V_{\mathrm{so}}\big((y_j,u_j)\big) &= V(y_j)^{\frac{1}{q}} + \beta\|u_j - u^*(y_j)\|_{L^s(\Sigma)} \\ &\geq V(y_j)^{\frac{1}{q}} + \beta\Big(\|u_j\|_{L^s(\Sigma)} - \|u^*(y_j)\|_{L^s(\Sigma)}\Big) \\ &\overset{(4.6)}{\geq} a_1^{\frac{1}{q}}\|y_j\|_{L^p(\Omega)} - \beta\|u^*(y_j)\|_{L^s(\Sigma)} + \beta\|u_j\|_{L^s(\Sigma)} \\ &\overset{(4.27)}{\geq} \Big(a_1^{\frac{1}{q}} - \beta\gamma\Big)\|y_j\|_{L^p(\Omega)} + \beta\|u_j\|_{L^s(\Sigma)} \\ &= \Big(a_1^{\frac{1}{q}} - \beta\gamma\Big)\|y_j - y^*\|_{L^p(\Omega)} + \beta\|u_j - u^*\|_{L^s(\Sigma)} \\ &\geq \min\Big\{a_1^{\frac{1}{q}} - \beta\gamma, \beta\Big\}\Big(\|y_j - y^*\|_{L^p(\Omega)} + \|u_j - u^*\|_{L^s(\Sigma)}\Big). \end{aligned}$$

Case: $a_1^{\frac{1}{q}} - \beta\gamma \leq 0$: Here, we need to make an additional case distinction for $\|y_j\|_{L^s(\Sigma)} - \frac{1}{\gamma}\|u_j\|_{L^s(\Sigma)}$. If $\|y_j\|_{L^s(\Sigma)} \leq \frac{1}{\gamma}\|u_j\|_{L^s(\Sigma)}$, we proceed as above to find

$$\begin{aligned} V_{\mathrm{so}}\big((y_j,u_j)\big) &\geq \Big(a_1^{\frac{1}{q}} - \beta\gamma\Big)\|y_j - y^*\|_{L^p(\Omega)} + \beta\|u_j - u^*\|_{L^s(\Sigma)} \\ &\geq \Big(a_1^{\frac{1}{q}} - \beta\gamma\Big) \sup_{\|y_j\|_{L^p(\Omega)} \leq \frac{1}{\gamma}\|u_j\|_{L^s(\Sigma)}} \|y_j\|_{L^p(\Omega)} + \beta\|u_j\|_{L^s(\Sigma)} \\ &= \Big(a_1^{\frac{1}{q}} - \beta\gamma\Big)\frac{1}{\gamma}\|u_j\|_{L^s(\Sigma)} + \beta\|u_j\|_{L^s(\Sigma)} = \frac{a_1^{\frac{1}{q}}}{\gamma}\|u_j\|_{L^s(\Sigma)}. \end{aligned}$$

If instead, $\|y_j\|_{L^s(\Sigma)} > \frac{1}{\gamma}\|u_j\|_{L^s(\Sigma)}$, we have

$$V_{\mathrm{so}}\big((y_j,u_j)\big) \geq V(y_j)^{\frac{1}{q}} \overset{(4.6)}{\geq} a_1^{\frac{1}{q}}\|y_j\|_{L^p(\Omega)} \geq \frac{a_1^{\frac{1}{q}}}{\gamma}\|u_j\|_{L^s(\Sigma)}.$$

Both times we thus have

$$V_{\mathrm{so}}\big((y_j,u_j)\big) \geq \frac{a_1^{\frac{1}{q}}}{\gamma}\|u_j\|_{L^s(\Sigma)} = \frac{a_1^{\frac{1}{q}}}{\gamma}\|u_j - u^*\|_{L^s(\Sigma)}$$

for all $(y_j,u_j) \in S$. Moreover, we have from Assumption 4.1 that

$$V_{\mathrm{so}}\big((y_j,u_j)\big) \geq V(y_j)^{\frac{1}{q}} \overset{(4.6)}{\geq} a_1^{\frac{1}{q}}\|y_j\|_{L^p(\Omega)} = a_1^{\frac{1}{q}}\|y_j - y^*\|_{L^p(\Omega)}.$$

Together we have established that

$$\begin{aligned} V_{\mathrm{so}}\big((y_j,u_j)\big) &\geq \frac{a_1^{\frac{1}{q}}}{2}\|y_j - y^*\|_{L^p(\Omega)} + \frac{a_1^{\frac{1}{q}}}{2\gamma}\|u_j - u^*\|_{L^s(\Sigma)} \\ &\geq \min\Big\{1, \frac{1}{\gamma}\Big\}a_1^{\frac{1}{q}}\Big(\|y_j - y^*\|_{L^p(\Omega)} + \|u_j - u^*\|_{L^s(\Sigma)}\Big). \end{aligned}$$

We set

$$C \coloneqq \begin{cases} \min\left\{a_1^{\frac{1}{q}} - \beta\gamma, \beta\right\}, & \text{if } a_1^{\frac{1}{q}} - \beta\gamma > 0, \\ \min\left\{1, \frac{1}{\gamma}\right\} a_1^{\frac{1}{q}}, & \text{else} \end{cases}$$

and conclude (i) by setting

$$\begin{aligned} &\alpha_1\Big(\|y_j - y^*\|_{L^p(\Omega)} + \|u_j - u^*\|_{L^s(\Sigma)}\Big) \\ &\qquad \coloneqq C\Big(\|y_j - y^*\|_{L^p(\Omega)} + \|u_j - u^*\|_{L^s(\Sigma)}\Big). \end{aligned}$$

(ii) For the upper bound on V_{so} we proceed by estimating

$$\begin{aligned} V_{\mathrm{so}}\big((y_j, u_j)\big) &\overset{(4.6)}{\leq} a_2^{\frac{1}{q}} \|y_j\|_{L^p(\Omega)} + \beta\Big(\|u_j\|_{L^s(\Sigma)} + \|u^*(y_j)\|_{L^s(\Sigma)}\Big) \\ &\overset{(4.27)}{\leq} \left(a_2^{\frac{1}{q}} + \gamma\beta\right)\|y_j\|_{L^p(\Omega)} + \beta\|u_j\|_{L^s(\Sigma)} \\ &\leq \max\left\{a_2^{\frac{1}{q}} + \gamma\beta, \beta\right\}\Big(\|y_j - y^*\|_{L^p(\Omega)} + \|u_j - u^*\|_{L^s(\Sigma)}\Big) \\ &=: \alpha_2\Big(\|y_j - y^*\|_{L^p(\Omega)} + \|u_j - u^*\|_{L^s(\Sigma)}\Big). \end{aligned}$$

(iii) For the final part (iii), we first note that

$$V_{\mathrm{so}}\big((y_{j+1}, u_{j+1})\big) = (1, \beta)\begin{pmatrix} V(y_{j+1})^{\frac{1}{q}} \\ \|u_{j+1} - u^*(y_{j+1})\|_{L^s(\Sigma)} \end{pmatrix}.$$

We note that the LYAPUNOV function $\tilde{V}$ is the same independent of the initial state and control in Λ. We can hence apply its properties to the dynamics (4.25) started in $V(y_j)^{\frac{1}{q}}$ and $\|u_j - u^*(y_j)\|_{L^s(\Sigma)}$. More precisely, we get with [208, Theorem 23]

[208]: Zanelli et al. (2021), "A Lyapunov function for the combined system-optimizer dynamics in inexact model predictive control"

$$\begin{aligned} &(1, \beta) A\begin{pmatrix} V(y_j)^{\frac{1}{q}} \\ \|u_j - u^*(y_j)\|_{L^s(\Sigma)} \end{pmatrix} = \tilde{V}\left(A\begin{pmatrix} V(y_j)^{\frac{1}{q}} \\ \|u_j - u^*(y_j)\|_{L^s(\Sigma)} \end{pmatrix}\right) \\ &\leq \tilde{V}\Big(V(y_j)^{\frac{1}{q}}, \|u_j - u^*(y_j)\|_{L^s(\Sigma)}\Big) - \widehat{d}\left\|\begin{pmatrix} V(y_j)^{\frac{1}{q}} \\ \|u_j - u^*(y_j)\|_{L^s(\Sigma)} \end{pmatrix}\right\|_1 \\ &= (1, \beta)\begin{pmatrix} V(y_j)^{\frac{1}{q}} \\ \|u_j - u^*(y_j)\|_{L^s(\Sigma)} \end{pmatrix} - \widehat{d}\left\|\begin{pmatrix} V(y_j)^{\frac{1}{q}} \\ \|u_j - u^*(y_j)\|_{L^s(\Sigma)} \end{pmatrix}\right\|_1. \end{aligned}$$

It then follows that

$$\begin{aligned} &(1, \beta)\begin{pmatrix} V(y_{j+1})^{\frac{1}{q}} \\ \|u_{j+1} - u^*(y_{j+1})\|_{L^s(\Sigma)} \end{pmatrix} \leq (1, \beta) A\begin{pmatrix} V(y_j)^{\frac{1}{q}} \\ \|u_j - u^*(y_j)\|_{L^s(\Sigma)} \end{pmatrix} \\ &\leq (1, \beta)\begin{pmatrix} V(y_j)^{\frac{1}{q}} \\ \|u_j - u^*(y_j)\|_{L^s(\Sigma)} \end{pmatrix} - \widehat{d}\left\|\begin{pmatrix} V(y_j)^{\frac{1}{q}} \\ \|u_j - u^*(y_j)\|_{L^s(\Sigma)} \end{pmatrix}\right\|_1. \end{aligned}$$

Therefore, we get

$$V_{\mathrm{so}}\big((y_{j+1},u_{j+1})\big) \leq V_{\mathrm{so}}\big((y_j,u_j)\big) - \widehat{d}\Big(V(y_j)^{\frac{1}{q}} + \|u_j - u^*(y_j)\|_{L^s(\Sigma)}\Big).$$

With the same computations as for (i), but with $\beta = 1$, we obtain

$$V(y_j)^{\frac{1}{q}} + \|u_j - u^*(y_j)\|_{L^s(\Sigma)} \geq \tilde{C}\Big(\|y_j - y^*\|_{L^p(\Omega)} + \|u_j - u^*(y_j)\|_{L^s(\Sigma)}\Big),$$

with

$$\tilde{C} := \begin{cases} \min\Big\{a_1^{\frac{1}{q}} - \gamma, 1\Big\}, & \text{if } a_1^{\frac{1}{q}} - \gamma > 0, \\ \min\Big\{1, \frac{1}{\gamma}\Big\} a_1^{\frac{1}{q}}, & \text{else.} \end{cases}$$

Defining

$$\alpha_3 := \widehat{d}\tilde{C}\Big(\|y_j - y^*\|_{L^p(\Omega)} + \|u_j - u^*(y_j)\|_{L^s(\Sigma)}\Big)$$

concludes the proof.

■

5. Smooth Multivariate Shape-Preserving Interpolation

In real-life engineering tasks, it is common to encounter situations where we lack a suitable functional formulation for the dependence of one quantity on another. In other cases, a functional formulation may exist, but its evaluation is too computationally expensive to be practical. A typical approach in such cases is to take measurements or evaluate the expensive formula at dedicated points and interpolate between these measurements. An example is the fuel consumption of an internal combustion engine as a function of speed and mean effective pressure, see, for example, [169, Bild 5-6]. We refer to such a collection of measurement or evaluation points, along with the corresponding function values, as a *Lookup Table (LUT)*. In fact, we also use LUTs in our OCP formulation for our Ecological Adaptive Cruise Control (EACC) system application, as described in Chapter 8.

[169]: Schnabel et al. (2011), *Grundlagen der Straßenverkehrstechnik und der Verkehrsplanung: Band 1-Straßenverkehrstechnik*

In particular, LUTs, as they appear in engineering applications, sometimes involve more than just one or two free variables. For interpolation, this means that we need to interpolate not only univariate or bivariate data but also multivariate data. Even if a complete model or all system properties are unknown, some information about the system modeled by a LUT is often available. For example, it may be known that the process state must be non-negative everywhere, such as when modeling concentrations, or monotonic with respect to a certain free variable. Consequently, there is a need for interpolation methods that preserve this kind of information, which is, of course, also encoded in the LUT. We refer to such interpolation methods as *shape-preserving interpolation methods*. We explain our understanding of shape-preservation in more detail in Section 5.1.

Another engineering requirement is that the interpolation method be a local scheme, meaning that the value of the interpolation function depends only on a few nearby data points.

The simplest interpolation method is linear interpolation of the data. For details, see standard textbooks on numerical analysis, such as [10, 180]. Although it is the simplest method, linear interpolation actually satisfies the requirements mentioned so far. It can be extended to multivariate settings, preserves most typical shapes, and works locally. Unfortunately, it does not fulfill a final requirement that arises when we want to perform optimal control or NMPC of systems modeled using LUTs. This last requirement is that the interpolating function must be sufficiently smooth, which at least means continuously differentiable—a property that linear interpolation does not possess.

[10]: Atkinson (1991), *An introduction to numerical analysis*
[180]: Süli et al. (2003), *An Introduction to Numerical Analysis*

While smooth shape-preserving interpolation methods for univariate and bivariate cases and smooth multivariate interpolation methods exist, to the best of our knowledge, a smooth, multivariate, and shape-preserving interpolation method has not yet been presented. One of the main contributions of this thesis is the development of such

We explain Coons' patches in Subsection 5.3.2.

We define SP2 in Definition 5.6.

[49]: Costantini (1988), "An algorithm for computing shape-preserving interpolating splines of arbitrary degree"

Theorem 5.1 contains the precise statement.

By uni-, bi-, or trivariate, we refer to situations where $i = 1$, $i = 2$, or $i = 3$, respectively.

an interpolation method. The main idea behind our novel interpolation method is to use Coons' patches to extend univariate interpolation methods to the multivariate case. Our proposed interpolation method has the following advantages:

- It works in the multivariate case for gridded data.
- It preserves shapes in an SP2 sense.
- It can preserve virtually all shapes of interest if a univariate method exists that preserves this shape.
- It has the same order of smoothness as the univariate interpolation method, allowing an arbitrarily high degree of smoothness to be achieved by utilizing, e.g., the univariate method presented in [49].
- It inherits locality if the univariate method is a local scheme.

The presentation of our novel method in this chapter is structured as follows. First, we clarify the problem at hand in Section 5.1. Next, we review existing approaches to shape-preserving and multivariate interpolation methods in Section 5.2. Section 5.3 contains the description of our novel method. Numerical results for our method on some test problems are reported in Section 5.5. A proof that our described method indeed produces shape-preserving interpolations is given in Section 5.4. A proof regarding the arbitrary smoothness for the general multivariate case is currently in progress.

To present the problem at hand and our method concisely, we make use of a multi-index notation.

Definition 5.1 An n-dimensional *multi-index* α is a tuple

$$\alpha = \left(\alpha^1, \dots, \alpha^n\right)^T \in \mathbb{N}_0^n. \tag{5.1}$$

Moreover, we write $\alpha \leq \gamma$ if and only if $\alpha^i \leq \gamma^i$ for all $i \in \{1, \dots, n\}$. For a fixed maximum multi-index $0 < \alpha_{\max} = \left(\alpha_{\max}^1, \dots, \alpha_{\max}^n\right)^T$, we use the abbreviation

$$\{\alpha \mid 0 \leq \alpha \leq \alpha_{\max}\} \coloneqq \left\{\alpha \in \mathbb{N}_0^n \;\middle|\; 0 \leq \alpha^i \leq \alpha_{\max}^i \; \forall i \in \{1, \dots, n\}\right\}.$$

We occasionally need to fix some indices in a multi-index and vary others. For that, we extend the multi-index notation of Definition 5.1 by the following definitions.

Definition 5.2 For a subset $\mathcal{M} \subset \{1, \dots, n\}$ with

$$1 \leq m \coloneqq |\mathcal{M}| \leq n - 1,$$

we define the *multi-index* $\alpha^{\mathcal{M}}$ *that corresponds to the dimensions* $\mathcal{M}$ by

$$\alpha^{\mathcal{M}} \coloneqq \left(\alpha^{i_1}, \dots, \alpha^{i_m}\right)^T \in \mathbb{N}_0^m,$$

where the dimension indices $i_1, \dots, i_m \in \{1, \dots, n\}$ satisfy

$$\bigcup_{k=1}^{m} \{i_k\} = \mathcal{M},$$

and, if $m > 1$, additionally

$$i_1 < \ldots < i_m.$$

Similarly, we denote by $\alpha^{\neg\mathcal{M}}$ the *multi-index that corresponds to dimensions not in* $\mathcal{M}$, which we define by

$$\alpha^{\neg\mathcal{M}} := \left(\alpha^{i_1}, \ldots, \alpha^{i_{n-m}}\right)^T \in \mathbb{N}_0^{n-m},$$

where the dimension indices $i_1, \ldots, i_{n-m} \in \{1, \ldots, n\}$ satisfy

$$\bigcup_{k=1}^{n-m} \{i_k\} = \{1, \ldots, n\} \setminus \mathcal{M},$$

and, if $m < n - 1$, additionally

$$i_1 < \ldots < i_{n-m}.$$

If $\mathcal{M} = \{i\}$, we write $\alpha^i \in \mathbb{N}_0$, which is consistent with α^i denoting a single index, and $\alpha^{\neg i} \in \mathbb{N}_0^{n-1}$ instead of $\alpha^{\{i\}}$ and $\alpha^{\neg\{i\}}$, respectively. We can also define a *recombination of* $\beta^{\mathcal{M}}$ *and* $\gamma^{\neg\mathcal{M}}$, which we denote by $\beta^{\mathcal{M}}|\gamma^{\neg\mathcal{M}} \in \mathbb{N}_0^n$ for given

$$\beta^{\mathcal{M}} = \left(\beta^{i_1}, \ldots, \beta^{i_m}\right)^T \in \mathbb{N}_0^m,$$
$$\gamma^{\neg\mathcal{M}} = \left(\gamma^{j_1}, \ldots, \gamma^{j_{n-m}}\right)^T \in \mathbb{N}_0^{n-m},$$

by setting

$$\beta^{\mathcal{M}}|\gamma^{\neg\mathcal{M}} := \left(\alpha^1, \ldots, \alpha^n\right)^T \quad \text{with} \quad \alpha^i := \begin{cases} \gamma^i & \text{if } i \in \mathcal{M}, \\ \beta^i & \text{else.} \end{cases}$$

Similarly, we can also split a given $\alpha \in \mathbb{N}_0^n$ into $\alpha^{\mathcal{M}} \in \mathbb{N}_0^m$ and $\alpha^{\neg\mathcal{M}} \in \mathbb{N}_0^{n-m}$, meaning that

$$\alpha = \alpha^{\mathcal{M}}|\alpha^{\neg\mathcal{M}}.$$

In other words, $\alpha^{\mathcal{M}}$ contains the components of α that correspond to the dimensions collected in $\mathcal{M}$, and $\alpha^{\neg\mathcal{M}}$ the components that correspond to dimensions other than the ones in $\mathcal{M}$.

If $i \in \mathcal{M}$, then there exists a $k \in \{1, \ldots, m\}$ such that $\gamma^i = \beta^{i_k}$. If $i \notin \mathcal{M}$, then there exists a $k \in \{1, \ldots, n-m\}$ such that $\gamma^i = \gamma^{\neg i_k}$.

If a multi-index is obtained by splitting, it inherits the same base symbol from the multi-index it was split off. Therefore, if multi-indices $\alpha^{\mathcal{M}}$, $\alpha^{\neg\mathcal{M}}$ occur together, they are obtained from the same parent multi-index α, whereas multi-indices with different base symbols, i.e., for example, β^i and β^j, do not have a parent multi-index.

5.1. Problem formulation

We are concerned with the interpolation of a gridded data set, which we define as follows.

Definition 5.3 For each dimension $i \in \{1, \ldots, n\}$, we assume an ordered *one-dimensional node set*

$$\mathcal{N}^i := \left\{x_0^i < \ldots < x_{\alpha_{\max}^i}^i\right\} \subset \mathbb{R}. \tag{5.2}$$

The convention regarding sub- and superscripts employed in this chapter is as follows: indices written as superscripts refer to the dimension, while indices written as subscripts enumerate elements within a fixed dimension.

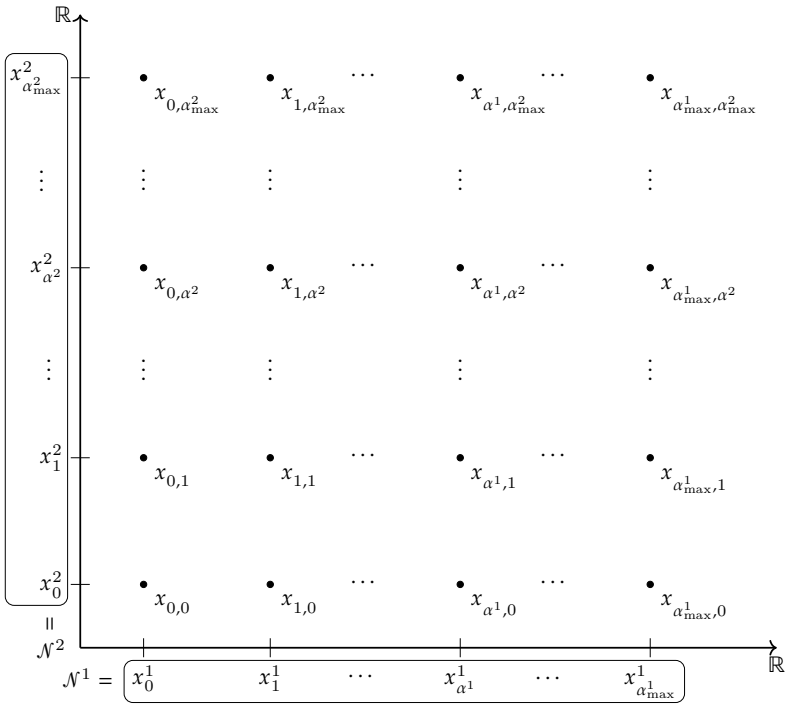


Figure 5.1.: The total node set $\mathcal{N}$, arising from the Cartesian product of $\mathcal{N}^1$ and $\mathcal{N}^2$, forms a grid in $\mathbb{R}^2$. The two-dimensional multi-index $\alpha = (\alpha^1, \alpha^2)^T$ is used to reference a specific node in $\mathcal{N}$.

The *(total) node set* $\mathcal{N} \subset \mathbb{R}^n$ is then defined as

$$\mathcal{N} := \mathcal{N}^1 \times \cdots \times \mathcal{N}^n.$$

Each *node* x_α in the total node set $\mathcal{N}$ can be referenced by defining

$$x_\alpha := \left(x^1_{\alpha^1}, \ldots, x^n_{\alpha^n}\right)^T \in \mathbb{R}^n$$

for each multi-index $0 \le \alpha \le \alpha_{\max} := (\alpha^1_{\max}, \ldots, \alpha^n_{\max})^T$. For each node x_α, a *data point* $d_\alpha \in \mathbb{R}$ is given. Since the nodes form an n-dimensional grid defined by the total node set $\mathcal{N}$, we refer to the *data set*

$$\mathcal{D} := \{(x_\alpha, d_\alpha) \in \mathbb{R}^n \times \mathbb{R} \mid 0 \le \alpha \le \alpha_{\max}\} \tag{5.3}$$

as a *gridded data set*.

Definition: $C^q(\mathbb{R}^n, \mathbb{R})$

$C^q(\mathbb{R}^n, \mathbb{R})$ denotes the *space of q-times continuously differentiable functions*, while $C^0(\mathbb{R}^n, \mathbb{R})$ denotes the *space of all continuous functions* from $\mathbb{R}^n$ to $\mathbb{R}$. For details, see standard textbooks on the subject, e.g., [1, Paragraph 1.26].

Frequently, a function is called smooth if it has a smoothness order $q > 1$. Whenever the smoothness order itself is not important, we omit it and only refer to smooth interpolations.

Figure 5.1 illustrates the gridded nature of the node set and the use of the multi-index notation in the bivariate case.

Our goal is to compute a smooth interpolation of the data set $\mathcal{D}$.

Definition 5.4 A function $p : \mathbb{R}^n \to \mathbb{R}$ is called an *interpolation of the data set $\mathcal{D}$ with smoothness of order q* if the *interpolation condition*

$$p(x_\alpha) = d_\alpha, \text{ for all } 0 \le \alpha \le \alpha_{\max}$$

is satisfied and if $p \in C^q(\mathbb{R}^n, \mathbb{R})$.

Additionally, the interpolation p should be shape-preserving. Unfortunately, the term shape-preserving is not used consistently throughout the literature. This inconsistency also applies to the term "shape."

In an attempt to provide a unifying definition, we state the rather general Definition 5.5.

Definition 5.5 A *shape* is a property of the data set $\mathcal{D}$, which is a discrete object, that can be reasonably generalized to the graph

$$\mathcal{G}_p := \{(x, p(x)) \mid x_0 \leq x \leq x_{\alpha_{\max}}\}$$

of the interpolation p, which is a continuous object. The interpolation is then said to be *shape-preserving* if all shapes of interest exhibited by the data set $\mathcal{D}$ are also exhibited by the graph $\mathcal{G}_p$.

Attention:

In some works, for example [125], shape-preserving interpolation is understood in a different sense, to which we refer to as fair or visually pleasing interpolation, cf. [89, Section 2.1]. We discuss this issue in more detail on p. 85.

Definition 5.5 obviously leaves some room for interpretation. Therefore, it is, to some degree, up to researchers and practitioners to state which shapes are of interest for their method or application and how this shape is defined for the data set and for the interpolation. This also applies to us, and therefore we provide details on our interpretation of Definition 5.5 in the following Subsection 5.1.1.

5.1.1. Categories of shape-preservation in multivariate settings

Our proposed method inherits its shape-preservation property from the method applied to solve the arising univariate interpolation problems. Therefore, we adopt the rather general Definition 5.5 of the term shape and only name typical shapes in Example 5.1.

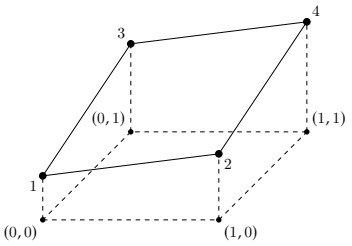
▪ **Example 5.1** The following shapes are typical choices for shapes that must be preserved in shape-preserving interpolation.

- **Positivity:** If all data points are positive, i.e., $d_\alpha \geq 0$ for all $0 \leq \alpha \leq \alpha_{\max}$, the interpolation should satisfy $p(x_\alpha) \geq 0$ for all $x_0 \leq x \leq x_{\alpha_{\max}}$. See, e.g., [113, 212].
- **Monotonicity:** If data points are monotonically increasing or decreasing, respectively, along one or more directions, the interpolation should also be monotonically increasing or decreasing, respectively, in this region. In a univariate setting, it is clear how monotonicity in both the data set and the interpolation should be understood. In multivariate settings, different interpretations are possible. We revisit this issue when defining our categorizations of shape-preservation in Definition 5.6. See, for example, [52, 114].
- **Convexity and concavity:** Similarly to monotonicity, different definitions of convexity and concavity, respectively, are used for both the data set and the interpolation. Nevertheless, convexity and concavity are common shapes of interest in shape-preserving interpolation. See, for example, [32, 70, 123].

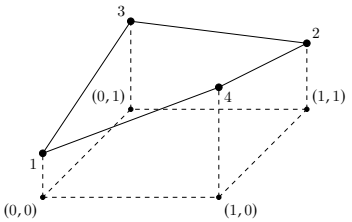
▪

[32]: Brodlie et al. (1991), "Preserving convexity using piecewise cubic interpolation"
[52]: Costantini et al. (1991), "A local scheme for bivariate co-monotone interpolation"
[70]: Dougherty et al. (1989), "Nonnegativity-, monotonicity-, or convexity-preserving cubic and quintic Hermite interpolation"
[113]: Hussain et al. (2008), "Positivity-preserving interpolation of positive data by rational cubics"
[114]: Hyman (1983), "Accurate Monotonicity Preserving Cubic Interpolation"
[123]: Kuijt (1998), "Convexity preserving interpolation: stationary nonlinear subdivision and splines"
[212]: Zhu (2018), "C^2 positivity-preserving rational interpolation splines in one and two dimensions"

In multivariate settings, some shapes allow for multiple conceivable definitions, as illustrated in the following Example 5.2.



(a) The example data set $\mathcal{D} = \{(\binom{0}{0}, 1), (\binom{1}{0}, 2), (\binom{0}{1}, 3), (\binom{1}{1}, 4)\}$ is monotonically increasing according to both definitions.



(b) The example data set $\mathcal{D} = \{(\binom{0}{0}, 1), (\binom{1}{0}, 4), (\binom{0}{1}, 3), (\binom{1}{1}, 2)\}$ is monotonically increasing only according to the first definition.

Figure 5.2.: Visualizations of two example data sets illustrating that the two monotonicity definitions in Example 5.2 can lead to different interpretations of the shape of a given data set.

$x \leq y$ is meant componentwise, i.e., it means $x^i \leq y^i \ \forall i \in \{1, 2\}$.

▪ **Example 5.2** To define monotonicity in a bivariate data set $\mathcal{D}$, we could consider the following two definitions. The first is based on the edges of the rectangles in the node grid, while the second considers the entire rectangles.

(i) The data set is monotonically increasing along the edge $[x^1_{\alpha^1}, x^1_{\alpha^1+1}] \times x^2_{\alpha^2}$ or the edge $x^1_{\alpha^1} \times [x^2_{\alpha^2}, x^2_{\alpha^2+1}]$, respectively, if

$$d_{(\alpha^1,\alpha^2)} \leq d_{(\alpha^1+1,\alpha^2)}, \quad \text{or} \quad d_{(\alpha^1,\alpha^2)} \leq d_{(\alpha^1,\alpha^2+1)}, \text{ resp.}$$

(ii) The data set is monotonically increasing on the rectangle $[x^1_{\alpha^1}, x^1_{\alpha^1+1}] \times [x^2_{\alpha^2}, x^2_{\alpha^2+1}]$ if

$$\begin{aligned} &d_{(\alpha^1,\alpha^2)} \leq d_{(\alpha^1+1,\alpha^2)}, && d_{(\alpha^1,\alpha^2+1)} \leq d_{(\alpha^1+1,\alpha^2+1)}, \\ &d_{(\alpha^1,\alpha^2)} \leq d_{(\alpha^1,\alpha^2+1)}, && d_{(\alpha^1+1,\alpha^2)} \leq d_{(\alpha^1+1,\alpha^2+1)}, \\ &d_{(\alpha^1,\alpha^2)} \leq d_{(\alpha^1+1,\alpha^2+1)}. \end{aligned}$$

If we examine the two example data sets illustrated in Figure 5.2, we observe that these two definitions can indeed lead to different interpretations of the shape of a given data set.

In Definition 5.5, we stated that the shape definitions must be reasonably transferable from the data set to the interpolation. Possible transfers of the two definitions given above to the interpolation are:

(i) The interpolation p is monotonically increasing along the edge $[x^1_{\alpha^1}, x^1_{\alpha^1+1}] \times x^2_{\alpha^2}$ or the edge $x^1_{\alpha^1} \times [x^2_{\alpha^2}, x^2_{\alpha^2+1}]$, respectively, if

$$p(x) \leq p(y) \text{ for all } x, y \in [x^1_{\alpha^1}, x^1_{\alpha^1+1}] \times x^2_{\alpha^2} \text{ with } x^1 \leq y^1,$$

or

$$p(x) \leq p(y) \text{ for all } x, y \in x^1_{\alpha^1} \times [x^2_{\alpha^2}, x^2_{\alpha^2+1}] \text{ with } x^2 \leq y^2, \text{ resp.}$$

(ii) The interpolation p is monotonically increasing on the rectangle $[x^1_{\alpha^1}, x^1_{\alpha^1+1}] \times [x^2_{\alpha^2}, x^2_{\alpha^2+1}]$ if

$$p(x) \leq p(y) \text{ for all } x, y \in [x^1_{\alpha^1}, x^1_{\alpha^1+1}] \times [x^2_{\alpha^2}, x^2_{\alpha^2+1}] \text{ with } x \leq y.$$

▪

Example 5.2 also demonstrates that different definitions of a specific shape can vary in strictness. It is likely agreeable that the second monotonicity definition in Example 5.2 is stricter than the first, as it considers entire rectangles for the shape definition instead of only the edges. Consequently, we propose the following categorization scheme for how interpolations can preserve shapes.

Definition 5.6 To define the categories, we introduce

$$\mathcal{N}^i \subset \bar{\mathcal{N}}^i := [x^i_0, x^i_{\alpha^i_{\max}}] \subset \mathbb{R},$$

i.e., the straight line from x^i_0 to $x^i_{\alpha^i_{\max}}$. We then propose the following *shape-preservation categories* for an interpolation p of a given data set $\mathcal{D}$ with node set $\mathcal{N}$.

- **Shape Preservation Category 1 (SP1):** Shapes are preserved along parallel grid lines of the grid formed by $\mathcal{N}$ for a single direction $i \in \{1, \dots, n\}$, i.e., on the set

$$\left(\bigtimes_{j=1}^{i-1} \mathcal{N}^j\right) \times \bar{\mathcal{N}}^i \times \left(\bigtimes_{j=i+1}^{n} \mathcal{N}^j\right).$$

- **Shape Preservation Category 2 (SP2):** Shapes are preserved along all grid lines of the grid formed by $\mathcal{N}$, i.e. on the set

$$\bigcup_{i=1}^{n}\left(\bigtimes_{j=1}^{i-1} \mathcal{N}^j\right) \times \bar{\mathcal{N}}^i \times \left(\bigtimes_{j=i+1}^{n} \mathcal{N}^j\right).$$

- **Shape Preservation Category 3 (SP3):** Shapes are preserved at each point

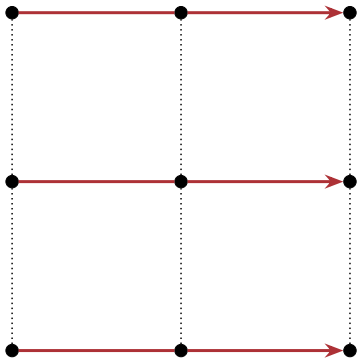
$$x \in \bar{\mathcal{N}} \coloneqq \bigtimes_{i=1}^{n} \bar{\mathcal{N}}^i$$

in each direction

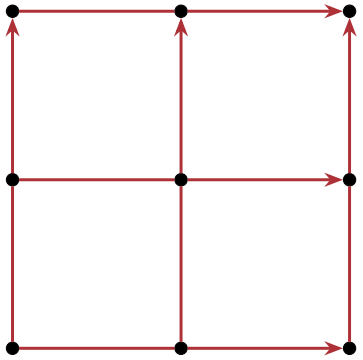
$$v \in \mathcal{V} \coloneqq \{v \in \mathbb{R}^n \mid 0 \leq v,\ \|v\|_= 1\}.$$

In other words, the interpolation is shape-preserving for each $x \in \bar{\mathcal{N}}$, $v \in \mathcal{V}$ along the capped line

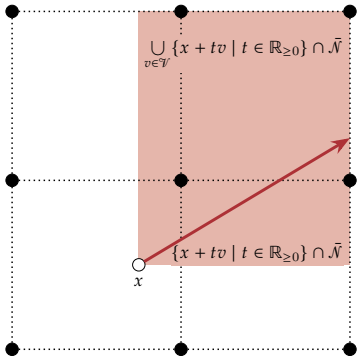
$$\{x + tv \mid t \in \mathbb{R}_{\geq 0}\} \cap \bar{\mathcal{N}}.$$



(a) SP1 directions for $i = 1$.



(b) SP2 directions.



(c) SP3 directions at some $x \in \bar{\mathcal{N}}$.

Figure 5.3.: Visualizations of the sets on which the interpolation p preserves shapes in the three shape-preservation categories in the bivariate case.

The category that most shape-preserving interpolation methods fall into is SP2. This includes our method, as proposed in Section 5.3.

5.2. Literature review

In this section, we give an overview of existing work related to smooth multivariate shape-preserving interpolation, divided into four parts. First, we examine smooth univariate shape-preserving methods in Subsection 5.2.1. These methods are of interest because our approach requires solving univariate interpolation problems. Especially, methods with the ability to preserve multiple shapes and a higher smoothness are of interest. Next, we focus on bivariate shape-preserving methods. While our approach can handle general multivariate interpolation problems, bivariate interpolation is a common case. Therefore, we list alternatives to our method for the bivariate case in Subsection 5.2.2. A common approach to bivariate interpolation in Computer Aided Geometric Design (CAGD) is the use of blending schemes. Since our method employs a blending scheme to combine univariate interpolations, we review popular works on blending schemes in Subsection 5.2.3. Finally, we discuss existing methods for multivariate interpolation in Subsection 5.2.4. To the best of our knowledge, no multivariate interpolation method exists that is both shape-preserving and achieves smoothness of order $q > 1$. Our proposed method is unique in meeting all these requirements.

Reminder: Smoothness order q

We say that a function f has a smoothness of order q if $C^q(\mathbb{R}^n, \mathbb{R})$, cf. Definition 5.4.

5.2.1. Smooth univariate shape-preserving interpolation methods

Shape-preserving interpolation methods for univariate data have been extensively studied. Akima's seminal work [2] presents an interpolation method that, in our understanding, falls into the category of visually pleasing interpolation (see the discussion on p. 85). Fritsch and Carlson introduced methods for monotonicity-preserving interpolation in [83, 84]. Other early works include [55, 142, 174, 179]. For additional references, we direct the reader to the survey [89] and the references therein.

[2]: Akima (1970), "A New Method of Interpolation and Smooth Curve Fitting Based on Local Procedures"
[55]: De Boor (1978), *A practical guide to splines*
[49]: Costantini (1988), "An algorithm for computing shape-preserving interpolating splines of arbitrary degree"
[84]: Fritsch et al. (1980), "Monotone Piecewise Cubic Interpolation"
[89]: Goodman (2002), "Shape preserving interpolation by curves"
[142]: Nielson (1974), "Some piecewise polynomial alternatives to splines under tension"
[174]: Schweikert (1966), "An interpolation curve using a spline in tension"
[179]: Späth (1974), *Spline algorithms for curves and surfaces*

Many shape-preserving interpolation methods developed over time produce interpolating functions in C^1. However, we are particularly interested in methods that yield interpolating functions in C^q, $q \geq 2$. A noteworthy paper on this topic is [49], where Costantini presents a method that preserves monotonicity and convexity/concavity while producing interpolating functions of arbitrary smoothness. Moreover, the method does not require derivative information, making it user-friendly. The method is formulated in a general setting, allowing extensions to preserve additional data properties. In the follow-up work [50], the method is extended to include broader settings, such as periodicity conditions.

5.2.2. Bivariate shape-preserving interpolation methods

The methods discussed above are designed for univariate interpolation. However, bivariate shape-preserving interpolation has also received considerable attention. One of the earliest works in this area is the extension of [2] presented in [3]. However, like its predecessor, this method only produces fair or visually pleasing interpolations. Carlson and Fritsch also explored monotonicity-preserving interpolation in the bivariate setting, publishing a series of works [41–43, 85]. These methods, however, produce interpolations in C^1. Notable bivariate shape-preserving interpolations with higher smoothness are presented in [51, 52, 108, 113, 167].

[3]: Akima (1974), "A Method of Bivariate Interpolation and Smooth Surface Fitting Based on Local Procedures"
[41]: Carlson et al. (1985), "Monotone Piecewise Bicubic Interpolation"
[51]: Costantini et al. (1990), "Shape-Preserving Bivariate Interpolation"
[85]: Fritsch et al. (1985), "Monotonicity preserving bicubic interpolation: A progress report"
[108]: Heß et al. (1994), "Positive quartic, monotone quintic C^2-spline interpolation in one and two dimensions"
[113]: Hussain et al. (2008), "Positivity-preserving interpolation of positive data by rational cubics"
[167]: Schmidt et al. (1993), "S-convex, monotone, and positive interpolation with rational bicubic splines of C^2-continuity"

5.2.3. Blending schemes

Bivariate interpolation is often referred to as the computation of interpolating surfaces, particularly in CAGD. One popular approach to creating interpolating surfaces is the use of blending schemes, which originated in CAGD research. The first widely known blending scheme is Coons' patches, introduced in [46]. Over the years, Coons' patches were further developed into Gregory squares [94], generalized into Gordon surfaces [90–93], and used to generate C^2 surfaces [14, 205]. We discuss Coons' patches in more detail in Subsection 5.3.2, as our approach is also based on them. Coons' patches have also been used for bivariate shape-preserving interpolations, e.g., [53, 54] for C^1 interpolations and [212] for positivity-preserving C^2 interpolations.

[46]: Coons (1964), *Surfaces for computer-aided design*
[14]: Barnhill (1983), "Computer aided surface representation and design"
[53]: Costantini et al. (1996), "A bicubic shape-preserving blending scheme"
[90]: Gordon (1969), "Distributive lattices and the approximation of multivariate functions"
[94]: Gregory (1974), "Smooth interpolation without twist constraints"
[205]: Worsey (1984), "A modified C^2 Coons' patch"
[212]: Zhu (2018), "C^2 positivity-preserving rational interpolation splines in one and two dimensions"

Although multivariate interpolation is less prominent in CAGD, there are some use cases, e.g., [16] and those listed in [15, p. 11]. Outside CAGD, multivariate interpolation methods based on GREGORY squares or GORDON surfaces have also emerged, e.g., [17, 93]. However, these methods are not designed to be shape-preserving.

[15]: Barnhill (1985), "Surfaces in computer aided geometric design: a survey with new results"
[93]: Gordon (1971), "Blending-function methods of bivariate and multivariate interpolation and approximation"

5.2.4. Multivariate interpolation methods

For multivariate interpolation, many methods have been developed beyond the CAGD-rooted approaches mentioned in Subsection 5.2.3. Techniques include polynomial interpolation, divided differences, KERGIN's interpolation theory, multivariate BIRKHOFF and HERMITE interpolation, radial basis functions, and more. For references on these techniques, see [86, 147].

[86]: Gasca et al. (2000), "Polynomial interpolation in several variables"
[147]: Olver (2006), "On Multivariate Interpolation"

However, none of these methods addresses shape preservation. In fact, multivariate shape-preserving interpolation has received little attention. Since LUTs with more than two or three variables are common in engineering, preserving shape in the multivariate case is still desirable. The only paper explicitly addressing multivariate shape-preserving interpolation is [125]. But, the author states on p. 334 :

> *"In bi- and multivariate situations just as in univariate situations, preserving linearity, monotonicity and convexity (concavity) is, contrary to a widespread assumption, not equivalent to preserving shape."*
>
> — LAVERY [125]

[125]: Lavery (2001), "Shape-preserving, multiscale interpolation by bi- and multivariate cubic L1 splines"

This statement clarifies that the method in [125] produces interpolations that are shape-preserving in a sense different from ours. While we agree that shape is not equivalent to the mentioned properties, we follow the argument in [89] that these properties are typically desirable in shape-preserving interpolation. In our view, LAVERY's interpretation of shape-preserving could be classified as fair or visually pleasing interpolation, as described in [89]. Furthermore, the interpolations computed by the method in [125] are in C^1.

5.3. Novel smooth multivariate shape-preserving interpolation method

We now present our novel method for smooth multivariate shape-preserving interpolation. The main idea of it is as follows.

Main idea — Smooth multivariate shape-preserving interpolation. We apply an arbitrary univariate interpolation method that is both shape-preserving and smooth to all the univariate interpolation problems arising along the grid lines of the node set $\mathcal{N}$. These interpolations form a curve network, which we extend into the interior of the hypercubes of the grid $\mathcal{N}$ by applying a multivariate generalization of COONS' patches.

The resulting interpolation preserves, in a SP2 sense, all the shapes preserved by the univariate method. Depending on the smoothness of the univariate method and the blending functions, the resulting interpolation can achieve any desired degree of smoothness.

We divide the presentation of our method into three steps. First, we describe the interpolation of the univariate interpolation problems along all grid lines, see Subsection 5.3.1. Then, we present Coons' patches in Subsection 5.3.2. Finally, we explain in Subsection 5.3.3 how we blend the univariate interpolation results together. The complete algorithm for our interpolation method is given in Algorithm 5.1.

5.3.1. Univariate interpolation along the grid lines

When moving along a grid line of the grid $\mathcal{D}$, the nodes and data points along that line form a univariate interpolation problem if we aim to interpolate the data along that line. The corresponding univariate data set for this interpolation problem is defined as follows.

There are $\prod_{j\in\{1,\dots,n\}\setminus\{i\}}\left(\alpha^j_{\max}+1\right)$ parallel grid lines and, accordingly, as many univariate interpolation problems in the direction i.

Definition 5.7 Let $i \in \{1,\dots,n\}$ be the dimension, or informally the direction, along which we follow the grid lines. Along each grid line with fixed multi-index $0 \le \alpha^{\neg i} \le \alpha^{\neg i}_{\max}$, a univariate interpolation problem arises with the *univariate data sets* $\mathcal{D}^i_{\alpha^{\neg i}}$ given as

$$\mathcal{D}^i_{\alpha^{\neg i}} := \left\{ \left(x_{\alpha^i}, d_{\alpha^i|\alpha^{\neg i}} \right) \in \mathbb{R}\times\mathbb{R} \;\middle|\; 0 \le \alpha^i \le \alpha^i_{\max} \right\}. \tag{5.4}$$

For our numerical results in Section 5.5, we have chosen the method presented in [49] as the univariate interpolation method.

As mentioned in our literature review in Section 5.2, numerous univariate shape-preserving interpolation methods already exist. Therefore, we do not develop our own method to solve the univariate interpolation problems. Instead, we use an existing method. The choice of the univariate interpolation method affects

- which shapes are preserved,
- the smoothness order,
- whether the multivariate interpolation is local,
- and the computational load.

From now on, we thus assume that a univariate interpolation

$$p^i_{\alpha^{\neg i}} : \mathbb{R} \to \mathbb{R}$$

of $\mathcal{D}^i_{\alpha^{\neg i}}$ can be computed for all $i \in \{1,\dots,n\}$, $0 \le \alpha^{\neg i} \le \alpha^{\neg i}_{\max}$.

5.3.2. Coons' patches

The first step in our method, i.e., the interpolation of the univariate data sets, results in a network of curves. Figure 5.4 shows such a curve network in the bivariate case.

The idea behind our multivariate method is to achieve SP2 shape-preservation by producing an interpolation that matches the univariate interpolations along the grid lines, thereby inheriting the shape-preservation of the univariate interpolations. To achieve this, we need a blending scheme that smoothly blends the curves of the curve network while preserving their values on the grid lines.

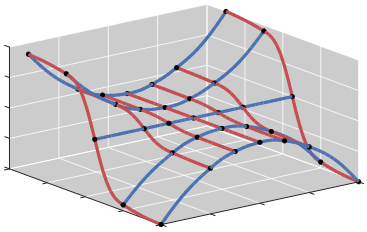


Figure 5.4.: A curve network resulting from the interpolation of the univariate data sets for an illustrative bivariate problem.

A similar problem – in a bivariate setting – arises in car design. Designers prescribe the shape of the car at specific points by specifying feature curves. The surface of the car should then match these feature curves and fill the space between them in a natural-looking way. A tool to solve such problems was developed in the automotive context by Coons, who consulted for Ford [71, p. 399]. Today, this tool is known as Coons' patches. Coons' patches were first introduced in [46]. In this subsection, we present the idea behind Coons' patches in the bivariate case. For our method, we extend the technique to the multivariate case in Subsection 5.3.3. For further reading on Coons' patches, we refer to [12, 13] and [71, Chapter 22].

[71]: Farin (2002), *Curves and Surfaces for CAGD*

[46]: Coons (1964), *Surfaces for computer-aided design*

[12]: Barnhill (1977), "Representation and Approximation of Surfaces"
[13]: Barnhill (1982), "Coon's patches"

Curve network To keep the explanation of Coons' patches simple, we focus on a single two-dimensional rectangle $Q := [s_0, s_1] \times [t_0, t_1] \subset \mathbb{R}^2$. Together with the rectangle Q we consider the four functions

$$\begin{aligned} h_k &: [s_0, s_1] \subset \mathbb{R} \to \mathbb{R}, \quad s \mapsto h_k(s), \quad k \in \{0, 1\}, \\ v_k &: [t_0, t_1] \subset \mathbb{R} \to \mathbb{R}, \quad t \mapsto v_k(t), \quad k \in \{0, 1\}, \end{aligned} \tag{5.5}$$

with

$$\begin{aligned} h_1(s_0) &= d_{01} = v_0(t_1), \quad h_1(s_1) = d_{11} = v_1(t_1), \\ h_0(s_0) &= d_{00} = v_0(t_0), \quad h_0(s_1) = d_{10} = v_0(t_0), \end{aligned} \tag{5.6}$$

for data points $d_{00}, d_{01}, d_{10}, d_{11} \in \mathbb{R}$. The functions and data points are visualized in Figure 5.5. The goal is to find a surface

$$p : Q \subset \mathbb{R}^2 \to \mathbb{R}, \quad (s, t) \mapsto p(s, t)$$

that is equal to the functions h_k, v_k, $k \in \{0, 1\}$ on the respective edge. In other words, p should satisfy

$$\begin{aligned} p(s, 0) &= h_0(s), \quad p(s, 1) = h_1(s), \quad \text{for all } s \in [s_0, s_1], \\ p(0, t) &= v_0(t), \quad p(1, t) = v_1(t), \quad \text{for all } t \in [t_0, t_1]. \end{aligned} \tag{5.7}$$

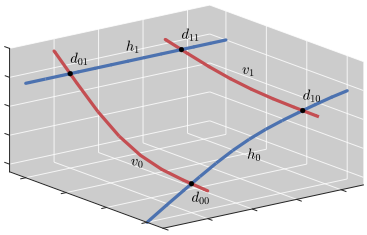


Figure 5.5.: The functions h_k correspond to the horizontal grid lines, while v_k correspond to the vertical grid lines, as illustrated.

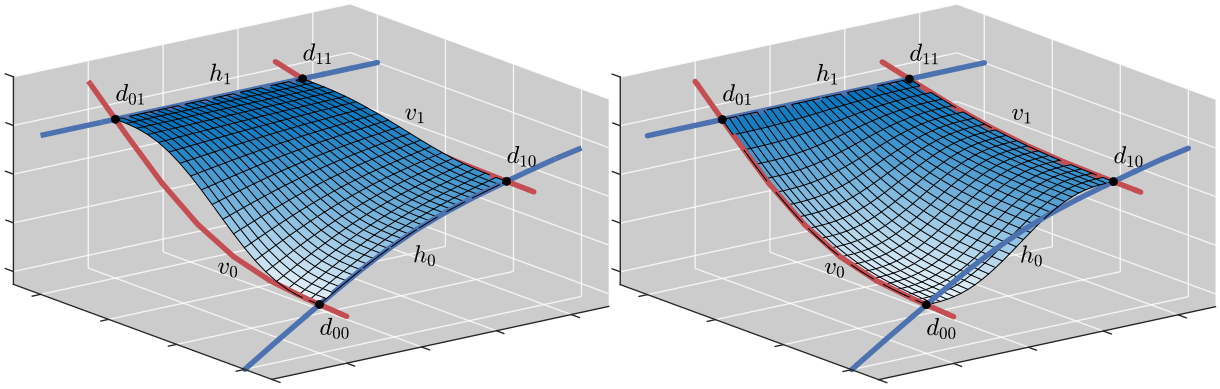
Lofted surfaces As an intermediate step to construct the surface p, we blend the parallel functions into each other, i.e., h_0 into h_1 and v_0 into v_1. This is done using a blending function w. We define blending functions as follows.

Definition 5.8 We call a function $w : [0, 1] \subset \mathbb{R} \to [0, 1] \subset \mathbb{R}$ a *blending function* if it satisfies

$$w(0) = 0, \quad w(1) = 1 \tag{5.8}$$

and is strictly monotously increasing.

We will discuss some common choices for the blending function together with the resulting variant of the Coons' patches in Example 5.3.



(a) Lofted surface p_h. (b) Lofted surface p_v.

Figure 5.6.: Lofted surfaces for an illustrative biviarate problem with the weight function $w(z) = 10z^3 - 15z^4 + 6z^5$.

Moreover, we say that a *blending function is of smoothness order* $q \in \mathbb{N}_0$ if $p \in C^q([0,1],[0,1])$ and additionally

$$\frac{\mathrm{d}^k}{\mathrm{d}^k z} w(0) = 0, \quad \frac{\mathrm{d}^k}{\mathrm{d}^k z} w(1) = 0 \quad \text{for all } k \in \mathbb{N} \text{ with } k \leq q. \tag{5.9}$$

Performing the mentioned blending of parallel functions yields us the two, so called lofted, surfaces $p_h, p_v : Q \subset \mathbb{R}^2 \to \mathbb{R}$ defined by

$$\begin{aligned} p_h(s,t) &:= \left(1 - w\left(\frac{t-t_0}{t_1-t_0}\right)\right) h_0(s) + w\left(\frac{t-t_0}{t_1-t_0}\right) h_1(s), \\ p_v(s,t) &:= \left(1 - w\left(\frac{s-s_0}{s_1-s_0}\right)\right) v_0(t) + w\left(\frac{s-s_0}{s_1-s_0}\right) v_1(t). \end{aligned}$$

The lofted surface p_h is now equal to h_0 and h_1 on the respective horizontal edge of the rectangle Q, but not equal to v_0 and v_1 on the other two sides. The same holds for p_v just with the sides switched. This can also be seen in our illustrative example shown in Figure 5.6.

Correction surface Let us examine the difference between each lofted surface and the univariate functions on the sides where they do not match the curve network. For p_h on the edge $\{s_0\} \times [t_0, t_1]$, we have

$$\begin{aligned} &v_0(t) - p_h(s_0,t) \\ &\quad = v_0(t) - \left(1 - w\left(\frac{t-t_0}{t_1-t_0}\right)\right) h_0(s_0) - w\left(\frac{t-t_0}{t_1-t_0}\right) h_1(s_0) \\ &\quad = v_0(t) - \left(1 - w\left(\frac{t-t_0}{t_1-t_0}\right)\right) d_{00} - w\left(\frac{t-t_0}{t_1-t_0}\right) d_{01}. \end{aligned} \tag{5.10}$$

In the second step, we have used the relation between the curves and the data points given in Equation (5.6).

Equation (5.10) is not very insightful so far, but if we exploit that

$$w\left(\frac{s_0 - s_0}{s_1 - s_0}\right) = 0,$$

we can expand Equation (5.10) to

$$\begin{aligned} & v_0(t) - p_h(s_0, t) \\ & \quad = \left(1 - w\left(\frac{s_0 - s_0}{s_1 - s_0}\right)\right) v_0(t) + w\left(\frac{s_0 - s_0}{s_1 - s_0}\right) v_1(t) \\ & \qquad - \begin{pmatrix} 1 - w\left(\frac{s_0 - s_0}{s_1 - s_0}\right) \\ w\left(\frac{s_0 - s_0}{s_1 - s_0}\right) \end{pmatrix}^T \begin{pmatrix} d_{00} & d_{01} \\ d_{10} & d_{11} \end{pmatrix} \begin{pmatrix} 1 - w\left(\frac{t - t_0}{t_1 - t_0}\right) \\ w\left(\frac{t - t_0}{t_1 - t_0}\right) \end{pmatrix} \\ & \quad = p_v(s_0, t) - p_{hv}(s_0, t), \end{aligned} \tag{5.11}$$

where we have introduced the correction surface p_{hv} defined as

$$p_{hv}(s,t) := \begin{pmatrix} 1 - w\left(\frac{s - s_0}{s_1 - s_0}\right) \\ w\left(\frac{s - s_0}{s_1 - s_0}\right) \end{pmatrix}^T \begin{pmatrix} d_{00} & d_{01} \\ d_{10} & d_{11} \end{pmatrix} \begin{pmatrix} 1 - w\left(\frac{t - t_0}{t_1 - t_0}\right) \\ w\left(\frac{t - t_0}{t_1 - t_0}\right) \end{pmatrix}.$$

Definition of Coons' patches If we proceed analogously for the other sides and then for the other lofted surface p_v, we find that

$$v_1(t) - p_h(s_1, t) = p_v(s_1, t) - p_{hv}(s_1, t), \tag{5.12}$$
$$h_0(s) - p_h(s, t_0) = p_v(s, t_0) - p_{hv}(s, t_0), \tag{5.13}$$
$$h_1(s) - p_h(s, t_1) = p_v(s, t_1) - p_{hv}(s, t_1). \tag{5.14}$$

From Equation (5.11) - Equation (5.14) we derive that adding the other lofting surface and subtracting the correction surface works as a remedy for all sides of the rectangle, which amounts to the definition of Coons' patches.

Definition 5.9 The *Generalized Coons' patch* p with a blending function w for the rectangle Q with the curve network h_k, v_k, $k \in \{0, 1\}$ and data points $d_{00}, d_{01}, d_{10}, d_{11} \in \mathbb{R}$ as related via Equation (5.5) and Equation (5.6) is defined by

$$p(s,t) := p_h(s,t) + p_v(s,t) - p_{hv}(s,t).$$

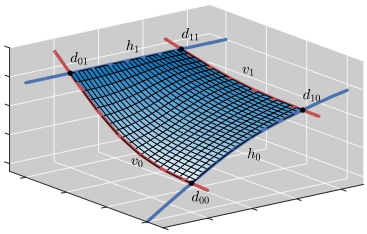


Figure 5.7.: Coons' patch for an illustrative bivariate problem with the weight function $w(z) = 10z^3 - 15z^4 + 6z^5$.

That the generalized Coons' patches are indeed equal to the curve network on the grid lines, i.e., satisfy condition (5.7), is a direct consequence of the property (5.8) of the blending functions and Equation (5.6).

By not specifying a particular blending function w, we have introduced generalized Coons' patches. In the following Example 5.3, we mention popular choices for the blending function.

▪ **Example 5.3** Common choices for the blending function w in Definition 5.9 include the following.

- Choosing $w(z) = z$ leads to a bilinearly blended Coons' patch, which was also the first type of Coons' patches to be developed and was originally presented in [46]. This linear blending function has the smoothness order $q = 0$ as

$$\frac{\mathrm{d}}{\mathrm{d}z} w(0) = \frac{\mathrm{d}}{\mathrm{d}z} w(1) = 1 \neq 0.$$

[46]: Coons (1964), *Surfaces for computer-aided design*

[71]: Farin (2002), *Curves and Surfaces for CAGD*

- Choosing $w(z) = 3z^2 - 2z^3$ leads to partially bicubically blended Coons' patches, cf. [71, Section 22.2]. This cubic Hermite polynomial blending function has a smoothness order $q = 1$.
- Choosing $w(z) = 10z^3 - 15z^4 + 6z^5$ leads to partially biquintically blended Coons' patches. This blending function has a smoothness order $q = 2$, which in combination with twice continuously differentiable univariate interpolations leads to a C^2 multivariate interpolation in our method and is used to compute the numerical results presented in Section 5.5.

▪

The zero corner twist issue applies only to Coons' patches with blending functions of smoothness order $q > 0$. For bilinearly blended Coons' patches, the disadvantage is that the interpolation surface is not differentiable across the rectangle boundaries, cf. [71, Section 22.2].

[71]: Farin (2002), *Curves and Surfaces for CAGD*

Zero corner twist issue Unfortunately, Coons' patches have one significant disadvantage: the mixed second derivatives at the corners of the rectangle are zero, i.e.,

$$\frac{\partial^2}{\partial s \partial t} p(s,t) = 0, \quad \text{for } s \in \{s_0, s_1\}, t \in \{t_0, t_1\}.$$

This deficiency is known as the zero corner twist. To avoid this issue, it is necessary to prescribe the cross-boundary derivatives, cf. [71, Chapter 22.2]. Much research has been conducted in this direction. However, for our method, we have decided to accept this shortcoming in exchange for the advantage of not requiring cross-boundary derivatives. This allows us to easily extend Coons' patches to the multivariate case. It also facilitates the use of our method in real engineering applications. This is because in engineering, the task of measuring the data is often decoupled from the use of the data. As a result, cross-boundary derivatives are typically not built into the data set. The situation is even more challenging when the data is automatically processed, e.g. on embedded hardware, for potentially different purposes.

5.3.3. Blending the univariate results together

In the first step of our proposed method, we interpolate all the univariate data sets as described in Subsection 5.3.1. This step results in a curve network. Unlike the situation described for Coons' patches in Subsection 5.3.2, this curve network spans all directions $i \in \{1, \dots, n\}$ rather than just two. Nevertheless, we can generalize the idea of Coons' patches for the bivariate case to the multivariate case. Essentially, we weight the univariate interpolations and data points such that their influence diminishes towards the parallel edges and other corners of the n-dimensional hypercubes of the grid $\mathcal{N}$.

Reminder: Interpolation p

The interpolation p is a function $p \in C^q(\mathbb{R}^n, \mathbb{R})$ that satisfies $p(x_\alpha) = d_\alpha$, for all $0 \le \alpha \le \alpha_{\max}$, cf. Definition 5.4.

In our method, we define the interpolation p of the data set $\mathcal{D}$ piecewise over the n-dimensional hyperrectangles Q_α. Specifically, we set

$$p(x) = p_\alpha(x), \quad \text{for all } x \in Q_\alpha, \tag{5.15}$$

for all $0 \le \alpha_{\max} - 1$, where Q_α is given by the following Definition 5.10.

Definition 5.10 The *n-dimensional hyperrectangles Q_α of the node grid $\mathcal{N}$* are defined as

$$Q_\alpha := \bigtimes_{i=1}^{n} \left[x^i_{\alpha^i}, x^i_{\alpha^i+1} \right].$$

for all $0 \leq \alpha_{\max} - 1$.

The construction of the functions p_α is the key task in this subsection. To define p_α concisely, we introduce the mappings I_0, I_1.

Definition 5.11 The mappings

$$I_0, I_1 \colon \bigcup_{i=1}^{n} \{0,1\}^i \to \mathcal{P}(\{1,\dots,n\})$$

are defined for a multi-index $\beta = \left(\beta^{i_1}, \dots, \beta^{i_m}\right)^T \in \{0,1\}^m$ by

$$\begin{aligned} I_0(\beta) &:= \left\{ j \in \{i_1,\dots,i_m\} \,\middle|\, \beta^j = 0 \right\}, \\ I_1(\beta) &:= \left\{ j \in \{i_1,\dots,i_m\} \,\middle|\, \beta^j = 1 \right\}. \end{aligned} \tag{5.16}$$

Definition: Power set $\mathcal{P}$

$\mathcal{P}(\{1,\dots,n\})$ is the *power set* of $\{1,\dots,n\}$.

In simple terms, I_0, I_1 provide the subsets of directions whose corresponding entries in the binary multi-index β are 0 and 1, respectively.

Moreover, we introduce the abbreviation

$$w^i_{\alpha^i}(x^i) := w\left(\frac{x^i - x^i_{\alpha^i}}{x^i_{\alpha^i+1} - x^i_{\alpha^i}} \right) \tag{5.17}$$

for all $i \in \{1,\dots,n\}$, $0 \leq \alpha^i \leq \alpha^i_{\max} - 1$. Using this, we define $p_\alpha(x)$.

Definition 5.12 The function $p_\alpha(x)$ is computed from the univariate interpolations $p^i_{\alpha^{\neg i}+\beta^{\neg i}}$, $\beta^{\neg i} \in \{0,1\}^{n-1}$, $i \in \{1,\dots,n\}$ and the data points $d_{\alpha+\beta}$, $\beta \in \{0,1\}^n$ that are adjacent to the hyperrectangle Q_α by generalizing the idea of Coons' patches to the multivariate case. Specifically,

$$\begin{aligned} p_\alpha(x) := &\sum_{i=1}^{n} \sum_{\beta^{\neg i} \in \{0,1\}^{n-1}} w_\alpha(\beta^{\neg i}, x)\, p^i_{\alpha^{\neg i}+\beta^{\neg i}}(x^i) \\ &- (n-1) \sum_{\beta \in \{0,1\}^n} w_\alpha(\beta, x)\, d_{\alpha+\beta}, \end{aligned} \tag{5.18}$$

The factor $(n-1)$ in front of the data points arises because n univariate interpolations meet at each corner of the hyperrectangle. As we sum up these univariate interpolations, we need to remove $(n-1)$ times the data points.

where the collective weight function

$$w_\alpha(\cdot,\cdot) \colon \bigcup_{k=1}^{n} \{0,1\}^k \times \mathbb{R}^n \to \mathbb{R}$$

is defined for a binary multi-index $\gamma \in \{0,1\}^k$ and $x \in Q_\alpha$ by

$$w_\alpha(\gamma, x) := \left(\prod_{j \in I_0(\gamma)} \left(1 - w^j_{\alpha^j}(x^j)\right) \right) \left(\prod_{j \in I_1(\gamma)} w^j_{\alpha^j}(x^j) \right). \tag{5.19}$$

We summarize our novel smooth multivariate shape-preserving interpolation method in Algorithm 5.1.

Algorithm 5.1: Smooth multivariate shape-preserving interpolation.

Input:

Data:

- One-dimensional node sets $\mathcal{N}^i,\ i \in \{1,\dots,n\}$, cf. eq. (5.2)
- Data set $\mathcal{D}$, cf. eq. (5.3)

User choices:

- Univariate shape-preserving interpolation method that yields interpolations $p^i_{\alpha^{\neg i}} \in C^q(\mathbb{R},\mathbb{R})$ of the univariate data sets $\mathcal{D}^i_{\alpha^{\neg i}}$ for all $0 \le \alpha^{\neg i} \le \alpha^{\neg i}_{\max},\ i \in \{1,\dots,n\}$
- Blending function w with smoothness of order q, cf. Def. 5.8

From application:

- Evaluation point $x \in \bigtimes_{i=1}^n \bar{\mathcal{N}}^i$

Output: $p(x)$, where p is an interpolation of the data set $\mathcal{D}$ with smoothness of order q that is shape-preserving in the SP2 sense, cf. Def. 5.4 and Def. 5.6

Optional preparational step:

0 $p^i_{\alpha^{\neg i}} \leftarrow$ Precompute all univariate interpolations for $0 \le \alpha^{\neg i} \le \alpha^{\neg i}_{\max},\ i \in \{1,\dots,n\}$

Main method:

1 $\alpha \leftarrow$ Find hyperrectangle $Q_\alpha \ni x$

2 $p^i_{\alpha^{\neg i}+\beta^{\neg i}}(x^i) \leftarrow$ Evaluate univariate interpolations adjacent to Q_α for $\beta^{\neg i} \in \{0,1\}^{n-1},\ i \in \{1,\dots,n\}$

3 $w^i_{\alpha^i}(x^i) \leftarrow$ Compute weights according to eq. (5.17) for $i \in \{1,\dots,n\}$

4 $p_\alpha(x) \leftarrow$ Evaluate eq. (5.18) and eq. (5.19)

5 $p(x) \leftarrow p_\alpha(x)$

Finally, we state that our novel interpolation method is indeed an interpolation and is shape-preserving in the SP2 sense, even in the multivariate case, as shown in Theorem 5.1. As mentioned at the beginning of the chapter on p. 78, a proof that our described method yields an interpolation with smoothness of order q in the multivariate case is still in progress and is intended to be published elsewhere.

Theorem 5.1 The function $p : \mathbb{R}^n \to \mathbb{R}$ constructed as described in this subsection, particularly according to Equation (5.15) and Equation (5.18), is an interpolation of the data set $\mathcal{D}$, as defined in Definition 5.3, according to Definition 5.4. Moreover, p is shape-preserving in the SP2 sense, as defined in Definition 5.6.

Proof. The statement is split into Theorem 5.2 and Corollary 5.1, whose proofs are given in Section 5.4. ∎

5.4. Proof of the interpolation and shape-preservation property

To prove that our proposed method yields an interpolation that is shape-preserving in the SP2 sense, i.e., to establish Theorem 5.1, we primarily need to demonstrate the shape-preservation property, which is stated in Theorem 5.2. The interpolation property then follows directly from the interpolation property of the univariate interpolations, as shown in Corollary 5.1.

In the first step, in Subsection 5.4.1, we establish auxiliary results concerning the product of blending functions, which, in essence, recover the intuition behind our method. Using these auxiliary results, we prove the shape-preservation property in Subsection 5.4.2. The interpolation property is demonstrated in Subsection 5.4.3.

Reminder: Intuition of our method

We weight the univariate interpolations and data points such that their influence diminishes towards the parallel edges and other corners, respectively, see p. 90.

5.4.1. Auxiliary results

The main task is to investigate the properties of the collective weight functions $w_\alpha(\cdot,\cdot)$, as defined in Equation (5.19), for evaluation points x located on the edges of the hyperrectangle. We begin by defining the edges of the hyperrectangles and the grid lines of the node set.

Definition 5.13 We denote by $K^i_{\alpha^i|\alpha^{\neg i}+\beta^{\neg i}}$ with $\beta^{\neg i} \in \{0,1\}^{n-1}$ the *edges of the hyperrectangle* Q_α, i.e.,

$$K^i_{\alpha^i|\alpha^{\neg i}+\beta^{\neg i}} := \bigtimes_{j=1}^{i-1}\left\{x^j_{\alpha^j+\beta^j}\right\} \times \left[x^i_{\alpha^i}, x^i_{\alpha^i+1}\right] \times \bigtimes_{j=i+1}^{n}\left\{x^j_{\alpha^j+\beta^j}\right\}.$$

A *grid line of the node set* is denoted for a fixed direction $i \in \{1,\dots,n\}$ and $0 \le \alpha^{\neg i} \le \alpha^{\neg i}_{\max}$ by $L^i_{\alpha^{\neg i}}$ and defined as

$$\begin{aligned} L^i_{\alpha^{\neg i}} &:= \bigtimes_{j=1}^{i-1}\left\{x^j_{\alpha^j}\right\} \times \left[x^i_0, x^i_{\alpha^i_{\max}}\right] \times \bigtimes_{j=i+1}^{n}\left\{x^j_{\alpha^j}\right\} \\ &= \bigcup_{\alpha^i=0}^{\alpha^i_{\max}-1} K^i_{\alpha^i|\alpha^{\neg i}}. \end{aligned}$$

The nodes and data points along the edge $K^i_{\alpha^i|\alpha^{\neg i}+\beta^{\neg i}}$ belong to the univariate data set $\mathcal{D}^i_{\alpha^{\neg i}+\beta^{\neg i}}$, and accordingly, the univariate interpolation $p^i_{\alpha^{\neg i}+\beta^{\neg i}}$ is applied on the edge $K^i_{\alpha^i|\alpha^{\neg i}+\beta^{\neg i}}$.

Our first auxiliary result, Lemma 5.1, states that the weight of $p^i_{\alpha^{\neg i}+\beta^{\neg i}}$ is one on the edge $K^i_{\alpha^i|\alpha^{\neg i}+\beta^{\neg i}}$ but zero on parallel edges.

Lemma 5.1 Let $i \in \{1,\dots,n\}$ and $0 \le \alpha \le \alpha_{\max}$, with $\alpha^{\neg i}$ constructed from α, and $\beta^{\neg i} \in \{0,1\}^{n-1}$. For all $x \in K^i_{\alpha^i|\alpha^{\neg i}+\beta^{\neg i}}$ it holds that

$$w_\alpha\left(\beta^{\neg i}, x\right) = 1, \tag{5.20}$$

but for $\gamma^{\neg i} \in \{0,1\}^{n-1}$ with $\beta^{\neg i} \neq \gamma^{\neg i}$, we have

$$w_\alpha\left(\gamma^{\neg i}, x\right) = 0. \tag{5.21}$$

Proof. We first prove Equation (5.20). Since $x \in K^i_{\alpha^i|\alpha^{\neg i}+\beta^{\neg i}}$, we have

$$\begin{aligned} x^j &= x^j_{\alpha^j}, \quad \forall j \in I_0\left(\beta^{\neg i}\right), \\ x^j &= x^j_{\alpha^j+1}, \quad \forall j \in I_1\left(\beta^{\neg i}\right). \end{aligned}$$

The immediate consequences are

Here, we use Equation (5.17) and Equation (5.8).

$$w^j_{\alpha^j}\left(x^j\right) = w^j_{\alpha^j}\left(x^j_{\alpha^j}\right) = w\left(\frac{x^j_{\alpha^j} - x^j_{\alpha^j}}{x^j_{\alpha^j+1} - x^j_{\alpha^j}}\right) = 0$$

and accordingly

$$1 - w^j_{\alpha^j}\left(x^j\right) = 1$$

for all $j \in I_0\left(\beta^{\neg i}\right)$, and

$$w^j_{\alpha^j}\left(x^j\right) = w^j_{\alpha^j}\left(x^j_{\alpha^j+1}\right) = w\left(\frac{x^j_{\alpha^j+1} - x^j_{\alpha^j}}{x^j_{\alpha^j+1} - x^j_{\alpha^j}}\right) = 1$$

for all $j \in I_1\left(\beta^{\neg i}\right)$. Therefore, we have

$$w_\alpha\left(\beta^{\neg i}, x\right) = \left(\prod_{j \in I_0\left(\beta^{\neg i}\right)} \left(1 - w^j_{\alpha^j}\left(x^j\right)\right)\right)\left(\prod_{j \in I_1\left(\beta^{\neg i}\right)} w^j_{\alpha^j}\left(x^j\right)\right) = 1,$$

which is Equation (5.20).

Now, we prove Equation (5.21). Let $k \in \{1,\dots,n\} \setminus \{i\}$ be one of the indices with $\beta^k \neq \gamma^k$. If $\gamma^k = 0$ and $\beta^k = 1$, we have $x^k = x^k_{\alpha^k+1}$ and thus

The intermediate step here is the same as above.

$$1 - w^k_{\alpha^k}\left(x^k\right) = 1 - w^k_{\alpha^k}\left(x^k_{\alpha^k+1}\right) = 0. \tag{5.22}$$

As $k \in I_0\left(\gamma^{\neg i}\right)$, we obtain

$$w_\alpha\left(\gamma^{\neg i}, x\right) = \underbrace{\left(1 - w^k_{\alpha^k}\left(x^k\right)\right)}_{\overset{(5.22)}{=}0}\left(\prod_{j \in I_0\left(\gamma^{\neg i}\right)\setminus\{k\}} \left(1 - w^j_{\alpha^j}\left(x^j\right)\right)\right)\left(\prod_{j \in I_1\left(\gamma^{\neg i}\right)} w^j_{\alpha^j}\left(x^j\right)\right) = 0.$$

In this case, we have $w^k_{\alpha^k}\left(x^k\right) = 0$ and $k \in I_1\left(\gamma^{\neg i}\right)$.

The case where $\gamma^{\neg k} = 1$ and $\beta^{\neg k} = 0$ works analogously. ∎

The following Lemma 5.2 addresses the weight of the univariate interpolations in the remaining directions.

Lemma 5.2 Let $i \in \{1,\dots,n\}$ and $0 \le \alpha \le \alpha_{\max}$, with $\alpha^{\neg i}$ constructed from α, and $\beta^{\neg i} \in \{0,1\}^{n-1}$. For all $x \in K^i_{\alpha^i|\alpha^{\neg i}+\beta^{\neg i}}$, $k \in \{1,\dots,n\}\setminus\{i\}$ and $\gamma^{\neg k} \in \{0,1\}^{n-1}$ it holds that

(i) $w_\alpha(\gamma^{\neg k}, x) = 1 - w^i_{\alpha^i}(x^i)$ if $\beta^j = \gamma^j$ for all $j \in \{1,\dots,n\}\setminus\{i,k\}$ and $\gamma^i = 0$,
(ii) $w_\alpha(\gamma^{\neg k}, x) = w^i_{\alpha^i}(x^i)$ if $\beta^j = \gamma^j$ for all $j \in \{1,\dots,n\}\setminus\{i,k\}$ and $\gamma^i = 1$,
(iii) $w_\alpha(\gamma^{\neg k}, x) = 0$ if $\beta^j \neq \gamma^j$ for a $j \in \{1,\dots,n\}\setminus\{i,k\}$.

Proof. We prove the three statements (i)–(iii) one by one:
(i): Since $\gamma^i = 0$, we have $i \in I_0(\gamma^{\neg k})$, and therefore

$$w_\alpha(\gamma^{\neg k}, x) = \left(1 - w^i_{\alpha^i}(x^i)\right)\left(\prod_{j\in I_0(\gamma^{\neg k})\setminus\{i\}}\left(1 - w^j_{\alpha^j}(x^j)\right)\right)\left(\prod_{j\in I_1(\gamma^{\neg k})\setminus\{i\}} w^j_{\alpha^j}(x^j)\right). \tag{5.23}$$

For all $j \in I_0(\gamma^{\neg k})\setminus\{i\}$, we have $\beta^j = \gamma^j = 0$, and thus $x^j = x^j_{\alpha^j}$, which implies $1 - w^j_{\alpha^j}(x^j) = 1$. For all $j \in I_1(\gamma^{\neg k})\setminus\{i\}$, we have $\beta^j = \gamma^j = 1$, and thus $x^j = x^j_{\alpha^j+1}$, which implies $w^j_{\alpha^j}(x^j) = 1$. Therefore, we obtain

The steps to show $w^j_{\alpha^j}(x^j) = 0,1$ are the same as in the proof of Lemma 5.1.

$$\left(\prod_{j\in I_0(\gamma^{\neg k})\setminus\{i\}}\left(1 - w^j_{\alpha^j}(x^j)\right)\right)\left(\prod_{j\in I_1(\gamma^{\neg k})\setminus\{i\}} w^j_{\alpha^j}(x^j)\right) = 1.$$

Thus, Equation (5.23) simplifies to

$$w_\alpha(\gamma^{\neg k}, x) = 1 - w^i_{\alpha^i}(x^i).$$

(ii): The proof follows analogously to that of (i).

For (ii), we have $\gamma^i = 1$, $i \in I_1(\gamma^{\neg k})$, and thus $w^i_{\alpha^i}(x^i)$ replaces $\left(1 - w^i_{\alpha^i}(x^i)\right)$ in Equation (5.23).

(iii): We proceed similarly to the proof of Equation (5.21). Let

$$l \in \{1,\dots,n\}\setminus\{i,k\} \quad \text{such that} \quad \beta^l \neq \gamma^l.$$

If $\gamma^l = 0$ and $\beta^l = 1$, we obtain

$$1 - w^l_{\alpha^l}(x^l) = 0. \tag{5.24}$$

Since $l \in I_0(\gamma^{\neg i})$, we have

$$w_\alpha(\gamma^{\neg k}, x) = \underbrace{\left(1 - w^l_{\alpha^l}(x^l)\right)}_{\overset{(5.24)}{=}0}\left(\prod_{j\in I_0(\gamma^{\neg k})\setminus\{l\}}\left(1 - w^j_{\alpha^j}(x^j)\right)\right)\left(\prod_{j\in I_1(\gamma^{\neg k})} w^j_{\alpha^j}(x^j)\right) = 0.$$

■

Finally, we also examine the influence of the data points for $x \in K^i_{\alpha^i|\alpha^{\neg i}+\beta^{\neg i}}$. The result is stated in Lemma 5.3.

Lemma 5.3 Let $i \in \{1,\dots,n\}$ and $0 \le \alpha \le \alpha_{\max}$, with $\alpha^{\neg i}$ constructed from α, and $\beta^{\neg i} \in \{0,1\}^{n-1}$. For all $x \in K^i_{\alpha^i|\alpha^{\neg i}+\beta^{\neg i}}$ and $\gamma \in \{0,1\}^n$ it holds that

(i) $w_\alpha(\gamma, x) = 1 - w^i_{\alpha^i}(x^i)$ if $\beta^j = \gamma^j$ for all $j \in \{1,\dots,n\} \setminus \{i\}$ and $\gamma^i = 0$,
(ii) $w_\alpha(\gamma, x) = w^i_{\alpha^i}(x^i)$ if $\beta^j = \gamma^j$ for all $j \in \{1,\dots,n\} \setminus \{i\}$ and $\gamma^i = 1$,
(iii) $w_\alpha(\gamma, x) = 0$ if $\beta^j \neq \gamma^j$ for some $j \in \{1,\dots,n\} \setminus \{i\}$.

The difference from Lemma 5.2 is that we now consider the weight for the data points. Therefore, we use the index γ instead of $\gamma^{\neg k}$.

Proof. The proof is entirely analogous to that of Lemma 5.2. The only difference is that $\gamma^{\neg k}$ is replaced by γ throughout. ∎

5.4.2. Shape-preservation property

We can now prove the shape-preservation property of our method.

Theorem 5.2 Let $i \in \{1,\dots,n\}$ and $0 \le \tilde{\alpha}^{\neg i} \le \alpha^{\neg i}_{\max}$ be fixed, and let the function $p : \mathbb{R}^n \to \mathbb{R}$ be constructed as described in Section 5.3, particularly according to Equation (5.15) and Equation (5.18). For all $x \in L^i_{\tilde{\alpha}^{\neg i}}$, it holds that

$$p(x) = p^i_{\tilde{\alpha}^{\neg i}}(x^i), \tag{5.25}$$

which means that p is shape-preserving in the SP2 sense, as defined in Definition 5.6.

In simple terms, Theorem 5.2 states that the multivariate interpolation reduces to the univariate interpolations on the edges of the hyperrectangles. Since these are shape-preserving, the multivariate interpolation is shape-preserving in the SP2 sense.

Proof. Let $0 \le \alpha^i \le \alpha^i_{\max} - 1$ such that $x^i \in \left[x^i_{\alpha^i}, x^i_{\alpha^i+1}\right]$. Moreover, let $0 \le \alpha^{\neg i} \le \alpha^{\neg i}_{\max} - 1$ and $\beta^{\neg i} \in \{0,1\}^{n-1}$ such that

$$\tilde{\alpha}^{\neg i} = \alpha^{\neg i} + \beta^{\neg i}. \tag{5.26}$$

We abbreviate $\alpha := \alpha^i|\alpha^{\neg i}$. Then, we have that

$$x \in K^i_{\alpha^i|\alpha^{\neg i}+\beta^{\neg i}} \subset Q_\alpha.$$

According to Equation (5.15) we thus set

$$p(x) = p_\alpha(x),$$

where p_α is given as in Equation (5.18). For this proof, we rearrange

There are 2^{n-1} many hyperrectangles which contain the edge $K^i_{\alpha^i|\alpha^{\neg i}+\beta^{\neg i}}$. So far, it could be that p is thus not uniquely defined on the edge $K^i_{\alpha^i|\alpha^{\neg i}+\beta^{\neg i}}$. However, the following steps in the proof show that p is indeed uniquely defined, with the values as given in Equation (5.25), as the following steps work for all these hyperrectangles.

the terms in Equation (5.18) to

$$\begin{aligned} p_\alpha(x) = & \underbrace{\sum_{\gamma^{\neg i}\in\{0,1\}^{n-1}} w_\alpha\left(\gamma^{\neg i}, x\right) p^i_{\alpha^{\neg i}+\gamma^{\neg i}}\left(x^i\right)}_{=:S_1(x)} \\ & + \underbrace{\sum_{j\in\{1,\dots,n\}\setminus\{i\}} \sum_{\gamma^{\neg j}\in\{0,1\}^{n-1}} w_\alpha\left(\gamma^{\neg j}, x\right) p^j_{\alpha^{\neg j}+\gamma^{\neg j}}\left(x^j\right)}_{=:S_2(x)} \\ & - \underbrace{(n-1)\sum_{\gamma\in\{0,1\}^n} w_\alpha(\gamma, x) d_{\alpha+\gamma}}_{=:S_3(x)}. \end{aligned} \tag{5.27}$$

S_1 contains the information from all univariate interpolations in the fixed direction i and S_2 the information from all univariate interpolations in the orthogonal directions. S_3 the contains the influence of the data points.

We investigate the three summands S_1, S_2 and S_3 individually.

$\mathbf{S_1(x)}$: From Lemma 5.1 we know that $w_\alpha\left(\gamma^{\neg i}, x\right) = 1$ if $\gamma^{\neg i} = \beta^{\neg i}$ and that $w_\alpha\left(\gamma^{\neg i}, x\right) = 0$ if not. Therefore, S_1 collapses to

$$S_1(x) = p^i_{\alpha^{\neg i}+\beta^{\neg i}}\left(x^i\right). \tag{5.28}$$

That means that along the direction i only the interpolation of the data on the edge on which x is located has an influence.

$\mathbf{S_2(x)}$: Lemma 5.2 yields that

$$S_2(x) = \sum_{j\in\{1,\dots,n\}\setminus\{i\}} \left(1 - w^i_{\alpha^i}\left(x^i\right)\right) p^j_{\alpha^{\neg j}+\gamma^{\neg j}}\left(x^j\right) + w^i_{\alpha^i}\left(x^i\right) p^j_{\alpha^{\neg j}+\delta^{\neg j}}\left(x^j\right),$$

where $\beta^{\neg j}, \delta^{\neg j} \in \{0,1\}^{n-1}$ such that

$$\beta^k = \gamma^k = \delta^k \quad \text{for all } k \in \{1,\dots,n\}\setminus\{i,j\}, \tag{5.29a}$$
$$\gamma^i = 0, \tag{5.29b}$$
$$\delta^i = 1. \tag{5.29c}$$

As $x^j = x^j_{\alpha^j+\beta^j}$ for all $j \in \{1,\dots,n\}\setminus i$, the interpolation property of the univariate interpolations yields

$$\begin{aligned} p^j_{\alpha^{\neg j}+\gamma^{\neg j}}\left(x^j\right) &= p^j_{\alpha^{\neg j}+\gamma^{\neg j}}\left(x^j_{\alpha^j+\beta^j}\right) = d_{\alpha^j+\beta^j|\alpha^{\neg j}+\gamma^{\neg j}}, \\ p^j_{\alpha^{\neg j}+\delta^{\neg j}}\left(x^j\right) &= p^j_{\alpha^{\neg j}+\delta^{\neg j}}\left(x^j_{\alpha^j+\beta^j}\right) = d_{\alpha^j+\beta^j|\alpha^{\neg j}+\delta^{\neg j}}, \end{aligned}$$

which we can write by defining $\gamma^j := \beta^j, \delta^j := \beta^j$ as

The other components of γ and δ are determined by Equation (5.29).

$$\begin{aligned} p^j_{\alpha^{\neg j}+\gamma^{\neg j}}\left(x^j\right) &= d_{\alpha+\gamma}, \\ p^j_{\alpha^{\neg j}+\delta^{\neg j}}\left(x^j\right) &= d_{\alpha+\delta}. \end{aligned}$$

We thus have found

$$\begin{aligned} S_2(x) &= \sum_{j\in\{1,\dots,n\}\setminus\{i\}} \left(1 - w^i_{\alpha^i}\left(x^i\right)\right) d_{\alpha+\gamma} + w^i_{\alpha^i}\left(x^i\right) d_{\alpha+\delta} \\ &= (n-1)\left(\left(1 - w^i_{\alpha^i}\left(x^i\right)\right) d_{\alpha+\gamma} + w^i_{\alpha^i}\left(x^i\right) d_{\alpha+\delta}\right). \end{aligned} \tag{5.30}$$

In other words: As the components of x for these directions are equal to node points, the influence of these interpolations is equal to the one of the data points.

$S_3(x)$: We apply Lemma 5.3 to directly obtain

$$S_3(x) = (n-1)\left(\left(1 - w^i_{\alpha^i}(x^i)\right)d_{\alpha+\gamma} + w^i_{\alpha^i}(x^i)d_{\alpha+\delta}\right) = S_2. \tag{5.31}$$

Plugging in our findings Equation (5.28) - Equation (5.31) for S_1, S_2 and S_3 back into Equation (5.27) and recalling the multi-index relation (5.26) yields the final statement

$$\begin{aligned} p_\alpha(x) &= S_1(x) + S_2(x) - S_3(x) \\ &= S_1(x) + S_2(x) - S_2(x) = S_1(x) \\ &= p^i_{\alpha^{\neg i}+\beta^{\neg i}}(x^i) = p^i_{\tilde{\alpha}^{\neg i}}(x^i). \end{aligned}$$

■

5.4.3. Interpolation property

Corollary 5.1 Let the function $p : \mathbb{R}^n \to \mathbb{R}$ be constructed as described in Section 5.3, particularly according to Equation (5.15) and Equation (5.18). Then p is an interpolation of the data set $\mathfrak{D}$, i.e.,

$$p(x_\alpha) = d_\alpha$$

holds for all $0 \le \alpha \le \alpha_{\max}$, i.e., for all $x_\alpha \in \mathcal{N}$.

Proof. For all $i \in \{1, \dots, n\}$ we have $x_\alpha \in L^i_{\alpha^{\neg i}}$ for $\alpha^{\neg i}$ obtained from α by removing its i-th component. From Theorem 5.2 we thus get

$$p(x_\alpha) = p^i_{\alpha^{\neg i}}\left(x^i_{\alpha^i}\right).$$

As the univariate interpolations are interpolations of the univariate data sets, we have

$$p^i_{\alpha^{\neg i}}\left(x^i_{\alpha^i}\right) = d_{\alpha^i|\alpha^{\neg i}} = d_\alpha.$$

■

5.5. Numerical results

Attention:

Although the examples are abstracted, we intentionally present them in a way that prevents conclusions about their origin.

In this section, we apply our novel interpolation method to two abstracted examples resulting from an industrial collaboration. The first example is a 3D case, meaning its data set $\mathfrak{D}$ is a subset of $\mathbb{R}^3 \times \mathbb{R}$. The second example is a 4D case.

For both examples, we used the method presented in [49] to solve the arising univariate interpolation problems. The hyperparameters of the method were chosen such that the resulting multivariate interpolation is twice continuously differentiable.

[49]: Costantini (1988), "An algorithm for computing shape-preserving interpolating splines of arbitrary degree"

As a blending function, we use $w(z) = 10z^3 - 15z^4 + 6z^5$ which has the smoothness order $q = 2$.

Naturally, we cannot visualize the full multivariate interpolation in this document. Therefore, we present the interpolation results along slices with two-dimensional planes.

5.5.1. 3D example

In Figure 5.8, we show the evaluation of the multivariate interpolation p computed with our proposed method along a slice where one of the three free variables x^1, x^2, x^3 is fixed to a value in its respective node set. E.g., in Figure 5.8a, for a fixed $0 \leq \alpha^1 \leq \alpha^1_{\max}$, we visualize the set

$$\left\{ (x^2, x^3, p(x)) \;\middle|\; x = \begin{pmatrix} x^1 \\ x^2 \\ x^3 \end{pmatrix} \text{ with } x^1 = x^1_{\alpha^1} \in \mathcal{N}^1, (x^2, x^3) \in \bar{\mathcal{N}}^2 \times \bar{\mathcal{N}}^3 \right\}.$$

As confirmed by our theoretical findings in Theorem 5.1, our method produces an interpolation that preserves the shape properties of the univariate interpolations, such as monotonicity and convexity, while interpolating the data set.

As mentioned earlier, we think that our method preserves the smoothness order of the univariate interpolations if the blending functions have the same smoothness order. However, a proof of this claim in the multivariate setting is still under development. With the chosen univariate method and blending function, the interpolation for the 3D example should be twice continuously differentiable. Our primary interest in the plots for the first and second derivatives is to verify whether the derivatives appear continuous.

In Figure 5.9, we show the first derivative of the interpolation with respect to x^2 and x^3, while keeping x^1 fixed to $x^1 = x^1_{\alpha^1} \in \mathcal{N}^1$. These plots in Figure 5.9 should be compared with Figure 5.8a. The first derivatives appear continuous, as claimed for our method.

The second derivatives, shown in Figure 5.10, also appear continuous, supporting our hypothesis. The zero corner twists for the mixed second derivatives are visible in Figure 5.10a and Figure 5.10b. The non-mixed second derivatives do not vanish, as shown in Figure 5.10c.

Attention:

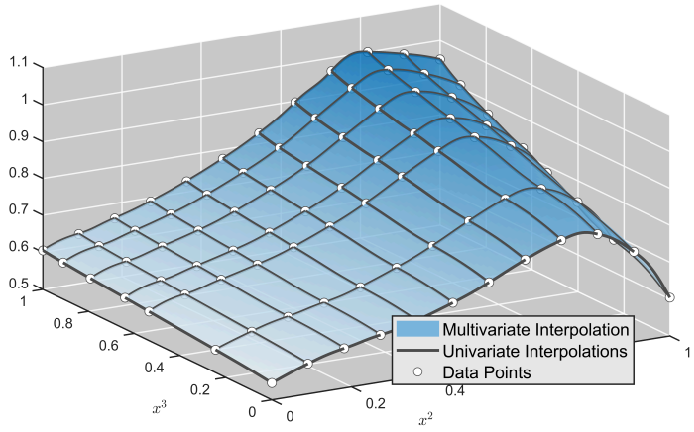
To obtain the derivative plots, we approximate the derivatives using finite differences at a large number of points. Unfortunately, scatterplots of these results are difficult to interpret. Therefore, we chose to plot surfaces instead. However, surface plots interpolate the calculated derivative results, potentially hiding discontinuities. Given the large number of evaluation points, we are confident that any discontinuities would manifest as steep changes in the surfaces.

Reminder: Zero corner twist

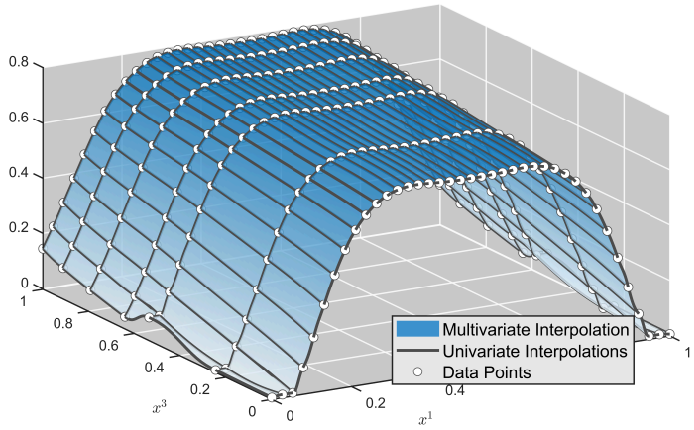
Zero corner twist means that the mixed second derivatives vanish at the corners of the hyperrectangles, cf. the paragraph about the zero corner twist on p. 90.

5.5.2. 4D example

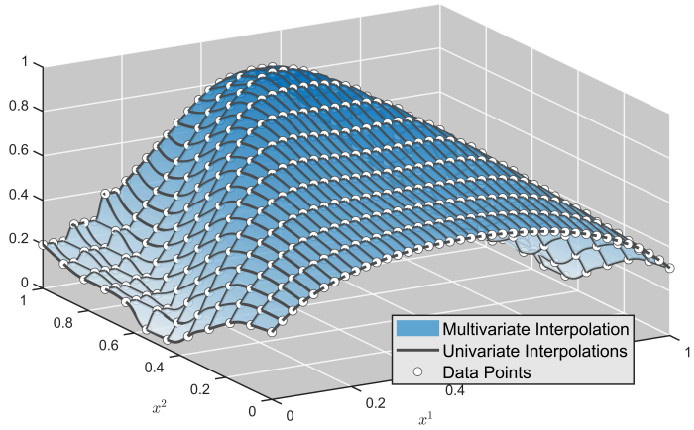
The following results of our method applied to a 4D example show the ability of our method to successfully perform shape-preserving multivariate interpolation. Similar to the 3D example results in Subsection 5.5.1, the following Figure 5.11 and Figure 5.12 show that the interpolation is indeed shape-preserving in the SP2 sense and appears to be twice continuously differentiable, but has zero corner twists.



(a) $x^1 = x^1_{\alpha^1} \in \mathcal{N}^1$ for a fixed $0 \leq \alpha^1 \leq \alpha^1_{\max}$.

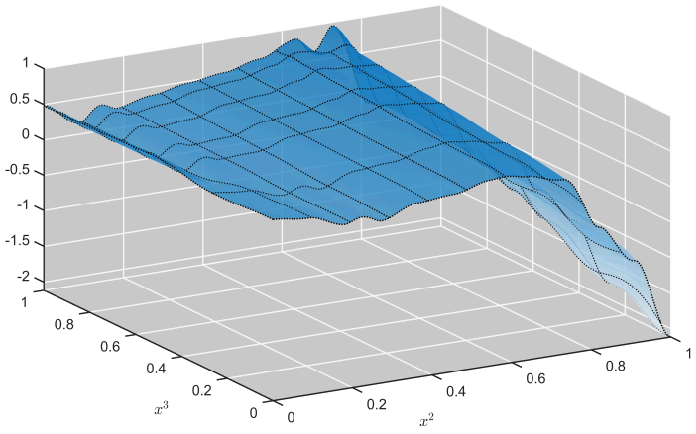


(b) $x^2 = x^2_{\alpha^2} \in \mathcal{N}^2$ for a fixed $0 \leq \alpha^2 \leq \alpha^2_{\max}$.

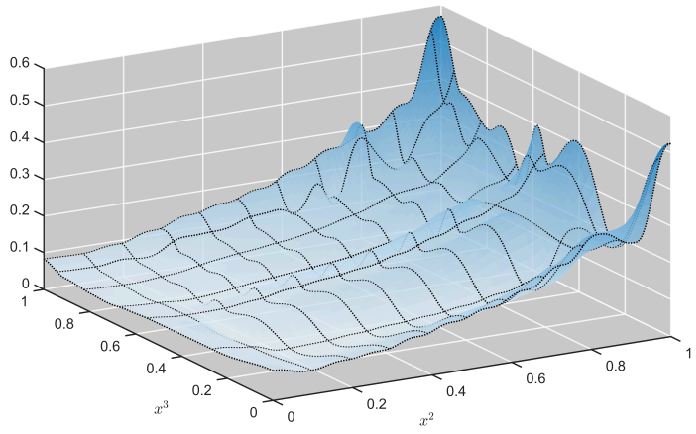


(c) $x^3 = x^3_{\alpha^3} \in \mathcal{N}^3$ for a fixed $0 \leq \alpha^3 \leq \alpha^3_{\max}$.

Figure 5.8.: Multivariate interpolation of a 3D example evaluated at slices where one of the three free variables x^1, x^2, x^3 is fixed to a value in its respective node set. The white dots represent the data points included in the slice. The dark gray lines are the results of the univariate interpolations of these data points, computed using the method presented in [49]. Our method produces an interpolation that preserves the shape properties of the univariate interpolations, such as monotonicity and convexity.

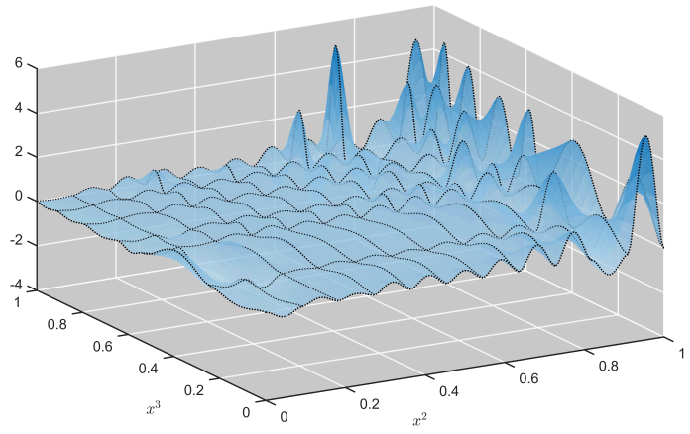


(a) First derivative w.r.t. x^2 with $x^1 = x^1_{\alpha^1} \in \mathcal{N}^1$ for a fixed $0 \leq \alpha^1 \leq \alpha^1_{\max}$.

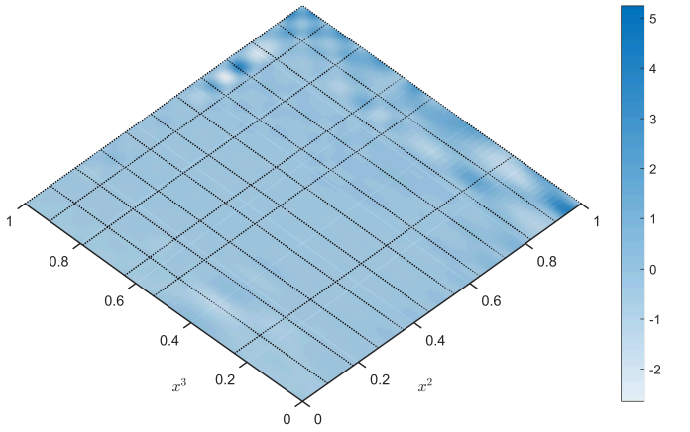


(b) First derivative w.r.t. x^3 with $x^1 = x^1_{\alpha^1} \in \mathcal{N}^1$ for a fixed $0 \leq \alpha^1 \leq \alpha^1_{\max}$.

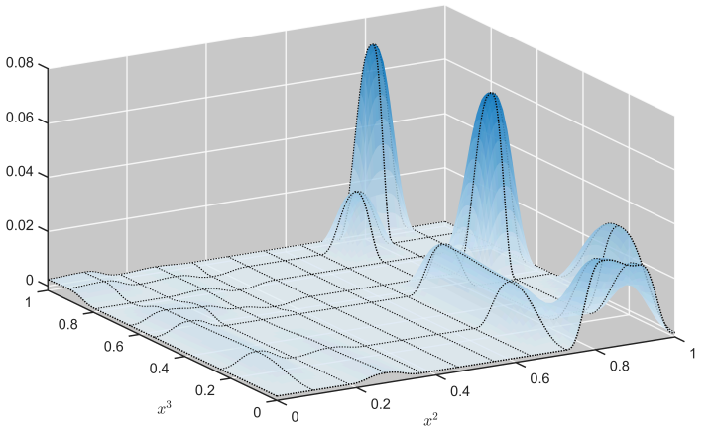
Figure 5.9.: Selected first derivatives of the multivariate interpolation of a 3D example evaluated at the slice used in Figure 5.8a. The dotted gray lines indicate the projection of the grid $\mathcal{N}$ onto the plotted surface. The first derivatives appear continuous, as claimed for our method.



(a) Mixed second derivative w.r.t. x^2 and then x^3 with $x^1 = x^1_{\alpha^1} \in \mathcal{N}^1$ for a fixed $0 \leq \alpha^1 \leq \alpha^1_{\max}$.

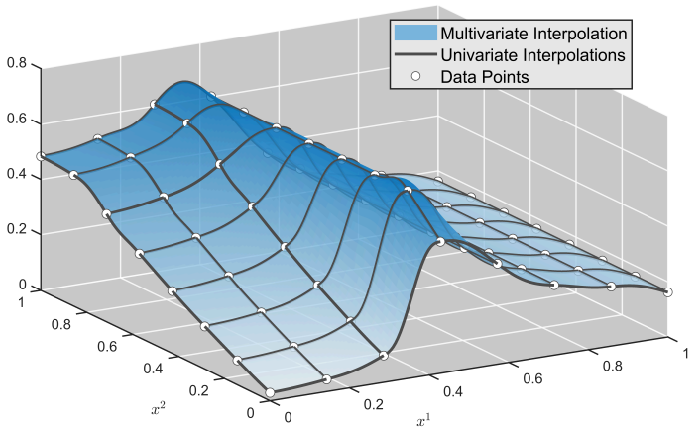


(b) Mixed second derivative w.r.t. x^2 and then x^3 with $x^1 = x^1_{\alpha^1} \in \mathcal{N}^1$ for a fixed $0 \leq \alpha^1 \leq \alpha^1_{\max}$ represented as heat map.

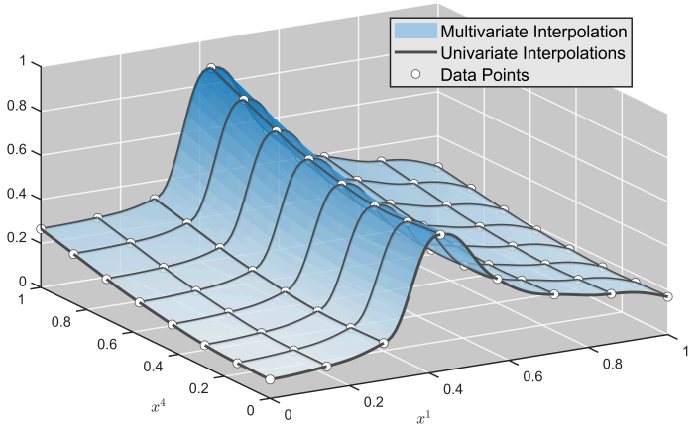


(c) Non-mixed derivative w.r.t. twice x^1 with $x^1 = x^1_{\alpha^1} \in \mathcal{N}^1$ for a fixed $0 \leq \alpha^1 \leq \alpha^1_{\max}$.

Figure 5.10.: Selected second derivatives of the multivariate interpolation of a 3D example evaluated at the slice used in Figure 5.8a. The dotted gray lines indicate the projection of the grid $\mathcal{N}$ onto the plotted surface. The second derivatives also appear continuous, supporting our hypothesis. The zero corner twists for the mixed second derivatives are visible. However, the non-mixed second derivatives do not vanish.

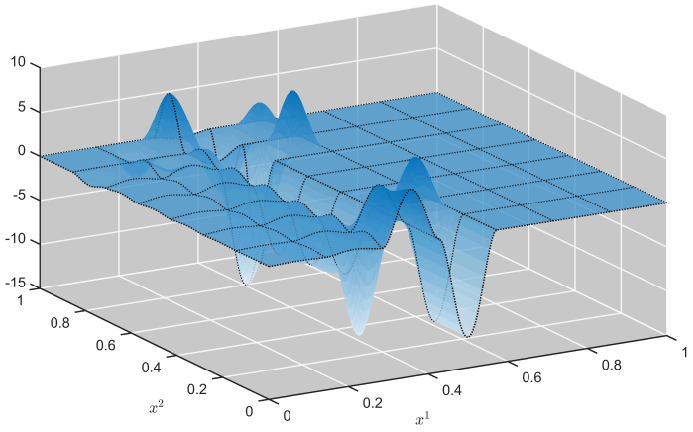


(a) $x^3 = x^3_{\alpha^3} \in \mathcal{N}^3, x^4 = x^4_{\alpha^4} \in \mathcal{N}^4$ for fixed $0 \leq \alpha^3 \leq \alpha^3_{\max}$ and $0 \leq \alpha^4 \leq \alpha^4_{\max}$.

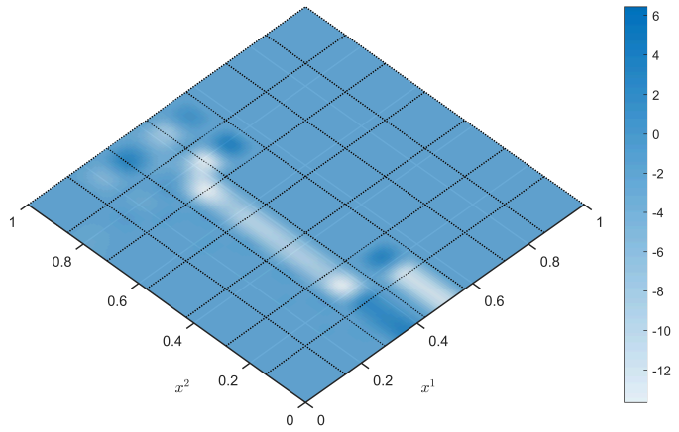


(b) $x^2 = x^2_{\alpha^2} \in \mathcal{N}^2, x^3 = x^3_{\alpha^3} \in \mathcal{N}^3$ for fixed $0 \leq \alpha^2 \leq \alpha^2_{\max}$ and $0 \leq \alpha^3 \leq \alpha^3_{\max}$.

Figure 5.11.: Multivariate interpolation of a 4D example evaluated at slices where two of the four free variables x^1, x^2, x^3, x^4 are fixed to values in their respective node set. The white dots represent the data points included in the slice. The dark gray lines are the results of the univariate interpolations of these data points, computed using the method presented in [49]. Our method produces an interpolation that preserves the shape properties of the univariate interpolations, such as monotonicity and convexity.



(a) Mixed second derivative w.r.t. x^1 and then x^2 with $x^3 = x^3_{\alpha^3} \in \mathcal{N}^3, x^4 = x^4_{\alpha^4} \in \mathcal{N}^4$ for fixed $0 \leq \alpha^3 \leq \alpha^3_{\max}$ and $0 \leq \alpha^4 \leq \alpha^4_{\max}$.



(b) Mixed second derivative w.r.t. x^1 and then x^2 with $x^3 = x^3_{\alpha^3} \in \mathcal{N}^3, x^4 = x^4_{\alpha^4} \in \mathcal{N}^4$ for fixed $0 \leq \alpha^3 \leq \alpha^3_{\max}$ and $0 \leq \alpha^4 \leq \alpha^4_{\max}$ represented as heat map.

Figure 5.12.: Selected second derivatives of the multivariate interpolation of a 4D example evaluated at the slice used in Figure 5.11a. The dotted gray lines indicate the projection of the grid $\mathcal{N}$ onto the plotted surface. The second derivatives also appear continuous. The zero corner twists for the mixed second derivatives are visible.

External Inputs in DMS, RTI, and MLI

6.

In Chapter 3, we presented efficient numerical methods for NMPC, where the OCP governing the NMPC scheme is given by

$$\begin{aligned} \min_{x(\cdot),\,u(\cdot)} \quad & \int_{t^j}^{t^j+T_{\mathrm{hor}}} \Psi(x(t),u(t))\mathrm{d}t + \Phi\big(x\big(t^j+T_{\mathrm{hor}}\big)\big) \\ \text{s.t.} \quad & \dot{x}(t) = f(x(t),u(t)), \quad t \in I_{\mathrm{hor}}\big(t^j\big), \\ & 0 \le h(x(t),u(t)), \quad t \in I_{\mathrm{hor}}\big(t^j\big), \\ & 0 = r^{\mathrm{e}}\big(x\big(t^j\big), x\big(t^j+T_{\mathrm{hor}}\big)\big), \\ & 0 \le r^{\mathrm{i}}\big(x\big(t^j\big), x\big(t^j+T_{\mathrm{hor}}\big)\big), \\ & x\big(t^j\big) = x^j. \end{aligned} \tag{6.1}$$

In theory, the OCP formulation (6.1) can also cover cases where the functions depend on constant parameters $\rho \in \mathbb{R}^{n_\rho}$ and external inputs $\sigma \in \mathbb{R}^{n_\sigma}$ that vary with the free variable t. To be precise, we define external inputs as follows.

Definition 6.1 *External inputs* are functions $\sigma\colon \mathbb{R} \to \mathbb{R}^{n_\sigma}$ that are typically not constant. Unlike the state x and the control u, external inputs are not free variables in the OCP but fixed. By

$$\sigma^j\colon I_{\mathrm{hor}}\big(t^j\big) \to \mathbb{R}^{n_\sigma},\ j \in \mathbb{N}_0$$

we denote the *external inputs for the next prediction horizon* $I_{\mathrm{hor}}\big(t^j\big)$. We do not require that

$$\sigma^j(t) \neq \sigma^k(t),\ \text{for } j \neq k \in \mathbb{N}_0,\ t \in I_{\mathrm{hor}}\big(t^j\big) \cap I_{\mathrm{hor}}\big(t^k\big).$$

Constant parameters can simply be considered as part of the function descriptions. External inputs can also be incorporated into the function descriptions by transforming them to be state-dependent, which is achieved by introducing a differential state that equals the free variable via

$$\begin{aligned} x_t(0) &= 0, \\ \dot{x}_t(t) &= 1,\ \text{for all } t \in \mathbb{R}_+. \end{aligned} \tag{6.2}$$

We then have

$$\sigma^j(t) = \sigma^j(x_t(t)) \text{ for all } t \in \mathbb{R}_+ \tag{6.3}$$

and can thus include the external inputs in OCP (6.1). In addition to external inputs, constant parameters $\rho \in \mathbb{R}^{n_\rho}$ can also play a role. We denote the constant parameters at the sampling time t^j as ρ^j.

In NMPC applications, however, it is beneficial to treat external inputs and constant parameters explicitly, as they can be subject to dis-

Reminder: OCP (6.1)

- OCP (6.1) appeared already as OCP (2.6) on p. 16 and OCP (3.1) on p. 23.
- $x(\cdot) \in W^{1,\infty}\big(I_{\mathrm{hor}}\big(t^j\big), \mathbb{R}^{n_x}\big)$ is the state trajectory.
- $u(\cdot) \in L^{\infty}\big(I_{\mathrm{hor}}\big(t^j\big), \mathbb{R}^{n_u}\big)$ is the control trajectory.
- t^j is the j-th sampling time, see Definition 2.1.
- T_{hor} is the length of the prediction horizon, see Definition 2.2.
- $x^j \in \mathbb{R}^{n_x}$ is the current system state, see Definition 2.4.
- $I_{\mathrm{hor}}(t^j) := [t^j, t^j + T_{\mathrm{hor}}]$.
- Ψ, Φ, f, h, r^{e}, and r^{i} are functions satisfying Assumption 3.2. See Equation (2.7) for their dimensions.

We introduce σ^j to reflect that external inputs are not necessarily predefined for the entire NMPC procedure but possibly only for the upcoming prediction horizon $I_{\mathrm{hor}}\big(t^j\big)$. This can, for example, occur when external inputs are obtained through measurements.

turbances and are typically estimated through measurements. Techniques for explicitly handling constant parameters are well developed, primarily in the context of Moving Horizon Estimation (MHE); see, for example, [104, 121]. In contrast, the default approach for handling external inputs relies on the aforementioned transformation. This approach is not well suited to handle disturbances in the external inputs and requires a formula for the mapping $t \mapsto \sigma(t)$. In most real-world applications, external inputs are measured, and this mapping is typically an interpolation of the measurements. A sufficiently smooth interpolation, however, is often expensive to compute and is typically piecewise defined. The latter can complicate the numerical solution of the IVPs (3.6), particularly the derivative computations in the DMS method.

[104]: Gutekunst et al. (2020), "Fast moving horizon estimation using multi-level iterations for microgrid control"
[121]: Kühl et al. (2011), "A real-time algorithm for moving horizon state and parameter estimation"

Therefore, in this chapter, we present strategies to explicitly incorporate external inputs into the DMS method and the RTI and MLI schemes. Our strategy has the advantage of accounting for disturbances in the external inputs in RTI- or MLI-based NMPC. Moreover, our strategies allow users to choose how to represent the external inputs, whether through accurate measurement representations or approximations. This flexibility enables users to avoid potentially expensive interpolations and select representations best suited to their applications. Finally, the strategies presented in this chapter can serve as an important building block for further numerical methods for NMPC. In particular, our novel Sensitivity and External Input Scenario based (SensEIS) feedback, introduced in Chapter 7, relies on the explicit treatment of external inputs and constant parameters. This is because we need derivatives of the optimal solution of the discretized version of OCP (6.1) with respect to the external inputs. Even if we approximate the external inputs using splines, as done in [136, 137], the number of coefficients for which we need to differentiate the optimal solution quickly becomes impractical.

[136]: Merino (2018), "Real-time optimization for estimation and control: Application to waste heat recovery for heavy duty trucks"
[137]: Merino et al. (2018), "A Nonlinear Model-Predictive Control Scheme for a Heavy Duty Truck's Waste Heat Recovery System Featuring Moving Horizon Estimation"

Within this thesis, we encounter external inputs in the context of the EACC application presented in Chapter 8. There, we account for information such as the elevation profile of the driving route or the velocity of a preceding vehicle. Both are functions of the free variable t, and we model these as external inputs.

The goals of this chapter are to extend the DMS method to discretize OCPs of the form

$$
\begin{aligned}
\min_{x(\cdot),\,u(\cdot)} \quad & \int_{t^j}^{t^j+T_{\mathrm{hor}}} \Psi\big(x(t),u(t);\rho^j,\sigma^j(t)\big)\mathrm{d}t + \Phi\big(x\big(t^j+T_{\mathrm{hor}}\big);\rho^j,\sigma^j\big(t^j+T_{\mathrm{hor}}\big)\big) \\
\text{s.t.} \quad & \dot{x}(t) = f\big(x(t),u(t);\rho^j,\sigma^j(t)\big), \quad t \in I_{\mathrm{hor}}\big(t^j\big), \\
& 0 \le h\big(x(t),u(t);\rho^j,\sigma^j(t)\big), \quad t \in I_{\mathrm{hor}}\big(t^j\big), \\
& 0 = r^{\mathrm{e}}\big(x\big(t^j\big),x\big(t^j+T_{\mathrm{hor}}\big);\rho^j,\sigma^j\big(t^j\big),\sigma^j\big(t^j+T_{\mathrm{hor}}\big)\big), \\
& 0 \le r^{\mathrm{i}}\big(x\big(t^j\big),x\big(t^j+T_{\mathrm{hor}}\big);\rho^j,\sigma^j\big(t^j\big),\sigma^j\big(t^j+T_{\mathrm{hor}}\big)\big), \\
& x\big(t^j\big) = x^j
\end{aligned}
\tag{6.4}
$$

and to adjust the RTI and MLI schemes to work with the resulting DMS NLP that includes external inputs.

In the following Section 6.1, we present our novel technique to incorporate external inputs into the DMS method. Afterwards, in Section 6.2, we propose two strategies to adjust the RTI and MLI schemes to work with the resulting DMS NLP that includes external inputs. One strategy is designed for cases where the upcoming external inputs are available before the next sampling time, while the other is for cases where they only become available together with the current state x^j. Both strategies have different interpretations in terms of PNLP methods.

6.1. Incorporating external inputs in DMS

Our extension of the DMS method to explicitly consider external inputs is based on the following main idea:

> **Main idea — External inputs in DMS.** External inputs are discretized using the same technique applied to discretize the control. Furthermore, instead of representing the external inputs exactly in the finite-dimensional DMS NLP, we approximate them. The user is provided with the option to define a mapping that transforms the external inputs into their approximation, allowing the user to design an approximation best suited to the specific application.

We first present the details of the external input discretization in Subsection 6.1.1. In Subsection 6.1.2, we explain the adjustments required for the state, constraint, and objective function discretization. The resulting NLP is described in Subsection 6.1.3.

6.1.1. External input discretization

Unlike the state and control discretization, external inputs are not free variables in the OCP. Instead, there is a ground truth, and we aim to formulate a representation of it which is characterized by a finite-dimensional vector $v_m \in \mathbb{R}^{n_{v_m}}$ for each shooting interval $[\tau_m, \tau_{m+1})$, $m = 0, \dots, M-1$. We assume that σ^j is our best available estimate of the true external inputs at the sampling time t^j, typically obtained as an interpolation of measurements of the ground truth.

Similarly to the control discretization (see Subsection 3.2.1), the user can choose the basis functions $\xi^{\sigma}_{m,i}$ for the external inputs. These basis functions are characterized by coefficients $v_{m,i}$. Additionally, and unlike state and control discretization, the user can define a mapping ζ that transforms the external inputs σ^j into the coefficients $v_{m,i}$. This flexibility allows the user to design an approximation of the external inputs tailored to the specific application. The following describes this process in more detail.

While the basis functions can differ for each component and interval, it is common to use the same basis function across all shooting intervals.

For each shooting interval $[\tau_m, \tau_{m+1})$, $m = 0, \dots, M-1$ and each component σ_i, $i = 0, \dots, n_\sigma - 1$ of the external input, we introduce a basis function

$$\xi^\sigma_{m,i} \colon [\tau_m, \tau_{m+1}) \times \mathbb{R}^{n_{v_{m,i}}} \to \mathbb{R}, \quad (\tau, v_{m,i}) \mapsto \xi^\sigma_{m,i}(\tau; v_{m,i}),$$

where $v_{m,i} \in \mathbb{R}^{n_{v_{m,i}}}$ are coefficients characterizing the basis function $\xi^\sigma_{m,i}$. Some common choices for basis functions are provided in Example 3.1 and Example 6.1. The coefficients corresponding to the same shooting interval are collected in a vector

It holds $n_{v_m} = \sum_{i=0}^{n_\sigma - 1} n_{v_{m,i}}$.

$$v_m \coloneqq (v_{m,0}^T, \dots, v_{m,n_\sigma-1}^T)^T \in \mathbb{R}^{n_{v_m}}.$$

Unlike to the controls, but similar to state variables, we also introduce an external input value $v_M \in \mathbb{R}^{n_\sigma}$ for the final shooting node τ_M. All external input coefficients are collected in v, i.e.,

It holds $n_v = n_\sigma + \sum_{m=0}^{M-1} n_{v_m}$.

$$v \coloneqq (v_0^T, \dots, v_M^T)^T \in \mathbb{R}^{n_v}.$$

The external input approximation characterized by the coefficients v_m for a given choice of basis functions is denoted by $\tilde{\sigma}(\cdot; v)$. Specifically, $\tilde{\sigma}(\cdot; v)$ is defined as

$$\tilde{\sigma}(\tau; v) \coloneqq \begin{pmatrix} \xi^\sigma_{m,0}(\tau; v_{m,0}) \\ \vdots \\ \xi^\sigma_{m,n_\sigma-1}(\tau; v_{m,n_\sigma-1}) \end{pmatrix},$$

for $\tau \in [\tau_m, \tau_{m+1})$, $m = 0, \dots, M-1$, and

$$\tilde{\sigma}(\tau_M; v) \coloneqq v_M.$$

Unlike control coefficients, external input coefficients are not free variables. Instead, they are determined by a user-provided mapping ζ that transforms the external inputs σ^j into external input coefficients v^j, such that

$$\sigma^j(t) \approx \tilde{\sigma}\left(t; v^j = \zeta\left(\sigma^j\right)\right)$$

for all $t \in I_{\mathrm{hor}}\left(t^j\right)$ for a given choice of basis functions. We also write $\tilde{\sigma}\left(\tau; v_m^j\right)$ for $\tau \in [\tau_m, \tau_{m+1})$ to emphasize that the approximation of the external inputs depends only on v_m^j for $\tau \in [\tau_m, \tau_{m+1})$.

As mentioned earlier, the user can choose the map ζ to ensure that the resulting approximation $\tilde{\sigma}(\cdot; v)$ is well suited to the specific application. Two simple choices for ζ are provided in Example 6.1.

▪ **Example 6.1** Two simple choices for the map ζ are as follows:

(i) Using constant basis functions, i.e.,

$$\xi^\sigma_{m,i}(\tau, v_{m,i}) = v_{m,i} \in \mathbb{R},$$

for all $m = 0, \dots, M-1$, $i = 0, \dots, n_\sigma - 1$, and $\tau \in [\tau_m, \tau_{m+1})$. The coefficients $v_{m,i}$ are set as the average of the external inputs

over each shooting interval. The map ζ is given by

$$v^j = \zeta(\sigma^j) = \begin{pmatrix} \frac{1}{\tau_1-\tau_0}\int_{\tau_0}^{\tau_1} \sigma^j(\tau)\mathrm{d}\tau \\ \vdots \\ \frac{1}{\tau_M-\tau_{M-1}}\int_{\tau_{M-1}}^{\tau_M} \sigma^j(\tau)\mathrm{d}\tau \\ \sigma^j(\tau_M) \end{pmatrix}.$$

(ii) Using linear basis functions: With $v_{m,i} = (v^{\mathrm{l}}_{m,i}, v^{\mathrm{r}}_{m,i})^T \in \mathbb{R}^2$, set

$$\xi^{\sigma}_{m,i}(\tau, v_{m,i}) = \frac{\tau_{m+1}-\tau}{\tau_{m+1}-\tau_m} v^{\mathrm{l}}_{m,i} + \frac{\tau-\tau_m}{\tau_{m+1}-\tau_m} v^{\mathrm{r}}_{m,i},$$

for all $m = 0, \dots, M-1$, $i = 0, \dots, n_\sigma - 1$, and $\tau \in [\tau_m, \tau_{m+1})$. This linearly interpolates the external input values at the shooting nodes. The map ζ is given by

$$v^j = \zeta(\sigma^j) = \begin{pmatrix} \begin{pmatrix}\sigma^j(\tau_0)\\ \sigma^j(\tau_1)\end{pmatrix} \\ \vdots \\ \begin{pmatrix}\sigma^j(\tau_{M-1})\\ \sigma^j(\tau_M)\end{pmatrix} \\ \sigma^j(\tau_M) \end{pmatrix}.$$

■

6.1.2. Adjusted DMS discretization

We still need to incorporate the discretized external inputs into the discretization of the states, constraints, and the objective function. For the state discretization, we need to adjust the formulation of the matching conditions (3.7), which ensure the continuity of the state trajectory over the entire control horizon. The matching conditions are now given by

$$x\left(\tau_{m+1}; s_m, q_m; \rho^j, v^j_m\right) - s_{m+1} = 0, \qquad m = 0, \dots, M-1,$$

where $x\left(\cdot; s_m, q_m; \rho^j, v^j_m\right)$ is the solution of the IVP.

$$\begin{aligned} \dot{x}(\tau) &= f\left(x(\tau), u(\tau; q); \rho^j, \tilde{\sigma}\left(\tau; v^j_m\right)\right), \quad \tau \in [\tau_m, \tau_{m+1}), \\ x(\tau_m) &= s_m. \end{aligned}$$

Attention:

As a consequence of only approximating the true external inputs, we must accept that the predicted trajectory deviates from the true trajectory, even if no further model errors or disturbances are present.

Mixed state-control constraints h are typically discretized in the DMS method by enforcing them pointwise at the shooting nodes, as explained in Subsection 3.2.3. We continue to utilize this strategy and modify the discretized mixed state-control constraints (3.9) to

$$0 \le h\left(s_m, q_m; \rho^j, \tilde{\sigma}\left(\tau_m; v^j_m\right)\right), \quad m = 0, \dots, M-1. \tag{6.5}$$

As indicated in the OCP formulation (6.4), the boundary constraints $r^{\mathrm{e}}, r^{\mathrm{i}}$ are now allowed to depend on the external inputs as well.

For example, in the case of airplanes, these boundary conditions could specify that the airplane must have the same elevation as the runway at the beginning and end of the flight.

The boundary constraints (3.11) and (3.12) are thus extended to

$$0 = r^{\mathrm{e}}\left(s_0, s_M; \rho^j, \tilde{\sigma}\left(\tau_0; v_0^j\right), \tilde{\sigma}\left(\tau_M; v_M^j\right)\right), \tag{6.6}$$

$$0 \leq r^{\mathrm{i}}\left(s_0, s_M; \rho^j, \tilde{\sigma}\left(\tau_0; v_0^j\right), \tilde{\sigma}\left(\tau_M; v_M^j\right)\right). \tag{6.7}$$

If we want to ensure that the constraints are also satisfied with σ^j at the shooting nodes, i.e.,

$$h\left(s_m, q_m; \rho^j, \tilde{\sigma}\left(\tau_m; v_m^j\right)\right) = h\left(s_m, q_m; \rho^j, \sigma^j(\tau_m)\right),$$

for all $m = 0, \ldots, M-1$, and

$$r^{\mathrm{e}}\left(s_0, s_M; \rho^j, \tilde{\sigma}\left(\tau_0; v_0^j\right), \tilde{\sigma}\left(\tau_M; v_M^j\right)\right) = r^{\mathrm{e}}\left(s_0, s_M; \rho^j, \sigma^j(\tau_0), \sigma^j(\tau_M)\right),$$

$$r^{\mathrm{i}}\left(s_0, s_M; \rho^j, \tilde{\sigma}\left(\tau_0; v_0^j\right), \tilde{\sigma}\left(\tau_M; v_M^j\right)\right) = r^{\mathrm{i}}\left(s_0, s_M; \rho^j, \sigma^j(\tau_0), \sigma^j(\tau_M)\right),$$

we need to choose the basis functions $\xi_{m,i}^{\sigma}$, $m = 0, \ldots, M-1$, $i = 0, \ldots, n_\sigma - 1$ such that

$$\tilde{\sigma}\left(\tau_m; v_m^j\right) = \sigma^j(\tau_m) \text{ for all } m = 0, \ldots, M-1.$$

One option where this is the case is the second choice for the map ζ presented in Example 6.1.

As in Subsection 3.2.3, we slightly overload our notation and simply write (6.5), (6.6) and (6.7) as

$$0 \leq h\left(s_m, q_m; \rho^j, v_m^j\right), \quad m = 0, \ldots, M-1,$$

and

$$0 = r^{\mathrm{e}}\left(s_0, s_M; \rho^j, v_0^j, v_M^j\right),$$

$$0 \leq r^{\mathrm{i}}\left(s_0, s_M; \rho^j, v_0^j, v_M^j\right).$$

For the objective function discretization, the extension to include external inputs works similarly. As with the boundary constraints, the MAYER term Φ can now additionally depend on the final external input. Thus, we replace the objective function (3.13) with

$$\sum_{m=0}^{M-1} \Psi_m\left(s_m, q_m; \rho^j, v_m^j\right) + \Phi\left(s_M; \rho^j, v_M^j\right).$$

As in Subsection 3.2.4, Ψ_m is either an exact representation of the LAGRANGE objective function or an approximation thereof.

6.1.3. Resulting Nonlinear Program

The NLP that results from the discretization of the OCP (6.4) using the DMS method, as described in Section 6.1 to incorporate external inputs, is given by

$$
\begin{aligned}
\min_{\substack{s\in\mathbb{R}^{n_s}\\ q\in\mathbb{R}^{n_q}}} \quad & \sum_{m=0}^{M-1} \Psi_m\left(s_m, q_m; \rho^j, v_m^j\right) + \Phi\left(s_M; \rho^j, v_M^j\right) \\
\text{s.t.} \quad & 0 = x\left(\tau_{m+1}; s_m, q_m; \rho^j, v_m^j\right) - s_{m+1}, && m = 0, \dots, M-1, \\
& 0 \le h\left(s_m, q_m; \rho^j, v_m^j\right), && m = 0, \dots, M-1, \\
& 0 = r^{\mathrm{e}}\left(s_0, s_M; \rho^j, v_0^j, v_M^j\right), \\
& 0 \le r^{\mathrm{i}}\left(s_0, s_M; \rho^j, v_0^j, v_M^j\right), \\
& 0 = x^j - s_0.
\end{aligned} \tag{6.8}
$$

Analogously to Assumption 3.2 for the DMS NLP (3.14), we make the following smoothness assumption to ensure that our SQP-based optimization algorithms presented in Section 6.2 and Chapter 7 can be applied to the NLP (6.8).

Assumption 6.1 All functions appearing in the NLP (6.8) are twice continuously differentiable with respect to the primal variables s and q and additionally with respect to the constant parameter ρ and the external inputs v In particular, we tighten Assumption 2.1 to require that the vector field $f : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_\rho} \times \mathbb{R}^{n_\sigma} \to \mathbb{R}^{n_x}$ is twice continuously differentiable with respect to x, u, ρ and σ.

6.2. Incorporating external inputs in the RTI and MLI scheme

As our focus remains on the development of efficient numerical methods for NMPC, the next step in our framework for handling external inputs is to investigate how the RTI and MLI schemes can be adapted to the novel DMS NLP (6.8). We propose two different strategies, each tailored to one of the following cases. In the first case, the external inputs for the next sampling time t^j are already available before that time t^j. In the second case, the external inputs only become available at or after the time t^j. The second case is realistic if the external inputs need to be measured online, whereas the first case applies to situations where external inputs can be retrieved from a database or similar sources.

In Subsection 6.2.1, we present strategies to incorporate external inputs into the RTI scheme. A comparison of these strategies is provided in Subsection 6.2.2. Subsequently, we discuss how the MLI scheme can be adjusted accordingly in Subsection 6.2.3.

Reminder: RTI and MLI

In the RTI scheme, two main ideas are implemented. First, only a single SQP iteration is performed per sampling time. Second, the computations are divided into a preparation phase, a feedback phase, and a transition phase, where only the duration of the feedback phase determines the feedback delay. In the MLI scheme, the RTI scheme is extended by updating the QP solved in the RTI scheme using different levels, thereby further reducing the required computation time. For details, see Section 3.4 and Section 3.5.

6.2.1. External inputs in the RTI scheme

To adapt the RTI scheme, we need to examine how the SQP method can be tailored to the DMS NLP (6.8) such that performing a single step, i.e., solving a single QP at each sampling time, can be achieved with minimal feedback delay while still providing effective feedback. In the following, we focus on computing the feedback for the sampling time t^j.

Strategy 1 – if external inputs are available a priori If the external inputs $\sigma^j(t)$ are known for all $t \in I_{\mathrm{hor}}(t^j)$ sufficiently in advance of the sampling time t^j, we can apply the tailored SQP method from Subsection 3.3.2 without any specific modifications to the DMS NLP (6.8). This involves setting up the QP

$$
\begin{aligned}
\min_{\substack{\Delta s=\left(\Delta s_0^T,\dots,\Delta s_M^T\right)^T\in\mathbb{R}^{n_s}\\ \Delta q=\left(\Delta q_0^T,\dots,\Delta q_{M-1}^T\right)^T\in\mathbb{R}^{n_q}}} \quad & \frac{1}{2}\begin{pmatrix}\Delta s\\ \Delta q\end{pmatrix}^T\begin{pmatrix}B^{ss} & B^{sq}\\ B^{qs} & B^{qq}\end{pmatrix}\begin{pmatrix}\Delta s\\ \Delta q\end{pmatrix}+\begin{pmatrix}b^s\\ b^q\end{pmatrix}^T\begin{pmatrix}\Delta s\\ \Delta q\end{pmatrix}\\
\text{s.t.} \quad & 0 = S_m^s\Delta s_m + S_m^q\Delta q_m - \Delta s_{m+1} + \delta_m, \quad m=0,\dots,M-1,\\
& 0 \le H_m^s\Delta s_m + H_m^q\Delta q_m + h_m, \quad m=0,\dots,M-1,\\
& 0 = R_{s_0}^{\mathrm{e}}\Delta s_0 + R_{s_M}^{\mathrm{e}}\Delta s_M + r^{\mathrm{e}},\\
& 0 = R_{s_0}^{\mathrm{i}}\Delta s_0 + R_{s_M}^{\mathrm{i}}\Delta s_M + r^{\mathrm{i}},\\
& 0 = x^j - s_0 - \Delta s_0,
\end{aligned}
\tag{6.9}
$$

with the key difference from QP (3.24) in Subsection 3.3.2 being that all quantities are evaluated at the already updated external inputs and constant parameters v^j, ρ^j. The state and control variables used for the evaluation are the same as in QP (3.24), i.e., s^{j-1}, q^{j-1}, which originate from the sampling time t^{j-1}. These evaluations can still be performed during the preparation phase of the RTI scheme, as the external inputs and constant parameters are available a priori. The QP (6.9) can then be condensed as described for QP (3.24) in Subsection 3.3.2. Consequently, the feedback and transition phases of the RTI scheme remain as described in Subsection 3.4.2.

Strategy 2 – if external inputs become available after the sampling time If the external inputs are not known in advance, the strategy from the previous case is no longer viable. This is because the derivatives cannot be computed during the preparation phase, and computing them during the feedback phase would result in unacceptably large feedback delays. To enable the computation of derivatives in a preparation phase before the sampling time t^j, the strategy in this case is to treat the external inputs similarly to the current state x^j. Specifically, the concept of the IVE discussed in Subsection 3.4.1 is extended to include the external inputs and constant parameters. This means that derivatives with respect to the external inputs and constant parameters are also computed, and their update steps Δv_m, $m = 0,\dots,M$, and $\Delta\rho$ are included as trivial optimization variables

in the QP (3.24). More precisely, the QP solved for the sampling time t^j in this strategy is given by

$$\min_{\substack{\Delta s=\left(\Delta s_0^T,\dots,\Delta s_M^T\right)^T\in\mathbb{R}^{n_s}\\ \Delta q=\left(\Delta q_0^T,\dots,\Delta q_{M-1}^T\right)^T\in\mathbb{R}^{n_q}\\ \Delta v^T=\left(\Delta v_0^T,\dots,\Delta v_M^T\right)^T\in\mathbb{R}^{n_v}\\ \Delta\rho\in\mathbb{R}^{n_\rho}}} \quad \frac{1}{2}\begin{pmatrix}\Delta s\\ \Delta q\\ \Delta\rho\\ \Delta v\end{pmatrix}^T\begin{pmatrix}B^{ss} & B^{sq} & B^{s\rho} & B^{sv}\\ B^{qs} & B^{qq} & B^{q\rho} & B^{qv}\\ B^{\rho s} & B^{\rho q} & B^{\rho\rho} & B^{\rho v}\\ B^{vs} & B^{vq} & B^{v\rho} & B^{vv}\end{pmatrix}\begin{pmatrix}\Delta s\\ \Delta q\\ \Delta\rho\\ \Delta v\end{pmatrix}+\begin{pmatrix}b^s\\ b^q\\ b^\rho\\ b^v\end{pmatrix}^T\begin{pmatrix}\Delta s\\ \Delta q\\ \Delta\rho\\ \Delta v\end{pmatrix} \tag{6.10a}$$

$$\text{s.t.} \quad 0 = S_m^s\Delta s_m + S_m^q\Delta q_m + S_m^\rho\Delta\rho + S_m^v\Delta v_m - \Delta s_{m+1} + \delta_m, \quad m=0,\dots,M-1, \tag{6.10b}$$

$$0 \le H_m^s\Delta s_m + H_m^q\Delta q_m + H_m^\rho\Delta\rho + H_m^v\Delta v_m + h_m, \qquad m=0,\dots,M-1, \tag{6.10c}$$

$$0 = R_{\mathrm{e}}^{s_0}\Delta s_0 + R_{\mathrm{e}}^{s_M}\Delta s_M + R_{\mathrm{e}}^{\rho}\Delta\rho + R_{\mathrm{e}}^{v_0}\Delta v_0 + R_{\mathrm{e}}^{v_M}\Delta v_M + r^{\mathrm{e}}, \tag{6.10d}$$

$$0 \le R_{\mathrm{i}}^{s_0}\Delta s_0 + R_{\mathrm{i}}^{s_M}\Delta s_M + R_{\mathrm{i}}^{\rho}\Delta\rho + R_{\mathrm{i}}^{v_0}\Delta v_0 + R_{\mathrm{i}}^{v_M}\Delta v_M + r^{\mathrm{i}}, \tag{6.10e}$$

$$0 = x^j - s_0^{j-1} - \Delta s_0, \tag{6.10f}$$

$$0 = \rho^j - \rho^{j-1} - \Delta\rho, \tag{6.10g}$$

$$0 = v_m^j - v_m^{j-1} - \Delta v_m, \qquad m=0,\dots,M. \tag{6.10h}$$

In addition to the notation defined in Equations (3.25) – (3.29) on p. 34, we have introduced in Equation (6.10) the Hessian blocks

$$\begin{pmatrix}\nabla^2_{s\rho}\mathscr{L} & \nabla^2_{sv}\mathscr{L}\\ \nabla^2_{q\rho}\mathscr{L} & \nabla^2_{qv}\mathscr{L}\end{pmatrix} =: \begin{pmatrix}B^{s\rho} & B^{sv}\\ B^{q\rho} & B^{qv}\end{pmatrix}, \tag{6.11a}$$

$$\begin{pmatrix}\nabla^2_{\rho s}\mathscr{L} & \nabla^2_{\rho q}\mathscr{L}\\ \nabla^2_{vs}\mathscr{L} & \nabla^2_{vq}\mathscr{L}\end{pmatrix} =: \begin{pmatrix}B^{\rho s} & B^{\rho q}\\ B^{vs} & B^{vq}\end{pmatrix}, \tag{6.11b}$$

$$\begin{pmatrix}\nabla^2_{\rho\rho}\mathscr{L} & \nabla^2_{\rho v}\mathscr{L}\\ \nabla^2_{v\rho}\mathscr{L} & \nabla^2_{vv}\mathscr{L}\end{pmatrix} =: \begin{pmatrix}B^{\rho\rho} & B^{\rho v}\\ B^{v\rho} & B^{vv}\end{pmatrix}, \tag{6.11c}$$

The Hessian blocks are evaluated at the primal-dual variables $s^{j-1}, q^{j-1}, \lambda^{j-1}, \mu^{j-1}$, which are the results from the previous sampling time t^{j-1}, and the parameter and external input values ρ^{j-1}, v^{j-1}.

the sensitivity matrices

$$S_m^\rho := \frac{\partial}{\partial\rho}x\left(\tau_{m+1}; s_m^{j-1}, q_m^{j-1}; \rho^{j-1}, v_m^{j-1}\right), \quad m=0,\dots,M-1, \tag{6.12a}$$

$$S_m^v := \frac{\partial}{\partial v}x\left(\tau_{m+1}; s_m^{j-1}, q_m^{j-1}; \rho^{j-1}, v_m^{j-1}\right), \quad m=0,\dots,M-1, \tag{6.12b}$$

the path constraints Jacobians

$$H_m^\rho := \frac{\partial}{\partial\rho}h\left(s^{j-1}, q^{j-1}; \rho^{j-1}, v_m^{j-1}\right), \qquad m=0,\dots,M-1, \tag{6.13a}$$

$$H_m^v := \frac{\partial}{\partial v}h\left(s^{j-1}, q^{j-1}; \rho^{j-1}, v_m^{j-1}\right), \qquad m=0,\dots,M-1, \tag{6.13b}$$

the boundary constraints Jacobians

$$R_{\mathrm{e}}^{\rho} \coloneqq \frac{\partial}{\partial \rho} r^{\mathrm{e}}\left(s_0^{j-1}, s_M^{j-1}; \rho^{j-1}, v_0^{j-1}, v_M^{j-1}\right), \tag{6.14a}$$

$$R_{\mathrm{i}}^{\rho} \coloneqq \frac{\partial}{\partial \rho} r^{\mathrm{i}}\left(s_0^{j-1}, s_M^{j-1}; \rho^{j-1}, v_0^{j-1}, v_M^{j-1}\right), \tag{6.14b}$$

$$R_{v_0}^{\mathrm{e}} \coloneqq \frac{\partial}{\partial v_0} r^{\mathrm{e}}\left(s_0^{j-1}, s_M^{j-1}; \rho^{j-1}, v_0^{j-1}, v_M^{j-1}\right), \tag{6.14c}$$

$$R_{v_0}^{\mathrm{i}} \coloneqq \frac{\partial}{\partial v_0} r^{\mathrm{i}}\left(s_0^{j-1}, s_M^{j-1}; \rho^{j-1}, v_0^{j-1}, v_M^{j-1}\right), \tag{6.14d}$$

$$R_{v_M}^{\mathrm{e}} \coloneqq \frac{\partial}{\partial v_M} r^{\mathrm{e}}\left(s_0^{j-1}, s_M^{j-1}; \rho^{j-1}, v_0^{j-1}, v_M^{j-1}\right), \tag{6.14e}$$

$$R_{v_M}^{\mathrm{i}} \coloneqq \frac{\partial}{\partial v_M} r^{\mathrm{i}}\left(s_0^{j-1}, s_M^{j-1}; \rho^{j-1}, v_0^{j-1}, v_M^{j-1}\right), \tag{6.14f}$$

and the boundary constraints residuals

$$r^{\mathrm{e}} \coloneqq r^{\mathrm{e}}\left(s_0^{j-1}, s_M^{j-1}; \rho^{j-1}, v_0^{j-1}, v_M^{j-1}\right), \tag{6.15a}$$

$$r^{\mathrm{i}} \coloneqq r^{\mathrm{i}}\left(s_0^{j-1}, s_M^{j-1}; \rho^{j-1}, v_0^{j-1}, v_M^{j-1}\right). \tag{6.15b}$$

The structures of the relevant Jacobians and Hessian blocks in QP (6.10) are listed in Appendix C. The QP (6.10) can be condensed into a smaller QP with the form

$$\begin{aligned}
\min_{\substack{\Delta s_0 \in \mathbb{R}^{n_x} \\ \Delta q = \left(\Delta q_0^T, \ldots, \Delta q_{M-1}^T\right)^T \in \mathbb{R}^{n_q} \\ \Delta v^T = \left(\Delta v_0^T, \ldots, \Delta v_M^T\right)^T \in \mathbb{R}^{n_v} \\ \Delta \rho \in \mathbb{R}^{n_\rho}}} \quad & \frac{1}{2}\begin{pmatrix} \Delta s_0 \\ \Delta q \\ \Delta \rho \\ \Delta v \end{pmatrix}^T \begin{pmatrix} \hat{B}^{s_0 s_0} & \hat{B}^{s_0 q} & \hat{B}^{s_0 \rho} & \hat{B}^{s_0 v} \\ \hat{B}^{q s_0} & \hat{B}^{q q} & \hat{B}^{q \rho} & \hat{B}^{q v} \\ \hat{B}^{\rho s_0} & \hat{B}^{\rho q} & \hat{B}^{\rho \rho} & \hat{B}^{\rho v} \\ \hat{B}^{v s_0} & \hat{B}^{v q} & \hat{B}^{v \rho} & \hat{B}^{v v} \end{pmatrix} \begin{pmatrix} \Delta s_0 \\ \Delta q \\ \Delta \rho \\ \Delta v \end{pmatrix} + \begin{pmatrix} \hat{b}^{s_0} \\ \hat{b}^{q} \\ \hat{b}^{\rho} \\ \hat{b}^{v} \end{pmatrix}^T \begin{pmatrix} \Delta s_0 \\ \Delta q \\ \Delta \rho \\ \Delta v \end{pmatrix} \\
\text{s.t.} \quad & 0 \leq H_0^s \Delta s_0 + H_0^q \Delta q_0 + H_0^\rho \Delta \rho + H_0^v \Delta v_0 + h_0, \\
& 0 \leq \hat{H}_m^s \Delta s_0 + \sum_{k=0}^{m} \hat{H}_m^{q_k} \Delta q_k + \hat{H}_m^{v_k} \Delta v_k + \hat{H}_m^\rho \Delta \rho + \hat{h}_m, \quad m = 1, \ldots, M-1, \\
& 0 = \hat{R}_0^{\mathrm{e}} \Delta s_0 + \sum_{k=0}^{M-1} \hat{R}_{q_k}^{\mathrm{e}} \Delta q_k + \hat{R}_{v_k}^{\mathrm{e}} \Delta v_k + R_{v_M}^{\mathrm{e}} \Delta v_M + \hat{R}_\rho^{\mathrm{e}} \Delta \rho + \hat{r}^{\mathrm{e}}, \\
& 0 \leq \hat{R}_0^{\mathrm{i}} \Delta s_0 + \sum_{k=0}^{M-1} \hat{R}_{q_k}^{\mathrm{i}} \Delta q_k + \hat{R}_{v_k}^{\mathrm{i}} \Delta v_k + R_{v_M}^{\mathrm{i}} \Delta v_M + \hat{R}_\rho^{\mathrm{i}} \Delta \rho + \hat{r}^{\mathrm{i}}, \\
& 0 = x^j - s_0^{j-1} - \Delta s_0, \\
& 0 = \rho^j - \rho^{j-1} - \Delta \rho, \\
& 0 = v_m^j - v_m^{j-1} - \Delta v_m, \qquad m = 0, \ldots, M.
\end{aligned} \tag{6.16}$$

The condensing procedure, which is an extended version of the one presented in Subsection 3.3.2, is detailed in Appendix D. Since all matrices in the condensed QP (6.16) depend solely on the primal-dual variables $s^{j-1}, q^{j-1}, \lambda^{j-1}, \mu^{j-1}$, which are results from the previous sampling time t^{j-1}, and the parameter and external input values ρ^{j-1}, v^{j-1}, the condensing can be performed during the preparation phase. Thus, the fast feedback of the RTI scheme is preserved.

6.2.2. Comparison of the strategies for the RTI scheme

The two strategies proposed in the previous Subsection 6.2.1 are generally not equivalent to each other. Furthermore, neither is equivalent to the default approach, where the artificial state x_t is introduced, as defined by the IVP (6.2), to treat the external inputs as functions of the differential state in the form of Equation (6.3), as described on p. 105. The RTI scheme is then applied as outlined in Section 3.4. In this section, we interpret the steps computed by our two novel strategies and the default approach.

Interpretation of the computed step For Strategy 1, the solution of QP (6.9) represents a step in the primal variables that is a generalized TP, as described in Subsection 3.4.3, and is equivalent to one step of the EULER predictor path-following method from PNLP if exact derivatives are used [57, p. 48], cf. also [103, Section 3.3, p. 73]. Regarding the parameter ρ^j and the external inputs v^j, a step of a predictor-corrector scheme is performed, with a trivial predictor and a single SQP iteration as the corrector step, cf. again [103, Section 3.3, p. 73]. Solving the QP (6.10) in Strategy 2 corresponds to performing a step of the EULER predictor path-following method with respect to both the current state x^j and the parameter ρ^j and the external inputs v^j. In both cases, the new values ρ^j, v^j are reached. In the default approach, the RTI scheme again results in a step of the EULER predictor path-following method [57, p. 48]. However, in this case, the solution manifold is parameterized only by the current state. Consequently, only a linear approximation of the novel external inputs is utilized. Thus, the computed solution approximates the optimal solution for

$$\rho^{j-1},\ \zeta\left(\sigma^{j-1}\left(t^{j-1}\right) + \frac{\partial}{\partial t}\sigma^{j-1}\left(t^{j-1}\right)\left(t^j - t^{j-1}\right)\right),$$

rather than for ρ^j, v^j.

[57]: Diehl (2001), "Real-time optimization for large scale nonlinear processes"
[103]: Guddat et al. (1990), *Parametric Optimization: Singularities, Pathfollowing and Jumps*

Computational effort The computational effort for the first strategy is nearly identical to that of the RTI scheme applied to systems without external inputs, with the only addition being the evaluation of the map ζ. For the second strategy, as well as when external inputs are transformed into functions of the states, the computational effort increases due to the need to compute derivatives with respect to the parameters and external inputs and to account for them in the condensing. However, the fact that the resulting QP (6.10) and its condensed form (6.16) include additional optimization variables and constraints does not significantly increase the computational effort, as these variables and trivial constraints can be eliminated from the QPs (6.10), (6.16) in a preprocessing step before solving the QP.

6.2.3. External inputs in the MLI scheme

Fortunately, both of our novel strategies for incorporating external inputs into the RTI scheme can be seamlessly integrated into the MLI scheme without significant challenges.

Strategy 1 – if external inputs are available a priori The intuitive extension of the strategy proposed for the RTI scheme, when external inputs are available a priori, to the MLI scheme involves solving a QP of the form (6.9), where the Hessian, linearizations, and residuals are evaluated at external inputs of varying ages, as they are reused from previous sampling times. The modified gradient used in Level C and below would still be computed using Equation (3.43), and the LAGRANGE gradient would still be approximated in Level B and A using Equation (3.44).

To determine whether this intuitive approach is theoretically justified, we recall that solving QP (6.9) was interpreted as a step of a trivial predictor combined with an SQP method as a corrector. As mentioned in Section 3.5, the MLI scheme is essentially an inexact SQP method. Incorporating our first strategy for external inputs into the MLI scheme in this intuitive manner results in a step that can be viewed as a combination of a trivial predictor and an inexact SQP method as a corrector. In our opinion, this combination – trivial predictor and inexact SQP method as a corrector – is the appropriate adaptation of the previous combination – trivial predictor and exact SQP method as a corrector.

Thus, to incorporate our first strategy for external inputs into the MLI scheme, we make the following adjustment: If a specific MLI level is scheduled at sampling time t^j, all quantities of the MLI QP (3.42) updated at this level are evaluated using the current external inputs v^j and parameters ρ^j. For all remaining components of the QP (3.42), the reference values communicated from other levels are used. As these components may have been evaluated at an outdated set of variables, the external inputs and parameters at which they were evaluated may also be outdated.

Strategy 2 – if external inputs become available after the sampling time Incorporating the second strategy for external inputs into the MLI scheme is also straightforward. Since the external inputs and parameters are included as optimization variables, the QP (6.10) that needs to be solved at each sampling time can be directly written in the form of the MLI QP (3.42). The existing MLI scheme, as described in Section 3.5, can then be used without modifications. Only the number of variables and constraints, and consequently the number of required derivatives, increases.

7. Sensitivity and External Input Scenario based Feedback

The overarching goal of this thesis is to develop efficient numerical methods to realize an NMPC-based Ecological Adaptive Cruise Control (EACC) system. To foster the use of our methods in real-life scenarios, we must ensure that our methods can be run on the onboard control units of vehicles reasonably fast. These onboard control units typically have much less computing power than personal computers, let alone high-performance computers. In particular, we have to deal with much less available memory and single-core CPUs. Despite the lack of computational power, it remains paramount that we are always able to provide feedback within the available computation time.

Even though the RTI and MLI schemes that we have presented in Section 3.4 and Section 3.5 already drastically reduce the computation time required to compute feedback, we strive to amend the MLI scheme by an additional level that allows for even shorter sampling times than Level A. The method proposed in this chapter can then either be scheduled as a regular MLI level or serve as a fallback option in case the MLI feedback could not be computed in time. It could also be used as a standalone feedback method.

For the development of this method, we make use of the following observation about our guiding use case, the EACC system. Ideally, our EACC system improves the efficiency and comfort of the drive at all times. Although we will face unforeseeable situations, we will also encounter some driving situations repeatedly with minor variations. In fact, we usually spend most of the driving time in such situations. Such situations can be, for example:

- driving at a constant speed of $30\,\mathrm{km\,h^{-1}}$, $50\,\mathrm{km\,h^{-1}}$, $70\,\mathrm{km\,h^{-1}}$ or $100\,\mathrm{km\,h^{-1}}$ without a preceding vehicle,
- accelerating after having stopped, for example at a traffic light, or after a speed limit has been lifted,
- decelerating to stop or to match an upcoming speed limit,
- following a preceding vehicle with a time gap of $2\,\mathrm{s}$, $5\,\mathrm{s}$ or $10\,\mathrm{s}$,
- driving uphill, downhill, or on an even level.

The velocities correspond to common German speed limits.

In this chapter, we propose a method that taps the potential of knowing these situations in advance. We call this method Sensitivity and External Input Scenario based (SensEIS) Feedback or Level. The main idea of our proposed SensEIS feedback is to harness the potential of knowing these situations or scenarios, as we will call them from now on, in advance. It can be summarized as follows.

The term that reads "sensei" in Japanese usually means teacher or is used for professionals. In our proposed SensEIS feedback, the feedback is based on the information obtained from the scenarios. In this sense, the scenarios teach us how to choose good feedback for new parameters. By naming our feedback SensEIS feedback, we capture this intuition in a memorable way.

Main idea — Sensitivity and External Input Scenario based (SensEIS) Feedback. We solve the OCP that we use in our NMPC scheme for several scenarios and set up a feedback matrix or a feedback-generating QP in an offline phase using results from parametric sensitivity analysis from Parametric Nonlinear Programming (PNLP). In the online phase, where we control the system of interest, the solutions of the scenarios together with the feedback matrix or the feedback-generating QP are used to compute feedback with only a single matrix-vector product, or a single QP solve, and vector-vector summations.

Our novel SensEIS feedback offers the following features:

- We can choose between feedback that can be computed with a single matrix-vector product and some vector-vector summations but cannot consider active set changes, or feedback that incorporates active set changes and is computed by a QP solve.
- All expensive computations are outsourced to an offline phase.
- We can not only react to changes in the finite-dimensional parameter vector and current state but also to changes in the continuous external inputs.
- It can serve both as a standalone method or as an extension of the MLI scheme.

Our presentation of the SensEIS feedback in this chapter is structured as follows. First, we outline the development of sensitivity-based techniques from nonlinear programming over optimal control to NMPC in Section 7.1. An important requirement for SensEIS feedback is the differentiability of the discretized OCP's solution. Therefore, we report relevant results in that field in Section 7.2. We then proceed to propose the SensEIS feedback in Section 7.3. Finally, Section 7.4 discusses arising difficulties together with strategies to solve or mitigate these difficulties and future directions of research.

7.1. Literature review

[39]: Büskens et al. (2001), "Sensitivity Analysis and Real-Time Optimization of Parametric Nonlinear Programming Problems"
[75]: Fiacco (1976), "Sensitivity analysis for nonlinear programming using penalty methods"
[76]: Fiacco (1983), *Introduction to Sensitivity and Stability Analysis in Nonlinear Programming*
[77]: Fiacco et al. (1968), *Nonlinear Programming: Sequential Unconstrained Minimization Techniques*
[163]: Robinson (1974), "Perturbed Kuhn-Tucker points and rates of convergence for a class of nonlinear-programming algorithms"

In general, the idea of using sensitivity-based techniques from PNLP to create fast online feedback methods is well established. The foundation for these techniques was laid with the research on sensitivity analysis for nonlinear programming. One of the key figures in the development of sensitivity analysis for nonlinear programming is Fiacco. In their influential book [77], Fiacco and McCormick developed Sequential Unconstrained Minimization Techniques. Particularly, the penalty method, which was first presented in [75], went on to become a cornerstone in sensitivity analysis. With his book [76], Fiacco published a reference work for stability and sensitivity analysis for nonlinear programming that is still frequently relied on today. Robinson also established the differentiability of solutions of NLPs using the Implicit Function Theorem (IFT) in [163], independently from Fiacco according to [39].

At the end of the 1990s and in the early 2000s, researchers started applying sensitivity-based techniques to OCPs as well. Noteworthy papers on this topic are contained in the collection [100]. In particular, the works [11, 134] that apply boundary value methods to compute approximate solutions quickly in an indirect fashion are contained in [100]. In the category of direct approaches, the collection [100] includes the works from Büskens and Maurer [38, 39] that build on their previous works [35–37, 135].

NMPC schemes that make use of sensitivity results followed soon. In fact, the RTI scheme first presented in 2001 [57, 67] exploits that the sequence of DMS NLPs is parametrized by the current state, also compare Section 3.4. Similarly, the MLI scheme, in particular its Level A, as we discussed it in Section 3.5, has a strong connection with results from PNLP. Further sensitivity-based NMPC schemes are surveyed in [24, 25]. Moreover, the review paper [204] does not only provide an excellent survey on fast regulatory and economic NMPC schemes in general, but in particular covers methods rooted in PNLP.

Our proposed SensEIS feedback bears similarities with the explicit feedback law that was proposed as an extension of the MLI scheme in [120] and the Level A of the MLI scheme. However, our method differs from these two approaches with respect to the linearization points and the ability to outsource the expensive computations to an offline phase. Moreover, we consider several scenarios in the offline phase, which allows us to choose from several precomputed feedback matrices online.

The fact that we precompute several feedback matrices, or feedback-generating QPs, in an offline phase places our approach between the feedback proposed in [120] and Level A on the one side and approaches from explicit multiparametric NMPC on the other side. Multiparametric NMPC methods apply techniques from multiparametric nonlinear programming. The goal in explicit multiparametric NMPC is to compute an explicit NMPC controller by partitioning the state space and computing feedback matrices on these partitions. This way, a piecewise approximation of the NLP that represents a suitable discretization of the NMPC OCP is constructed. For more information on explicit multiparametric NMPC, we refer to [69]. The contrast to our method is that we do not compute feedback matrices for the entire state space but only for selected scenarios, i.e., points in the parameter space.

Reminder: Indirect vs. direct

In direct approaches, we apply optimization methods to directly solve the optimization problem, whereas in indirect approaches, we solve the optimization problems indirectly by solving their optimality conditions. Cf. our overview of solution approaches in Section 3.1 starting on p. 23.

[11]: Augustin et al. (2001), "Sensitivity Analysis and Real-Time Control of a Container Crane under State Constraints"
[24]: Biegler (2013), "A Survey on Sensitivity-based Nonlinear Model Predictive Control"
[35]: Büskens (1998), "Optimierungsmethoden und Sensitivitätsanalyse für optimale Steuerprozesse mit Steuer- und Zustands-Beschränkungen"
[38]: Büskens et al. (2001), "Sensitivity Analysis and Real-Time Control of Parametric Optimal Control Problems Using Nonlinear Programming Methods"
[57]: Diehl (2001), "Real-time optimization for large scale nonlinear processes"
[69]: Domínguez et al. (2011), "Recent Advances in Explicit Multiparametric Nonlinear Model Predictive Control"
[100]: Grötschel et al. (2001), *Online Optimization of Large Scale Systems*
[120]: Kirches et al. (2010), "Efficient Numerics for Nonlinear Model Predictive Control"
[134]: Maurer et al. (2001), "Sensitivity Analysis and Real-Time Control of Parametric Optimal Control Problems Using Boundary Value Methods"
[135]: Maurer et al. (1995), "Solution differentiability for parametric nonlinear control problems with control-state constraints"
[204]: Wolf et al. (2016), "Fast NMPC schemes for regulatory and economic NMPC – A review"

7.2. Sensitivity theorem

Underlying the main idea of SensEIS feedback is the assumption that the optimal solution of the NLP (6.8) depends continuously on the external inputs and the constant parameters in the vicinity of the scenarios. In particular, SensEIS feedback relies on a first-order Taylor approximation of the mapping from the external inputs and constant parameters to the optimal solution of the NLP (6.8).

In this section, we present the main sensitivity theorem (Theorem 7.1) of PNLP. The sensitivity theorem is the cornerstone of the proposed SensEIS feedback and many existing sensitivity-based methods as it establishes under which conditions the optimal solution of an NLP depends continuously on the parameters. We consider the following family of NLPs throughout this section.

Definition 7.1 For at least twice continuously differentiable functions

$$\begin{aligned} J &: \mathbb{R}^{n_z} \times \mathbb{R}^{n_p} \to \mathbb{R}, \\ c &: \mathbb{R}^{n_z} \times \mathbb{R}^{n_p} \to \mathbb{R}^{n_c}, \\ d &: \mathbb{R}^{n_z} \times \mathbb{R}^{n_p} \to \mathbb{R}^{n_d} \end{aligned}$$

we consider the *parametrized NLP*

$$\begin{aligned} \min_{z \in \mathbb{R}^{n_z}} \quad & J(z,p) && \text{(7.1a)} \\ \text{s.t.} \quad & 0 = c(z,p), && \text{(7.1b)} \\ & 0 \le d(z,p) && \text{(7.1c)} \end{aligned}$$

which is parametrized by $p \in \mathbb{R}^{n_p}$. If it exists, we denote the solution of NLP (7.1) by $z^*(p) \in \mathbb{R}^{n_z}$.

In our case, the NLP (7.1) will be the DMS NLP (6.8) of the OCP (6.4), where constant parameters ρ^j and external inputs v^j have been included, which together with the current state x^j take the role of p in the NLP (7.1).

We assume that for a reference parameter $\bar{p} \in \mathbb{R}^{n_p}$ the NLP (7.1) has a unique solution $z^*(\bar{p}) \in \mathbb{R}^{n_z}$. Our goal in the following is to establish under which conditions the NLP (7.1) has a unique solution $z^*(p)$ for p in a neighborhood of $\bar{p}$ and when and how we can compute the Jacobian $\frac{\partial}{\partial p} z^*(\bar{p})$ so that we can approximate $z^*(p)$ by

$$z^*(p) \approx z^*(\bar{p}) + \frac{\partial}{\partial p} z^*(\bar{p})(p - \bar{p}). \tag{7.2}$$

These questions have been thoroughly investigated in the literature. The key to answering this question is the main sensitivity or stability theorem of PNLP as it was first proven in [75, Theorem 2.1]. In fact, there is not just one version of the theorem. Instead, a number of versions with slightly different requirements exist today. We present the theorem using the notion of the critical cone and strongly regular local solutions, similarly to [88, Theorem 6.1.4]. We thus first define the critical cone and strongly regular local solutions and then state the sensitivity theorem (Theorem 7.1).

[75]: Fiacco (1976), "Sensitivity analysis for nonlinear programming using penalty methods"

[88]: Gerdts (2024), *Optimal Control of ODEs and DAEs*

In the following, it will be important to distinguish between the components of the inequality constraint (7.1c) in NLP (7.1) that are equal to zero and those that are greater than zero for a given pair (z,p). To that end, we specify that

$$d(z,p) =: \begin{pmatrix} d_1(z,p) \\ \vdots \\ d_{n_d}(z,p) \end{pmatrix}$$

with $d_i : \mathbb{R}^{n_z} \times \mathbb{R}^{n_p} \to \mathbb{R}$ for all $i = 1, \dots, n_d$ and make the following definition.

Definition 7.2 For a given pair $(z,p) \in \mathbb{R}^{n_z} \times \mathbb{R}^{n_p}$ the *index set of active inequality constraints*, or short *active set*, is given by

$$\mathcal{A}(z,p) := \left\{ i \in \{1,\dots,n_d\} \;\middle|\; d_i(z,p) = 0 \right\}$$

and the *index set of inactive inequality constraints*, or *inactive set*, by

$$\mathcal{I}(z,p) := \left\{ i \in \{1,\dots,n_d\} \;\middle|\; d_i(z,p) > 0 \right\}$$

The inequality constraint

$$d_i(z,p) \geq 0$$

is called *active* if $i \in \mathcal{A}(z,p)$ and *inactive* if $i \in \mathcal{I}(z,p)$. By $d_{\mathcal{A}}$ and $d_{\mathcal{I}}$ we denote the *active* and *inactive constraints*, respectively, formally given as

$$d_{\mathcal{A}}(z,\rho) := \begin{pmatrix} d_{i_1}(z,p) \\ \vdots \\ d_{i_{n_{\mathcal{A}}}}(z,p) \end{pmatrix},$$

with $i_j \in \mathcal{A}(z,p)$ for all $j = 1,\dots,n_{\mathcal{A}} := \left|\mathcal{A}(z,p)\right|$ and

$$d_{\mathcal{I}}(z,\rho) := \begin{pmatrix} d_{i_1}(z,p) \\ \vdots \\ d_{i_{n_{\mathcal{I}}}}(z,p) \end{pmatrix},$$

with $i_j \in \mathcal{I}(z,p)$ for all $j = 1,\dots,n_{\mathcal{I}} := \left|\mathcal{I}(z,p)\right|$. The *LAGRANGE multipliers corresponding to the active and inactive constraints*, respectively, are similarly denoted by $\mu_{\mathcal{A}} \in \mathbb{R}^{n_{\mathcal{A}}}$ and $\mu_{\mathcal{I}} \in \mathbb{R}^{n_{\mathcal{I}}}$.

Sometimes, we will simply write $\mathcal{A}$ and $\mathcal{I}$ for brevity when it is clear to which (z,p) the index sets belong.

Next, we define the critical cone and afterwards strongly regular local solutions. Both definitions are taken from [88, Definition 6.1.2].

[88]: Gerdts (2024), *Optimal Control of ODEs and DAEs*

Definition 7.3 The *critical cone $\mathcal{T}_C(z,p)$ of NLP (7.1)* is defined as

$$\mathcal{T}_C(z,p) := \left\{ \Delta z \in \mathbb{R}^{n_z} \;\middle|\; \begin{array}{ll} \frac{\partial}{\partial z} c(z,p)\Delta z = 0, & \\ \frac{\partial}{\partial z} d_i(z,p)\Delta z \geq 0, & i \in \mathcal{A}(z,p),\ \mu_i = 0, \\ \frac{\partial}{\partial z} d_i(z,p)\Delta z = 0, & i \in \mathcal{A}(z,p),\ \mu_i > 0 \end{array} \right\}.$$

An explanation of why the set $\mathcal{T}_C$ is called the critical cone can be found in [88, p. 310].

Definition 7.4 A local minimum $\bar{z}$ of NLP (7.1) with parameter $\bar{p}$ is a *strongly regular local solution*, if the following properties hold.

(i) $\bar{z}$ is feasible, i.e. $c(\bar{z},\bar{p}) = 0$ and $d(\bar{z},\bar{p}) \geq 0$.

(ii) $\bar{z}$ satisfies the Linear Independence Constraint Qualification (LICQ), i.e., the Jacobian

$$\begin{pmatrix} \frac{\partial}{\partial z} c(\bar{z},\bar{p}) \\ \frac{\partial}{\partial z} d_{\mathcal{A}}(\bar{z},\bar{p}) \end{pmatrix}$$

has full row rank.

The LAGRANGE multipliers are unique due to the LICQ.

We have encountered the LICQ and PD before for equality constrained NLPs in Assumption 3.1.

(iii) The KKT conditions hold at $(\bar{z}, \bar{\lambda}, \bar{\mu})$, where $\bar{\lambda} \in \mathbb{R}^{n_c}$ and $\bar{\mu} \in \mathbb{R}^{n_d}$ are the LAGRANGE multipliers for the equality and inequality constraints, respectively.

(iv) The Strict Complementarity Condition (SCC) holds, i.e,

$$\bar{\mu} + d(\bar{z}, \bar{p}) > 0. \tag{7.3}$$

(v) The Positive Definiteness Condition (PD) is satisfied, i.e., the Hessian of the Lagrangian of NLP (7.1) is positive definite on the critical cone, i.e.,

$$\Delta z^T \nabla^2_{zz} \mathscr{L}(\bar{z}, \bar{\lambda}, \bar{\mu}, \bar{p}) \Delta z > 0$$

holds for all $\Delta z \in \mathscr{T}_C(\bar{z}, \bar{p}) \setminus \{0\}$, where the Lagrangian of NLP (7.1) is defined as

$$\mathscr{L}(z, \lambda, \mu, p) := J(z, p) - \lambda^T c(z, p) - \mu^T d(z, p).$$

[88]: Gerdts (2024), *Optimal Control of ODEs and DAEs*

We now state the sensitivity theorem, which is the cornerstone of SensEIS feedback. We present the theorem similarly to [88, Theorem 6.1.4], but tailor the expression for the solution derivatives to our needs.

Theorem 7.1 — Sensitivity Theorem.
Let J, c, d of NLP (7.1) be twice continuously differentiable. Let further $\bar{z}$ be a strongly regular local solution of NLP (7.1) with a nominal parameter $\bar{p}$ and let $\bar{\lambda}, \bar{\mu}$ be the associated LAGRANGE multipliers, compare Definition 7.4. Then there exist balls

$$\mathscr{B}_\varepsilon(\bar{p}) \subset \mathbb{R}^{n_p},$$
$$\mathscr{B}_\delta(\bar{z}, \bar{\lambda}, \bar{\mu}) \subset \mathbb{R}^{n_z} \times \mathbb{R}^{n_c} \times \mathbb{R}^{n_d},$$

with radii $\varepsilon, \delta > 0$, such that NLP (7.1) has a unique strongly regular local solution

$$(z^*(p), \lambda^*(p), \mu^*(p)) \in \mathscr{B}_\delta(\bar{z}, \bar{\lambda}, \bar{\mu})$$

for all $p \in \mathscr{B}_\varepsilon(\bar{p})$. Moreover, the active set remains the same, i.e.

$$\mathscr{A}(z^*(p), p) = \mathscr{A}(\bar{z}, \bar{p}) \quad \text{for all } p \in \mathscr{B}_\varepsilon(\bar{p}).$$

In addition, the mapping

$$p \mapsto (z^*(p), \lambda^*(p), \mu^*(p))$$

is continuously differentiable with respect to p with

$$\begin{pmatrix} \frac{\partial}{\partial p} z^*(\bar{p}) \\ -\frac{\partial}{\partial p} \lambda^*(\bar{p}) \\ -\frac{\partial}{\partial p} \mu^*_{\mathscr{A}}(\bar{p}) \end{pmatrix} = -\begin{pmatrix} \nabla^2_{zz} \mathscr{L} & \left(\frac{\partial}{\partial z} c\right)^T & \left(\frac{\partial}{\partial z} d_{\mathscr{A}}\right)^T \\ \frac{\partial}{\partial z} c & 0 & 0 \\ \frac{\partial}{\partial z} d_{\mathscr{A}} & 0 & 0 \end{pmatrix}^{-1} \begin{pmatrix} \nabla^2_{zp} \mathscr{L} \\ \frac{\partial}{\partial p} c \\ \frac{\partial}{\partial p} d_{\mathscr{A}} \end{pmatrix}, \tag{7.4a}$$

$$\frac{\partial}{\partial p} \mu^*_{\mathscr{I}}(\bar{p}) = 0, \tag{7.4b}$$

for all $p \in \mathscr{B}_\varepsilon(\bar{p})$, where all Jacobians on the right-hand side are evaluated at $(\bar{z}, \bar{p})$ and the Hessian blocks at $(\bar{z}, \bar{\lambda}, \bar{\mu}, \bar{p})$.

Proof. The main proof is the one of Theorem 6.1.4 in [88] and is based on the application of the Implicit Function Theorem (IFT). However, we have tailored the derivative expressions to our later need, so we shall prove the correctness of our presentation in the following. From [88, Theorem 6.1.4] we get, after reordering and accounting for the different convention regarding the sign of the inequalities and the LAGRANGE multipliers of the equality constraints in [88], that the derivatives are obtained as the solution of the uniquely solvable linear system

[88]: Gerdts (2024), *Optimal Control of ODEs and DAEs*

$$\left(\begin{array}{c|c|c} \nabla_{zz}^2 \mathscr{L} & \left(\frac{\partial}{\partial z} c\right)^T & -\left(\frac{\partial}{\partial z} d\right)^T \\ \hline \frac{\partial}{\partial z} c & 0 & 0 \\ \hline -\bar{\Xi} \cdot \frac{\partial}{\partial z} d & 0 & -\bar{\Gamma} \end{array}\right) \begin{pmatrix} \frac{\partial}{\partial p} z^*(\bar{p}) \\ -\frac{\partial}{\partial p} \lambda^*(\bar{p}) \\ \frac{\partial}{\partial p} \mu^*(\bar{p}) \end{pmatrix} = -\begin{pmatrix} \nabla_{zp}^2 \mathscr{L} \\ \frac{\partial}{\partial p} c \\ -\bar{\Xi} \cdot \frac{\partial}{\partial p} d \end{pmatrix}, \tag{7.5}$$

where

$$\bar{\Xi} := \operatorname{diag}(\mu_1, \ldots, \mu_{n_d}), \qquad \bar{\Gamma} := \operatorname{diag}(d_1, \ldots, d_{n_d}),$$

where the inequality constraints are evaluated at $(\bar{z}, \bar{p})$. We then equivalently write Equation (7.5) as

$$\left(\begin{array}{c|c|c} \nabla_{zz}^2 \mathscr{L} & \left(\frac{\partial}{\partial z} c\right)^T & \left(\frac{\partial}{\partial z} d\right)^T \\ \hline \frac{\partial}{\partial z} c & 0 & 0 \\ \hline \bar{\Xi} \cdot \frac{\partial}{\partial z} d & 0 & -\bar{\Gamma} \end{array}\right) \begin{pmatrix} \frac{\partial}{\partial p} z^*(\bar{p}) \\ -\frac{\partial}{\partial p} \lambda^*(\bar{p}) \\ -\frac{\partial}{\partial p} \mu^*(\bar{p}) \end{pmatrix} = -\begin{pmatrix} \nabla_{zp}^2 \mathscr{L} \\ \frac{\partial}{\partial p} c \\ \bar{\Xi} \cdot \frac{\partial}{\partial p} d \end{pmatrix}, \tag{7.6}$$

Without loss of generality, we assume that the inequality constraints are ordered such that

That $\mathscr{A}(\bar{z}, \bar{p}) \cup \mathscr{I}(\bar{z}, \bar{p}) = \{1, \ldots, n_d\}$ is ensured by the fact that $\bar{z}$ is feasible as a strongly regular local solution and thus satisfies $d(\bar{z}, \bar{p}) \geq 0$.

$$\mathscr{A}(\bar{z}, \bar{p}) = \{1, \ldots, n_{\mathscr{A}}\}, \qquad \mathscr{I}(\bar{z}, \bar{p}) = \{n_{\mathscr{A}} + 1, \ldots, n_d\}.$$

Due to the SCC (7.3) we get in this case that

$$\begin{aligned} &\mu_i > 0, && d_i = 0 && \text{for all } i \in \mathscr{A}(z, p), \\ &\mu_i = 0, && d_i > 0 && \text{for all } i \in \mathscr{I}(z, p). \end{aligned}$$

Therefore, we can write

$$\bar{\Xi} = \begin{pmatrix} \bar{\Xi}_{\mathscr{A}} & 0 \\ 0 & 0 \end{pmatrix}, \qquad \bar{\Gamma} = \begin{pmatrix} 0 & 0 \\ 0 & \bar{\Gamma}_{\mathscr{I}} \end{pmatrix},$$

with invertible blocks

$$\bar{\Xi}_{\mathscr{A}} := \operatorname{diag}(\mu_1, \ldots, \mu_{n_{\mathscr{A}}}), \qquad \bar{\Gamma}_{\mathscr{I}} := \operatorname{diag}(d_{n_{\mathscr{A}}+1}, \ldots, d_{n_d}).$$

The linear system (7.6) thus can equivalently be written as

$$\left(\begin{array}{c|c|cc} \nabla_{zz}^2 \mathscr{L} & \left(\frac{\partial}{\partial z} c\right)^T & \left(\frac{\partial}{\partial z} d_{\mathscr{A}}\right)^T & \left(\frac{\partial}{\partial z} d_{\mathscr{I}}\right)^T \\ \hline \frac{\partial}{\partial z} c & 0 & 0 & 0 \\ \hline \bar{\Xi}_{\mathscr{A}} \cdot \frac{\partial}{\partial z} d_{\mathscr{A}} & 0 & 0 & 0 \\ 0 & 0 & 0 & -\bar{\Gamma}_{\mathscr{I}} \end{array}\right) \begin{pmatrix} \frac{\partial}{\partial p} z^*(\bar{p}) \\ \frac{\partial}{\partial p} \lambda^*(\bar{p}) \\ -\frac{\partial}{\partial p} \mu_{\mathscr{A}}^*(\bar{p}) \\ -\frac{\partial}{\partial p} \mu_{\mathscr{I}}^*(\bar{p}) \end{pmatrix} = -\begin{pmatrix} \nabla_{zp}^2 \mathscr{L} \\ \frac{\partial}{\partial p} c \\ \bar{\Xi}_{\mathscr{A}} \cdot \frac{\partial}{\partial p} d_{\mathscr{A}} \\ 0 \end{pmatrix}. \tag{7.7}$$

The last block row shows that

$$\frac{\partial}{\partial p}\mu^*_{\mathcal{F}}(\bar{p}) = 0,$$

which is Equation (7.4b). The third block row is equivalent to

$$\frac{\partial}{\partial z} d_{\mathcal{A}} \frac{\partial}{\partial p} z^*(\bar{p}) = -\frac{\partial}{\partial p} d_{\mathcal{A}}.$$

Therefore, we obtain from the linear system (7.7) that

$$\begin{pmatrix} \nabla^2_{zz}\mathcal{L} & \left(\frac{\partial}{\partial z}c\right)^T & \left(\frac{\partial}{\partial z}d_{\mathcal{A}}\right)^T \\ \frac{\partial}{\partial z}c & 0 & 0 \\ \frac{\partial}{\partial z}d_{\mathcal{A}} & 0 & 0 \end{pmatrix} \begin{pmatrix} \frac{\partial}{\partial p}z^*(\bar{p}) \\ -\frac{\partial}{\partial p}\lambda^*(\bar{p}) \\ -\frac{\partial}{\partial p}\mu^*_{\mathcal{A}}(\bar{p}) \end{pmatrix} = -\begin{pmatrix} \nabla^2_{zp}\mathcal{L} \\ \frac{\partial}{\partial p}c \\ \frac{\partial}{\partial p}d_{\mathcal{A}} \end{pmatrix}. \tag{7.8}$$

Non-singularity of the saddle-point system on the left hand side of the linear system Equation (7.8) is a consequence of the LICQ and PD, compare Definition 7.4, and can be shown as in [88, p. 315] in the proof of Theorem 6.1.4. As a consequence, Equation (7.8) is equivalent to Equation (7.4a) which concludes the proof. ■

[88]: Gerdts (2024), *Optimal Control of ODEs and DAEs*

In the upcoming Section 7.3 we will explain how we leverage Theorem 7.1 in our proposed SensEIS feedback.

7.3. SensEIS feedback

As stated in the introduction of this chapter, the overarching goal of this thesis is to develop efficient numerical methods to realize an NMPC-based EACC system. The core task in this context is to compute an approximate solution of the OCP

$$\begin{aligned} \min_{x(\cdot),u(\cdot)} \quad & \int_{t^j}^{t^j+T_{\mathrm{hor}}} \Psi\big(x(t),u(t);\rho^j,\sigma^j(t)\big)\mathrm{d}t + \Phi\big(x\big(t^j+T_{\mathrm{hor}}\big);\rho,\sigma^j\big(t^j+T_{\mathrm{hor}}\big)\big) \\ \text{s.t.} \quad & \dot{x}(t) = f\big(x(t),u(t);\rho^j,\sigma^j(t)\big), \quad t\in I_{\mathrm{hor}}\big(t^j\big), \\ & 0 \le h\big(x(t),u(t);\rho^j,\sigma^j(t)\big), \quad t\in I_{\mathrm{hor}}\big(t^j\big), \\ & 0 = r^{\mathrm{e}}\big(x\big(t^j\big),x\big(t^j+T_{\mathrm{hor}}\big);\rho^j,\sigma^j\big(t^j\big),\sigma^j\big(t^j+T_{\mathrm{hor}}\big)\big), \\ & 0 \le r^{\mathrm{i}}\big(x\big(t^j\big),x\big(t^j+T_{\mathrm{hor}}\big);\rho^j,\sigma^j\big(t^j\big),\sigma^j\big(t^j+T_{\mathrm{hor}}\big)\big), \\ & x\big(t^j\big) = x^j, \end{aligned} \tag{7.9}$$

efficiently at each sampling time t^j. Note that the OCP (7.9) is not governed by a Differential Algebraic Equation (DAE), but only an ODE. This is because the external inputs v^j and the constant parameter ρ^j are not optimization variables but fixed.

Reminder: OCP formulation

The OCP (7.9) is an adapted version of the NMPC OCP (2.6), where we have added a current parameter vector ρ and external inputs σ. The OCP (7.9) was also considered in Chapter 6 as OCP (6.4). The suitable spaces for the optimization variables are as for OCP (2.6) and given in Equation (2.8).

We use a "first discretize, then optimize" approach. For the discretization of the OCP (7.9) we use the DMS method as described in Section 3.2 and its extension to treat external inputs in Section 6.1. As a result, we have to compute an approximate solution of an NLP of the form

$$\min_{\substack{s\in\mathbb{R}^{n_s}\\ q\in\mathbb{R}^{n_q}}} \quad \sum_{m=0}^{M-1} \Psi_m\left(s_m, q_m; \rho, v_m\right) + \Phi\left(s_M; \rho, v_M\right) \tag{7.10a}$$

$$\text{s.t.} \quad 0 = x\left(\tau_{m+1}; s_m, q_m; \rho, v_m\right) - s_{m+1}, \quad m = 0, \dots, M-1, \tag{7.10b}$$

$$0 \le h\left(s_m, q_m; \rho, v_m\right), \quad m = 0, \dots, M-1, \tag{7.10c}$$

$$0 = r^{\mathrm{e}}\left(s_0, s_M; \rho, v_0, v_M\right), \tag{7.10d}$$

$$0 \le r^{\mathrm{i}}\left(s_0, s_M; \rho, v_0, v_M\right), \tag{7.10e}$$

$$0 = \hat{x} - s_0. \tag{7.10f}$$

Reminder: First discretize then optimize approach

In the "first discretize, then optimize approach" we first discretize the infinite-dimensional 3.1 to obtain a finite-dimensional NLP, see Section 3.1 for more information.

In the NMPC case: $\rho = \rho^j$, $v = v^j$ and $\hat{x} = x^j$.

The NLP (7.10) is parametrized by

- the discretized external inputs $v = \begin{pmatrix} v_0 \\ \vdots \\ v_M \end{pmatrix} \in \mathbb{R}^{n_v}$,
- the parameter $\rho \in \mathbb{R}^{n_\rho}$,
- the state $\hat{x} \in \mathbb{R}^{n_x}$,

which we collect in a single parameter vector

$$p := \begin{pmatrix} \hat{x} \\ \rho \\ v \end{pmatrix} \in \mathbb{R}^{n_p},$$

with $n_p = n_x + n_\rho + n_v$. In that sense, the NLP (7.10) can be understood as an NLP parametrized by p in the sense of Definition 7.1.

As described in the main idea of SensEIS, we aim to leverage results from sensitivity analysis to compute the first control coefficient q_0 and from it the NMPC feedback value u^j. More specifically, we want to apply Theorem 7.1.

In the SensEIS feedback we provide two different variants to compute q_0 from the NLP (7.10) based on Theorem 7.1. The first is based on the explicit computation of the matrix on the right-hand side of Equation (7.4a) and then computing q_0 using a matrix-vector product including this matrix. For the second approach, we interpret the linear system posed by Equation (7.4a) as the KKT system of a suitably formulated QP. We then extend the QP to consider the inactive constraints, too. Solving the resulting QP thus constitutes the second approach. We present both approaches in the following subsections and state the full algorithm in Subsection 7.3.3. We conclude by presenting three options for how SensEIS feedback can be combined with the MLI scheme in Subsection 7.3.4.

7.3.1. Variant 1: Using the feedback matrix

Our proposed SensEIS feedback can be divided into an offline phase that takes place once before we start the NMPC procedure and an online phase that is carried out at each sampling time t^j. Before we delve into further details of the two phases, we make the following abbreviating definition.

Definition 7.5 For a nominal parameter $\bar{p}$, let $K(\bar{p})$ be defined as

$$K(\bar{p}) := -\begin{pmatrix} \nabla^2_{zz}\mathcal{L} & \left(\frac{\partial}{\partial z}c\right)^T & \left(\frac{\partial}{\partial z}d_{\mathcal{A}}\right)^T \\ \frac{\partial}{\partial z}c & 0 & 0 \\ \frac{\partial}{\partial z}d_{\mathcal{A}} & 0 & 0 \end{pmatrix}^{-1} \begin{pmatrix} \nabla^2_{zp}\mathcal{L} \\ \frac{\partial}{\partial p}c \\ \frac{\partial}{\partial p}d_{\mathcal{A}} \end{pmatrix}.$$

We refer to $K(\bar{p})$ as the ***feedback matrix***, compare Equation (7.4a).

For the DMS NLP (7.10), the equality constraints c consist of the matching conditions (7.10b), the equality boundary constraint (7.10d), and the initial value constraint (7.10f). The inequality constraints d accordingly consist of the inequality mixed state-control constraints (7.10c) and the inequality boundary constraint (7.10e). The optimization variables $z \in \mathbb{R}^{n_z}$ comprise the node values $s \in \mathbb{R}^{n_s}$ and the control coefficients $q \in \mathbb{R}^{n_q}$. We thus write the feedback matrix $K(\bar{p})$ corresponding to the DMS NLP (7.10) with parameter vector $\bar{p}$ as

$$K(\bar{p}) = -\begin{pmatrix} \nabla^2_{ss}\mathcal{L} & \nabla^2_{sq}\mathcal{L} & \left(\frac{\partial}{\partial s}c\right)^T & \left(\frac{\partial}{\partial s}d\right)^T \\ \nabla^2_{qs}\mathcal{L} & \nabla^2_{qq}\mathcal{L} & \left(\frac{\partial}{\partial q}c\right)^T & \left(\frac{\partial}{\partial q}d\right)^T \\ \frac{\partial}{\partial s}c & \frac{\partial}{\partial q}c & 0 & 0 \\ \frac{\partial}{\partial s}d & \frac{\partial}{\partial q}d & 0 & 0 \end{pmatrix}^{-1} \begin{pmatrix} \nabla^2_{s\hat{x}}\mathcal{L} & \nabla^2_{s\rho}\mathcal{L} & \nabla^2_{sv}\mathcal{L} \\ \nabla^2_{q\hat{x}}\mathcal{L} & \nabla^2_{q\rho}\mathcal{L} & \nabla^2_{qv}\mathcal{L} \\ \frac{\partial}{\partial \hat{x}}c & \frac{\partial}{\partial \rho}c & \frac{\partial}{\partial v}c \\ 0 & \frac{\partial}{\partial \rho}d & \frac{\partial}{\partial v}d \end{pmatrix} \tag{7.11}$$

with

$$\begin{aligned} \nabla^2_{ss}\mathcal{L} &\in \mathbb{R}^{n_s \times n_s}, & \nabla^2_{qs}\mathcal{L} &\in \mathbb{R}^{n_q \times n_s}, \\ \nabla^2_{sq}\mathcal{L} &\in \mathbb{R}^{n_s \times n_q}, & \nabla^2_{qq}\mathcal{L} &\in \mathbb{R}^{n_q \times n_q}, \\ \nabla^2_{s\hat{x}}\mathcal{L} &\in \mathbb{R}^{n_s \times n_x}, & \nabla^2_{q\hat{x}}\mathcal{L} &\in \mathbb{R}^{n_q \times n_x}, \\ \nabla^2_{s\rho}\mathcal{L} &\in \mathbb{R}^{n_s \times n_\rho}, & \nabla^2_{q\rho}\mathcal{L} &\in \mathbb{R}^{n_q \times n_\rho}, \\ \nabla^2_{sv}\mathcal{L} &\in \mathbb{R}^{n_s \times n_v}, & \nabla^2_{qv}\mathcal{L} &\in \mathbb{R}^{n_q \times n_v}, \end{aligned}$$

The Hessian blocks are evaluated at the reference parameter values $p = \bar{p} = \left((\hat{\bar{x}})^T, (\bar{\rho})^T, (\bar{v})^T\right)^T$ and the primal-dual solution $\bar{s}, \bar{q}, \bar{\lambda}, \bar{\mu}$ of the NLP (7.10) with $\bar{p}$.

and

$$\begin{aligned}
\frac{\partial}{\partial s} c &\in \mathbb{R}^{n_s + n_{r^e} \times n_s}, & \frac{\partial}{\partial s} d &\in \mathbb{R}^{M n_h + n_{r^i} \times n_s}, \\
\frac{\partial}{\partial q} c &\in \mathbb{R}^{n_s + n_{r^e} \times n_q}, & \frac{\partial}{\partial q} d &\in \mathbb{R}^{M n_h + n_{r^i} \times n_q}, \\
\frac{\partial}{\partial \hat{x}} c &\in \mathbb{R}^{n_s + n_{r^e} \times n_x}, & 0 = \frac{\partial}{\partial \hat{x}} d &\in \mathbb{R}^{M n_h + n_{r^i} \times n_x}, \\
\frac{\partial}{\partial \rho} c &\in \mathbb{R}^{n_s + n_{r^e} \times n_\rho}, & \frac{\partial}{\partial \rho} d &\in \mathbb{R}^{M n_h + n_{r^i} \times n_\rho}, \\
\frac{\partial}{\partial v} c &\in \mathbb{R}^{n_s + n_{r^e} \times n_v}, & \frac{\partial}{\partial v} d &\in \mathbb{R}^{M n_h + n_{r^i} \times n_v}.
\end{aligned}$$

The Jacobians are evaluated at the reference parameter values $p = \bar{p} = \left((\hat{\bar{x}})^T, (\bar{\rho})^T, (\bar{v})^T\right)^T$ and the primal solution $\bar{s}, \bar{q}$ of the NLP (7.10) with $\bar{p}$.

The detailed structure of the blocks in the feedback matrix (7.11) for the DMS NLP is given in Appendix C.

Offline phase A fundamental component that sets our method apart from the explicit MLI feedback proposed in [120] and explicit multi-parametric NMPC is that we compute the feedback matrix K neither for only a single nominal parameter $\bar{p}$ nor do we map out the entire space $\mathcal{P} \subset \mathbb{R}^{n_p}$ from which the parameter can be chosen. Instead, we compute $K(\bar{p}^i)$ for selected scenarios, i.e., selected

[120]: Kirches et al. (2010), "Efficient Numerics for Nonlinear Model Predictive Control"

$$\bar{p}^i = \begin{pmatrix} \hat{\bar{x}}^i \\ \bar{\rho}^i \\ \bar{v}^i \end{pmatrix} \in \mathcal{P}, \ i = 0, \dots, n_{\text{scen}}.$$

The selection of suitable scenarios is primarily a task for the user, who should make use of problem-specific domain knowledge. However, we believe that developing automated scenario selection strategies is a promising direction for future research.

Reminder: EACC scenarios

At the beginning of the chapter, we have mentioned scenarios which could be used for the external inputs in an EACC system.

For each scenario $\bar{p}^i \in \mathcal{P}$, we then solve the DMS NLP (7.10) to obtain the respective primal variables $\bar{s}^i, \bar{q}^i$ and associated dual variables $\bar{\lambda}^i, \bar{\mu}^i_{\mathcal{A}}$. In general, any suitable NLP solver can be applied to solve the NLP (7.10). Our recommendation, though, is to use the tailored SQP method as described in Subsection 3.3.2. The advantage of this tailored SQP method over other general-purpose NLP solvers is that it efficiently exploits the structure of NLP (7.10).

The final step in the offline phase for each scenario is to explicitly construct the feedback matrix $K(\bar{p}^i)$ according to Equation (7.11) using suitable numerical linear algebra techniques, for which we refer to textbooks about numerical linear algebra like [56, 182]. Of course, explicit inversion of the first matrix on the right-hand side of Equation (7.11) should be avoided. Fortunately, we can compute the feedback matrix without doing so by solving the matrix linear system posed by Equation (7.11). This is especially efficient if

[182]: Trefethen et al. (1997), *Numerical Linear Algebra*
[56]: Demmel (1997), *Applied Numerical Linear Algebra*

$$n_p = n_x + n_\rho + n_v \ll 2(M+1)n_x + M(n_q + n_h) + n_{r^e} + n_{r^i}.$$

In other words, the linear system can be solved if the number of parameters and external inputs is much smaller than the number of rows/columns of the inverse on the right-hand side of Equation (7.11).

Even though the duration of the offline phase is less relevant, we still recommend that future research addresses the question of whether structure exploitation is possible to accelerate the solution of the linear system (7.11).

If we only need the feedback value u^j and thus only q_0 at each sampling time, we can decrease the storage requirement and the required computation in the online phase by only storing the block row of $K(\bar{p}^i)$ that yields $\frac{\partial}{\partial p} q_0$, i.e., the reduced feedback matrix

$$\hat{K}(\bar{p}^i) := (0, \mathbb{I}, 0) K(\bar{p}^i), \tag{7.12}$$

where the first zero block has the size $n_{q_0} \times (M+1)n_x$, the identity block the size $n_{q_0} \times n_{q_0}$, and the second zero block the size

$$n_{q_0} \times n_q - n_{q_0} + (M+1)n_x + Mn_h + n_{r^e} + n_{r^i}.$$

The external input discretization v^j might be known already before the sampling time. But, the current state x^j and the parameter ρ^j are only available from the sampling time t^j onwards.

The external inputs are scaled down in order to avoid that they dominate the scenario selection as there is a vector for the external input for each shooting interval, whereas the current state and the parameter are only a single vector for the entire horizon each.

Online phase At the sampling time t^j, the parametrizing quantities, i.e., the current state x^j, the parameter ρ^j, and the external input discretization v^j become known. The task at hand at this point is to choose a suitable scenario, i.e., a $\bar{p}^i \in \mathcal{P}$, whose feedback matrix $K\left(\bar{p}^{\bar{i}}\right)$ we use to compute the feedback. We suggest the simple procedure of defining the closest scenario $\bar{p}^{\bar{i}}$ by

$$\bar{i} := \underset{i \in \{0,\dots,n_{\text{scen}}\}}{\arg\min} \left\|\hat{\bar{x}}^i - x^j\right\|_{W_x} + \left\|\bar{\rho}^i - \rho^j\right\|_{W_\rho} + \frac{1}{M+1}\left\|\bar{v}^i - v^j\right\|_{W_v}, \tag{7.13}$$

where $\|\cdot\|_W$ is a weighted Euclidean norm with weight matrix W. The weight matrices are introduced to account for different scaling of the different components. Moreover, the weight matrices can be used to put more weight on some of the components based on expert knowledge regarding the application.

Different selection strategies are also conceivable. In particular, we are choosing a single scenario at each sampling time. A different idea is to consider the feedback based on multiple scenarios at the same time and combine the results in an appropriate way. We suggest exploring this idea in future research.

Once the reference scenario $\bar{p}^{\bar{i}}$ has been selected, it only remains to compute the update step either as

$$\begin{pmatrix} \Delta s \\ \Delta q \\ -\Delta\lambda \\ -\Delta\mu_{\mathcal{A}} \end{pmatrix} = K\left(\bar{p}^{\bar{i}}\right) \begin{pmatrix} \hat{\bar{x}}^{\bar{i}} - x^j \\ \bar{\rho}^{\bar{i}} - \rho^j \\ \bar{v}^{\bar{i}} - v^j \end{pmatrix} \tag{7.14}$$

or, in the reduced case, as

$$\Delta q_0 = \hat{K}\left(\bar{p}^{\bar{i}}\right) \begin{pmatrix} \hat{\bar{x}}^{\bar{i}} - x^j \\ \bar{\rho}^{\bar{i}} - \rho^j \\ \bar{v}^{\bar{i}} - v^j \end{pmatrix}. \tag{7.15}$$

Following Equation (7.2), the steps are then added to the solutions obtained in the scenarios, i.e., we set

$$\begin{pmatrix} s \\ q \\ \lambda \\ \mu_{\mathcal{A}(z,p)} \end{pmatrix} = \begin{pmatrix} \bar{s}^{\bar{i}} \\ \bar{q}^{\bar{i}} \\ \bar{\lambda}^{\bar{i}} \\ \bar{\mu}^{\bar{i}}_{\mathcal{A}} \end{pmatrix} + \begin{pmatrix} \Delta s \\ \Delta q \\ \Delta\lambda \\ \Delta\mu_{\mathcal{A}} \end{pmatrix} \tag{7.16}$$

or, in the reduced case,

$$q_0 = \bar{q}_0^{\bar{i}} + \Delta q_0. \tag{7.17}$$

7.3.2. Variant 2: Using the feedback generating QP

The variable step Δz, $\Delta\lambda$, $\Delta\mu_{\mathcal{A}}$ computed using the feedback matrix $K(\bar{p})$ according to

$$\begin{pmatrix} \Delta z \\ -\Delta\lambda \\ -\Delta\mu_{\mathcal{A}} \end{pmatrix} = K(\bar{p})\Delta p$$

solves the linear system

$$\begin{pmatrix} \nabla^2_{zz}\mathcal{L} & \left(\frac{\partial}{\partial z}c\right)^T & \left(\frac{\partial}{\partial z}d_{\mathcal{A}}\right)^T \\ \frac{\partial}{\partial z}c & 0 & 0 \\ \frac{\partial}{\partial z}d_{\mathcal{A}} & 0 & 0 \end{pmatrix} \begin{pmatrix} \Delta z \\ -\Delta\lambda \\ -\Delta\mu_{\mathcal{A}} \end{pmatrix} = -\begin{pmatrix} \nabla^2_{zp}\mathcal{L} \\ \frac{\partial}{\partial p}c \\ \frac{\partial}{\partial p}d_{\mathcal{A}} \end{pmatrix} \Delta p$$

or equivalently

$$\begin{pmatrix} \nabla^2_{zz}\mathcal{L} & \left(\frac{\partial}{\partial z}c\right)^T & \left(\frac{\partial}{\partial z}d_{\mathcal{A}}\right)^T \\ \frac{\partial}{\partial z}c & 0 & 0 \\ \frac{\partial}{\partial z}d_{\mathcal{A}} & 0 & 0 \end{pmatrix} \begin{pmatrix} \Delta z \\ -(\bar{\lambda}+\Delta\lambda) \\ -(\bar{\mu}_{\mathcal{A}}+\Delta\mu_{\mathcal{A}}) \end{pmatrix} = -\begin{pmatrix} \nabla^2_{zp}\mathcal{L}\Delta p + \left(\frac{\partial}{\partial z}c(\bar{z},\bar{p})\right)^T\bar{\lambda} + \left(\frac{\partial}{\partial z}d_{\mathcal{A}}(\bar{z},\bar{p})\right)^T\bar{\mu}_{\mathcal{A}} \\ \frac{\partial}{\partial p}c\Delta p \\ \frac{\partial}{\partial p}d_{\mathcal{A}}\Delta p \end{pmatrix}. \tag{7.18}$$

The reference variables $\bar{z}$, $\bar{\lambda}$, $\bar{\mu}_{\mathcal{A}}$ are the optimal solution and multipliers of the NLP

$$\begin{aligned} \min_{z\,\in\,\mathbb{R}^{n_z}} \quad & J(z,\bar{p}) \\ \text{s.t.} \quad & 0 = c(z,\bar{p}), \\ & 0 = d_{\mathcal{A}}(z,\bar{p}) \end{aligned} \tag{7.19}$$

i.e., the solution of NLP (7.1) with the active inequalities considered as equality constraints. Thus, the reference variables satisfy the KKT conditions for NLP (7.19) and thus in particular

$$0 = \nabla_z J(\bar{z},\bar{p}) - \left(\frac{\partial}{\partial z}c(\bar{z},\bar{p})\right)^T\bar{\lambda} - \left(\frac{\partial}{\partial z}d_{\mathcal{A}}(\bar{z},\bar{p})\right)^T\bar{\mu}_{\mathcal{A}}.$$

Therefore, we can equivalently write the linear system (7.18) as

$$\begin{pmatrix} \nabla^2_{zz}\mathscr{L} & \left(\frac{\partial}{\partial z}c\right)^T & \left(\frac{\partial}{\partial z}d_{\mathscr{A}}\right)^T \\ \frac{\partial}{\partial z}c & 0 & 0 \\ \frac{\partial}{\partial z}d_{\mathscr{A}} & 0 & 0 \end{pmatrix} \begin{pmatrix} \Delta z \\ -(\bar{\lambda}+\Delta\lambda) \\ -(\bar{\mu}_{\mathscr{A}}+\Delta\mu_{\mathscr{A}}) \end{pmatrix} = -\begin{pmatrix} \nabla^2_{zp}\mathscr{L}\Delta p + \nabla_z J \\ \frac{\partial}{\partial p}c\Delta p \\ \frac{\partial}{\partial p}d_{\mathscr{A}}\Delta p \end{pmatrix}. \tag{7.20}$$

Under the stated smoothness assumptions, the Hessian of the Lagrangian is symmetric and thus $\left(\nabla^2_{zp}\mathscr{L}\right)^T = \nabla^2_{pz}\mathscr{L}$.

Similar to the derivation of the SQP method in Subsection 3.3.1, we interpret the linear system (7.20) as the KKT optimality system of a QP. The suitable QP in the present case is given by

$$\begin{aligned} \min_{\Delta z \in \mathbb{R}^{n_z}} \quad & \frac{1}{2}\Delta z^T \nabla^2_{zz}\mathscr{L}(\bar{z},\bar{\lambda},\bar{\mu}_{\mathscr{A}},\bar{p})\Delta z + \Delta p^T \nabla^2_{pz}\mathscr{L}(\bar{z},\bar{\lambda},\bar{\mu}_{\mathscr{A}},\bar{p})\Delta z + \nabla_z J(\bar{z},\bar{p})^T \Delta z \\ \text{s.t.} \quad & 0 = \frac{\partial}{\partial z}c(z,\bar{p})\Delta z + \frac{\partial}{\partial p}c(z,\bar{p})\Delta p, \\ & 0 = \frac{\partial}{\partial z}d_{\mathscr{A}}(z,\bar{p})\Delta z + \frac{\partial}{\partial p}d_{\mathscr{A}}(z,\bar{p})\Delta p. \end{aligned} \tag{7.21}$$

We continue to follow the ideas used in the derivation of the SQP method and propose to extend the QP (7.21) such that all inequalities are considered. This leads to the QP

$$\begin{aligned} \min_{\Delta z \in \mathbb{R}^{n_z}} \quad & \frac{1}{2}\Delta z^T \nabla^2_{zz}\mathscr{L}(\bar{z},\bar{\lambda},\bar{\mu}_{\mathscr{A}},\bar{p})\Delta z + \Delta p^T \nabla^2_{pz}\mathscr{L}(\bar{z},\bar{\lambda},\bar{\mu}_{\mathscr{A}},\bar{p})\Delta z + \nabla_z J(\bar{z},\bar{p})^T \Delta z \\ \text{s.t.} \quad & 0 = \frac{\partial}{\partial z}c(z,\bar{p})\Delta z + \frac{\partial}{\partial p}c(z,\bar{p})\Delta p, \\ & 0 \le \frac{\partial}{\partial z}d(z,\bar{p})\Delta z + \frac{\partial}{\partial p}d(z,\bar{p})\Delta p. \end{aligned} \tag{7.22}$$

In our case where the general NLP (7.1) is given by the NLP (7.10) that arises from the DMS discretization of OCP (7.9) with a reference parameter vector $\bar{p}$, the QP (7.22) for a current parameter vector p^j is given by

$$\min_{\substack{\Delta s=\left(\Delta s_0^T,\dots,\Delta s_M^T\right)^T\in\mathbb{R}^{n_s} \\ \Delta q=\left(\Delta q_0^T,\dots,\Delta q_{M-1}^T\right)^T\in\mathbb{R}^{n_q}}} \quad \frac{1}{2}\begin{pmatrix}\Delta s\\ \Delta q\end{pmatrix}^T \begin{pmatrix} B^{ss} & B^{sq} \\ B^{qs} & B^{qq}\end{pmatrix}\begin{pmatrix}\Delta s\\ \Delta q\end{pmatrix} + \begin{pmatrix}\Delta x^j\\ \Delta\rho^j\\ \Delta v^j\end{pmatrix}^T \begin{pmatrix} B^{\hat{x}s} & B^{\hat{x}q} \\ B^{\rho s} & B^{\rho q} \\ B^{vs} & B^{vq}\end{pmatrix}\begin{pmatrix}\Delta s\\ \Delta q\end{pmatrix} + \begin{pmatrix}b^s\\ b^q\end{pmatrix}^T\begin{pmatrix}\Delta s\\ \Delta q\end{pmatrix} \tag{7.23a}$$

$$\text{s.t.} \quad 0 = S^s_m \Delta s_m + S^q_m \Delta q_m + S^\rho_m \Delta\rho^j + S^v_m \Delta v^j_m - \Delta s_{m+1}, \quad m = 0,\dots,M-1, \tag{7.23b}$$

$$0 \le H^s_m \Delta s_m + H^q_m \Delta q_m + H^\rho_m \Delta\rho^j + H^v_m \Delta v^j_m + \bar{h}_m, \quad m = 0,\dots,M-1, \tag{7.23c}$$

$$0 = R^{\mathrm{e}}_{s_0}\Delta s_0 + R^{\mathrm{e}}_{s_M}\Delta s_M + R^{\mathrm{e}}_{\rho}\Delta\rho^j + R^{\mathrm{e}}_{v_0}\Delta v^j_0 + R^{\mathrm{e}}_{v_M}\Delta v^j_M, \tag{7.23d}$$

$$0 \le R^{\mathrm{i}}_{s_0}\Delta s_0 + R^{\mathrm{i}}_{s_M}\Delta s_M + R^{\mathrm{i}}_{\rho}\Delta\rho^j + R^{\mathrm{i}}_{v_0}\Delta v^j_0 + R^{\mathrm{i}}_{v_M}\Delta v^j_M + \bar{r}^{\mathrm{i}}, \tag{7.23e}$$

$$0 = \Delta x^j - \Delta s_0. \tag{7.23f}$$

In addition to the notation defined in Equations (3.25) – (3.29) on p. 34 and in Equations (6.11) – (6.15) on p. 113, we have introduced in Equation (7.23) the steps

$$\begin{pmatrix}\Delta x^j\\ \Delta\rho^j\\ \Delta v^j\end{pmatrix} \coloneqq \begin{pmatrix}x^j - \hat{\bar{x}}\\ \rho^j - \bar{\rho}\\ v^j - \bar{v}\end{pmatrix},$$

and the Hessian blocks are now evaluated at the reference parameter values

$$p = \bar{p} = \begin{pmatrix}\hat{\bar{x}}\\ \bar{\rho}\\ \bar{v}\end{pmatrix}$$

and the primal-dual solution $\bar{s}$, $\bar{q}$, $\bar{\lambda}$, $\bar{\mu}$ of the NLP (7.10) with $\bar{p}$. Similarly, the sensitivity matrices, constraints Jacobians, and residuals are evaluated at $\bar{s}$, $\bar{q}$, and $\bar{p}$.

As we have done for the QP that arises in the SQP method tailored to the DMS NLP (3.14), we can condense the QP (7.23) to a QP that has only Δs_0 and Δq as optimization variables. We describe the condensing procedure for QP (7.23) in Appendix E.

The resulting condensed QP has the same form as the QP (3.41) that we have encountered in the tailored SQP method for the DMS NLP. That means the condensed form of QP (7.23) is given by

$$\begin{aligned}
\min_{\substack{\Delta s_0\in\mathbb{R}^{n_x}\\ \Delta q=\left(\Delta q_0^T,\dots,\Delta q_{M-1}^T\right)^T\in\mathbb{R}^{n_q}}} \quad & \frac{1}{2}\begin{pmatrix}\Delta s_0\\ \Delta q\end{pmatrix}^T\begin{pmatrix}\hat{B}^{s_0s_0} & \hat{B}^{s_0q}\\ \hat{B}^{qs_0} & \hat{B}^{qq}\end{pmatrix}\begin{pmatrix}\Delta s_0\\ \Delta q\end{pmatrix} + \begin{pmatrix}\hat{b}^{s_0}\left(\rho^j, v^j\right)\\ \hat{b}^{q}\left(\rho^j, v^j\right)\end{pmatrix}^T\begin{pmatrix}\Delta s_0\\ \Delta q\end{pmatrix}\\
\text{s.t.} \quad & 0 \le H_0^s\Delta s_0 + H_0^q\Delta q_0 + h_0\left(\rho^j, v^j\right),\\
& 0 \le \hat{H}_m^s\Delta s_0 + \sum_{j=0}^{m}\hat{H}_m^{q_j}\Delta q_j + \hat{h}_m\left(\rho^j, v^j\right), \quad m = 1,\dots,M-1,\\
& 0 = \hat{R}_0^{\mathrm{e}}\Delta s_0 + \sum_{j=0}^{M-1}\hat{R}_{q_j}^{\mathrm{e}}\Delta q_j + \hat{r}^{\mathrm{e}}\left(\rho^j, v^j\right),\\
& 0 \le \hat{R}_0^{\mathrm{i}}\Delta s_0 + \sum_{j=0}^{M-1}\hat{R}_{q_j}^{\mathrm{i}}\Delta q_j + \hat{r}^{\mathrm{i}}\left(\rho^j, v^j\right),\\
& 0 = \Delta x^j\left(x^j\right) - s_0 - \Delta s_0,
\end{aligned} \tag{7.24}$$

where we have explicitly highlighted that only

- the residuals h_0, $\hat{h}_m$, $\hat{r}^{\mathrm{e}}$, and $\hat{r}^{\mathrm{i}}$,
- the gradients $\hat{b}^{s_0}$ and $\hat{b}^q$,
- the step Δx^j in the initial value

depend on the new parameters x^j, ρ^j, v^j. In fact, these are the only quantities that we still need to compute online to set up the QP (7.24). All condensing matrices can be computed in the offline phase as they only depend on the reference parameters that characterize the scenarios. Therefore, the computational effort required for the condensing in SensEIS feedback is comparable to the one for Level B in the MLI scheme. The complete adjusted condensing procedure is given in Appendix E.

Let now Δs, Δq denote the primal solution of the QP (7.23), or the blown up solution of QP (7.24), and λ_{QP}, μ_{QP} the associated dual variables. Let further $\bar{i}$ be the index of the selected reference scenario according to Equation (7.13). We then again follow Equation (7.2) and compute our approximate solution to NLP (7.10) by setting

$$\begin{aligned}\begin{pmatrix} s \\ q \end{pmatrix} &= \begin{pmatrix} \bar{s}^{\bar{i}} \\ \bar{q}^{\bar{i}} \end{pmatrix} + \begin{pmatrix} \Delta s \\ \Delta q \end{pmatrix}, \\ \begin{pmatrix} \lambda \\ \mu \end{pmatrix} &= \begin{pmatrix} \lambda_{\mathrm{QP}} \\ \mu_{\mathrm{QP}} \end{pmatrix},\end{aligned} \tag{7.25}$$

or, in the reduced case, again only

$$q_0 = \bar{q}_0^{\bar{i}} + \Delta q_0. \tag{7.26}$$

In the reduced case, we do not even need to blow up the solution of the condensed QP (7.24).

Reminder: Generalized Tangential Predictor (TP)

We discussed the generalized TP in Subsection 3.4.3 as the RTI scheme produces steps that are approximated generalized TPs. We said there that in simple terms, a TP is a first-order approximation at smooth parts of the solution manifold and a generalized TP can even "jump" over the non-differentiable points of the solution manifold by taking active set changes into account.

As we solve the NLP (7.10) until convergence for the scenarios, the step $\Delta s, \Delta q$ that we compute by solving the QP (7.22) is a generalized TP. Therefore, active set changes can be taken into account.

7.3.3. Full algorithm

In the offline phase of SensEIS feedback, we perform Algorithm 7.1. In the online phase, we perform Algorithm 7.2 at each sampling time t^j when the current parameters

$$p^j := \begin{pmatrix} x^j \\ \rho^j \\ v^j \end{pmatrix}$$

become known. As we have illustrated in the previous two subsections, there are two different variants of SensEIS feedback. In the first, we use the feedback matrix, and in the second, we use the feedback generating QP. In both variants, we can choose whether we compute an approximation for all primal-dual variables or only for the first control coefficients q_0, which are the only ones we need to provide feedback to the system.

7.3.4. Combination with the MLI scheme

So far, we have presented SensEIS feedback as a standalone method. However, we can also use SensEIS feedback to extend the MLI scheme that we presented in Section 3.5. For that, we can think of three different approaches, which we present in the following paragraphs. Hybrids between these three approaches are also conceivable. However, we will not discuss these further.

SensEIS feedback as a fallback In this approach, we exploit that the SensEIS feedback for one sampling time t^j does not depend on variables from any other sampling time. In contrast, the MLI scheme also requires variables and linearizations from previous sampling times.

Algorithm 7.1: Offline phase for SensEIS feedback.

Input: A set of scenarios $\bar{p}^i = \begin{pmatrix} \hat{\bar{x}}^i \\ \bar{\rho}^i \\ \bar{v}^i \end{pmatrix},\ i = 0, \dots, n_{\mathrm{scen}}$.

Output: For all $i = 0, \dots, n_{\mathrm{scen}}$

Variant 1 with full feedback:

- the optimal primal-dual solution $\bar{s}^{\bar{i}}, \bar{q}^{\bar{i}}, \bar{\lambda}^{\bar{i}}, \bar{\mu}^{\bar{i}}_{\mathcal{A}}$ for the active set $\mathcal{A}\left(\bar{s}^{\bar{i}}, \bar{q}^{\bar{i}}, \bar{p}^i\right)$,
- the feedback matrix $K\left(\bar{p}^i\right)$ given by Equation (7.11)

Variant 1 with reduced feedback:

- the optimal first control coefficients $\bar{q}^{\bar{i}}_0$,
- the reduced feedback matrix $\hat{K}\left(\bar{p}^i\right)$ given by Equation (7.12)

Variant 2 with full feedback:

- the optimal primal-dual solution $\bar{s}^{\bar{i}}, \bar{q}^{\bar{i}}, \bar{\lambda}^{\bar{i}}, \bar{\mu}^{\bar{i}}_{\mathcal{A}}$ for the active set $\mathcal{A}\left(\bar{s}^{\bar{i}}, \bar{q}^{\bar{i}}, \bar{p}^i\right)$,
- all components needed to set up QP (7.22) or its condensed form (7.24)

Variant 2 with reduced feedback:

- the first optimal control coefficients $\bar{q}^{\bar{i}}_0$,
- all components needed to set up QP (7.22) or its condensed form (7.24)

for $i \leftarrow 0$ **to** n_{scen} **do**

$\bar{s}^{\bar{i}}, \bar{q}^{\bar{i}}, \bar{\lambda}^{\bar{i}}, \bar{\mu}^{\bar{i}}_{\mathcal{A}} \leftarrow$ Solve NLP (7.10) with $\bar{p} = \bar{p}^i$

Compute Jacobians of constraints w.r.t. $s, q, \hat{x}, \rho, v$

Compute Hessian of Lagrangian w.r.t. $s, q, \hat{x}, \rho, v$

Variant 1:

$\mathcal{A}\left(\bar{s}^{\bar{i}}, \bar{q}^{\bar{i}}, \bar{p}^i\right) \leftarrow$ Identify active set according to Definition 7.2

$K\left(\bar{p}^i\right) \leftarrow$ Compute full feedback matrix using Equation (7.11)

Only for reduced feedback:

$\hat{K}\left(\bar{p}^i\right) \leftarrow$ Compute reduced feedback matrix using Equation (7.12)

Variant 2:

$$
\begin{aligned}
E_m^{s_0}, E_m^{q_j} &\leftarrow \text{Compute matrices (3.31), (3.32),} \\
\hat{H}_m^{q_j}, \hat{H}_m^{s} &\leftarrow \text{Condense constraints (3.24c) w. (3.39),} \\
\hat{R}^{\mathrm{e}}_{q_m}, \hat{R}^{\mathrm{e}}_0 &\leftarrow \text{Condense constraints (3.24d) w. (3.40),} \\
\hat{R}^{\mathrm{i}}_{q_m}, \hat{R}^{\mathrm{i}}_0 &\leftarrow \text{Condense constraints (3.24e) w. (3.40)} \\
\hat{B}^{s_0 s_0}, \hat{B}^{s_0 q} & \\
\hat{B}^{q s_0}, \hat{B}^{q q} &\leftarrow \text{Condense Hessian w. (3.35), (3.36)}
\end{aligned}
$$

Algorithm 7.2: Online phase for SensEIS feedback.

Input:

- a set of scenarios $\begin{pmatrix} \hat{\bar{x}}^i \\ \bar{\rho}^i \\ \bar{v}^i \end{pmatrix}$, $i = 0, \dots, n_{\text{scen}}$,
- the output of Algorithm 7.1

Output: The control coefficients q_0 and, for the full feedback, additionally s, q, λ, μ

1 $\bar{i} \leftarrow$ Identify closest scenario using (7.13)

Variant 1 with full feedback:

2 $\Delta s, \Delta q, \Delta\lambda, \Delta\mu_{\mathcal{A}} \leftarrow$ Compute update step using full feedback matrix $K\left(\bar{p}^{\bar{i}}\right)$ using (7.14)

3 $s, q, \lambda, \mu_{\mathcal{A}(z,p)} \leftarrow$ Compute approximate solution using (7.16)

Variant 1 with reduced feedback:

2 $\Delta q_0 \leftarrow$ Compute update step using reduced feedback matrix $\hat{K}\left(\bar{p}^{\bar{i}}\right)$ using (7.15)

3 $q_0 \leftarrow$ Compute approximate solution using (7.17)

Variant 2 with full feedback:

2 Complete setting up condensed QP (7.24)

3 $\Delta s, \Delta q, \lambda_{\text{QP}}, \mu_{\text{QP}} \leftarrow$ Solve QP (7.24) and blow up its solution

4 $s, q, \lambda, \mu \leftarrow$ Compute approximate solution using (7.25)

Variant 2 with reduced feedback:

2 Complete setting up condensed QP (7.24)

3 $\Delta q_0 \leftarrow$ Solve QP (7.24)

4 $q_0 \leftarrow$ Compute approximate solution using (7.26)

In SensEIS feedback, this information is taken from the precomputed scenarios. Moreover, SensEIS feedback is very fast, especially if the feedback matrix is used. Thus, SensEIS feedback can serve well as a fallback option when applying the MLI scheme. This means that we use SensEIS feedback whenever a QP in the MLI scheme could not be solved successfully in time. This can either happen due to very strict limitations on the available computation time or because a QP happens to be infeasible. The latter is an inherent difficulty of the SQP method, as demonstrated in [143, Section 18.3].

[143]: Nocedal et al. (2006), *Numerical Optimization*

If the MLI feedback could not be computed successfully, we then execute the online phase of SensEIS feedback as given in Algorithm 7.2 after the failure has happened. For this purpose, the feedback matrix-based variant with reduced SensEIS feedback is favorable as this variant has the lowest computation time of all variants. To make sure that SensEIS feedback is, in fact, available in time, the time limit for the MLI feedback phase should be shortened such that this fallback strategy can still be realized.

SensEIS feedback as a provider of derivatives and variables The idea of this approach is that the Hessian blocks, linearizations, and primal-dual variables that have been computed for the scenarios are used to update the respective components in the MLI QP. In that sense, SensEIS feedback takes a similar role as Level D in the MLI scheme, with the difference that its feedback is not necessarily passed to the system. The advantage over the use of Level D or C is that the derivatives do not have to be computed online as they are already available from the scenarios. Therefore, new derivatives can be used for faster lower MLI levels without having to use the slower Levels D and C.

In this approach, we only perform online the first step of Algorithm 7.2, i.e., the scenario selection, and then communicate the derivatives and variables from the scenarios to the MLI levels.

If the derivatives provided by the offline scenarios are used without the respective primal-dual variables as current variables in the MLI scheme, we are in an inexact SQP method setting. Hence, modified gradients have to be used, as in Level C and the lower levels.

We explained the idea behind the modified gradients in the description of MLI Level C on p. 46.

SensEIS feedback as a scheduled MLI level This approach extends the previous approach. Here, SensEIS feedback does not only provide derivative and variable information to other MLI levels, but we also use it to provide feedback. Accordingly, SensEIS feedback is treated the same as the other MLI levels. That means that SensEIS feedback can be scheduled with other MLI levels and communicates its derivative and variable information to other MLI levels according to the chosen communication scheme. A difference from the existing MLI levels is that the SensEIS level does not obtain derivatives and usually no variables from the other MLI levels. The reason for that is that the SensEIS level has good derivative and variable information at its disposal from the scenarios.

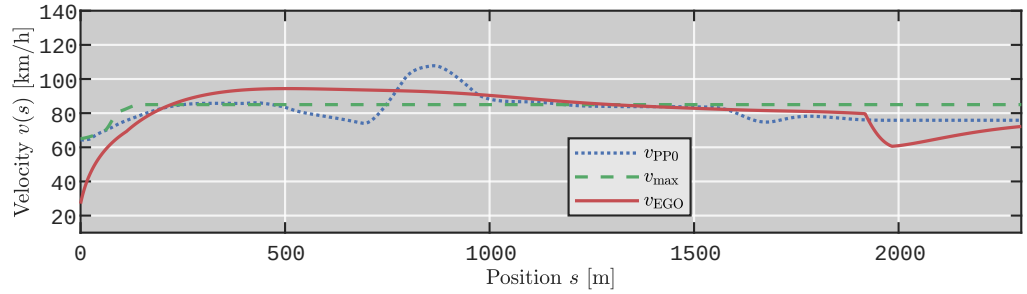
Reminder: MLI communication schemes

We described the different MLI communication schemes in Subsection 3.5.2.

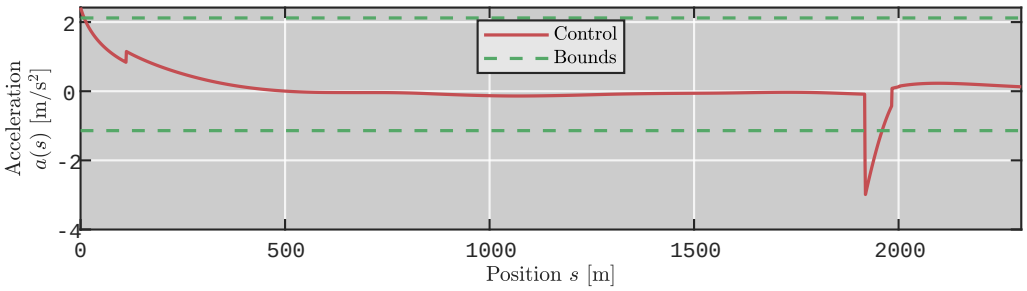
7.4. Challenges and future directions of research

In this section, we turn our attention to challenges that we have observed when applying SensEIS feedback to the EACC system application presented in Chapter 8. For each challenge, we discuss remedies that we have found or future research directions required to overcome these challenges.

Inequality violations with Variant 1 The feedback matrix-based variant, i.e., Variant 1, of SensEIS feedback potentially violates inequality constraints that are inactive in the optimal solution of the selected reference scenario. In Figure 7.1, we can see, for example, that the maximum velocity and the minimal acceleration are violated in the EACC system application if we use Variant 1 of SensEIS feedback.



(a) Velocity trajectory



(b) Control trajectory

Figure 7.1.: Variant 1 of SensEIS feedback violates upper speed limits (a) and lower control bounds (b) in the EACC system application.

One remedy is to use the second variant of SensEIS feedback, which is based on the feedback generating QP (7.22). While still being a very fast feedback strategy, this second variant is slower than the first. Therefore, we might want to use Variant 1 as often as possible instead of Variant 2. To achieve that, we suggest two strategies. The first is to only follow the computed step direction until the first inactive constraint becomes active. The full strategy is presented for the QP (7.22) in Algorithm 7.3. To state Algorithm 7.3, let for a fixed reference index $\bar{i} \in \{1, \dots, n_{\text{scen}}\}$ and fixed new parameters ρ^j, v^j be $\Delta s_0 = \Delta x^j$ and Δq be the step computed by Variant 1 of SensEIS feedback. For these, we define

b captures the fixed effect of the parameters in the linearized constraints. a captures the one of the step in the controls, which we can choose. The step size is then chosen such that the effect of the step in the controls compensates the effect of the parameters such that feasibility is achieved – if possible.

$$a = \begin{pmatrix} a_1 \\ \vdots \\ a_{Mn_d+n_{r\mathrm{i}}} \end{pmatrix} := \begin{pmatrix} H_0^q \Delta q_0 \\ \sum_{j=0}^{1} \hat{H}_1^{q_j} \Delta q_j \\ \vdots \\ \sum_{j=0}^{M-1} \hat{H}_{M-1}^{q_j} \Delta q_j \\ \sum_{j=0}^{M-1} \hat{R}_{q_j}^{\mathrm{i}} \Delta q_j \end{pmatrix}, \tag{7.27}$$

$$b = \begin{pmatrix} b_1 \\ \vdots \\ b_{Mn_d+n_{r\mathrm{i}}} \end{pmatrix} := \begin{pmatrix} H_0^s \Delta s_0 + h_0(\rho^j, v^j) \\ \hat{H}_1^s \Delta s_0 + \hat{h}_1(\rho^j, v^j) \\ \vdots \\ \hat{H}_{M-1}^s \Delta s_0 + \hat{h}_{M-1}(\rho^j, v^j) \\ \hat{R}_0^{\mathrm{i}} \Delta s_0 + \hat{r}^{\mathrm{i}}(\rho^j, v^j) \end{pmatrix}. \tag{7.28}$$

The inspiration for this strategy is the homotopy step length determination in the QP solution algorithm qpOASES presented in [74].

[74]: Ferreau et al. (2014), "qpOASES: a parametric active-set algorithm for quadratic programming"

Algorithm 7.3: Step size strategy for Variant 1 of SensEIS feedback.

Input:

- new parameters x^j, ρ^j, v^j,
- reference index $\bar{i} \in \{1, \dots, n_{\text{scen}}\}$ selected in step 1 in Algorithm 7.2,
- the step $\Delta s_0 = \Delta x^j$ and Δq computed by Variant 1 of SensEIS feedback

Output: Step length $\kappa \in [0, 1]$ for adjusted step $\begin{pmatrix} \Delta x^j \\ \kappa \Delta q \end{pmatrix}$

1 $a_i, b_i \leftarrow$ Evaluate (7.27), (7.28)
2 **if** $a_i + b_i < 0$ *and* $b_i < 0$ *hold for some* $i \in \{1, \dots, Mn_d + n_{r^i}\}$ **then**
3 $\quad$ **return** *no feasible step size found*
4 Set $\chi_+ := \{i \mid b_i < 0, a_i + b_i \geq 0\}$, $\chi_- := \{i \mid b_i \geq 0, a_i + b_i < 0\}$
5 Set $\kappa_{\min} := \begin{cases} \max_{i \in \chi_+} \frac{-b_i}{a_i}, & \text{if } \chi_+ \neq \emptyset, \\ 0, & \text{else} \end{cases}$, $\kappa_{\max} := \begin{cases} \min_{i \in \chi_-} \frac{-b_i}{a_i}, & \text{if } \chi_- \neq \emptyset, \\ 1, & \text{else} \end{cases}$
6 **if** $\kappa_{\min} \leq \kappa_{\max}$ **then**
7 $\quad$ **return** $\kappa = \kappa_{\max}$
8 **else**
9 $\quad$ **return** *no feasible step size found*

Even if a feasible step size κ could be found, this heuristic can have the following two drawbacks. First, while an inequality constraint violation can be avoided, we might pay for this by violating equality constraints occurring in QP (7.24). We have deliberately designed Algorithm 7.3 such that the linearized initial value constraint is satisfied though. Moreover, to execute Algorithm 7.3, we need some matrices that occur in QP (7.24), which leads to a higher storage requirement than Variant 1 typically has. It also should be noted that only violations of the linearized inequalities are avoided, not necessarily of the original nonlinear inequality constraints.

While Algorithm 7.3 can help mitigate the challenge of inequality violations, it is not a fully satisfying remedy. Using Variant 2 of SensEIS feedback instead is a more reliable approach to avoid inequality violations. To still use Variant 1 as often as possible, we suggest adaptively switching between the two variants as follows. If a step computed by Variant 1 leads to a violation of the (linearized) inequality constraints, we switch to Variant 2 for the next sampling time where SensEIS feedback is used and possibly apply Algorithm 7.3 to adjust the step from Variant 1. If the active set has stayed the same for several sampling times with Variant 2 SensEIS feedback, we switch back to Variant 1.

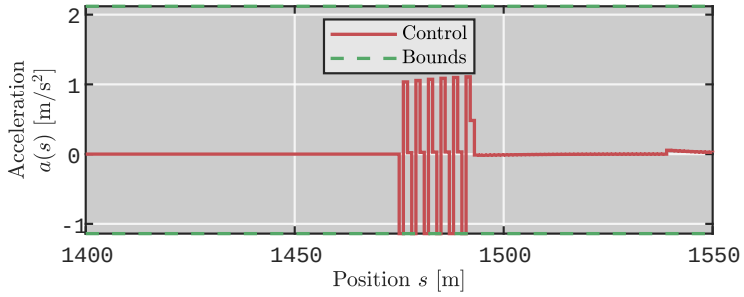


Figure 7.2.: The SensEIS feedback chatters due to rapid alternation between reference scenarios in the EACC system application.

Chattering due to scenario selection If we use SensEIS feedback at several sampling times in quick succession, we can encounter the situation where SensEIS feedback leads to chattering in the control variables. Chattering means that the control frequently switches between two or more identical or very similar values. Figure 7.2 shows such a situation for the EACC system application.

This problem is caused by the way we select the reference scenario, i.e., by Equation (7.13). If the distances of two or more scenarios to the current parameters are very similar, i.e.,

$$\begin{aligned}&\left\|\hat{\bar{x}}^l - x^j\right\|_{W_x} + \left\|\bar{\rho}^l - \rho^j\right\|_{W_\rho} + \frac{1}{M+1}\left\|\bar{v}^l - v^j\right\|_{W_v}\\ &\approx \left\|\hat{\bar{x}}^i - x^j\right\|_{W_x} + \left\|\bar{\rho}^i - \rho^j\right\|_{W_\rho} + \frac{1}{M+1}\left\|\bar{v}^i - v^j\right\|_{W_v}\end{aligned}$$

for some $i \neq l \in \{1, \dots, n_{\text{scen}}\}$, then $\bar{i}$ can quickly alternate between two or more values.

Fortunately, this challenge can usually be overcome by introducing a discount for the latest reference index. That means that we modify our reference index selection (7.13) to

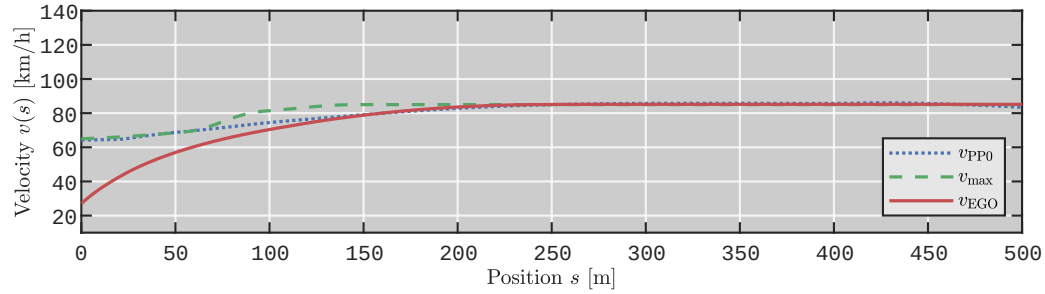
$$\bar{i} := \underset{i \in \{0,\dots,n_{\text{scen}}\}}{\arg\min}\ \omega\left(i, \bar{i}^{\text{old}}\right)\left(\left\|\hat{\bar{x}}^i - x^j\right\|_{W_x} + \left\|\bar{\rho}^i - \rho^j\right\|_{W_\rho} + \frac{1}{M+1}\left\|\bar{v}^i - v^j\right\|_{W_v}\right), \tag{7.29}$$

where $\bar{i}^{\text{old}}$ denotes the previously chosen reference index and where

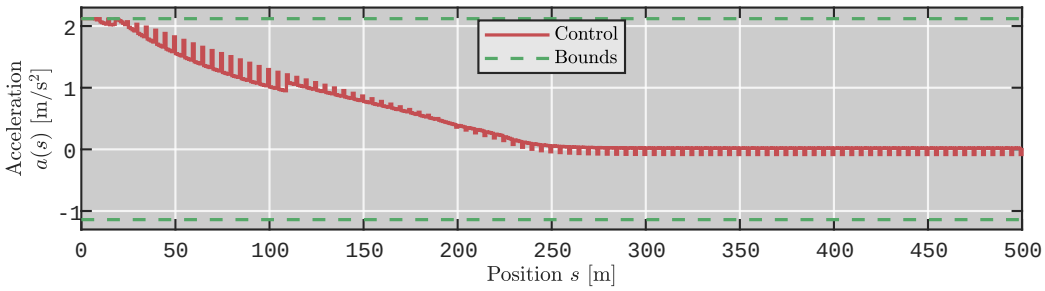
$$\omega\left(i, \bar{i}^{\text{old}}\right) := \begin{cases}\gamma, & \text{if } i = \bar{i}^{\text{old}},\\ 1, & \text{else,}\end{cases}$$

with a constant $\gamma \in (0, 1)$. In our experiments for the EACC system application, $\gamma = 0.8$ worked well. A second possible remedy is to smoothly blend the feedback from several nearby scenarios.

Chattering in combination with the MLI scheme If SensEIS feedback is scheduled as an MLI level, we can again encounter chattering. In this case, the chattering occurs if both MLI Level D and SensEIS level are used. An example is shown in Figure 7.3. The reason is that Level D and SensEIS level use linearizations that are independent of each other. Therefore, the feedback that Level D and SensEIS level compute differs slightly even if SensEIS level communicates its computed variables to Level D after each sampling time.



(a) Velocity trajectory



(b) Control trajectory

Figure 7.3.: Scheduling both Level D and the SensEIS level causes chattering in the EACC system application. Here, Level D is used at every 5th sampling time and otherwise SensEIS level is used.

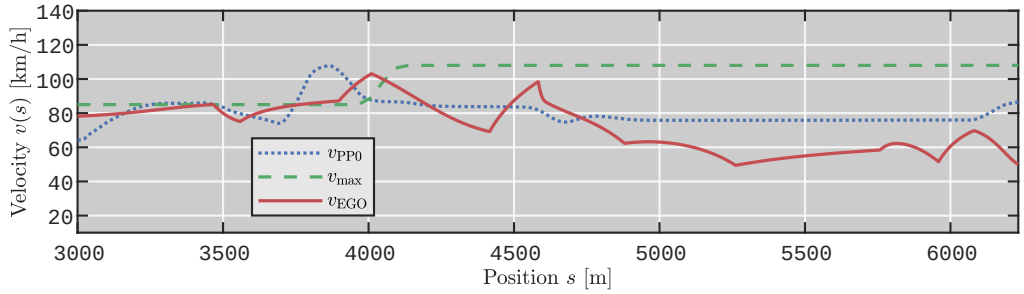
As a consequence, the control that is realized alternates between the control profiles that stem from Level D and SensEIS level. However, the chattering is less pronounced than the one discussed in the previous paragraph. We can also see in Figure 7.3 that the chattering can vanish again if the linearizations used by Level D and SensEIS level are close enough to each other. This again is the case if the current parameters are sufficiently close to a scenario and if the primal variables are close enough to the solution of this scenario.

We can mitigate this type of chattering by making sure that Level D and SensEIS level take into account the variables computed by each other. A strategy to achieve this is the following: We treat the latest control variables that were computed by the MLI scheme or SensEIS level as external inputs and penalize deviations from these controls. Let q^{old} be the latest control variables computed by either the MLI scheme or SensEIS level. We then change the objective function of the DMS NLP (7.10) to

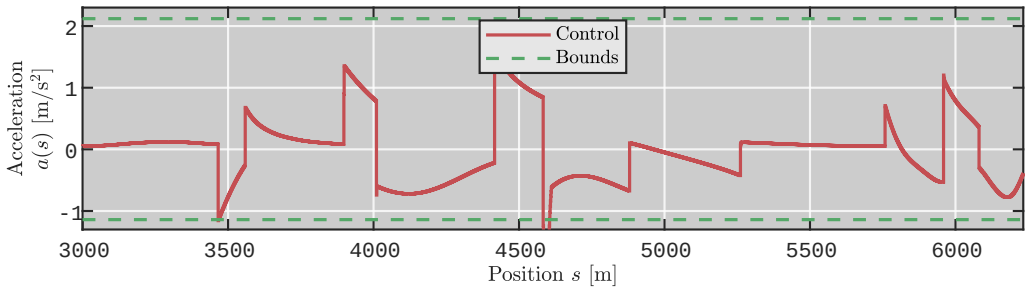
$$\sum_{m=0}^{M-1} \Psi_m\left(s_m, q_m; \rho, v_m\right) + \frac{1}{2}\left\|q_m - q_m^{\mathrm{old}}\right\|_{W_m}^2 + \Phi\left(s_M; \rho, v_M\right) \tag{7.30}$$

with suitable weight matrices $W_m,\ m = 0, \ldots, M-1$.

However, this approach is only a heuristic. It also requires the user to select appropriate weight matrices and introduces further external inputs that must be covered by scenarios.



(a) Velocity trajectory



(b) Control trajectory

Figure 7.4.: Big jumps can be seen in the control profile computed by using only Variant 1 of SensEIS feedback for the EACC system application even though we consider 5 different external inputs and one constant parameter and have solved 3600 scenarios in the offline phase.

Large numbers of scenarios required The heuristic proposed above to mitigate the chattering caused by using both Level D and SensEIS level requires us to treat the control profile from a previous sampling time as an external input. This increases the number of parameters that have to be considered in SensEIS feedback. Moreover, we have mentioned for both chattering types that they are also an artifact of scenarios not being close enough to the current parameters. Both issues point us to the challenge that SensEIS feedback requires a large number of scenarios to produce good feedback for all parameters in the online phase. The number of scenarios is driven by the number of parameters in the OCP (7.9) that should be considered and how many values for each parameter should be considered.

In Figure 7.4, we see the state and control profile for the EACC system application from Chapter 8 where we use only Variant 1 of SensEIS feedback. There, we have considered 5 different external inputs and one constant parameter for which we have solved 3600 scenarios in the offline phase. Despite this large scenario number, we can see big jumps in the control profile. These happen when the reference scenario is switched. The fact that these jumps are big indicates that the number of scenarios is still too small to obtain a satisfying control using SensEIS feedback only.

In our experience, the main challenge of SensEIS feedback is the large number of scenarios required to produce good feedback for all parameters in the online phase. Although the duration of the offline phase is not a critical issue for many applications, storing the data from the offline phase can become a problem. In the EACC system application, e.g., the memory on the embedded system is limited.

Therefore, to make SensEIS feedback well applicable to problems with a large number of parameters, future research should address the challenge of reducing the number of scenarios required to produce satisfactory feedback. We suggest two directions for this.

For the first, we recall that the impulse for the development of SensEIS feedback is the fact that in applications like the EACC system, some scenarios occur repeatedly. In fact, the vision was not that SensEIS feedback should be used during all driving situations, but only in situations that are similar to these scenarios. Furthermore, we assumed that there are only a few dozen, not thousands, of such scenarios. Moreover, if we wished to cover (almost) the entire parameter space, we should rather use techniques developed in explicit multiparametric NMPC, see for example the aforementioned work [69]. Therefore, we propose to precompute only typical scenarios and then apply SensEIS feedback if and only if the current parameters are sufficiently close to a precomputed scenario and otherwise rely on the MLI scheme. The computation time that is saved when SensEIS feedback is used can then be used to perform additional computations that improve the performance of the MLI scheme, like computing new derivatives, performing additional SQP iterations, or preparing upcoming sampling times in an advanced step fashion, cf. [145, 211].

[69]: Domínguez et al. (2011), "Recent Advances in Explicit Multiparametric Nonlinear Model Predictive Control"

[145]: Nurkanović et al. (2019), "The Advanced Step Real Time Iteration for NMPC"
[211]: Zavala et al. (2009), "The advanced-step NMPC controller: Optimality, stability and robustness"

The second proposal is to develop strategies to smoothly blend the feedback based on different scenarios together. The way we currently choose the reference scenario, see Equation (7.13) and Equation (7.29), we select only a single scenario as a reference and fully commit to the feedback resulting from this scenario. However, if the current parameters fall between multiple scenarios, it may be beneficial to consider the feedback from all scenarios that are close enough and then merge them. As we have already mentioned in the respective paragraph, we expect that such an approach will also solve the challenge of chattering caused by rapid reference scenario switches.

8. Application: Ecological Adaptive Cruise Control System

Throughout the entire thesis, the guiding application has been Ecological Adaptive Cruise Control (EACC) systems. EACC is a variant of Adaptive Cruise Control (ACC) that places a stronger emphasis on achieving an ecological driving style. ACC is an Advanced Driver-Assistance System (ADAS) that, as an enhancement of simple Cruise Control (CC) systems that maintain a user-defined velocity, adjusts the velocity of a vehicle to maintain a safe distance from the preceding vehicle (PP0). The goal of EACC systems is not only to maintain a safe distance from the PP0 but also to do so in an energy-efficient manner. At the same time, the driving style should remain comfortable for the passengers. Moreover, constraints such as speed limits or limitations of the vehicle's powertrain must be taken into account. Therefore, NMPC is a suitable control technique for EACC systems, as it allows us to incorporate all these requirements into the OCP (6.4) that governs the NMPC scheme. To implement NMPC-based EACC systems in real vehicles, we need highly efficient numerical methods for NMPC. Throughout this thesis, we have developed such numerical methods. In this chapter, we will apply these numerical methods to a realistic formulation of an EACC system for an electric vehicle.

We have explained NMPC in detail in Chapter 2. As described there, central components of any NMPC scheme include a model for the system of interest, an objective function to minimize, and possibly constraints on the system state and control. In our case, the model is represented by an ODE. These components are combined in an OCP of the form (6.4), which serves as the cornerstone of the NMPC scheme as described in Subsection 2.1.3. We have collaborated closely with an industrial partner to set up an OCP formulation that, while being an abstracted problem based on standard vehicle models, still captures several challenges arising in realistic EACC scenarios. Consequently, a successful application of our numerical methods to this abstracted problem serves as a compelling demonstration of their capability to address real-life applications.

In this chapter, we will first review other published applications of NMPC to real-life electric vehicle control problems in Section 8.1. To prepare for the formulation of the OCP, we will next present the underlying vehicle model in Section 8.2. Subsequently, we will develop the OCP (6.4) for our EACC system application in Section 8.3. Finally, we will present numerical results of our numerical methods applied to the EACC system application in Section 8.4.

Attention:

Out of consideration for our industrial partner, we cannot disclose some details regarding the OCP formulation. In particular, this concerns the LUTs and constants used, as well as details about certain terms in the objective function.

8.1. Literature review

Given the importance of vehicle traffic for both individuals and the economy, as well as its significant contribution to greenhouse gas emissions and global energy consumption, it is not surprising that the development of control techniques for vehicles has been a highly active area of research over the past two decades. A driving factor in this context has been the advent of increasingly powerful yet efficient mathematical control methods and computing units. These advancements have enabled the application of mathematical control techniques to real-life vehicle control problems. Consequently, a large body of literature has been published on this topic. For readers interested in exploring this field, we recommend the surveys [102, 149], which cover perception, planning, control, and coordination for connected and automated vehicles.

An important branch of vehicle control techniques is based on MPC methods. MPC-based methods have also been extensively studied. Therefore, we do not provide a detailed review of MPC-based techniques for vehicle control here. Instead, we refer interested readers to a comprehensive survey on this topic, such as [111].

As mentioned at the beginning of this chapter, our work focuses on an ecological driving assistance system, i.e., an Advanced Driver-Assistance System (ADAS). A wide range of MPC-based methods have been applied to ADASs, and we refer readers to the survey paper [141]. Several of these works specifically apply NMPC techniques to develop ecological Adaptive Cruise Control (ACC) systems. For a recent review of the development of ecological ACC systems, we recommend [148]. Notable examples leveraging NMPC include [168, 186, 187, 193, 198]. Typically, tracking NMPC formulations are employed, where either a desired distance to the PP0 is tracked [187, 193] or a precomputed speed profile is followed [186, 198].

The aforementioned works, along with most references cited therein, primarily focus on problem formulation rather than on numerical methods for efficiently solving the resulting NMPC problems. Instead, toolkits such as **FORCESPRO** [207], which builds on the solver **FORCES** [68], or **ACADO** [109] are often used to automatically generate solvers.

Works that place a stronger emphasis on numerical methods similar to ours and target similar applications include the following. In [118], heavy-duty trucks are considered, with the main numerical challenge being the treatment of mixed-integer NMPC problems. Heavy-duty trucks are also the focus of [136, 137], where the waste heat recovery system is of particular interest. As the RTI scheme described in Section 3.4 is also implemented in **ACADO**, it has been utilized for NMPC in autonomous vehicles, for example, in [168, 176, 209]. Further works applying the RTI scheme to similar applications include [79, 80]. To the best of our knowledge, the MLI scheme described in Section 3.5 has not yet been used to implement an ecological driving assistance system. In [81], an extension of the MLI scheme was applied to stabilize a vehicle after a disturbance, but not for ecological driving. In [136, p. 118], it was proposed as future research to use the MLI scheme for the presented problem, but this has not yet been realized.

[79]: Fors et al. (2023), "Long-Horizon Vehicle Planning and Control Through Real-Time Iterations"
[80]: Frasch et al. (2013), "An auto-generated nonlinear MPC algorithm for real-time obstacle avoidance of ground vehicles"
[81]: Frasch et al. (2012), "Mixed-Level Iteration Schemes for Nonlinear Model Predictive Control"
[102]: Guanetti et al. (2018), "Control of connected and automated vehicles: State of the art and future challenges"
[111]: Hrovat et al. (2012), "The development of Model Predictive Control in automotive industry: A survey"
[118]: Kirches et al. (2013), "Mixed-integer NMPC for predictive cruise control of heavy-duty trucks"
[136]: Merino (2018), "Real-time optimization for estimation and control: Application to waste heat recovery for heavy duty trucks"
[137]: Merino et al. (2018), "A Nonlinear Model-Predictive Control Scheme for a Heavy Duty Truck's Waste Heat Recovery System Featuring Moving Horizon Estimation"
[141]: Musa et al. (2021), "A Review of Model Predictive Controls Applied to Advanced Driver-Assistance Systems"
[148]: Pan et al. (2022), "A review of the development trend of adaptive cruise control for ecological driving"
[149]: Pendleton et al. (2017), "Perception, Planning, Control, and Coordination for Autonomous Vehicles"
[168]: Schmied et al. (2015), "Nonlinear MPC for Emission Efficient Cooperative Adaptive Cruise Control"
[176]: Siampis et al. (2018), "A Real-Time Nonlinear Model Predictive Control Strategy for Stabilization of an Electric Vehicle at the Limits of Handling"
[186]: Turri et al. (2017), "Cooperative Look-Ahead Control for Fuel-Efficient and Safe Heavy-Duty Vehicle Platooning"
[187]: Turri et al. (2017), "A model predictive controller for non-cooperative eco-platooning"
[193]: Vajedi et al. (2016), "Ecological Adaptive Cruise Controller for Plug-In Hybrid Electric Vehicles Using Nonlinear Model Predictive Control"
[198]: Weißmann et al. (2017), "Energy-Optimal Adaptive Cruise Control based on Model Predictive Control"
[209]: Zanon et al. (2014), "Model Predictive Control of Autonomous Vehicles"

Our contributions in this chapter are thus twofold. First, we present the first application of the MLI scheme to an ecological driving assistance system, thereby demonstrating its potential for deployment in real vehicle control. Second, we provide a proof of concept for SensEIS feedback from Chapter 7, along with our strategy for handling external inputs as described in Chapter 6.

8.2. Underlying vehicle model

For the movement of the road vehicle under consideration, we focus solely on its longitudinal dynamics. For its description in Subsection 8.2.1, we follow the popular textbook [105, Subsection 2.1.1]. In Subsection 8.2.2, we introduce the powers relevant to our OCP formulation. Table 8.1 summarizes all constants referenced in the following sections.

[105]: Guzzella et al. (2013), *Vehicle Propulsion Systems*

Table 8.1.: Constants used in the model of the vehicle dynamics.

Symbol	Meaning	Unit
A_f	Frontal area of the vehicle	m^2
c_d	Aerodynamic drag coefficient	dimensionless
c_r	Rolling friction coefficient	dimensionless
F_d	Disturbance forces	N
g	Standard gravity	$\mathrm{m\,s^{-2}}$
γ	Transmission ratio	dimensionless
m_v	Vehicle mass	kg
P_aux	Auxiliary power for HVAC	W
r_w	Radius of the wheels	m
ρ_a	Density of ambient air	$\mathrm{kg\,m^{-3}}$
Θ_e	Moment of inertia of engine components	$\mathrm{kg\,m^2}$
Θ_w	Moment of inertia of wheels	$\mathrm{kg\,m^2}$

8.2.1. Vehicle movement

If time t in s is used as the independent variable, the longitudinal position $s \in \mathbb{R}$ in m and velocity v in $\mathrm{m\,s^{-1}}$, depending on the acceleration a in $\mathrm{m\,s^{-2}}$, are given for $t_0 \leq t \in \mathbb{R}$ by the IVP

$$\begin{pmatrix} \frac{\mathrm{d}}{\mathrm{d}t} s(t) \\ \frac{\mathrm{d}}{\mathrm{d}t} v(t) \end{pmatrix} = \begin{pmatrix} v(t) \\ a(t) \end{pmatrix},$$
$$\begin{pmatrix} s(t_0) \\ v(t_0) \end{pmatrix} = \begin{pmatrix} s_0 \\ v_0 \end{pmatrix}.$$

If instead the position s is used as the independent variable, the time $t \in \mathbb{R}$ and velocity v, depending on the acceleration a, are given for $s_0 \leq s \in \mathbb{R}$ by the IVP

$$\begin{pmatrix} \frac{\mathrm{d}}{\mathrm{d}s} t(s) \\ \frac{\mathrm{d}}{\mathrm{d}s} v(s) \end{pmatrix} = \begin{pmatrix} \frac{1}{v(s)} \\ \frac{a(s)}{v(s)} \end{pmatrix}, \tag{8.1a}$$
$$\begin{pmatrix} t(s_0) \\ v(s_0) \end{pmatrix} = \begin{pmatrix} t_0 \\ v_0 \end{pmatrix}. \tag{8.1b}$$

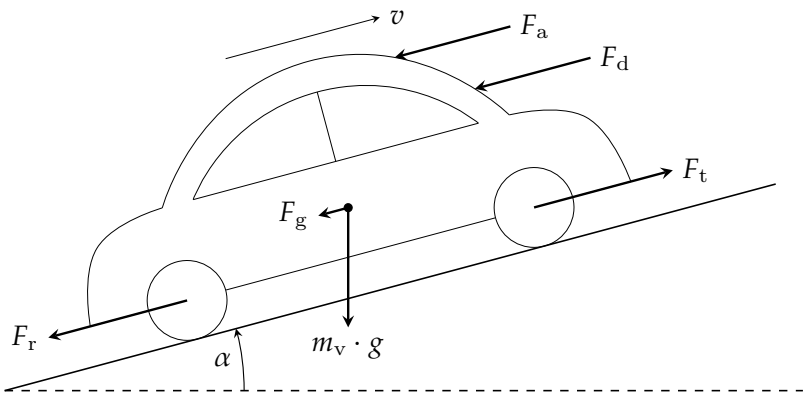


Figure 8.1.: Illustration of the balance of forces (8.2). Adapted from [105, Fig. 2.1]. The traction force F_t actively accelerates or decelerates the vehicle. The aerodynamic and rolling friction forces F_a and F_r, along with the disturbance forces F_d, decelerate the vehicle. The gravitational force F_g can either accelerate, decelerate, or have no effect, depending on the road inclination α.

In both cases, the acceleration a is determined by the balance of forces

$$m_\mathrm{v} a = F_\mathrm{t} - \left(F_\mathrm{a} + F_\mathrm{r} + F_\mathrm{g} + F_\mathrm{d}\right), \tag{8.2}$$

where

- F_t is the traction force,
- F_a is the aerodynamic friction force,
- F_r is the rolling friction force,
- F_g is the force caused by the gravitation if the vehicle is driving up- or downhill,
- F_d collects any other disturbing forces.

The rolling friction force and the gravitational force F_g depend on the road inclination α. Figure 8.1 schematically illustrates how the different forces act on the vehicle. Below, we briefly present the equations for these forces. For more details, refer to [105, Subsection 2.1.1]. The disturbance forces F_d are treated as constant.

[105]: Guzzella et al. (2013), *Vehicle Propulsion Systems*

Traction force F_t The traction force F_t is the force used to actively accelerate or decelerate the vehicle. In our application, where we consider an electric vehicle, the source of the traction force F_t is the force F_em exerted by the electric motor. However, part of the force F_em is used to accelerate the wheels and rotating parts within the powertrain. The force F_em is given as

Friction losses are accounted for as disturbance forces F_d.

$$F_\mathrm{em} = \frac{\gamma}{r_\mathrm{w}} T_\mathrm{em},$$

The constants γ and r_w are defined in Table 8.1.

where T_em is the torque exerted by the electric motor. The forces F_in that are exerted to accelerate the rotating parts are called inertial forces and can be written as

We assume no wheel slip. The constants $\gamma, r_\mathrm{w}, \Theta_\mathrm{w}, \Theta_\mathrm{e}$ are defined in Table 8.1.

$$F_\mathrm{in} = \left(\frac{1}{r_\mathrm{w}^2}\Theta_\mathrm{w} + \frac{\gamma^2}{r_\mathrm{w}^2}\Theta_\mathrm{e}\right)\frac{\mathrm{d}}{\mathrm{d}t}v.$$

The traction force F_t is then given as

$$\begin{aligned} F_\mathrm{t} &= F_\mathrm{em} - F_\mathrm{in} \\ &= \frac{\gamma}{r_\mathrm{w}} T_\mathrm{em} - \left(\frac{1}{r_\mathrm{w}^2}\Theta_\mathrm{w} + \frac{\gamma^2}{r_\mathrm{w}^2}\Theta_\mathrm{e}\right)\frac{\mathrm{d}}{\mathrm{d}t}v. \end{aligned} \tag{8.3}$$

Aerodynamic friction force F_a For the aerodynamic friction force F_a, we use the simplified expression

$$F_a = \frac{1}{2}\rho_a A_f c_d v^2.$$

The aerodynamic drag coefficient c_d is generally not constant and must be determined experimentally. However, it is common practice to assume it to be constant [105, p. 15].

As in [105, Section 2.1.1], we assume the vehicle has a prismatic shape with frontal area A_f. The constants ρ_a, A_f, c_d are defined in Table 8.1.

[105]: Guzzella et al. (2013), *Vehicle Propulsion Systems*

Rolling friction force F_r Similarly to the aerodynamic friction force, we use a simplified model for the rolling friction force F_r. Specifically, we assume the rolling friction coefficient c_r to be constant, which is a common simplification [105, p. 15]. For positive velocities, the rolling friction force F_r is modeled as

$$F_r = c_r m_v g \cos(\alpha),$$

where α is the road inclination in rad.

[105]: Guzzella et al. (2013), *Vehicle Propulsion Systems*

Gravitational force F_g The gravitational force is as usual given as

$$F_g = m_v g \sin(\alpha),$$

where again α is the inclination of the road in rad.

8.2.2. Computation of relevant powers

To compute the relevant powers for our OCP formulation, we first determine the torque T_{em} that the electric motor must deliver to achieve a desired acceleration or traction force F_t. The torque T_{em} can be derived from the balance of forces (8.2). First, we solve (8.2) for the traction force F_t and then use

$$T_{em} = \frac{r_w}{\gamma} F_{em}.$$

As indicated in Figure 8.2, the power required by the electric motor is computed from the torque T_{em} and the current angular speed ω_{em}, which is given by

$$\omega_{em} = \frac{\gamma}{r_w} v$$

using a bivariate LUT.

Reminder: Bivariate LUT

A bivariate Lookup Table (LUT) is a data set

$$\mathcal{D} := \{(x_\alpha, d_\alpha) \in \mathbb{R}^2 \times \mathbb{R} \mid 0 \le \alpha \le \alpha_{\max}\},$$

cf. Definition 5.3. Such LUTs can be interpolated in a shape-preserving way using our novel interpolation method proposed in Section 5.3.

In an electric vehicle, the power P_{em} of the electric motor is supplied by the battery. The battery also needs to provide auxiliary power P_{aux} for heating, ventilation, air conditioning (HVAC). We assume P_{aux} to be constant. The total power P_{batt} required from the battery is thus given by

$$P_{batt} = P_{em} + P_{aux}.$$

If P_{aux} were not constant, it could still be incorporated as an external input. We proposed novel techniques for incorporating external inputs into DMS and the RTI and MLI schemes in Chapter 6.

Delivering the power P_{batt} also incurs losses. These power losses P_{loss} in the battery depend on P_{batt} and the state of charge (SoC) of the

battery. This dependency is typically measured and provided as a bivariate LUT. The SoC could be included in the dynamics using the COULOMB counting method. For details, see [105, p. 113]. However, we assume a constant SoC in our model.

[105]: Guzzella et al. (2013), *Vehicle Propulsion Systems*

8.3. OCP formulation

We examine the three main components of OCP (6.4) separately. First, we specify our choice for the controls and differential states of interest in Subsection 8.3.1. Subsection 8.3.2 provides details on the structure of the objective function used. In Subsection 8.3.3, we present the constraints considered in our OCP formulation.

Figure 8.2 summarizes the most important relations between

- the control, denoted by u,
- the differential states, which are the velocity v and the time t,
- forces, denoted by F,
- powers, denoted by P,
- torques, denoted by T,
- the external inputs, which include information about the route and the first preceding vehicle (PP0),
- and the components of OCP (6.4).

These relationships are described in the previous Section 8.2 and the present section.

8.3.1. Choice for the control and differential states

As the LUT used to compute the electric motor power P_{em} cannot be evaluated in reverse (i.e., to compute the torque T_{em} from P_{em}), our options for the control choice are somewhat limited. The two most relevant options are the torque T_{em} and the vehicle's acceleration. Choosing the torque T_{em} has the advantage of being closer to what can be controlled in a real vehicle. However, in this case, bounds on the acceleration a would become nonlinear mixed state-control constraints. If we instead choose the acceleration as the control, these constraints become simple box constraints. We thus decide to use the acceleration as the control, i.e., we choose

$$u(\cdot) = a(\cdot),$$

The unit of the control u is thus $\mathrm{m\,s^{-2}}$.

where $a(s) \in \mathbb{R}$, in the OCP formulation (6.4).

Next, we need to decide whether to use time t or position s as the independent variable. Both options are common and have been used in similar work. We choose the position s as the independent variable for three reasons. First, the way we have, in discussion with our industrial partner, decided to model some comfort aspects in the objective function requires this choice. Second, bounds on the vehicle's velocity are given as functions of the position s. Choosing the position s as the free variable turns these bounds into simple box constraints instead of nonlinear mixed state-control constraints. Finally, external

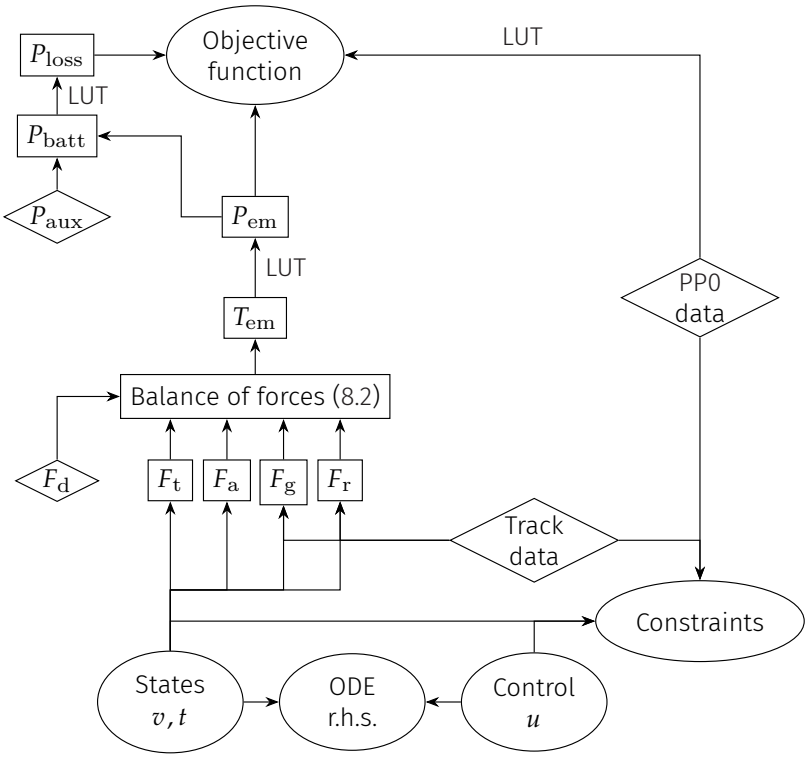


Figure 8.2.: Overview of the relationships between the most important quantities used in the EACC problem formulation. Ellipses denote components of the OCP (6.4). Wherever Lookup Tables (LUTs) are used, the arrows are annotated with "LUT ". Other parameters and external inputs (diamond-shaped), such as the auxiliary power P_{aux}, the disturbance force F_{d}, and the track and PP0 data, are omitted for clarity.

inputs, such as the road's inclination α, are also functions of the position s. Thus, to leverage our novel SensEIS feedback presented in Chapter 7, we need to choose the position s as the free variable. The drawback of this choice is that the velocity $v(s)$ must remain positive at all times due to the form of the ODE (8.1a). Therefore, stopping the vehicle is not considered in our experiments.

The differential states x in the OCP (6.4) are thus

$$x(\cdot) = \begin{pmatrix} v(\cdot) \\ t(\cdot) \end{pmatrix} \in \mathbb{R}^2.$$

8.3.2. Optimization criteria

The goal of an ecological driving assistance system is to achieve highly efficient driving behavior, i.e., to reduce energy consumption. At the same time, we must ensure that the resulting driving behavior is acceptable to a human driver. To achieve this, we include comfort aspects in the OCP (6.4). In close collaboration with our industrial partner, we have developed an objective function that incorporates such comfort aspects. These include, but are not limited to, smooth following of the PP0 and avoidance of rapid velocity changes. The objective function Ψ is thus given by

$$\Psi = w_0 P_{\text{em}} + w_1 P_{\text{loss}} + \sum_{i=1}^{k} w_{i+1} \Psi_i^{\text{c}}, \tag{8.4}$$

where $k \in \mathbb{N}$ and $w_i \in \mathbb{R},\ i = 0, \dots, k+1$ are constant weights, and $\Psi_i^{\text{c}},\ i = 1, \dots, k$ are functions introduced for the comfort aspects.

From a mathematical perspective, the objective function Ψ as given in Equation (8.4) presents the following challenges. First, external inputs in the form of track data and PP0 data influence all its components. Second, LUTs are used in modeling some components, such as the power P_{em}. Third, in its original form, some comfort aspects Ψ_i^c are not differentiable. Fortunately, we have the tools to address these challenges. We have extended the methodology of the DMS method and the RTI and MLI schemes to include external inputs in Chapter 6. For evaluating the LUTs, we employ our novel shape-preserving interpolation method from Chapter 5. For the non-differentiable comfort aspects, we use differentiable approximations.

8.3.3. Constraints

We have already mentioned some relevant constraints. These include bounds for the acceleration

$$a_{\min} \leq u(s) \leq a_{\max},$$

for all $s_0 \leq s \in \mathbb{R}$, where $a_{\min} < a_{\max} \in \mathbb{R}$, and upper speed limits

$$v(s) \leq v_{\max}(s)$$

for all $s_0 \leq s \in \mathbb{R}$. Additionally, we enforce a minimal time gap between the controlled vehicle and the PP0. Let $t_{\mathrm{PP0}}(s)$ be the time at which the PP0 is at position s, and let $\Delta t_{\min} > 0$ be a constant minimal time gap. The respective constraint can then be formulated as

$$0 \leq t(s) - t_{\mathrm{PP0}}(s) - \Delta t_{\min}.$$

Finally, we can theoretically include nonlinear mixed state-control constraints to account for limitations on other quantities, such as the torque T_{em}. However, in our numerical experiments, the solution has always remained safely within these limitations. Therefore, we have ultimately omitted these constraints.

8.4. Numerical results

In this section, we present numerical results for the application of our numerical methods to the EACC system problem as described in the previous subsections. We report results for four different choices of numerical methods, which differ as follows:

(i) We apply the standard MLI scheme as described in Section 3.5. The external inputs are not explicitly treated, instead, they are interpreted as functions of the differential states based on Equations (6.2) and (6.3). A novel aspect is the use of our newly developed interpolation method from Chapter 5 for interpolating the multivariate LUTs.

(ii) We additionally use our novel strategy to incorporate a priori known external inputs into the MLI scheme, as detailed in Subsection 6.2.3.

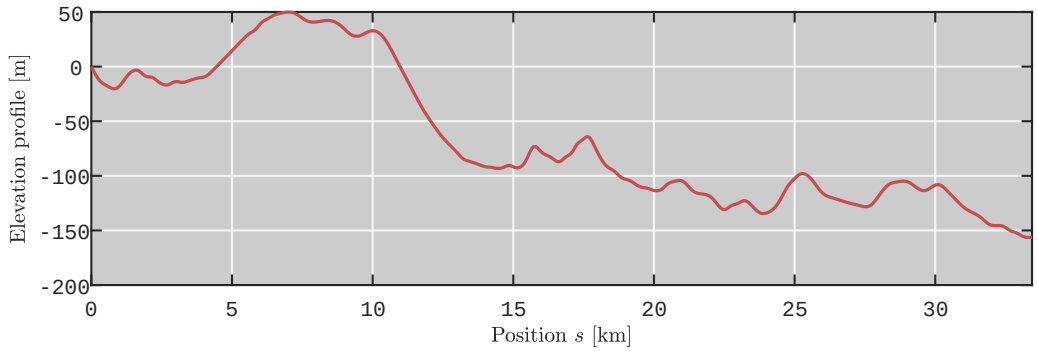


Figure 8.3.: Elevation profile of the test drive route.

(iii) We employ our novel SensEIS level from Chapter 7 within the MLI scheme.
(iv) We use SensEIS feedback as a standalone method.

More details on the specific method choices are provided in the respective subsections. In all our numerical experiments, the road inclination α and the driving behavior of the PP0 are derived from post-processed measurements of a real-world test drive provided by our industrial partner. Similarly, the maximum velocity $v_{\max}$ is based on this test drive. Figure 8.3 shows the elevation profile of the test drive route. We make the simplifying assumption that all external inputs, including the PP0 information, are known a priori. Looking ahead to scenarios where we control fleets of vehicles, this simplification is not overly unrealistic. If the PP0 is driven by a human driver, we can only predict its future velocity. As vehicle velocity prediction is an important task for many ADASs, it has been addressed by several researchers. For a concise overview of common approaches to vehicle velocity prediction, we refer to [161].

[161]: Rezaei et al. (2015), "Prediction of Vehicle Velocity for Model Predictive Control"

8.4.1. MLI without explicit treatment of external inputs

In our first experiment, we apply the standard MLI scheme as described in Section 3.5.The external inputs are not explicitly treated, instead, they are interpreted as functions of the differential states based on Equations (6.2) and (6.3). The NMPC prediction horizon length T_{hor} is $1060\,\text{m}$, divided into 53 shooting intervals of $20\,\text{m}$ each. The sampling period T is $2\,\text{m}$. Our control variable, as described in Subsection 8.3.1, is the acceleration a of the vehicle, given in $\text{m}\,\text{s}^{-2}$, with bounds $a_{\min} = -1.14\,\text{m}\,\text{s}^{-2}$ and $a_{\max} = 2.12\,\text{m}\,\text{s}^{-2}$. We use constant basis functions for the control and the MLI schedule $\text{A}^1\text{B}^3\text{C}^4\text{D}^7$.

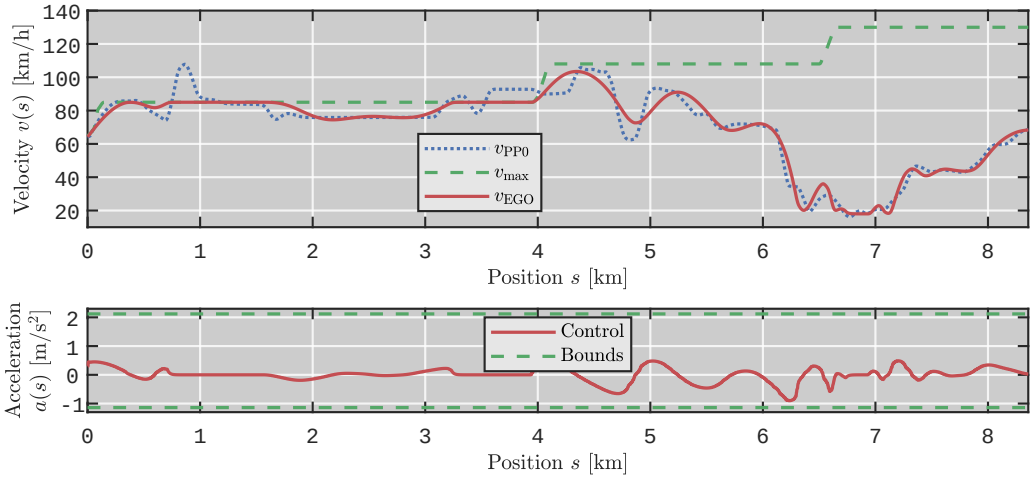
Reminder: NMPC and DMS

The sampling period T and the prediction horizon T_{hor} are defined in Definitions 2.1 – 2.2. Shooting intervals are defined in Definition 3.1, and constant control basis functions are described in Example 3.1. The MLI schedule is explained in Subsection 3.5.2.

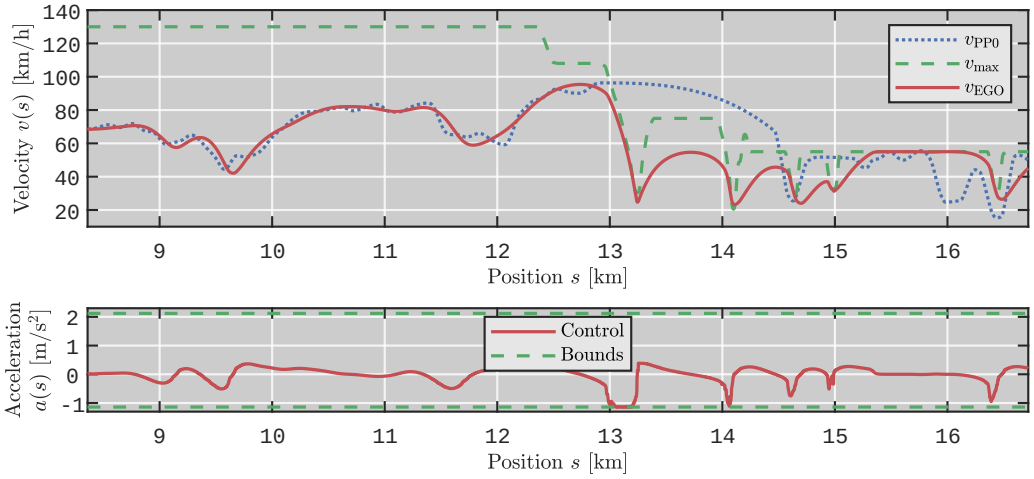
The initial position s_0 is $0\,\text{m}$. The initial differential states are

$$t_0 = 2\,\text{s} = t_{\text{PP0}}(s_0) + 2\,\text{s},$$
$$v_0 = 17.8667\,\text{m}\,\text{s}^{-1} = v_{\text{PP0}}(t_0).$$

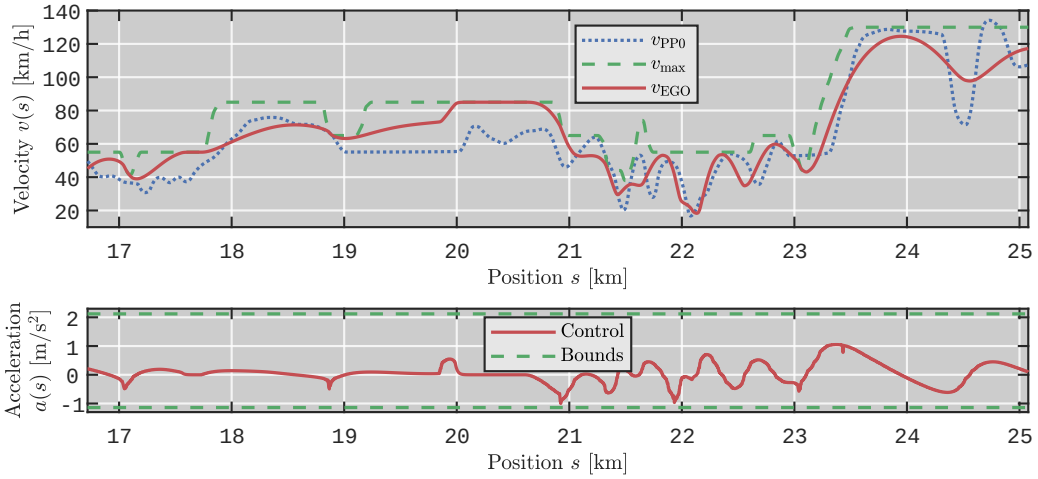
The initial time gap between our controlled vehicle and the PP0 is thus $2\,\text{s}$, and both vehicles move with the same initial velocity.



(a) Results for the section of the driving route from $s = 0\,\mathrm{m}$ to $s = 8358\,\mathrm{m}$.

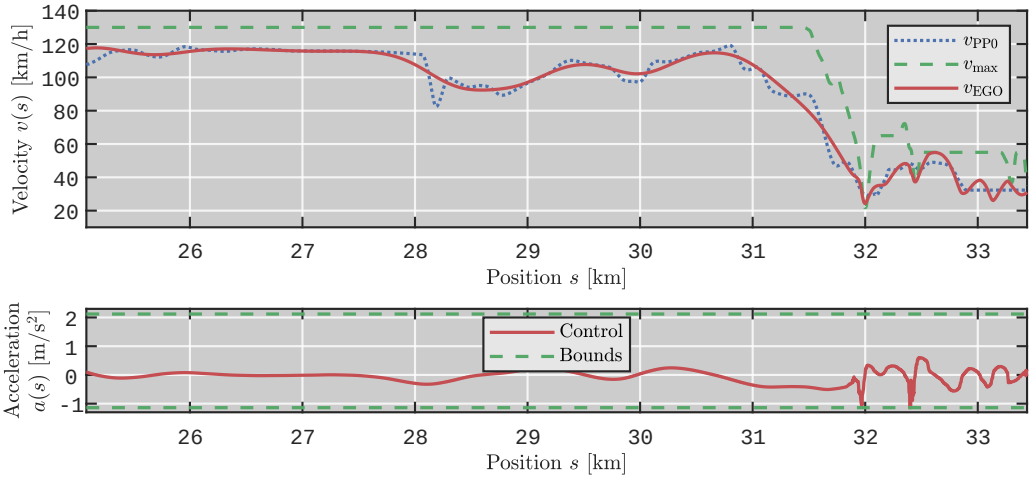


(b) Results for the section of the driving route from $s = 8360\,\mathrm{m}$ to $s = 16\,718\,\mathrm{m}$.



(c) Results for the section of the driving route from $s = 16\,720\,\mathrm{m}$ to $s = 25\,078\,\mathrm{m}$.

Figure 8.4.: Velocity and acceleration trajectories computed using the MLI scheme without explicit treatment of external inputs, as described in Subsection 8.4.1. The involved LUTs are interpolated using our novel method presented in Chapter 5. The trajectories for the entire driving route are split into four parts, displayed in Figures 8.4a – 8.4d.



(d) Results for the section of the driving route from $s = 25\,080\,\text{m}$ to $s = 33\,438\,\text{m}$.

Velocity and acceleration trajectory computed using the MLI scheme without explicit treatment of external inputs as described in Subsection 8.4.1. The involved LUTs are interpolated using our novel method that we presented in Chapter 5. The trajectories for the entire driving route are split into four parts which are displayed in Figures 8.4a – 8.4d.

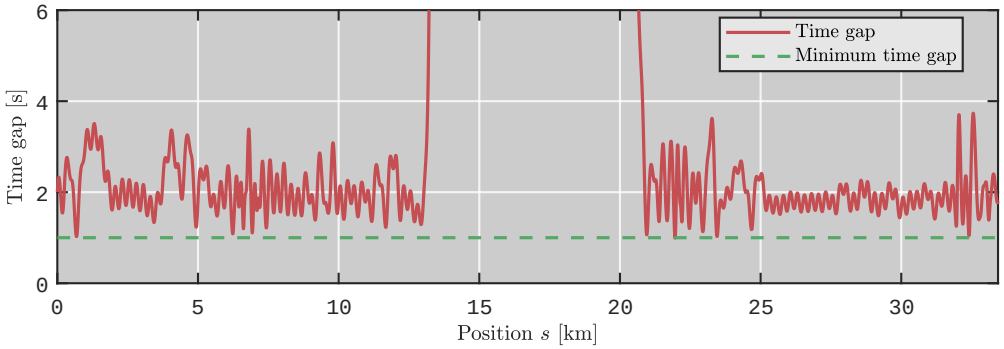


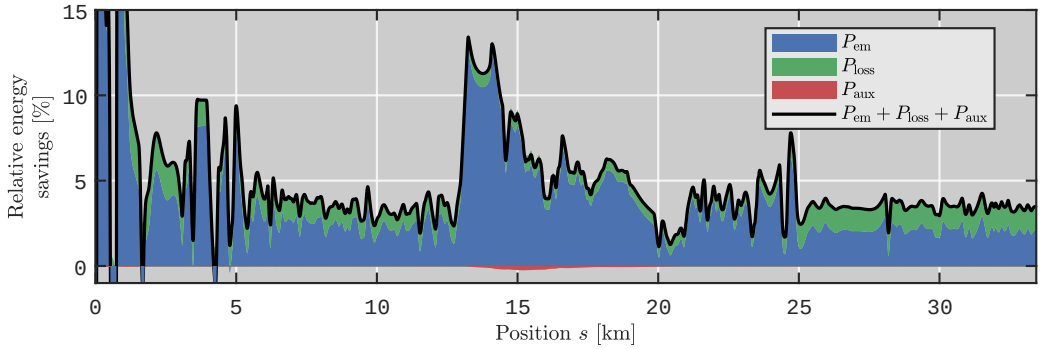
Figure 8.5.: The time gap $t(s) - t_{\text{PP0}}(s)$ between our controlled vehicle and the PP0 in the first numerical experiment using the MLI scheme without explicit treatment of external inputs. The minimum time gap is never violated. The initial time gap is $2\,\text{s}$, and the final time gap is approximately $1.8\,\text{s}$.

Figure 8.4 shows the results for the velocity and control (i.e., acceleration) trajectories. We observe that the resulting velocity adheres to its upper bound most of the time. Only upon close inspection, as seen in Figure 8.4b at $s \approx 14\,\text{km}$, do we notice slight violations. These violations occur because some of the arising QPs in this part of the driving route are infeasible. In such cases, we reuse the result from the last successful sampling time. The control bounds are satisfied throughout the entire driving route.

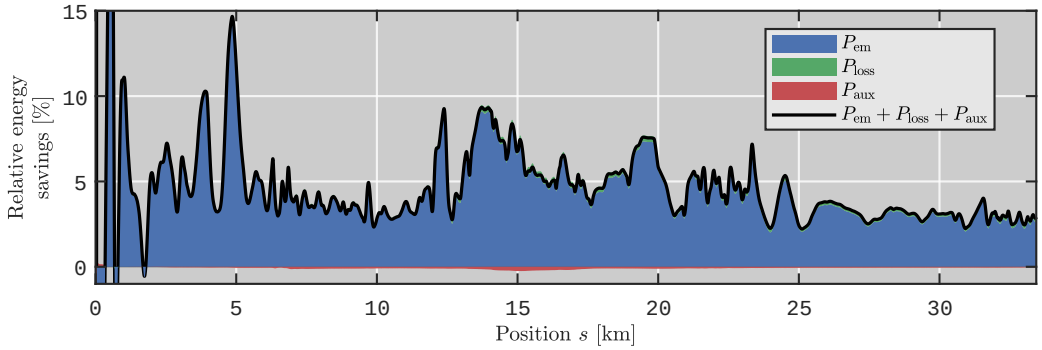
If several subsequent QPs are infeasible, we shift the control variables accordingly.

Furthermore, Figure 8.5 shows that our controlled vehicle maintains the minimum time gap $\Delta t_{\min} = 1\,\text{s}$ at all times. Moreover, the final time gap is approximately $1.8\,\text{s}$, about $0.2\,\text{s}$ shorter than the initial time gap of $2\,\text{s}$. In other words, the travel time of our controlled vehicle is slightly shorter than that of the PP0. Thus, the energy savings of more[1] than $3.4\,\%$ achieved compared to the PP0 are not simply due to driving slower on average but rather due to driving more efficiently. Additionally, we compare our energy consumption to that of a reference velocity, which is the speed profile of a human driver who followed the PP0 during the measurement drive.

1: The acceleration used by the PP0 leads to power requests to the electric motor that exceed the bounds of the LUT available for the battery losses. In such cases, we used the maximum loss given in the LUT. As losses tend to increase with larger power requests, we expect the actual savings to exceed $3.4\,\%$.



(a) Comparison to the PP0. The final relative energy savings are approximately $3.4\,\%$.



(b) Comparison to the reference velocity. The final relative energy savings are approximately $2.9\,\%$.

Figure 8.6.: Relative energy savings of our controlled vehicle separated into the three components P_{em}, P_{loss} and P_{aux}. The total energy consumption of our controlled vehicle is about $4.22\,\text{kW h}$.

The respective relative energy savings over the driving route are presented in Figure 8.6. As we do not possess any information about the vehicle type of the PP0, we use the same model to compute the energy consumption of the PP0 as for our controlled vehicle.

We observe that the energy savings are primarily due to reductions in the power P_{em} requested from the electric motor. Around $s = 15\,\text{km}$, the PP0 violates the velocity bounds, thereby building up a time gap to our vehicle, as also seen in Figure 8.5. Consequently, the PP0 saves auxiliary power P_{aux} compared to our vehicle.

In summary, the control and velocity profile computed using our numerical methods result in a driving behavior that

- obeys all constraints, except for a few meters,
- is slightly faster than the PP0,
- and still achieves energy savings of about $3.4\,\%$.

Reminder: Settings 1st experiment

- prediction horizon $T_{\text{hor}} = 1060\,\text{m}$,
- 53 shooting intervals of $20\,\text{m}$ length each,
- sampling period $T = 2\,\text{m}$,
- control: a in m s^{-2} with bounds $a_{\min} = -1.14\,\text{m s}^{-2}$ and $a_{\max} = 2.12\,\text{m s}^{-2}$,
- MLI schedule: $\text{A}^1\text{B}^3\text{C}^4\text{D}^7$,
- $t_0 = 2\,\text{s} = t_{\text{PP0}}(s_0) + 2\,\text{s}$,
- $v_0 = 17.8667\,\text{m} = v_{\text{PP0}}(t_0)$

8.4.2. MLI with explicit treatment of external inputs

In our second numerical experiment, we treat the external inputs explicitly using the technique described in Chapter 6, particularly in Subsection 6.2.3. We assume that the external inputs are known a priori. Constant basis functions and the same mapping ζ as in Example 6.1 are used for external input discretization. The remaining settings are identical to those in the first experiment from Subsection 8.4.1.

Figure 8.7 shows the results obtained with the explicit treatment of external inputs. The results are mostly very similar to those displayed in Figure 8.4 for the first numerical experiment. However, examining the time gap between our controlled vehicle and the PP0 shown in Figure 8.8, we observe that the minimum time gap $\Delta t_{\min} = 1\,\mathrm{s}$ is occasionally violated. Our observation is that this treatment of external inputs slightly increases the likelihood of obtaining infeasible QPs, likely due to additional disturbances caused by updating the external input discretization.

On the positive side, explicitly treating the external inputs increases the relative energy savings, shown in Figure 8.9, by up to about $4.3\,\%$ compared to the PP0 and $3.8\,\%$ compared to the measurement drive.

An obvious conjecture is that the increased savings compared to the first numerical experiment are only achieved because the time gap constraint is occasionally violated. However, the comparison of energy consumption without and with explicit treatment of external inputs shown in Figure 8.10 reveals that the increased savings are primarily achieved when the minimum time gap is not violated. In fact, Figure 8.10 suggests that intermediate violations of the minimum time gap are disadvantageous because the controller quickly decelerates our vehicle to satisfy the time gap constraint again. This behavior can also be observed when comparing the control profiles from Figure 8.4a and Figure 8.7a between $s = 6\,\mathrm{km}$ and $s = 7\,\mathrm{km}$.

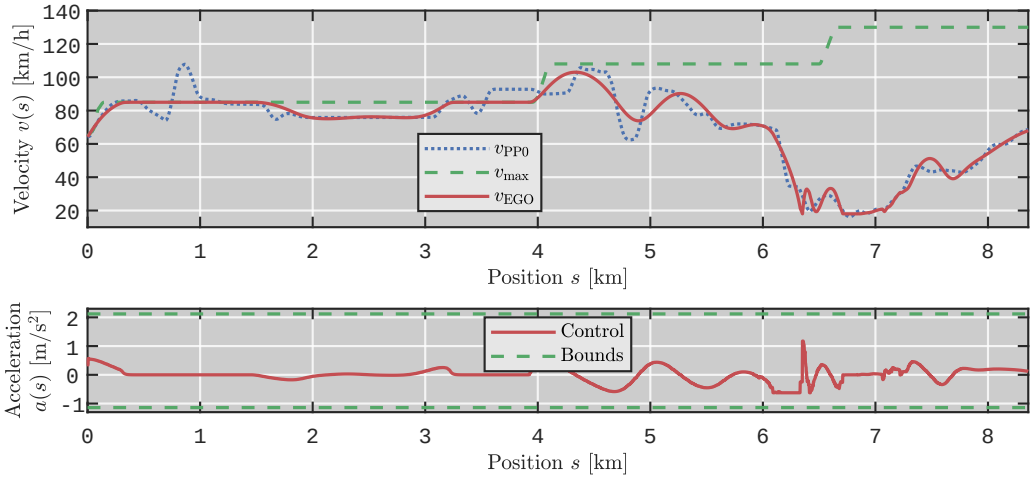
In summary, our findings from the second numerical experiment are that treating external inputs as described in Subsection 6.2.3 leads to

- a slight increase in the number of sampling times where the QP is infeasible, likely causing occasional violations of the time gap constraints,
- but even greater energy savings compared to the first numerical experiment.

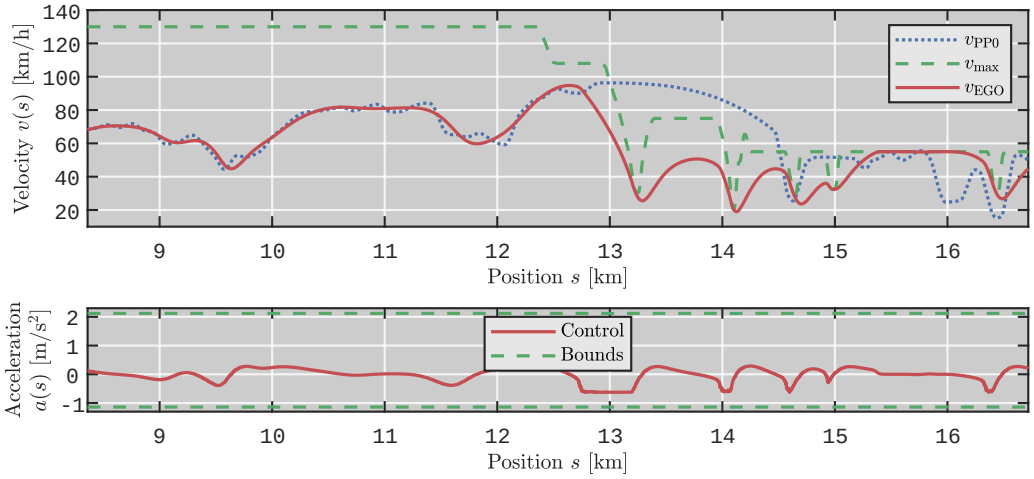
Thus, we conclude that treating the external inputs using our proposed approach is promising. However, further research is required to avoid constraint violations.

8.4.3. MLI with SensEIS level

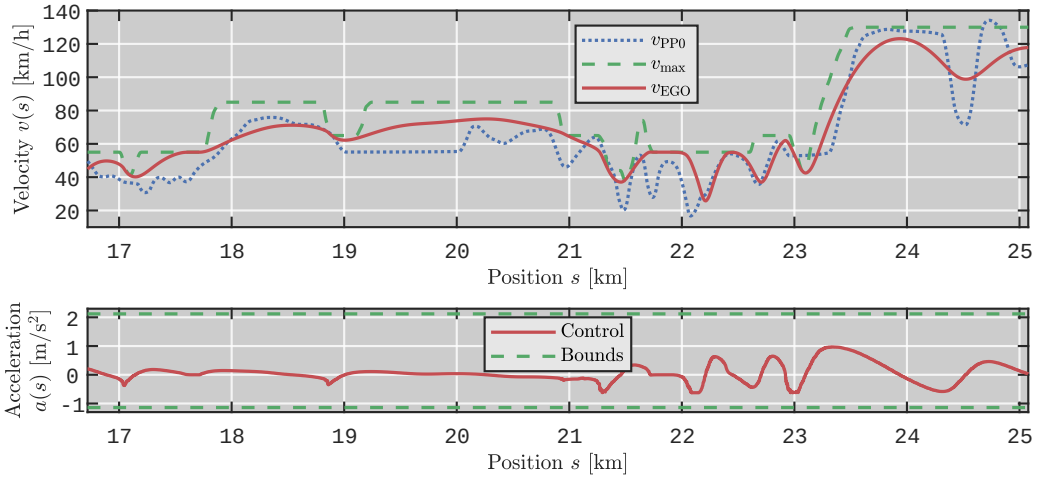
We now investigate the performance of an MLI schedule where SensEIS level is also used, as described in Subsection 7.3.4. In this experiment, the NMPC prediction horizon T_{hor} is $1050\,\mathrm{m}$, divided into 75 shooting intervals of $14\,\mathrm{m}$ each. The sampling period remains $T = 2\,\mathrm{m}$. Again, the acceleration a in $\mathrm{m\,s^{-2}}$ is used as the control. The MLI schedule is modified such that Level D is performed at every 4th sampling time, while SensEIS level is used at all other sampling times.



(a) Results for the section of the driving route from $s = 0\,\mathrm{m}$ to $s = 8358\,\mathrm{m}$.

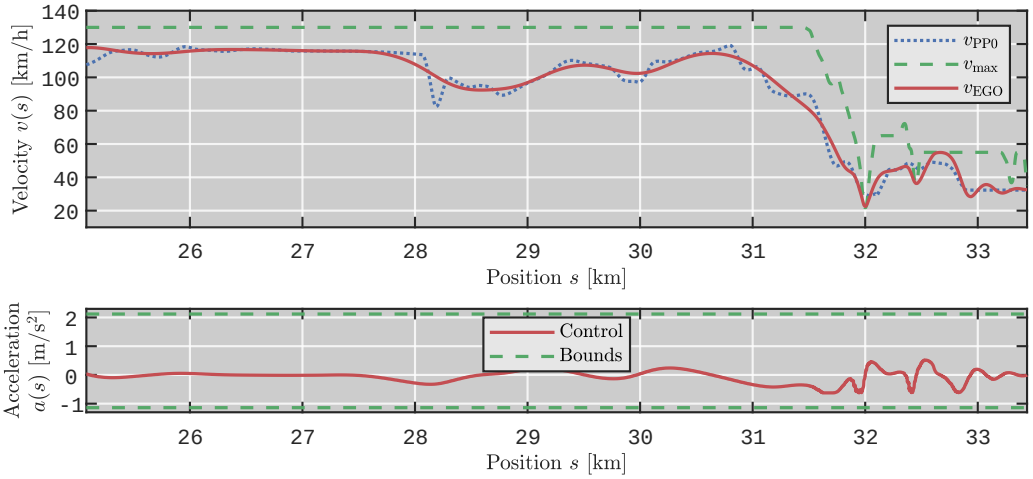


(b) Results for the section of the driving route from $s = 8360\,\mathrm{m}$ to $s = 16\,718\,\mathrm{m}$.



(c) Results for the section of the driving route from $s = 16\,720\,\mathrm{m}$ to $s = 25\,078\,\mathrm{m}$.

Figure 8.7.: Velocity and acceleration trajectories computed using the MLI scheme with explicit treatment of external inputs as described in Subsection 8.4.2. The involved LUTs are interpolated using our novel method presented in Chapter 5. The trajectories for the entire driving route are split into four parts, displayed in Figures 8.7a – 8.7d.



(d) Results for the section of the driving route from $s = 25\,080\,\mathrm{m}$ to $s = 33\,438\,\mathrm{m}$.

Velocity and acceleration trajectories computed using the MLI scheme with explicit treatment of external inputs as described in Subsection 8.4.2. The involved LUTs are interpolated using our novel method presented in Chapter 5. The trajectories for the entire driving route are split into four parts, displayed in Figures 8.7a – 8.7d.

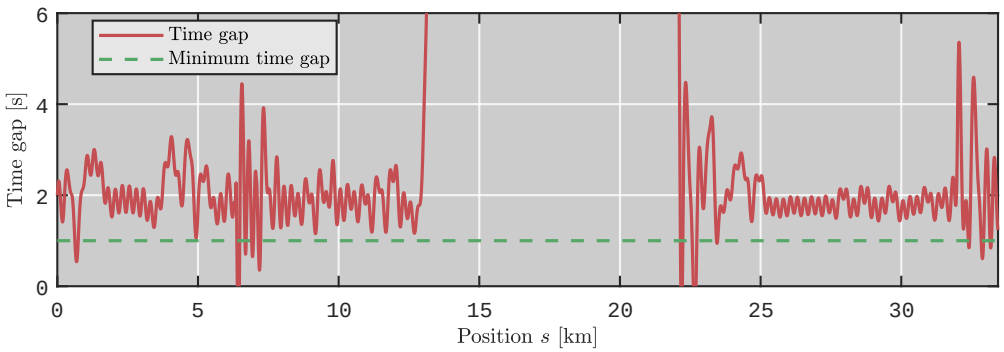


Figure 8.8.: The time gap $t(s) - t_{\mathrm{PP0}}(s)$ between our controlled vehicle and the PP0 in the second numerical experiment using the MLI scheme with explicit treatment of external inputs. The minimum time gap is occasionally violated. The initial time gap is $2\,\mathrm{s}$, and the final time gap is approximately $1.2\,\mathrm{s}$.

In the scenarios, the free variable s ranges from $s = 0$ to $s = T_{\mathrm{hor}}$. For external inputs used in SensEIS level, we choose

- the road inclination α,
- the maximum velocity $v_{\max}$,
- the velocity v_{PP0} of the PP0,
- the relative PP0 time $\tilde{t}_{\mathrm{PP0}}$, defined as the time gap plus the time elapsed since the PP0 was at the sampling position s^j, i.e., $t_{\mathrm{PP0}}(s^j) - t(s^j) + \int_0^{T_{\mathrm{hor}}} \frac{1}{v_{\mathrm{PP0}}(s)} \mathrm{d}s$

and as constant parameters

- the current velocity of our controlled vehicle $v(s)$,
- the control from the previous sampling time, i.e., the current acceleration of our vehicle $a(s)$.

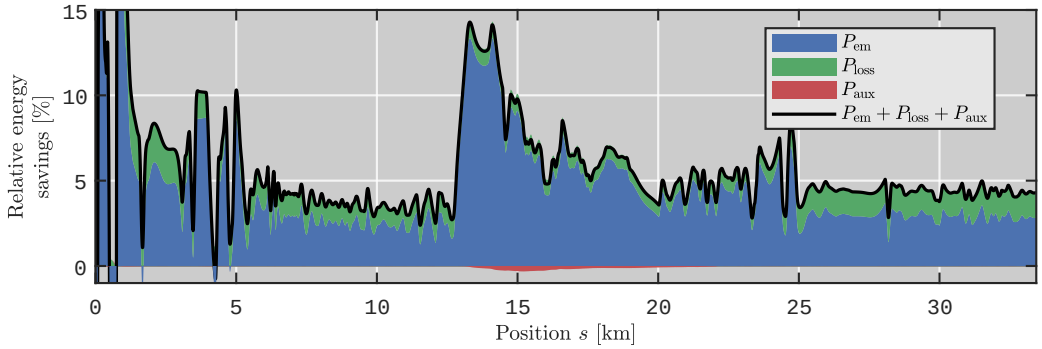
For the external inputs, we use constant basis functions and the map ζ as in Example 6.1. In the scenarios, we further use the same value for each shooting interval, i.e., we set

$$v_0 = v_1 = \ldots = v_M.$$

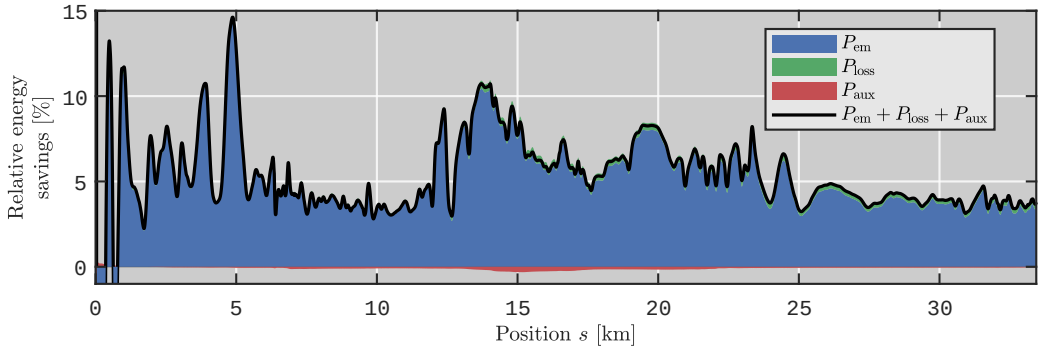
It is important that the external inputs are chosen such that they do not increase (or decrease) over the time the vehicle is controlled. For example, choosing the time $t_{\mathrm{PP0}}(s)$ at which the PP0 reaches the position s is a poor choice, as this value constantly increases. Instead, it is more appropriate to consider the time gap between our vehicle and the PP0 plus the time elapsed since the PP0 was at the sampling position, which is exactly what we do.

Reminder: v_m

v_m denotes the coefficients of the basis functions of the external inputs for the shooting interval with index m, cf. Subsection 6.1.1.



(a) Comparison to the PP0. The final relative energy savings are approximately $4.3\,\%$.



(b) Comparison to the reference velocity. The final relative energy savings are approximately $3.8\,\%$.

Figure 8.9.: Relative energy savings of our controlled vehicle with explicit treatment of external inputs, separated into the three components P_{em}, P_{loss}, and P_{aux}. The total energy consumption of our controlled vehicle is about $4.18\,\mathrm{kW\,h}$.

The values for each external input and constant parameter used in the scenarios are given in Table 8.2. Since these quantities are independent of each other, we precompute a scenario for each possible combination of values, resulting in 3600 scenarios. Additionally, Table 8.2 includes the inverse of the weights used in the scenario selection for each external input and constant parameter, cf. Equation (7.13).

Moreover, we

- apply the step size strategy Algorithm 7.3 to avoid inequality violations when using the feedback matrix-based variant,
- adaptively switch between the QP- and the feedback matrix-based variant, as described in the first paragraph of Section 7.4,
- use the modified reference index selection Equation (7.29) with $\gamma = 0.8$,
- and use the modified objective Equation (7.30) to better align the MLI feedback and SensEIS feedback.

Figure 8.11 shows the results obtained with the resulting controller. Unfortunately, we were only able to control the vehicle up to $s = 4836\,\mathrm{m}$. Beyond this point, infeasible QPs occur too frequently, leading to unreasonable controller behavior. Additionally, minor chattering in the controls persists because the MLI feedback and SensEIS feedback slightly disagree, even with the modified objective Equation (7.30). At least, Figure 8.12 shows that the minimum time gap is maintained except for a brief distance.

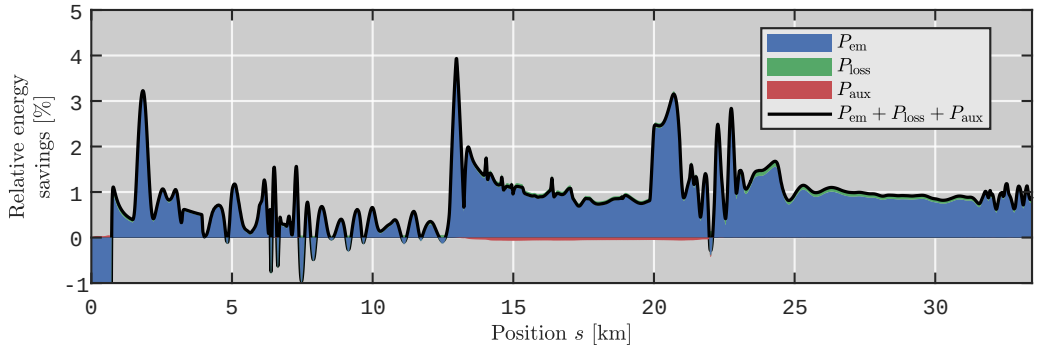


Figure 8.10.: Relative energy savings of our controlled vehicle from the second numerical experiment compared to the first numerical experiment. The energy savings are split into the three components P_{em}, P_{loss}, and P_{aux}. The final relative energy savings are approximately $0.9\,\%$.

Table 8.2.: Values for the external inputs and constant parameters in the precomputed scenarios for SensEIS level. Here, s^j denotes the current sampling position.

Quantity	Values	Inverse weight	Unit
road inclination α	$\{-0.025,\ 0.0, 0.025\}$	0.015	rad
maximum velocity $v_{\max}$	$\{15.2778, 23.6111\}$	23.2	$\mathrm{m\,s^{-1}}$
velocity v_{PP0} of the PP0	$\{9.0,\ 15.0,\ 22.5, 31.0\}$	23.6	$\mathrm{m\,s^{-1}}$
relative PP0 time $\tilde{t}_{\mathrm{PP0}}(s)$	$\{1.25,\ 1.75,\ 2.5,\ 4.0,\ 7.5, 15.0\} + \int_0^s \frac{1}{v_{\mathrm{PP0}}(\sigma)} \mathrm{d}\sigma$	22.9	s
current velocity $v(s^j)$	$\{0.5,\ 0.7,\ 0.9,\ 1.0, 1.1\} \cdot v_{\max}$	20.0	$\mathrm{m\,s^{-1}}$
previous control	$\{-1.0,\ -0.5,\ 0.0,\ 0.5, 1.25\}$	1.0	$\mathrm{m\,s^{-2}}$

In summary, the third numerical experiment confirms the challenges of SensEIS level discussed in Section 7.4, particularly the chattering behavior of the control when SensEIS level and other MLI levels are used together.

8.4.4. SensEIS feedback as standalone method

Finally, we use SensEIS feedback as a standalone method to control our vehicle in the described EACC system application. The setup is as described for the third numerical experiment in Subsection 8.4.3 except for two changes. First, we now only use SensEIS feedback and no MLI level at all sampling points. Second, we reduce the sampling period to $T = 1\,\mathrm{m}$. This decision can be justified as SensEIS feedback is significantly faster than Level D of the MLI scheme and thus we can use a shorter sampling period and still expect that the feedback would be available in time, even on vehicle hardware.

Figures 8.13 and 8.14 show the results that we obtain with SensEIS feedback. Comparing these results to the one for MLI with SensEIS level, we see that the overall behaviour is similar, but without chattering in the control. Unfortunately, also SensEIS feedback leads to infeasible QPs shortly after $s = 4.8\,\mathrm{km}$ after which the controls become unusable.

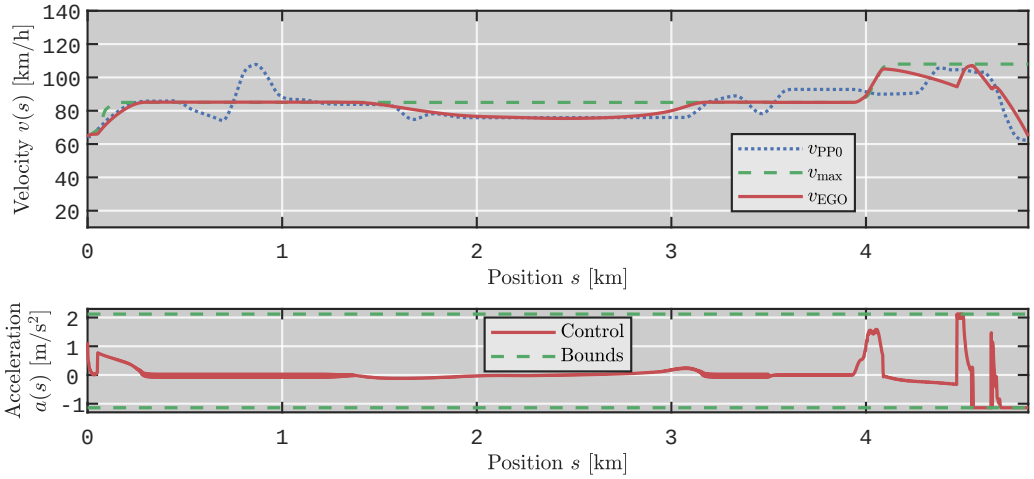


Figure 8.11.: Velocity and acceleration trajectories computed using the MLI scheme where Level D is used at every 4th sampling time and SensEIS level at all other sampling times. The involved LUTs are interpolated using our novel method presented in Chapter 5. After $s = 4836\,\mathrm{m}$, the controller produces unusable results.

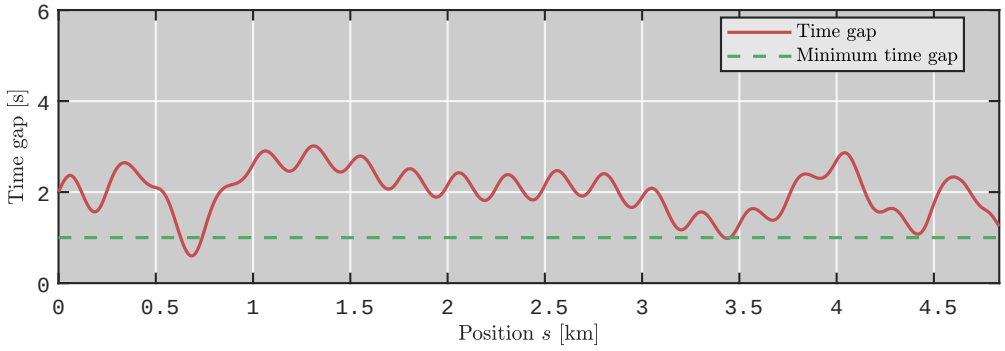


Figure 8.12.: The time gap $t(s) - t_{PP0}(s)$ between our controlled vehicle and the PP0 in the third numerical experiment where we use the MLI scheme with SensEIS level. The minimum time gap is briefly undercut once.

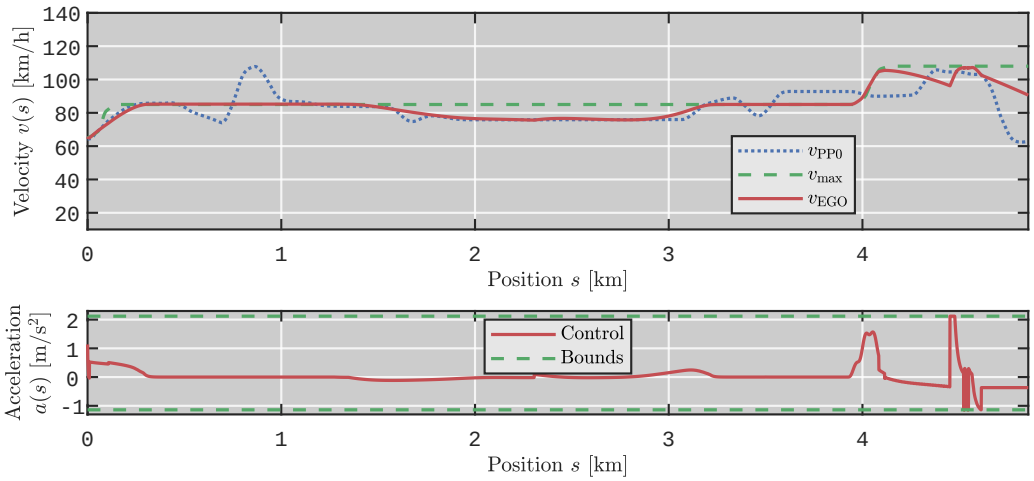


Figure 8.13.: Velocity and acceleration trajectory computed using SensEIS feedback at all sampling times. The involved LUTs are interpolated using our novel method that we presented in Chapter 5. After $s = 4857\,\mathrm{m}$ the controller leads to unusable results.

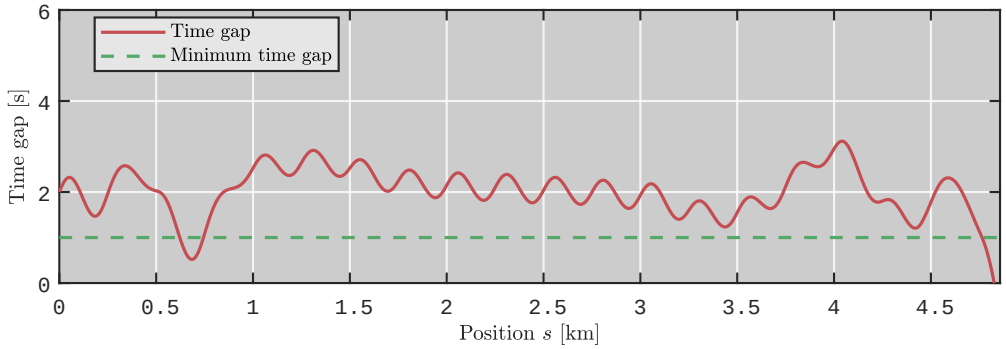


Figure 8.14.: The time gap $t(s) - t_{\mathrm{PP0}}(s)$ between the our controlled vehicle and the PP0 in the fourth numerical experiment where we use SensEIS feedback at all sampling times. The minimum time gap is briefly undercut once and again towards the end, where SensEIS feedback leads to unusable controls.

Both, the third numerical experiment discussed in Subsection 8.4.3 and the current experiment suggest that for such an application with many relevant external inputs and constant parameters even 3600 scenarios are not sufficiently many. This strengthens our previous conclusion from Section 7.4 that the challenge of considering enough parameters in sufficient detail to get good feedback from SensEIS feedback for all parameters, without exploding the number of scenarios to solve, is the Achilles heel of SensEIS feedback. Finding remedies for this problem has to be the primary task for future research in order to really capitalize on the potential of SensEIS feedback.

9. Conclusion

The goal of this thesis project was to find solutions to the challenges that still need to be overcome if we want to leverage the Multi-Level Iterations (MLI) scheme to realize an Ecological Adaptive Cruise Control (EACC) system based on Nonlinear Model Predictive Control (NMPC) in practice. In the following, we summarize our main developments that we presented in this thesis and outline future research directions arising from our work.

Summary

NMPC is an advanced control method. In essence, to perform NMPC, we repeatedly update our control by solving an Optimal Control Problem (OCP) that takes into account the current situation. The general challenge for numerical NMPC is to solve the OCPs sufficiently fast to achieve a high frequency of control updates. Since these OCPs are infinite-dimensional optimization problems, this requires sophisticated numerical methods. In the "first discretize, then optimize" approach, we first discretize the OCPs, which leads to finite-dimensional Nonlinear Programs (NLPs) that are then solved. A specific combination of methods that has been successfully used in several challenging NMPC applications is to use the Direct Multiple Shooting (DMS) method for the discretization and then the Real-Time Iterations (RTI) scheme or the Multi-Level Iterations (MLI) scheme to efficiently compute solutions to the sequence of NLPs. Both schemes can allow very high sampling frequencies by exploiting the structure of the Quadratic Programs (QPs) to be solved when a tailored Sequential Quadratic Programming (SQP) method is used to solve the NLPs.

The EACC application that we are considering is particularly challenging for three reasons.

First, multivariate Lookup Tables (LUTs) are commonly used in reality, both in the vehicle model and the OCP formulation. Therefore, our problem formulation also includes LUTs. For the optimization process we need to interpolate these LUTs. The challenge lays in computing a multivariate interpolation that is sufficiently smooth and additionally preserves certain patterns of the data. To the best of our knowledge, such an interpolation method has not been presented before. In this thesis, we have developed a smooth multivariate shape-preserving interpolation method. The main idea of our interpolation method is to apply smooth univariate shape-preserving interpolation methods to univariate subproblems and then aptly blend them together.

Second, external inputs such as the road elevation or the behaviour of a preceding vehicle (PP0) enter the OCPs and have to be handled appropriately. In principle, there exists a transformation that allows to transform the OCPs with external inputs back into a form that can be

handled by the existing methods. However, in our opinion explicitly considering external inputs in the DMS, RTI, and MLI holds further potential and opens new possibilities. Therefore, we presented novel strategies for incorporating external inputs into DMS, RTI, and MLI. In particular, these novel strategies are an indispensable component for our contribution for overcoming the third challenge.

Third, the computing power available in real vehicles is rather limited. Thus, although the MLI scheme provides levels that are already quite inexpensive, we strive to develop even faster feedback methods. Our contribution to this third challenge is the development of a new Sensitivity and External Input Scenario based (SensEIS) feedback. SensEIS feedback takes advantage of the fact that many driving scenarios are recurrent and can be precomputed. The connection to external inputs and thus the second challenge is that these driving situations are in particular characterized by external inputs. We have developed two different variants of SensEIS feedback, where the faster variant essentially requires only a matrix-vector product to compute the next control. The application purpose of SensEIS feedback is not limited to the EACC problem but extends to all applications where reasonable scenarios can be formed and precomputed.

Finally, from both a control theoretical and a practitioners point of view, it is important to know how reliable the inexact NMPC schemes realized by our numerical methods are. Regarding this question, we have provided, to the best of our knowledge, the first stability result for inexact NMPC for Partial Differential Equations (PDEs). Specifically, we proved asymptotic stability of the origin of the system-optimizer dynamics for a class of semilinear parabolic PDEs.

Ultimately, we combined our new developments and the MLI scheme and applied them to our EACC problem, which was equipped with data from a real test drive. In these numerical experiments, we demonstrated the potential to save more than $3.4\,\%$ of energy compared to the preceding vehicle from the test drive with only negligible constraint violations. If we extrapolate these savings to the total energy consumption of the transportation sector of about $698\,\mathrm{TW\,h}$ as reported in [191], our numerical methods have the potential to save about $23.7\,\mathrm{TW\,h}$ of energy. This is equivalent to the energy produced by burning about 3.9 million metric tons of coal [188]! And that's just for Germany...

[191]: Umweltbundesamt (2024), *Energieverbrauch nach Energieträgern und Sektoren*
[188]: U.S. Energy Information Administration (EIA) (2025), *Energy Conversion Calculators*

Future directions of research

From our stability results for inexact NMPC for a class of semilinear parabolic PDEs, we see three immediate directions for the future. First, our results require an estimate of how the optimal solution of a PDE-constrained OCP depends on the initial state. Establishing such estimates is an area of research for which very few results are available. More research in this direction may also benefit our stability results by possibly discovering appropriate estimates that are easier to verify. The second direction is to investigate generalizations of our results to other classes of PDEs.

Finally, we hope that our work will stimulate the development of adaptations of the RTI or MLI scheme to NMPC for PDEs. Given the success of these methods in NMPC for ODEs, we believe this is a particularly promising research question.

But also for NMPC for ODEs this thesis provides starting points for further developments. In our view, our novel extension of DMS to external inputs could be well suited for NMPC with DMS for long horizons, because it solves a challenge that arises when using a DMS grid with coarser intervals toward the end of the prediction horizon. To illustrate this challenge, consider the following example. Consider controlling a heavy truck in a hilly area. Suppose the engine torque control is chosen and discretized with constant basis functions, and the road elevation, which is the external input, is not approximated. The optimizer may then have to assume the same torque both up and down a hill, which may make the climb infeasible or cause speed violations downhill. Approximating the external inputs would be a remedy. But with the traditional approach, this would introduce perturbations in the model equations that would not be accounted for. Our method could address this by approximating the external inputs over long intervals, and taking changes in the approximation into account in the same way as changes in the current state. Longer prediction horizons should also further improve the performance of our numerical methods in the EACC application.

While the newly developed SensEIS feedback provides even faster feedback than the MLI scheme, we also identified challenges that need to be addressed. We discussed them in Section 7.4. The most important one that needs to be worked on is how to keep the number of precomputed scenarios low while still providing good feedback. A possible remedy might be to develop strategies to smoothly blend the feedback from different scenarios together. Furthermore, the interplay with the MLI scheme can still be improved. In particular, we suggest for future research to more thoroughly investigate the idea of using the scenarios only as providers of derivatives for the MLI scheme. This has the potential to completely remove online derivative computations from the MLI scheme, while still being able to react to changing variables.

Finally, as mentioned before, a proof that our interpolation method can realize an arbitrary degree of smoothness also for the general multivariate case is currently in progress.

By addressing these open challenges, future research can further add to the 3.9 million metric tons of coal that our simulations suggest could be saved from being burned if our methods were widely adopted.

Appendix

A. Proof for Example 3.2

Proof. We exploit the LIPSCHITZ continuity of h and estimate

$$\begin{aligned}\left\|h\left(x_m\left(\hat{\tau};s_m,q_m\right),q_m\right)\right\| &= \left\|h\left(x_m\left(\hat{\tau};s_m,q_m\right),q_m\right)-h\left(s_m,q_m\right)\right\| \\ &\leq \tilde{L}\left\|x_m\left(\hat{\tau};s_m,q_m\right)-s_m\right\|.\end{aligned}$$

We assumed $h(s_m,q_m)=0$ at the beginning of the example. Moreover, the controls do not appear on the right-hand side of our first LIPSCHITZ continuity estimation, as we use piecewise constant controls.

The state $x_m\left(\hat{\tau};s_m,q_m\right)$ is the solution of the IVP 2.4 and thus

$$x_m\left(\hat{\tau};s_m,q_m\right)=s_m+\int_{\tau_m}^{\hat{\tau}} f\left(x_m\left(\tau;s_m,q_m\right),q_m\right)\mathrm{d}\tau.$$

This allows us to estimate the deviation $x_m\left(\tau;s_m,q_m\right)-s_m$ by

$$\begin{aligned}\left\|x_m\left(\hat{\tau};s_m,q_m\right)-s_m\right\| &= \left\|\int_{\tau_m}^{\hat{\tau}} f\left(x_m\left(\tau;s_m,q_m\right),q_m\right)\mathrm{d}\tau\right\| \\ &\leq \int_{\tau_m}^{\hat{\tau}} \left\|f\left(x_m\left(\tau;s_m,q_m\right),q_m\right)\right\|\mathrm{d}\tau \\ &\leq \int_{\tau_m}^{\hat{\tau}} \left\|f\left(x_m\left(\tau;s_m,q_m\right),q_m\right)-f(0,0)\right\|\mathrm{d}\tau.\end{aligned} \tag{A.1}$$

We assumed $f(0,0)=0$ at the beginning of the example.

At this point, we leverage the LIPSCHITZ continuity of f with positive constants $\tilde{L}_x,\tilde{L}_u$ and obtain

$$\left\|f\left(x_m\left(\tau;s_m,q_m\right),q_m\right)-f(0,0)\right\| \leq \tilde{L}_x\left\|x_m\left(\tau;s_m,q_m\right)\right\|+\tilde{L}_u\left\|q_m\right\|. \tag{A.2}$$

To apply GRONWALL's Lemma [96] in the next step, we estimate

$$\begin{aligned}\left\|x_m\left(\tau;s_m,q_m\right)\right\| &= \left\|x_m\left(\tau;s_m,q_m\right)-s_m+s_m\right\| \\ &\leq \left\|x_m\left(\tau;s_m,q_m\right)-s_m\right\|+\left\|s_m\right\|.\end{aligned} \tag{A.3}$$

[96]: Gronwall (1919), "Note on the Derivatives with Respect to a Parameter of the Solutions of a System of Differential Equations"

Definition: GRONWALL's Lemma

One *integral form of GRONWALL's Lemma* states: Let $\gamma\colon[\tau_m,\tau_{m+1}]\to\mathbb{R}$ be a continuous function, $\alpha\colon[\tau_m,\tau_{m+1}]\to\mathbb{R}$ a non-decreasing function whose negative part is integrable on every closed and bounded subset of $[\tau_m,\tau_{m+1}]$, and $L\geq 0$. If

$$\gamma(\hat{\tau})\leq\alpha(\hat{\tau})+L\int_{\tau_m}^{\hat{\tau}}\gamma(\tau)\mathrm{d}\tau$$

for all $\hat{\tau}\in[\tau_m,\tau_{m+1}]$, then

$$\gamma(\hat{\tau})\leq\alpha(\hat{\tau})\mathrm{e}^{L(\hat{\tau}-\tau_m)}$$

holds for all $\hat{\tau}\in[\tau_m,\tau_{m+1}]$.

Combining estimates (A.2) and (A.3) in (A.1) yields

$$\begin{aligned}&\left\|x_m\left(\hat{\tau};s_m,q_m\right)-s_m\right\| \\ &\quad\leq \int_{\tau_m}^{\hat{\tau}} \tilde{L}_x\left\|x_m\left(\tau;s_m,q_m\right)-s_m\right\|+\tilde{L}_x\|s_m\|+\tilde{L}_u\left\|q_m\right\|\mathrm{d}\tau \\ &\quad= \int_{\tau_m}^{\hat{\tau}} \tilde{L}_x\left\|x_m\left(\tau;s_m,q_m\right)-s_m\right\|\mathrm{d}\tau+(\hat{\tau}-\tau_m)\left(\tilde{L}_x\|s_m\|+\tilde{L}_u\left\|q_m\right\|\right).\end{aligned} \tag{A.4}$$

The estimate (A.4) is now in a suitable form to apply the integral form of GRONWALL's Lemma, leading to

$$\begin{aligned}\left\|x_m\left(\hat{\tau};s_m,q_m\right)-s_m\right\| &\leq (\hat{\tau}-\tau_m)\mathrm{e}^{\tilde{L}_x(\hat{\tau}-\tau_m)}\left(\tilde{L}_x\|s_m\|+\tilde{L}_u\left\|q_m\right\|\right) \\ &\leq (\hat{\tau}-\tau_m)\mathrm{e}^{\tilde{L}_x(\tau_{m+1}-\tau_m)}\left(\tilde{L}_x\|s_m\|+\tilde{L}_u\left\|q_m\right\|\right).\end{aligned}$$

The subtle change in the second line is that we estimate

$$\mathrm{e}^{\tilde{L}_x(\hat{\tau}-\tau_m)}\leq\mathrm{e}^{\tilde{L}_x(\tau_{m+1}-\tau_m)}.$$

Defining $L_x:=\mathrm{e}^{\tilde{L}_x(\tau_{m+1}-\tau_m)}\tilde{L}_x$ and $L_u:=\mathrm{e}^{\tilde{L}_x(\tau_{m+1}-\tau_m)}\tilde{L}_u$ concludes the proof. ∎

Proof of Smoothness of our Interpolation Method in the Trivariate Case

B.

In the following we show that the interpolation p computed by our novel interpolation method that we presented in Chapter 5 in the trivariate case, i.e. $n = 3$ is twice continuously differentiable if the univariate interpolations are twice continuously differentiable and a blending function of smoothness order $q = 2$ is used.

In the interior of each hyperrectangle Q_α the function p_α is twice continuously differentiable as sum and product of twice continuously differentiable functions. The main task lays in proving the same at the intersection of two neighbouring hyperrectangles Q_α and $Q_{\alpha+\delta}$ with $0 \neq \delta \in \{-1, 0, 1\}^3$.

We write the multiindex α that we use for enumeration in this trivariate setting as

$$\alpha = (i, j, k)$$

and the multiindex δ as

$$\delta = (\delta^1, \delta^2, \delta^3)$$

The expanded form of the representation (5.18) for $p_{i,j,k}$ evaluated at

$$x = (x^1, x^2, x^3)^T \in Q_{i,j,k}$$

is

$$\begin{aligned}
p_{i,j,k}(x) = \quad & & (1-w_j^2(x^2)) & (1-w_k^3(x^3)) & p_{j,k}^1(x^1) \\
+ \quad & & (1-w_j^2(x^2)) & w_k^3(x^3) & p_{j,k+1}^1(x^1) \\
+ \quad & & w_j^2(x^2) & (1-w_k^3(x^3)) & p_{j+1,k}^1(x^1) \\
+ \quad & & w_j^2(x^2) & w_k^3(x^3) & p_{j+1,k+1}^1(x^1) \\
+ \quad & (1-w_i^1(x^1)) & & (1-w_k^3(x^3)) & p_{i,k}^2(x^2) \\
+ \quad & (1-w_i^1(x^1)) & & w_k^3(x^3) & p_{i,k+1}^2(x^2) \\
+ \quad & w_i^1(x^1) & & (1-w_k^3(x^3)) & p_{i+1,k}^2(x^2) \\
+ \quad & w_i^1(x^1) & & w_k^3(x^3) & p_{i+1,k+1}^2(x^2) \\
+ \quad & (1-w_i^1(x^1)) & (1-w_j^2(x^2)) & & p_{i,j}^3(x^3) \\
+ \quad & (1-w_i^1(x^1)) & w_j^2(x^2) & & p_{i,j+1}^3(x^3) \\
+ \quad & w_i^1(x^1) & (1-w_j^2(x^2)) & & p_{i+1,j}^3(x^3) \\
+ \quad & w_i^1(x^1) & w_j^2(x^2) & & p_{i+1,j+1}^3(x^3) \\
- 2 \quad & (1-w_i^1(x^1)) & (1-w_j^2(x^2)) & (1-w_k^3(x^3)) & d_{i,j,k} \\
- 2 \quad & (1-w_i^1(x^1)) & (1-w_j^2(x^2)) & w_k^3(x^3) & d_{i,j,k+1} \\
- 2 \quad & (1-w_i^1(x^1)) & w_j^2(x^2) & (1-w_k^3(x^3)) & d_{i,j+1,k} \\
- 2 \quad & (1-w_i^1(x^1)) & w_j^2(x^2) & w_k^3(x^3) & d_{i,j+1,k+1}
\end{aligned}$$

$$
\begin{array}{llllll}
-2 & w_i^1(x^1) & (1-w_j^2(x^2)) & (1-w_k^3(x^3)) & d_{i+1,j,k} \\
-2 & w_i^1(x^1) & (1-w_j^2(x^2)) & w_k^3(x^3) & d_{i+1,j,k+1} \\
-2 & w_i^1(x^1) & w_j^2(x^2) & (1-w_k^3(x^3)) & d_{i+1,j+1,k} \\
-2 & w_i^1(x^1) & w_j^2(x^2) & w_k^3(x^3) & d_{i+1,j+1,k+1}.
\end{array}
$$

The two hyperrectangles Q_α and $Q_{\alpha+\delta}$ can intersect in three different ways.

(i) They intersect at a face. In this case we have

$$|\delta^1| + |\delta^2| + |\delta^3| = 1.$$

Without loss of generality, we consider the case where $\delta^1 = \delta^2 = 0$ in the following. The other two cases work analogously. The intersection is then given by

$$\mathcal{S}^2_{\alpha,\delta} := Q_\alpha \cap Q_{\alpha+\delta} = \left\{ \begin{pmatrix} x^1 \\ x^2 \\ x^3_{\hat{k}} \end{pmatrix} \middle| \begin{array}{l} x_i^1 \le x^1 \le x_{i+1}^1, \\ x_j^2 \le x^2 \le x_{j+1}^2, \\ \hat{k} := \max\{k, k+\delta^3\} \end{array} \right\}.$$

(ii) They intersect at an edge. In this case we have

$$|\delta^1| + |\delta^2| + |\delta^3| = 2.$$

Without loss of generality, we consider the case where $\delta^1 = 0$ in the following. The other two cases work analogously. The intersection is then given by

$$\mathcal{S}^1_{\alpha,\delta} := Q_\alpha \cap Q_{\alpha+\delta} = \left\{ \begin{pmatrix} x^1 \\ x^2_{\hat{j}} \\ x^3_{\hat{k}} \end{pmatrix} \middle| \begin{array}{l} x_i^1 \le x^1 \le x_{i+1}^1, \\ \hat{j} := \max\{j, j+\delta^2\}, \\ \hat{k} := \max\{k, k+\delta^3\} \end{array} \right\}.$$

(iii) They intersect at a vertex. In this case we have

$$|\delta^1| + |\delta^2| + |\delta^3| = 3$$

and the intersection is given by

$$\mathcal{S}^0_{\alpha,\delta} := Q_\alpha \cap Q_{\alpha+\delta} = \left\{ \begin{pmatrix} x^1_{\hat{i}} \\ x^2_{\hat{j}} \\ x^3_{\hat{k}} \end{pmatrix} \middle| \begin{array}{l} \hat{i} := \max\{i, i+\delta^1\}, \\ \hat{j} := \max\{j, j+\delta^2\}, \\ \hat{k} := \max\{k, k+\delta^3\} \end{array} \right\}.$$

In the following we show continuity of p in Section B.1, then continuous differentiability in Section B.2 and finally twice continuous differentiability in Section B.3. Each time, we will go through the three different ways that the hyperrectangles can intersect that we have described above.

B.1. Continuity

(i) Intersection at a face We have that

$$w_k^3\left(x_{\hat{k}}^3\right) = \begin{cases} 0, & \text{if } \delta^3 = -1, \\ 1, & \text{if } \delta^3 = 1 \end{cases} = 1 - \begin{cases} 1, & \text{if } \delta^3 = -1, \\ 0, & \text{if } \delta^3 = 1 \end{cases} = 1 - w_{k+\delta^3}^3\left(x_{\hat{k}}^3\right).$$

Here we use the definition (5.17) of $w_k^3(x^3)$ and $w(0) = 0$, cf. Equation (5.8). Further we have defined $\hat{k} := \max\{k, k+\delta^3\}$ above. Further it is $\delta^3 \neq 0$ because we focus on the case where $\delta^1 = \delta^2 = 0$.

Therefore, we have

$$\begin{aligned}
p_{i,j,k}(x) = \quad & & & (1-w_j^2(x^2)) & p_{j,\hat{k}}^1(x^1) \\
+ \quad & & & w_j^2(x^2) & p_{j+1,\hat{k}}^1(x^1) \\
+ \quad & (1-w_i^1(x^1)) & & & p_{i,\hat{k}}^2(x^2) \\
+ \quad & w_i^1(x^1) & & & p_{i+1,\hat{k}}^2(x^2) \\
+ \quad & (1-w_i^1(x^1)) & (1-w_j^2(x^2)) & & p_{i,j}^3(x^3) \\
+ \quad & (1-w_i^1(x^1)) & w_j^2(x^2) & & p_{i,j+1}^3(x^3) \\
+ \quad & w_i^1(x^1) & (1-w_j^2(x^2)) & & p_{i+1,j}^3(x^3) \\
+ \quad & w_i^1(x^1) & w_j^2(x^2) & & p_{i+1,j+1}^3(x^3) \\
- 2 \quad & (1-w_i^1(x^1)) & (1-w_j^2(x^2)) & & d_{i,j,\hat{k}} \\
- 2 \quad & (1-w_i^1(x^1)) & w_j^2(x^2) & & d_{i,j+1,\hat{k}} \\
- 2 \quad & w_i^1(x^1) & (1-w_j^2(x^2)) & & d_{i+1,j,\hat{k}} \\
- 2 \quad & w_i^1(x^1) & w_j^2(x^2) & & d_{i+1,j+1,\hat{k}} \\
= \quad & p_{i+\delta^1,j+\delta^2,k+\delta^3}(x), & & &
\end{aligned}$$

which ensures continuity if the hyperrectangles intersect at a face.

(ii) Intersection at an edge We have already established in form of Theorem 5.2 that p is equal to the univariate interpolation on the edges. Thus, we have for all $x \in \mathcal{S}_{\alpha,\delta}^1$ that

As mentioned above, we consider without loss of generality the case where $\delta^1 = 0$. The other two cases work analogously.

$$p_{i,j,k}(x) = p_{\hat{j},\hat{k}}^1(x^1) = p_{i+\delta^1,j+\delta^2,k+\delta^3}(x),$$

which ensures continuity if the hyperrectangles intersect at an edge.

(iii) Intersection at a vertex We have already established in form of Corollary 5.1 that p is an interpolation. Thus, we have

$$p_{i,j,k}\left(x_{\hat{i},\hat{j},\hat{k}}\right) = d_{\hat{i},\hat{j},\hat{k}} = p_{i+\delta^1,j+\delta^2,k+\delta^3}\left(x_{\hat{i},\hat{j},\hat{k}}\right),$$

which ensures continuity if the hyperrectangles intersect at a vertex.

B.2. Continuous differentiability

We show continuity of the first derivative exemplarily for the derivative with respect to x^1. The other two cases work analogously. For $x \in Q_{i,j,k}$ the derivative of $p_{i,j,k}$ with respect to x^1 is given by

$$
\begin{aligned}
\frac{\partial}{\partial x^1} p_{i,j,k}(x) = \quad & & (1-w_j^2(x^2)) & (1-w_k^3(x^3)) & \frac{\partial}{\partial x^1} p_{j,k}^1(x^1) \\
+ \quad & & (1-w_j^2(x^2)) & w_k^3(x^3) & \frac{\partial}{\partial x^1} p_{j,k+1}^1(x^1) \\
+ \quad & & w_j^2(x^2) & (1-w_k^3(x^3)) & \frac{\partial}{\partial x^1} p_{j+1,k}^1(x^1) \\
+ \quad & & w_j^2(x^2) & w_k^3(x^3) & \frac{\partial}{\partial x^1} p_{j+1,k+1}^1(x^1) \\
- \quad & \frac{\partial}{\partial x^1} w_i^1(x^1) & & (1-w_k^3(x^3)) & p_{i,k}^2(x^2) \\
- \quad & \frac{\partial}{\partial x^1} w_i^1(x^1) & & w_k^3(x^3) & p_{i,k+1}^2(x^2) \\
+ \quad & \frac{\partial}{\partial x^1} w_i^1(x^1) & & (1-w_k^3(x^3)) & p_{i+1,k}^2(x^2) \\
+ \quad & \frac{\partial}{\partial x^1} w_i^1(x^1) & & w_k^3(x^3) & p_{i+1,k+1}^2(x^2) \\
- \quad & \frac{\partial}{\partial x^1} w_i^1(x^1) & (1-w_j^2(x^2)) & & p_{i,j}^3(x^3) \\
- \quad & \frac{\partial}{\partial x^1} w_i^1(x^1) & w_j^2(x^2) & & p_{i,j+1}^3(x^3) \\
+ \quad & \frac{\partial}{\partial x^1} w_i^1(x^1) & (1-w_j^2(x^2)) & & p_{i+1,j}^3(x^3) \\
+ \quad & \frac{\partial}{\partial x^1} w_i^1(x^1) & w_j^2(x^2) & & p_{i+1,j+1}^3(x^3) \\
+ 2 \quad & \frac{\partial}{\partial x^1} w_i^1(x^1) & (1-w_j^2(x^2)) & (1-w_k^3(x^3)) & d_{i,j,k} \\
+ 2 \quad & \frac{\partial}{\partial x^1} w_i^1(x^1) & (1-w_j^2(x^2)) & w_k^3(x^3) & d_{i,j,k+1} \\
+ 2 \quad & \frac{\partial}{\partial x^1} w_i^1(x^1) & w_j^2(x^2) & (1-w_k^3(x^3)) & d_{i,j+1,k} \\
+ 2 \quad & \frac{\partial}{\partial x^1} w_i^1(x^1) & w_j^2(x^2) & w_k^3(x^3) & d_{i,j+1,k+1} \\
- 2 \quad & \frac{\partial}{\partial x^1} w_i^1(x^1) & (1-w_j^2(x^2)) & (1-w_k^3(x^3)) & d_{i+1,j,k} \\
- 2 \quad & \frac{\partial}{\partial x^1} w_i^1(x^1) & (1-w_j^2(x^2)) & w_k^3(x^3) & d_{i+1,j,k+1} \\
- 2 \quad & \frac{\partial}{\partial x^1} w_i^1(x^1) & w_j^2(x^2) & (1-w_k^3(x^3)) & d_{i+1,j+1,k} \\
- 2 \quad & \frac{\partial}{\partial x^1} w_i^1(x^1) & w_j^2(x^2) & w_k^3(x^3) & d_{i+1,j+1,k+1}.
\end{aligned}
\tag{B.1}
$$

We will again investigate the three different intersection types (i) - (iii) described above. However, we will split our investigation into two parts. In the first, we have $|\delta^1| = 1$ and in the second $\delta^1 = 0$.

Case 1: $|\delta^1| = 1$

Here, we have that $x^1 = x^1_{\hat{i}}$ for all three intersection types. It holds that

$$\frac{\partial}{\partial x^1} w^1_i(x^1_i) = \frac{\partial}{\partial x^1} w^1_i(x^1_{i+1}) = 0 \tag{B.2}$$

cf. Equation (5.9). Thus, the expression for $\frac{\partial}{\partial x^1} p_{i,j,k}(x)$ collapses for $x^1 = x^1_{\hat{i}}$ to

$$\begin{aligned}
\frac{\partial}{\partial x^1} p_{i,j,k}(x) = \quad & (1-w^2_j(x^2)) \quad (1-w^3_k(x^3)) \quad \frac{\partial}{\partial x^1} p^1_{j,k}\left(x^1_{\hat{i}}\right) \\
+ \quad & (1-w^2_j(x^2)) \quad w^3_k(x^3) \quad \frac{\partial}{\partial x^1} p^1_{j,k+1}\left(x^1_{\hat{i}}\right) \\
+ \quad & w^2_j(x^2) \quad (1-w^3_k(x^3)) \quad \frac{\partial}{\partial x^1} p^1_{j+1,k}\left(x^1_{\hat{i}}\right) \\
+ \quad & w^2_j(x^2) \quad w^3_k(x^3) \quad \frac{\partial}{\partial x^1} p^1_{j+1,k+1}\left(x^1_{\hat{i}}\right)
\end{aligned} \tag{B.3}$$

and the one for $\frac{\partial}{\partial x^1} p_{i+\delta^1,j+\delta^2,k+\delta^3}(x)$ to

$$\begin{aligned}
\frac{\partial}{\partial x^1} p_{i+\delta^1,j+\delta^2,k+\delta^3}(x) = \quad & (1-w^2_{j+\delta^2}(x^2)) \quad (1-w^3_{k+\delta^3}(x^3)) \quad \frac{\partial}{\partial x^1} p^1_{j+\delta^2,k+\delta^3}\left(x^1_{\hat{i}}\right) \\
+ \quad & (1-w^2_{j+\delta^2}(x^2)) \quad w^3_{k+\delta^3}(x^3) \quad \frac{\partial}{\partial x^1} p^1_{j+\delta^2,k+1+\delta^3}\left(x^1_{\hat{i}}\right) \\
+ \quad & w^2_{j+\delta^2}(x^2) \quad (1-w^3_{k+\delta^3}(x^3)) \quad \frac{\partial}{\partial x^1} p^1_{j+1+\delta^2,k+\delta^3}\left(x^1_{\hat{i}}\right) \\
+ \quad & w^2_{j+\delta^2}(x^2) \quad w^3_{k+\delta^3}(x^3) \quad \frac{\partial}{\partial x^1} p^1_{j+1+\delta^2,k+1+\delta^3}\left(x^1_{\hat{i}}\right).
\end{aligned} \tag{B.4}$$

(i) Intersection at a face This intersection type is characterized by the fact that $|\delta^1| + |\delta^2| + |\delta^3| = 1$. As we are currently focussing on the case where $|\delta^1| = 1$, we have that $\delta^2 = \delta^3 = 0$. Considering this fact immediately reveals the equality of the expressions (B.3) and (B.4).

(ii) Intersection at an edge Here, we have that $|\delta^2| + |\delta^3| = 1$. We show the case $|\delta^2| = 1$ explicitly. The case $|\delta^3| = 1$ works analogously.

If $\delta^2 = 1$, then $x^2 = x^2_{\hat{j}} = x^2_{j+1}$ and thus $w^2_j(x^2) = 1$ and $w^2_{j+\delta^2}(x^2) = 0$. Therefore, we have that

$$\begin{aligned}
\frac{\partial}{\partial x^1} p_{i,j,k}(x) &= (1 - w^3_k(x^3)) \frac{\partial}{\partial x^1} p^1_{j+1,k}\left(x^1_{\hat{i}}\right) + w^3_k(x^3) \frac{\partial}{\partial x^1} p^1_{j+1,k+1}\left(x^1_{\hat{i}}\right) \\
&= (1 - w^3_{k+\delta^3}(x^3)) \frac{\partial}{\partial x^1} p^1_{j+\delta^2,k+\delta^3}\left(x^1_{\hat{i}}\right) + w^3_{k+\delta^3}(x^3) \frac{\partial}{\partial x^1} p^1_{j+\delta^2,k+1+\delta^3}\left(x^1_{\hat{i}}\right) \\
&= \frac{\partial}{\partial x^1} p_{i+\delta^1,j+\delta^2,k+\delta^3}(x).
\end{aligned}$$

If instead $\delta^2 = -1$, then $x^2 = x^2_{\hat{j}} = x^2_j$ and thus $w^2_j(x^2) = 0$ and $w^2_{j+\delta^2}(x^2) = 1$ in which case we obtain

$$\begin{aligned}\frac{\partial}{\partial x^1} p_{i,j,k}(x) &= (1 - w^3_k(x^3)) \frac{\partial}{\partial x^1} p^1_{j,k}\left(x^1_{\hat{i}}\right) + w^3_k(x^3) \frac{\partial}{\partial x^1} p^1_{j,k+1}\left(x^1_{\hat{i}}\right) \\ &= (1 - w^3_{k+\delta^3}(x^3)) \frac{\partial}{\partial x^1} p^1_{j+1+\delta^2,k+\delta^3}\left(x^1_{\hat{i}}\right) + w^3_{k+\delta^3}(x^3) \frac{\partial}{\partial x^1} p^1_{j+1+\delta^2,k+1+\delta^3}\left(x^1_{\hat{i}}\right) \\ &= \frac{\partial}{\partial x^1} p_{i+\delta^1,j+\delta^2,k+\delta^3}(x).\end{aligned} \tag{B.5}$$

(iii) Intersection at a vertex We first take a look at the values of the different weights.

- If $\delta^2 = 1$, then $x^2 = x^2_{\hat{j}} = x^2_{j+1}$ and thus $w^2_j(x^2) = 1$ and $w^2_{j+\delta^2}(x^2) = 0$.
- If $\delta^2 = -1$, then $x^2 = x^2_{\hat{j}} = x^2_j$ and thus $w^2_j(x^2) = 0$ and $w^2_{j+\delta^2}(x^2) = 1$.
- If $\delta^3 = 1$, then $x^3 = x^3_{\hat{k}} = x^3_{k+1}$ and thus $w^3_k(x^3) = 1$ and $w^3_{k+\delta^3}(x^3) = 0$.
- If $\delta^3 = -1$, then $x^3 = x^3_{\hat{k}} = x^3_k$ and thus $w^3_k(x^3) = 0$ and $w^3_{k+\delta^3}(x^3) = 1$.

From that we obtain that

$$\begin{aligned}\frac{\partial}{\partial x^1} p_{i,j,k}(x) &= \begin{cases} \frac{\partial}{\partial x^1} p^1_{j+1,k+1}\left(x^1_{\hat{i}}\right), & \text{if } \delta^2 = \delta^3 = 1, \\ \frac{\partial}{\partial x^1} p^1_{j+1,k}\left(x^1_{\hat{i}}\right), & \text{if } \delta^2 = 1,\ \delta^3 = -1, \\ \frac{\partial}{\partial x^1} p^1_{j,k+1}\left(x^1_{\hat{i}}\right), & \text{if } \delta^2 = -1,\ \delta^3 = 1, \\ \frac{\partial}{\partial x^1} p^1_{j,k}\left(x^1_{\hat{i}}\right), & \text{if } \delta^2 = \delta^3 = -1 \end{cases} \\ &= \begin{cases} \frac{\partial}{\partial x^1} p^1_{j+\delta^2,k+\delta^3}\left(x^1_{\hat{i}}\right), & \text{if } \delta^2 = \delta^3 = 1, \\ \frac{\partial}{\partial x^1} p^1_{j+\delta^2,k+1+\delta^3}\left(x^1_{\hat{i}}\right), & \text{if } \delta^2 = 1,\ \delta^3 = -1, \\ \frac{\partial}{\partial x^1} p^1_{j+1+\delta^2,k+\delta^3}\left(x^1_{\hat{i}}\right), & \text{if } \delta^2 = -1,\ \delta^3 = 1, \\ \frac{\partial}{\partial x^1} p^1_{j+1+\delta^2,k+1+\delta^3}\left(x^1_{\hat{i}}\right), & \text{if } \delta^2 = \delta^3 = -1 \end{cases} \\ &= \frac{\partial}{\partial x^1} p_{i+\delta^1,j+\delta^2,k+\delta^3}(x).\end{aligned}$$

Case 2: $\delta^1 = 0$

Here, we only know that $x^1_i \le x^1 \le x^1_{i+1}$. Therefore, the terms for $\frac{\partial}{\partial x^1} p_{i,j,k}(x)$ and $\frac{\partial}{\partial x^1} p_{i+\delta^1,j+\delta^2,k+\delta^3}(x)$ do not collapse into a form similar to Equations (B.3) and (B.4). In exchange, we do not need to consider the case where the hyperrectangles intersect at a vertex, as this would require that $|\delta^1| = 1$.

(i) Intersection at a face As $\delta^1 = 0$, we have here that $|\delta^2| + |\delta^3| = 1$. We show the case $|\delta^2| = 1$ explicitly. The case $|\delta^3| = 1$ works analogously.

If $\delta^2 = 1$, then $x^2 = x^2_{\hat{j}} = x^2_{j+1}$ and thus $w^2_j(x^2) = 1$ and $w^2_{j+\delta^2}(x^2) = 0$. Therefore, we have that

$$\begin{aligned}
\frac{\partial}{\partial x^1} p_{i,j,k}(x) &= (1 - w^3_k(x^3)) \frac{\partial}{\partial x^1} p^1_{j+1,k}(x^1) + w^3_k(x^3) \frac{\partial}{\partial x^1} p^1_{j+1,k+1}(x^1) \\
&\quad + \frac{\partial}{\partial x^1} w^1_i(x^1)(1 - w^3_k(x^3)) \Big(p^2_{i+1,k}(x^2) - p^2_{i,k}(x^2) \Big) \\
&\quad + \frac{\partial}{\partial x^1} w^1_i(x^1) w^3_k(x^3) \Big(p^2_{i+1,k+1}(x^2) - p^2_{i,k+1}(x^2) \Big) \\
&\quad + \frac{\partial}{\partial x^1} w^1_i(x^1) \Big(p^3_{i+1,j+1}(x^3) - p^3_{i,j+1}(x^3) \Big) \\
&\quad + 2 \frac{\partial}{\partial x^1} w^1_i(x^1)(1 - w^3_k(x^3)) (d_{i,j+1,k} - d_{i+1,j+1,k}) \\
&\quad + 2 \frac{\partial}{\partial x^1} w^1_i(x^1) w^3_k(x^3) (d_{i,j+1,k+1} - d_{i+1,j+1,k+1}) \\
&= (1 - w^3_{k+\delta^3}(x^3)) \frac{\partial}{\partial x^1} p^1_{j+\delta^2,k+\delta^3}(x^1) + w^3_{k+\delta^3}(x^3) \frac{\partial}{\partial x^1} p^1_{j+\delta^2,k+1+\delta^3}(x^1) \\
&\quad + \frac{\partial}{\partial x^1} w^1_{i+\delta^1}(x^1)(1 - w^3_{k+\delta^3}(x^3)) \Big(p^2_{i+1+\delta^1,k+\delta^3}(x^2) - p^2_{i+\delta^1,k+\delta^3}(x^2) \Big) \\
&\quad + \frac{\partial}{\partial x^1} w^1_{i+\delta^1}(x^1) w^3_{k+\delta^3}(x^3) \Big(p^2_{i+1+\delta^1,k+1+\delta^3}(x^2) - p^2_{i+\delta^1,k+1+\delta^3}(x^2) \Big) \\
&\quad + \frac{\partial}{\partial x^1} w^1_{j+1+\delta^2}(x^1) \Big(p^3_{i+\delta^1,j+\delta^2}(x^3) - p^3_{j+\delta^2,j+\delta^2}(x^3) \Big) \\
&\quad + 2 \frac{\partial}{\partial x^1} w^1_{i+\delta^1}(x^1)(1 - w^3_{k+\delta^3}(x^3)) \Big(d_{i+\delta^1,j+\delta^2,k+\delta^3} - d_{i+1+\delta^1,j+\delta^2,k+\delta^3} \Big) \\
&\quad + 2 \frac{\partial}{\partial x^1} w^1_{i+\delta^1}(x^1) w^3_{k+\delta^3}(x^3) \Big(d_{i+\delta^1,j+\delta^2,k+1+\delta^3} - d_{i+1+\delta^1,j+\delta^2,k+1+\delta^3} \Big) \\
&= \frac{\partial}{\partial x^1} p_{i+\delta^1,j+\delta^2,k+\delta^3}(x).
\end{aligned}$$

If instead $\delta^2 = -1$, then $x^2 = x^2_{\hat{j}} = x^2_j$ and thus $w^2_j(x^2) = 0$ and $w^2_{j+\delta^2}(x^2) = 1$ in which case we obtain

$$\begin{aligned}
\frac{\partial}{\partial x^1} p_{i,j,k}(x) &= (1 - w^3_k(x^3)) \frac{\partial}{\partial x^1} p^1_{j,k}(x^1) + w^3_k(x^3) \frac{\partial}{\partial x^1} p^1_{j,k+1}(x^1) \\
&\quad + \frac{\partial}{\partial x^1} w^1_i(x^1)(1 - w^3_k(x^3)) \Big(p^2_{i+1,k}(x^2) - p^2_{i,k}(x^2) \Big) \\
&\quad + \frac{\partial}{\partial x^1} w^1_i(x^1) w^3_k(x^3) \Big(p^2_{i+1,k+1}(x^2) - p^2_{i,k+1}(x^2) \Big) \\
&\quad + \frac{\partial}{\partial x^1} w^1_i(x^1) \Big(p^3_{i+1,j}(x^3) - p^3_{i,j}(x^3) \Big) \\
&\quad + 2 \frac{\partial}{\partial x^1} w^1_i(x^1)(1 - w^3_k(x^3)) (d_{i,j,k} - d_{i+1,j,k}) \\
&\quad + 2 \frac{\partial}{\partial x^1} w^1_i(x^1) w^3_k(x^3) (d_{i,j,k+1} - d_{i+1,j,k+1})
\end{aligned}$$

$$
\begin{aligned}
&= (1 - w^3_{k+\delta^3}(x^3))\frac{\partial}{\partial x^1} p^1_{j+1+\delta^2,k+\delta^3}(x^1) + w^3_{k+\delta^3}(x^3)\frac{\partial}{\partial x^1} p^1_{j+1+\delta^2,k+1+\delta^3}(x^1) \\
&+ \frac{\partial}{\partial x^1} w^1_{i+\delta^1}(x^1)(1 - w^3_{k+\delta^3}(x^3))\left(p^2_{i+1+\delta^1,k+\delta^3}(x^2) - p^2_{i+\delta^1,k+\delta^3}(x^2)\right) \\
&+ \frac{\partial}{\partial x^1} w^1_{i+\delta^1}(x^1) w^3_{k+\delta^3}(x^3)\left(p^2_{i+1+\delta^1,k+1+\delta^3}(x^2) - p^2_{i+\delta^1,k+1+\delta^3}(x^2)\right) \\
&+ \frac{\partial}{\partial x^1} w^1_{j+1+\delta^2}(x^1)\left(p^3_{i+\delta^1,j+1+\delta^2}(x^3) - p^3_{j+1+\delta^2,j+1+\delta^2}(x^3)\right) \\
&+ 2\frac{\partial}{\partial x^1} w^1_{i+\delta^1}(x^1)(1 - w^3_{k+\delta^3}(x^3))\left(d_{i+\delta^1,j+1+\delta^2,k+\delta^3} - d_{i+1+\delta^1,j+1+\delta^2,k+\delta^3}\right) \\
&+ 2\frac{\partial}{\partial x^1} w^1_{i+\delta^1}(x^1) w^3_{k+\delta^3}(x^3)\left(d_{i+\delta^1,j+1+\delta^2,k+1+\delta^3} - d_{i+1+\delta^1,j+1+\delta^2,k+1+\delta^3}\right) \\
&= \frac{\partial}{\partial x^1} p_{i+\delta^1,j+\delta^2,k+\delta^3}(x).
\end{aligned}
$$

(ii) Intersection at an edge As $\delta^1 = 0$, we have here that $|\delta^2| + |\delta^3| = 2$. Using the results for the values of the different weights that we have listed above for the intersection at a vertex in the first case, we obtain the following terms.

If $\delta^2 = \delta^3 = 1$, we have

$$
\begin{aligned}
\frac{\partial}{\partial x^1} p_{i,j,k}(x) &= \frac{\partial}{\partial x^1} p^1_{j+1,k+1}(x^1) \\
&+ \frac{\partial}{\partial x^1} w^1_i(x^1)\left(p^2_{i+1,k+1}\left(x^2_{\hat{j}}\right) - p^2_{i,k+1}\left(x^2_{\hat{j}}\right)\right) \\
&+ \frac{\partial}{\partial x^1} w^1_i(x^1)\left(p^3_{i+1,j+1}\left(x^2_{\hat{j}}\right) - p^3_{i,j+1}\left(x^3_{\hat{k}}\right)\right) \\
&+ 2\frac{\partial}{\partial x^1} w^1_i(x^1)\left(d_{i+1,j+1,k+1} - d_{i,j+1,k+1}\right) \\
&= \frac{\partial}{\partial x^1} p^1_{j+\delta^2,k+\delta^3}(x^1) \\
&+ \frac{\partial}{\partial x^1} w^1_{i+\delta^1}(x^1)\left(p^2_{i+1+\delta^1,k+\delta^3}\left(x^2_{\hat{j}}\right) - p^2_{i+\delta^1,k+\delta^3}\left(x^2_{\hat{j}}\right)\right) \\
&+ \frac{\partial}{\partial x^1} w^1_{i+\delta^1}(x^1)\left(p^3_{i+1+\delta^1,j+\delta^2}\left(x^2_{\hat{j}}\right) - p^3_{i+\delta^1,j+\delta^2}\left(x^3_{\hat{k}}\right)\right) \\
&+ 2\frac{\partial}{\partial x^1} w^1_{i+\delta^1}(x^1)\left(d_{i+1+\delta^1,j+\delta^2,k+\delta^3} - d_{i+\delta^1,j+\delta^2,k+\delta^3}\right) \\
&= \frac{\partial}{\partial x^1} p_{i+\delta^1,j+\delta^2,k+\delta^3}(x).
\end{aligned}
$$

If $\delta^2 = 1$ and $\delta^3 = -1$, we have

$$\begin{aligned}
\frac{\partial}{\partial x^1} p_{i,j,k}(x) &= \frac{\partial}{\partial x^1} p^1_{j+1,k}\left(x^1\right) \\
&\quad + \frac{\partial}{\partial x^1} w^1_i\left(x^1\right)\left(p^2_{i+1,k}\left(x^2_{\hat{j}}\right) - p^2_{i,k}\left(x^2_{\hat{j}}\right)\right) \\
&\quad + \frac{\partial}{\partial x^1} w^1_i\left(x^1\right)\left(p^3_{i+1,j+1}\left(x^2_{\hat{j}}\right) - p^3_{i,j+1}\left(x^3_{\hat{k}}\right)\right) \\
&\quad + 2\frac{\partial}{\partial x^1} w^1_i\left(x^1\right)\left(d_{i+1,j+1,k} - d_{i,j+1,k}\right) \\
&= \frac{\partial}{\partial x^1} p^1_{j+\delta^2,k+1+\delta^3}\left(x^1\right) \\
&\quad + \frac{\partial}{\partial x^1} w^1_{i+\delta^1}\left(x^1\right)\left(p^2_{i+1+\delta^1,k+1+\delta^3}\left(x^2_{\hat{j}}\right) - p^2_{i+\delta^1,k+1+\delta^3}\left(x^2_{\hat{j}}\right)\right) \\
&\quad + \frac{\partial}{\partial x^1} w^1_{i+\delta^1}\left(x^1\right)\left(p^3_{i+1+\delta^1,j+\delta^2}\left(x^2_{\hat{j}}\right) - p^3_{i+\delta^1,j+\delta^2}\left(x^3_{\hat{k}}\right)\right) \\
&\quad + 2\frac{\partial}{\partial x^1} w^1_{i+\delta^1}\left(x^1\right)\left(d_{i+1+\delta^1,j+\delta^2,k+1+\delta^3} - d_{i+\delta^1,j+\delta^2,k+1+\delta^3}\right) \\
&= \frac{\partial}{\partial x^1} p_{i+\delta^1,j+\delta^2,k+\delta^3}(x).
\end{aligned}$$

If $\delta^2 = -1$ and $\delta^3 = 1$, we have

$$\begin{aligned}
\frac{\partial}{\partial x^1} p_{i,j,k}(x) &= \frac{\partial}{\partial x^1} p^1_{j,k+1}\left(x^1\right) \\
&\quad + \frac{\partial}{\partial x^1} w^1_i\left(x^1\right)\left(p^2_{i+1,k+1}\left(x^2_{\hat{j}}\right) - p^2_{i,k+1}\left(x^2_{\hat{j}}\right)\right) \\
&\quad + \frac{\partial}{\partial x^1} w^1_i\left(x^1\right)\left(p^3_{i+1,j}\left(x^2_{\hat{j}}\right) - p^3_{i,j}\left(x^3_{\hat{k}}\right)\right) \\
&\quad + 2\frac{\partial}{\partial x^1} w^1_i\left(x^1\right)\left(d_{i+1,j,k+1} - d_{i,j,k+1}\right) \\
&= \frac{\partial}{\partial x^1} p^1_{j+1+\delta^2,k+\delta^3}\left(x^1\right) \\
&\quad + \frac{\partial}{\partial x^1} w^1_{i+\delta^1}\left(x^1\right)\left(p^2_{i+1+\delta^1,k+\delta^3}\left(x^2_{\hat{j}}\right) - p^2_{i+\delta^1,k+\delta^3}\left(x^2_{\hat{j}}\right)\right) \\
&\quad + \frac{\partial}{\partial x^1} w^1_{i+\delta^1}\left(x^1\right)\left(p^3_{i+1+\delta^1,j+1+\delta^2}\left(x^2_{\hat{j}}\right) - p^3_{i+\delta^1,j+1+\delta^2}\left(x^3_{\hat{k}}\right)\right) \\
&\quad + 2\frac{\partial}{\partial x^1} w^1_{i+\delta^1}\left(x^1\right)\left(d_{i+1+\delta^1,j+1+\delta^2,k+\delta^3} - d_{i+\delta^1,j+1+\delta^2,k+\delta^3}\right) \\
&= \frac{\partial}{\partial x^1} p_{i+\delta^1,j+\delta^2,k+\delta^3}(x).
\end{aligned}$$

If $\delta^2 = \delta^3 = -1$, we have

$$\begin{aligned}
\frac{\partial}{\partial x^1} p_{i,j,k}(x) &= \frac{\partial}{\partial x^1} p^1_{j,k}(x^1) \\
&\quad + \frac{\partial}{\partial x^1} w^1_i(x^1)\left(p^2_{i+1,k}\left(x^2_{\hat{j}}\right) - p^2_{i,k}\left(x^2_{\hat{j}}\right)\right) \\
&\quad + \frac{\partial}{\partial x^1} w^1_i(x^1)\left(p^3_{i+1,j}\left(x^2_{\hat{j}}\right) - p^3_{i,j}\left(x^3_{\hat{k}}\right)\right) \\
&\quad + 2\frac{\partial}{\partial x^1} w^1_i(x^1)\left(d_{i+1,j,k} - d_{i,j,k}\right) \\
&= \frac{\partial}{\partial x^1} p^1_{j+1+\delta^2,k+1+\delta^3}(x^1) \\
&\quad + \frac{\partial}{\partial x^1} w^1_{i+\delta^1}(x^1)\left(p^2_{i+1+\delta^1,k+1+\delta^3}\left(x^2_{\hat{j}}\right) - p^2_{i+\delta^1,k+1+\delta^3}\left(x^2_{\hat{j}}\right)\right) \\
&\quad + \frac{\partial}{\partial x^1} w^1_{i+\delta^1}(x^1)\left(p^3_{i+1+\delta^1,j+1+\delta^2}\left(x^2_{\hat{j}}\right) - p^3_{i+\delta^1,j+1+\delta^2}\left(x^3_{\hat{k}}\right)\right) \\
&\quad + 2\frac{\partial}{\partial x^1} w^1_{i+\delta^1}(x^1)\left(d_{i+1+\delta^1,j+1+\delta^2,k+1+\delta^3} - d_{i+\delta^1,j+1+\delta^2,k+1+\delta^3}\right) \\
&= \frac{\partial}{\partial x^1} p_{i+\delta^1,j+\delta^2,k+\delta^3}(x).
\end{aligned}$$

B.3. Twice continuous differentiability

We exemplarily show that the second derivatives are continuous across the boundaries of the hyperrectangles for $\frac{\partial^2}{\partial x^1 \partial x^1} p_{i,j,k}(x)$ and for $\frac{\partial^2}{\partial x^1 \partial x^2} p_{i,j,k}(x)$. The proofs for the derivatives $\frac{\partial}{\partial x^2} p_{i,j,k}(x)$, $\frac{\partial^2}{\partial x^1 \partial x^1} p_{i,j,k}(x)$ and $\frac{\partial^2}{\partial x^1 \partial x^3} p_{i,j,k}(x)$ and $\frac{\partial^2}{\partial x^2 \partial x^3} p_{i,j,k}(x)$ work analogously. As $p_{i,j,k}$ is twice continuously differentiable within the hyperrectangle $Q_{i,j,k}$, we know that

$$\begin{aligned}
\frac{\partial^2}{\partial x^1 \partial x^2} p_{i,j,k}(x) &= \frac{\partial^2}{\partial x^2 \partial x^1} p_{i,j,k}(x), \\
\frac{\partial^2}{\partial x^1 \partial x^3} p_{i,j,k}(x) &= \frac{\partial^2}{\partial x^3 \partial x^1} p_{i,j,k}(x), \\
\frac{\partial^2}{\partial x^2 \partial x^3} p_{i,j,k}(x) &= \frac{\partial^2}{\partial x^3 \partial x^2} p_{i,j,k}(x).
\end{aligned}$$

Thus, we do not need to consider the derivatives on the right side explicitly.

We first turn our attention to $\frac{\partial^2}{\partial x^1 \partial x^1} p_{i,j,k}(x)$.

For $p_{i,j,k}$ and $x \in Q_{i,j,k}$ we have that

$$
\begin{aligned}
\frac{\partial^2}{\partial x^1 \partial x^1} p_{i,j,k}(x) = \quad & & (1-w_j^2(x^2)) & (1-w_k^3(x^3)) & \frac{\partial}{\partial x^1} p_{j,k}^1(x^1) \\
+ \quad & & (1-w_j^2(x^2)) & w_k^3(x^3) & \frac{\partial}{\partial x^1} p_{j,k+1}^1(x^1) \\
+ \quad & & w_j^2(x^2) & (1-w_k^3(x^3)) & \frac{\partial}{\partial x^1} p_{j+1,k}^1(x^1) \\
+ \quad & & w_j^2(x^2) & w_k^3(x^3) & \frac{\partial}{\partial x^1} p_{j+1,k+1}^1(x^1) \\
- \quad & \frac{\partial}{\partial x^1} w_i^1(x^1) & & (1-w_k^3(x^3)) & p_{i,k}^2(x^2) \\
- \quad & \frac{\partial}{\partial x^1} w_i^1(x^1) & & w_k^3(x^3) & p_{i,k+1}^2(x^2) \\
+ \quad & \frac{\partial}{\partial x^1} w_i^1(x^1) & & (1-w_k^3(x^3)) & p_{i+1,k}^2(x^2) \\
+ \quad & \frac{\partial}{\partial x^1} w_i^1(x^1) & & w_k^3(x^3) & p_{i+1,k+1}^2(x^2) \\
- \quad & \frac{\partial}{\partial x^1} w_i^1(x^1) & (1-w_j^2(x^2)) & & p_{i,j}^3(x^3) \\
- \quad & \frac{\partial}{\partial x^1} w_i^1(x^1) & w_j^2(x^2) & & p_{i,j+1}^3(x^3) \\
+ \quad & \frac{\partial}{\partial x^1} w_i^1(x^1) & (1-w_j^2(x^2)) & & p_{i+1,j}^3(x^3) \\
+ \quad & \frac{\partial}{\partial x^1} w_i^1(x^1) & w_j^2(x^2) & & p_{i+1,j+1}^3(x^3) \\
+ 2 \quad & \frac{\partial}{\partial x^1} w_i^1(x^1) & (1-w_j^2(x^2)) & (1-w_k^3(x^3)) & d_{i,j,k} \\
+ 2 \quad & \frac{\partial}{\partial x^1} w_i^1(x^1) & (1-w_j^2(x^2)) & w_k^3(x^3) & d_{i,j,k+1} \\
+ 2 \quad & \frac{\partial}{\partial x^1} w_i^1(x^1) & w_j^2(x^2) & (1-w_k^3(x^3)) & d_{i,j+1,k} \\
+ 2 \quad & \frac{\partial}{\partial x^1} w_i^1(x^1) & w_j^2(x^2) & w_k^3(x^3) & d_{i,j+1,k+1} \\
- 2 \quad & \frac{\partial}{\partial x^1} w_i^1(x^1) & (1-w_j^2(x^2)) & (1-w_k^3(x^3)) & d_{i+1,j,k} \\
- 2 \quad & \frac{\partial}{\partial x^1} w_i^1(x^1) & (1-w_j^2(x^2)) & w_k^3(x^3) & d_{i+1,j,k+1} \\
- 2 \quad & \frac{\partial}{\partial x^1} w_i^1(x^1) & w_j^2(x^2) & (1-w_k^3(x^3)) & d_{i+1,j+1,k} \\
- 2 \quad & \frac{\partial}{\partial x^1} w_i^1(x^1) & w_j^2(x^2) & w_k^3(x^3) & d_{i+1,j+1,k+1}.
\end{aligned}
\tag{B.6}
$$

Comparing Equation (B.6) with Equation (B.1) for the first derivative $\frac{\partial}{\partial x^1} p_{i,j,k}(x)$, we see that they have the same structure. The only difference is that in Equation (B.6) the operator $\frac{\partial}{\partial x^1}$ replaces the operator $\frac{\partial}{\partial x^1}$ in Equation (B.1). As analogously to Equation (B.2) it holds that

$$
\frac{\partial}{\partial x^1} w_i^1(x_i^1) = \frac{\partial}{\partial x^1} w_i^1(x_{i+1}^1) = 0,
$$

Cf. again Equation (5.9).

the proof for the continuity of $\frac{\partial^2}{\partial x^1 \partial x^1} p_{i,j,k}(x)$ across the boundaries of the hyperrectangles proceeds completely analogously to the one of the contiuity of $\frac{\partial}{\partial x^1} p_{i,j,k}(x)$.

Next, we investigate $\frac{\partial^2}{\partial x^1 \partial x^2} p_{i,j,k}(x)$. For $x \in Q_{i,j,k}$ the derivative $\frac{\partial^2}{\partial x^1 \partial x^2} p_{i,j,k}(x)$ is given by

$$
\begin{aligned}
\frac{\partial^2}{\partial x^1 \partial x^2} p_{i,j,k}(x) = & & -\frac{\partial}{\partial x^2} w_j^2(x^2) & (1-w_k^3(x^3)) & \frac{\partial}{\partial x^1} p_{j,k}^1(x^1) \\
& - & \frac{\partial}{\partial x^2} w_j^2(x^2) & w_k^3(x^3) & \frac{\partial}{\partial x^1} p_{j,k+1}^1(x^1) \\
& + & \frac{\partial}{\partial x^2} w_j^2(x^2) & (1-w_k^3(x^3)) & \frac{\partial}{\partial x^1} p_{j+1,k}^1(x^1) \\
& + & \frac{\partial}{\partial x^2} w_j^2(x^2) & w_k^3(x^3) & \frac{\partial}{\partial x^1} p_{j+1,k+1}^1(x^1) \\
& - \frac{\partial}{\partial x^1} w_i^1(x^1) & & (1-w_k^3(x^3)) & \frac{\partial}{\partial x^2} p_{i,k}^2(x^2) \\
& - \frac{\partial}{\partial x^1} w_i^1(x^1) & & w_k^3(x^3) & \frac{\partial}{\partial x^2} p_{i,k+1}^2(x^2) \\
& + \frac{\partial}{\partial x^1} w_i^1(x^1) & & (1-w_k^3(x^3)) & \frac{\partial}{\partial x^2} p_{i+1,k}^2(x^2) \\
& + \frac{\partial}{\partial x^1} w_i^1(x^1) & & w_k^3(x^3) & \frac{\partial}{\partial x^2} p_{i+1,k+1}^2(x^2) \\
& + \frac{\partial}{\partial x^1} w_i^1(x^1) & \frac{\partial}{\partial x^2} w_j^2(x^2) & & p_{i,j}^3(x^3) \\
& - \frac{\partial}{\partial x^1} w_i^1(x^1) & \frac{\partial}{\partial x^2} w_j^2(x^2) & & p_{i,j+1}^3(x^3) \\
& - \frac{\partial}{\partial x^1} w_i^1(x^1) & \frac{\partial}{\partial x^2} w_j^2(x^2) & & p_{i+1,j}^3(x^3) \\
& + \frac{\partial}{\partial x^1} w_i^1(x^1) & \frac{\partial}{\partial x^2} w_j^2(x^2) & & p_{i+1,j+1}^3(x^3) \\
& - 2 \frac{\partial}{\partial x^1} w_i^1(x^1) & \frac{\partial}{\partial x^2} w_j^2(x^2) & (1-w_k^3(x^3)) & d_{i,j,k} \\
& - 2 \frac{\partial}{\partial x^1} w_i^1(x^1) & \frac{\partial}{\partial x^2} w_j^2(x^2) & w_k^3(x^3) & d_{i,j,k+1} \\
& + 2 \frac{\partial}{\partial x^1} w_i^1(x^1) & \frac{\partial}{\partial x^2} w_j^2(x^2) & (1-w_k^3(x^3)) & d_{i,j+1,k} \\
& + 2 \frac{\partial}{\partial x^1} w_i^1(x^1) & \frac{\partial}{\partial x^2} w_j^2(x^2) & w_k^3(x^3) & d_{i,j+1,k+1} \\
& + 2 \frac{\partial}{\partial x^1} w_i^1(x^1) & \frac{\partial}{\partial x^2} w_j^2(x^2) & (1-w_k^3(x^3)) & d_{i+1,j,k} \\
& + 2 \frac{\partial}{\partial x^1} w_i^1(x^1) & \frac{\partial}{\partial x^2} w_j^2(x^2) & w_k^3(x^3) & d_{i+1,j,k+1} \\
& - 2 \frac{\partial}{\partial x^1} w_i^1(x^1) & \frac{\partial}{\partial x^2} w_j^2(x^2) & (1-w_k^3(x^3)) & d_{i+1,j+1,k} \\
& - 2 \frac{\partial}{\partial x^1} w_i^1(x^1) & \frac{\partial}{\partial x^2} w_j^2(x^2) & w_k^3(x^3) & d_{i+1,j+1,k+1}.
\end{aligned}
\tag{B.7}
$$

We distinguish again between the three different intersection types.

(i) Intersection at a face Let first be $\delta^3 = 0$. We show exemplarily the case where $\left|\delta^1\right| = 1$. The case $\left|\delta^2\right| = 1$ works analogously. It is either $x^1 = x_i^1$ or $x^1 = x_{i+1}^1$. As in both cases we have

$$\frac{\partial}{\partial x^1} w_i^1(x^1) = \frac{\partial}{\partial x^1} w_{i+\delta^1}^1(x^1) = 0,$$

the expression (B.7) reduces to

$$\begin{aligned}\frac{\partial^2}{\partial x^1 \partial x^2} p_{i,j,k}(x) = \frac{\partial}{\partial x^2} w_j^2(x^2)(1 - w_k^3(x^3))\left(\frac{\partial}{\partial x^1} p_{j+1,k}^1(x^1) - \frac{\partial}{\partial x^1} p_{j,k}^1(x^1)\right) \\ + \frac{\partial}{\partial x^2} w_j^2(x^2) w_k^3(x^3)\left(\frac{\partial}{\partial x^1} p_{j+1,k+1}^1(x^1) - \frac{\partial}{\partial x^1} p_{j,k+1}^1(x^1)\right).\end{aligned}$$

Remembering that $\delta^2 = \delta^3 = 0$, we see that thus

$$\begin{aligned}\frac{\partial^2}{\partial x^1 \partial x^2} p_{i,j,k}(x) &= \frac{\partial}{\partial x^2} w_{j+\delta^2}^2(x^2)(1 - w_{k+\delta^3}^3(x^3))\left(\frac{\partial}{\partial x^1} p_{j+1+\delta^2,k+\delta^3}^1(x^1) - \frac{\partial}{\partial x^1} p_{j+\delta^2,k+\delta^3}^1(x^1)\right) \\ &\quad + \frac{\partial}{\partial x^2} w_{j+\delta^2}^2(x^2) w_{k+\delta^3}^3(x^3)\left(\frac{\partial}{\partial x^1} p_{j+1+\delta^2,k+1+\delta^3}^1(x^1) - \frac{\partial}{\partial x^1} p_{j+\delta^2,k+1+\delta^3}^1(x^1)\right) \\ &= \frac{\partial^2}{\partial x^1 \partial x^2} p_{i+\delta^1,j+\delta^2,k+\delta^3}(x).\end{aligned}$$

Let now be $\delta^3 = 1$ and accordingly $\delta^1 = \delta^2 = 0$. Here, we have that

$$x^3 = \begin{cases} x_k^2, & \text{if } \delta^3 = -1, \\ x_{k+1}^2, & \text{if } \delta^3 = 1 \end{cases}.$$

Therefore, we get

$$w_k^3(x^3) = \begin{cases} 0, & \text{if } \delta^3 = -1, \\ 1, & \text{if } \delta^3 = 1 \end{cases},$$

$$w_{k+\delta^3}^3(x^3) = \begin{cases} 1, & \text{if } \delta^3 = -1, \\ 0, & \text{if } \delta^3 = 1 \end{cases}.$$

For $\delta^3 = 1$ we thus get

$$
\begin{aligned}
&\frac{\partial^2}{\partial x^1 \partial x^2} p_{i,j,k}(x) \\
&= \frac{\partial}{\partial x^2} w_j^2(x^2) \left(\frac{\partial}{\partial x^1} p_{j+1,k+1}^1(x^1) - \frac{\partial}{\partial x^1} p_{j,k+1}^1(x^1) \right) \\
&+ \frac{\partial}{\partial x^1} w_i^1(x^1) \left(\frac{\partial}{\partial x^1} p_{i+1,k+1}^2(x^2) - \frac{\partial}{\partial x^1} p_{i,k+1}^2(x^2) \right) \\
&+ \frac{\partial}{\partial x^1} w_i^1(x^1) \frac{\partial}{\partial x^2} w_j^2(x^2) \left(p_{i,j}^3(x^3) - p_{i,j+1}^3(x^3) + d_{i,j+1,k+1} - d_{i,j,k+1} \right) \\
&- \frac{\partial}{\partial x^1} w_i^1(x^1) \frac{\partial}{\partial x^2} w_j^2(x^2) \left(p_{i+1,j}^3(x^3) - p_{i+1,j+1}^3(x^3) + d_{i+1,j+1,k+1} - d_{i+1,j,k+1} \right) \\
&= \frac{\partial}{\partial x^2} w_{j+\delta^2}^2(x^2) \left(\frac{\partial}{\partial x^1} p_{j+1+\delta^2,k+\delta^3}^1(x^1) - \frac{\partial}{\partial x^1} p_{j+\delta^2,k+\delta^3}^1(x^1) \right) \\
&+ \frac{\partial}{\partial x^1} w_{i+\delta^1}^1(x^1) \left(\frac{\partial}{\partial x^1} p_{i+1+\delta^1,k+\delta^3}^2(x^2) - \frac{\partial}{\partial x^1} p_{i+\delta^1,k+\delta^3}^2(x^2) \right) \\
&+ \frac{\partial}{\partial x^1} w_{i+\delta^1}^1(x^1) \frac{\partial}{\partial x^2} w_{j+\delta^2}^2(x^2) \left(p_{i+\delta^1,j+\delta^2}^3(x^3) - p_{i+\delta^1,j+1+\delta^2}^3(x^3) + d_{i+\delta^1,j+1+\delta^2,k+\delta^3} - d_{i+\delta^1,j+\delta^2,k+\delta^3} \right) \\
&- \frac{\partial}{\partial x^1} w_{i+\delta^1}^1(x^1) \frac{\partial}{\partial x^2} w_{j+\delta^2}^2(x^2) \left(p_{i+1+\delta^1,j+\delta^2}^3(x^3) - p_{i+1+\delta^1,j+1+\delta^2}^3(x^3) + d_{i+1+\delta^1,j+1+\delta^2,k+\delta^3} - d_{i+1+\delta^1,j+\delta^2,k+\delta^3} \right) \\
&= \frac{\partial^2}{\partial x^1 \partial x^2} p_{i+\delta^1,j+\delta^2,k+\delta^3}(x).
\end{aligned}
$$

For $\delta^3 = -1$ we get

$$
\begin{aligned}
&\frac{\partial^2}{\partial x^1 \partial x^2} p_{i,j,k}(x) \\
&= \frac{\partial}{\partial x^2} w_j^2(x^2) \left(\frac{\partial}{\partial x^1} p_{j+1,k}^1(x^1) - \frac{\partial}{\partial x^1} p_{j,k}^1(x^1) \right) \\
&+ \frac{\partial}{\partial x^1} w_i^1(x^1) \left(\frac{\partial}{\partial x^1} p_{i+1,k}^2(x^2) - \frac{\partial}{\partial x^1} p_{i,k}^2(x^2) \right) \\
&+ \frac{\partial}{\partial x^1} w_i^1(x^1) \frac{\partial}{\partial x^2} w_j^2(x^2) \left(p_{i,j}^3(x^3) - p_{i,j+1}^3(x^3) + d_{i,j+1,k} - d_{i,j,k} \right) \\
&- \frac{\partial}{\partial x^1} w_i^1(x^1) \frac{\partial}{\partial x^2} w_j^2(x^2) \left(p_{i+1,j}^3(x^3) - p_{i+1,j+1}^3(x^3) + d_{i+1,j+1,k} - d_{i+1,j,k} \right) \\
&= \frac{\partial}{\partial x^2} w_{j+\delta^2}^2(x^2) \left(\frac{\partial}{\partial x^1} p_{j+1+\delta^2,k+1+\delta^3}^1(x^1) - \frac{\partial}{\partial x^1} p_{j+\delta^2,k+1+\delta^3}^1(x^1) \right) \\
&+ \frac{\partial}{\partial x^1} w_{i+\delta^1}^1(x^1) \left(\frac{\partial}{\partial x^1} p_{i+1+\delta^1,k+1+\delta^3}^2(x^2) - \frac{\partial}{\partial x^1} p_{i+\delta^1,k+1+\delta^3}^2(x^2) \right) \\
&+ \frac{\partial}{\partial x^1} w_{i+\delta^1}^1(x^1) \frac{\partial}{\partial x^2} w_{j+\delta^2}^2(x^2) \left(p_{i+\delta^1,j+\delta^2}^3(x^3) - p_{i+\delta^1,j+1+\delta^2}^3(x^3) + d_{i+\delta^1,j+1+\delta^2,k+1+\delta^3} - d_{i+\delta^1,j+\delta^2,k+1+\delta^3} \right) \\
&- \frac{\partial}{\partial x^1} w_{i+\delta^1}^1(x^1) \frac{\partial}{\partial x^2} w_{j+\delta^2}^2(x^2) \left(p_{i+1+\delta^1,j+\delta^2}^3(x^3) - p_{i+1+\delta^1,j+1+\delta^2}^3(x^3) + d_{i+1+\delta^1,j+1+\delta^2,k+1+\delta^3} - d_{i+1+\delta^1,j+\delta^2,k+1+\delta^3} \right) \\
&= \frac{\partial^2}{\partial x^1 \partial x^2} p_{i+\delta^1,j+\delta^2,k+\delta^3}(x).
\end{aligned}
$$

(ii) Intersection at an edge Let first be $\delta^3 = 0$ and accordingly $\left|\delta^1\right| = \left|\delta^2\right| = 1$. Because of this, it is either $x^1 = x_i^1$ or $x^1 = x_{i+1}^1$ and $x^2 = x_j^2$ or $x^2 = x_{j+1}^2$. As a consequence, we have

$$\begin{aligned}\frac{\partial}{\partial x^1} w_i^1(x^1) &= \frac{\partial}{\partial x^1} w_{i+\delta^1}^1(x^1) = 0,\\ \frac{\partial}{\partial x^2} w_j^2(x^2) &= \frac{\partial}{\partial x^2} w_{j+\delta^2}^2(x^2) = 0.\end{aligned}$$

As one of these derivatives occurs as a factor in each summand of $\frac{\partial^2}{\partial x^1 \partial x^2} p_{i,j,k}(x)$, cf. Equation (B.7), we obtain

$$\frac{\partial^2}{\partial x^1 \partial x^2} p_{i,j,k}(x) = 0 = \frac{\partial^2}{\partial x^1 \partial x^2} p_{i+\delta^1,j+\delta^2,k+\delta^3}(x).$$

This is the zero corner twist issue that we discussed at the end of Subsection 5.3.2.

Let now be $\delta^3 = 1$. We exemplarily show the case $\left|\delta^1\right| = 1$. The case where $\left|\delta^2\right| = 1$ works analogously. As again $x^1 = x_{i+1}^1$ and $x^2 = x_j^2$ we again have that

$$\frac{\partial}{\partial x^1} w_i^1(x^1) = \frac{\partial}{\partial x^1} w_{i+\delta^1}^1(x^1) = 0.$$

This leads to

$$\begin{aligned}\frac{\partial^2}{\partial x^1 \partial x^2} p_{i,j,k}(x) &= \frac{\partial}{\partial x^2} w_j^2(x^2)(1 - w_k^3(x^3))\left(\frac{\partial}{\partial x^1} p_{j+1,k}^1(x^1) - \frac{\partial}{\partial x^1} p_{j,k}^1(x^1)\right)\\ &\quad + \frac{\partial}{\partial x^2} w_j^2(x^2) w_k^3(x^3)\left(\frac{\partial}{\partial x^1} p_{j+1,k+1}^1(x^1) - \frac{\partial}{\partial x^1} p_{j,k+1}^1(x^1)\right)\\ &= \begin{cases}\frac{\partial}{\partial x^2} w_j^2(x^2)\left(\frac{\partial}{\partial x^1} p_{j+1,k}^1(x^1) - \frac{\partial}{\partial x^1} p_{j,k}^1(x^1)\right), & \text{if } \delta^3 = -1,\\ \frac{\partial}{\partial x^2} w_j^2(x^2)\left(\frac{\partial}{\partial x^1} p_{j+1,k+1}^1(x^1) - \frac{\partial}{\partial x^1} p_{j,k+1}^1(x^1)\right), & \text{if } \delta^3 = 1\end{cases}\\ &= \begin{cases}\frac{\partial}{\partial x^2} w_{j+\delta^2}^2(x^2)\left(\frac{\partial}{\partial x^1} p_{j+1+\delta^2,k+1+\delta^3}^1(x^1) - \frac{\partial}{\partial x^1} p_{j+\delta^2,k+1+\delta^3}^1(x^1)\right), & \text{if } \delta^3 = -1,\\ \frac{\partial}{\partial x^2} w_{j+\delta^2}^2(x^2)\left(\frac{\partial}{\partial x^1} p_{j+1+\delta^2,k+\delta^3}^1(x^1) - \frac{\partial}{\partial x^1} p_{j+\delta^2,k+\delta^3}^1(x^1)\right), & \text{if } \delta^3 = 1\end{cases}\\ &= \frac{\partial}{\partial x^2} w_{j+\delta^2}^2(x^2) w_{k+\delta^3}^3(x^3)\left(\frac{\partial}{\partial x^1} p_{j+1+\delta^2,k+1+\delta^3}^1(x^1) - \frac{\partial}{\partial x^1} p_{j+\delta^2,k+1+\delta^3}^1(x^1)\right)\\ &\quad + \frac{\partial}{\partial x^2} w_{j+\delta^2}^2(x^2)(1 - w_{k+\delta^3}^3(x^3))\left(\frac{\partial}{\partial x^1} p_{j+1+\delta^2,k+\delta^3}^1(x^1) - \frac{\partial}{\partial x^1} p_{j+\delta^2,k+\delta^3}^1(x^1)\right)\\ &= \frac{\partial^2}{\partial x^1 \partial x^2} p_{i+\delta^1,j+\delta^2,k+\delta^3}(x).\end{aligned}$$

(iii) Intersection at a vertex As for the intersection at a edge in the case where $\delta^3 = 0$, we have here again that

$$\frac{\partial}{\partial x^1} w_i^1(x^1) = \frac{\partial}{\partial x^1} w_{i+\delta^1}^1(x^1) = 0, \quad \frac{\partial}{\partial x^2} w_j^2(x^2) = \frac{\partial}{\partial x^2} w_{j+\delta^2}^2(x^2) = 0.$$

and accordingly that

$$\frac{\partial^2}{\partial x^1 \partial x^2} p_{i,j,k}(x) = 0 = \frac{\partial^2}{\partial x^1 \partial x^2} p_{i+\delta^1,j+\delta^2,k+\delta^3}(x).$$

This is again the zero corner twist issue that we discussed at the end of Subsection 5.3.2.

C. Gradient and Hessian of Lagrangian of DMS NLP with External Inputs

In the following we present the structure of the gradient and the Hessian of the Lagrangian of the DMS NLP that corresponds to the OCP (6.4). We do this to aid in the understanding and implementation of the numerical methods that are presented in Chapter 3 and Chapters 6 – 7 and make use of the gradient and Hessian blocks and their structure that we present in the following. We consider the most general OCP (6.4), i.e. where external inputs and both mixed state-control and boundary constraints are present. The gradient and Hessian blocks for simpler cases can then be obtained by just dropping dependencies and constraints that are not present in these cases.

For ease of presentation we assume that the OCP has been transformed such that only the MAYER Φ objective term is present. The NLP that we consider in the following is thus a slight simplification of NLP (6.8) and given by

For information on this transformation we refer to [138, Section 5.1].

$$
\begin{aligned}
\min_{\substack{s\in\mathbb{R}^{n_s}\\ q\in\mathbb{R}^{n_q}}} \quad & \Phi\left(s_M;\rho,v_M\right)\\
\text{s.t.} \quad & 0 = x\left(\tau_{m+1};s_m,q_m;\rho^j,v_m^j\right) - s_{m+1}, && m=0,\dots,M-1,\\
& 0 \le h\left(s_m,q_m;\rho^j,v_m^j\right), && m=0,\dots,M-1,\\
& 0 = r^{\mathrm{e}}\left(s_0,s_M;\rho^j,v_0^j,v_M^j\right),\\
& 0 \le r^{\mathrm{i}}\left(s_0,s_M;\rho^j,v_0^j,v_M^j\right),\\
& 0 = x^j - s_0.
\end{aligned}
\tag{C.1}
$$

The Lagrangian $\mathscr{L}$ of the NLP (C.1) is given by

We drop the dependencies of the Lagrangian in the remaining part of this chapter for readibility.

$$
\begin{aligned}
\mathscr{L}\left(s,q,x^j,\rho^j,v^j,\lambda^{\mathrm{MC}},\lambda^r,\lambda^{\mathrm{IVE}},\mu^h,\mu^r\right) &:= \Phi\left(s_M;\rho,v_M\right)\\
&- \sum_{m=0}^{M-1}\left(\lambda_m^{\mathrm{MC}}\right)^T\left(x\left(\tau_{m+1};s_m,q_m;\rho^j,v_m^j\right) - s_{m+1}\right)\\
&- \left(\lambda^r\right)^T\left(r^{\mathrm{e}}\left(s_0,s_M;\rho^j,v_0^j,v_M^j\right)\right) - \left(\lambda^{\mathrm{IVE}}\right)^T\left(x^j - s_0\right)\\
&- \sum_{m=0}^{M-1}\left(\mu_m^h\right)^T\left(h\left(s_m,q_m;\rho^j,v_m^j\right)\right) - \left(\mu^r\right)^T\left(r^{\mathrm{i}}\left(s_0,s_M;\rho^j,v_0^j,v_M^j\right)\right)
\end{aligned}
$$

with the LAGRANGE multipliers

$$\lambda^{\mathrm{MC}} = \begin{pmatrix} \lambda_0^{\mathrm{MC}} \\ \vdots \\ \lambda_{M-1}^{\mathrm{MC}} \end{pmatrix} \in \mathbb{R}^{Mn_x}, \quad \lambda^r \in \mathbb{R}^{n_{r^{\mathrm{e}}}}, \quad \lambda^{\mathrm{IVE}} \in \mathbb{R}^{n_x},$$
$$\mu^h = \begin{pmatrix} \mu_0^h \\ \vdots \\ \mu_{M-1}^h \end{pmatrix} \in \mathbb{R}^{Mn_h}, \quad \mu^r \in \mathbb{R}^{n_{r^{\mathrm{i}}}}. \tag{C.2}$$

In the following we first present the structure of the gradient of the Lagrangian $\mathscr{L}$ given in Equation (C.2), which includes the Jacobians of the constraints of the NLP (C.1), and then the structure of the Hessian. For Chapter 3 and Chapters 6 – 7 the derivatives with respect to the state variables s, the control variables q, the constant parameters ρ and the external inputs v are of interest.

C.1. Gradient of the Lagrangian

The gradient of the Lagrangian $\mathscr{L}$ with respect to the state variables s, the control variables q, the constant parameters ρ and the external inputs v is given by

$$\nabla \mathscr{L} = \begin{pmatrix} \nabla_s \mathscr{L} \\ \nabla_q \mathscr{L} \\ \nabla_\rho \mathscr{L} \\ \nabla_{v_m} \mathscr{L} \end{pmatrix}.$$

The dimensions of the blocks of the gradient are:

- $\nabla_s \mathscr{L} \in \mathbb{R}^{(M+1)n_x}$,
- $\nabla_q \mathscr{L} \in \mathbb{R}^{Mn_q}$,
- $\nabla_\rho \mathscr{L} \in \mathbb{R}^{n_\rho}$,
- $\nabla_v \mathscr{L} \in \mathbb{R}^{n_v}$.

The structures of these gradients are given by

$$\nabla_s \mathscr{L} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ \nabla_{s_M} \Phi\left(s_M; \rho, v_M\right) \end{pmatrix} - \begin{pmatrix} -\mathbb{I} & \left(S_0^s\right)^T & 0 & \cdots & 0 \\ 0 & -\mathbb{I} & \left(S_1^s\right)^T & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & -\mathbb{I} & \left(S_{M-1}^s\right)^T \\ 0 & \cdots & 0 & 0 & -\mathbb{I} \end{pmatrix} \begin{pmatrix} \lambda^{\mathrm{IVE}} \\ \lambda_0^{\mathrm{MC}} \\ \lambda_1^{\mathrm{MC}} \\ \vdots \\ \lambda_{M-1}^{\mathrm{MC}} \end{pmatrix}$$
$$- \begin{pmatrix} \left(\frac{\partial}{\partial s_0} h\left(s_0, q_0; \rho^j, v_0^j\right)\right)^T \mu_0^h \\ \left(\frac{\partial}{\partial s_1} h\left(s_1, q_1; \rho^j, v_1^j\right)\right)^T \mu_1^h \\ \vdots \\ \left(\frac{\partial}{\partial s_{M-1}} h\left(s_{M-1}, q_{M-1}; \rho^j, v_{M-1}^j\right)\right)^T \mu_{M-1}^h \\ 0 \end{pmatrix}$$
$$- \begin{pmatrix} \left(\frac{\partial}{\partial s_0} r^{\mathrm{e}}\left(s_0, s_M; \rho^j, v_0^j, v_M^j\right)\right)^T \lambda^r + \left(\frac{\partial}{\partial s_0} r^{\mathrm{i}}\left(s_0, s_M; \rho^j, v_0^j, v_M^j\right)\right)^T \mu^r \\ 0 \\ \vdots \\ 0 \\ \left(\frac{\partial}{\partial s_M} r^{\mathrm{e}}\left(s_0, s_M; \rho^j, v_0^j, v_M^j\right)\right)^T \lambda^r + \left(\frac{\partial}{\partial s_M} r^{\mathrm{i}}\left(s_0, s_M; \rho^j, v_0^j, v_M^j\right)\right)^T \mu^r \end{pmatrix},$$

$$\nabla_q \mathcal{L} = \begin{pmatrix} \left(S_0^q\right)^T \lambda_0^{\mathrm{MC}} \\ \vdots \\ \left(S_{M-1}^q\right)^T \lambda_{M-1}^{\mathrm{MC}} \end{pmatrix} - \begin{pmatrix} \left(\frac{\partial}{\partial q_0} h\left(s_0, q_0; \rho^j, v_0^j\right)\right)^T \mu_0^h \\ \vdots \\ \left(\frac{\partial}{\partial q_{M-1}} h\left(s_{M-1}, q_{M-1}; \rho^j, v_{M-1}^j\right)\right)^T \mu_{M-1}^h \end{pmatrix},$$

$$\begin{aligned} \nabla_\rho \mathcal{L} = \nabla_\rho \Phi\left(s_M; \rho, v_M\right) &- \left(\frac{\partial}{\partial \rho} r^{\mathrm{e}}\left(s_0, s_M; \rho^j, v_0^j, v_M^j\right)\right)^T \lambda^r - \left(\frac{\partial}{\partial \rho} r^{\mathrm{i}}\left(s_0, s_M; \rho^j, v_0^j, v_M^j\right)\right)^T \mu^r \\ &- \sum_{m=0}^{M-1} \left(S_m^\rho\right)^T \lambda_m^{\mathrm{MC}} + \left(\frac{\partial}{\partial \rho} h\left(s_m, q_m; \rho^j, v_m^j\right)\right)^T \mu_m^h, \end{aligned}$$

$$\nabla_v \mathcal{L} = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 0 \\ \nabla_{v_M} \Phi\left(s_M; \rho, v_M\right) \end{pmatrix} - \begin{pmatrix} \left(S_0^v\right)^T \lambda_0^{\mathrm{MC}} \\ \vdots \\ \left(S_{M-1}^v\right)^T \lambda_{M-1}^{\mathrm{MC}} \\ 0 \end{pmatrix} - \begin{pmatrix} \left(\frac{\partial}{\partial v_0} h\left(s_0, q_0; \rho^j, v_0^j\right)\right)^T \mu_0^h \\ \vdots \\ \left(\frac{\partial}{\partial v_{M-1}} h\left(s_{M-1}, q_{M-1}; \rho^j, v_{M-1}^j\right)\right)^T \mu_{M-1}^h \end{pmatrix}.$$

C.2. Hessian of the Lagrangian

The Hessian of the Lagrangian $\mathcal{L}$ with respect to the state variables s, the control variables q, the constant parameters ρ and the external inputs v is given by

$$\nabla^2 \mathcal{L} = \begin{pmatrix} \nabla_{ss}^2 \mathcal{L} & \nabla_{sq}^2 \mathcal{L} & \nabla_{s\rho}^2 \mathcal{L} & \nabla_{sv}^2 \mathcal{L} \\ \nabla_{qs}^2 \mathcal{L} & \nabla_{qq}^2 \mathcal{L} & \nabla_{q\rho}^2 \mathcal{L} & \nabla_{qv}^2 \mathcal{L} \\ \nabla_{\rho s}^2 \mathcal{L} & \nabla_{\rho q}^2 \mathcal{L} & \nabla_{\rho\rho}^2 \mathcal{L} & \nabla_{\rho v}^2 \mathcal{L} \\ \nabla_{vs}^2 \mathcal{L} & \nabla_{vq}^2 \mathcal{L} & \nabla_{v\rho}^2 \mathcal{L} & \nabla_{vv}^2 \mathcal{L} \end{pmatrix},$$

where the dimensions of the blocks of the Hessian are given by

$$\begin{aligned} &\nabla_{ss}^2 \mathcal{L} \in \mathbb{R}^{(M+1)n_x \times (M+1)n_x}, && \nabla_{sq}^2 \mathcal{L} \in \mathbb{R}^{(M+1)n_x \times Mn_q}, \\ &\nabla_{s\rho}^2 \mathcal{L} \in \mathbb{R}^{(M+1)n_x \times n_\rho}, && \nabla_{sv}^2 \mathcal{L} \in \mathbb{R}^{(M+1)n_x \times n_v}, \\ &\nabla_{qq}^2 \mathcal{L} \in \mathbb{R}^{Mn_q \times Mn_q}, && \nabla_{q\rho}^2 \mathcal{L} \in \mathbb{R}^{Mn_q \times n_\rho}, \\ &\nabla_{qv}^2 \mathcal{L} \in \mathbb{R}^{Mn_q \times n_v}, && \nabla_{\rho\rho}^2 \mathcal{L} \in \mathbb{R}^{n_\rho \times n_\rho}, \\ &\nabla_{\rho v}^2 \mathcal{L} \in \mathbb{R}^{n_\rho \times n_v}, && \nabla_{vv}^2 \mathcal{L} \in \mathbb{R}^{n_v \times n_v}. \end{aligned}$$

Under the common assumption that the Lagrangian is twice continuously differentiable, the Hessian of the Lagrangian is symmetric. We therefore only present the upper triangular part of the Hessian. From here on, the index l is used to denote the scalar components of the vectors.

$\nabla^2_{ss}\mathscr{L}$

The Hessian block $\nabla^2_{ss}\mathscr{L}$ is given by

$$\nabla^2_{ss}\mathscr{L} = \begin{pmatrix} \nabla^2_{s_0s_0}\mathscr{L} & 0 & \cdots & 0 & \nabla^2_{s_0s_M}\mathscr{L} \\ 0 & \nabla^2_{s_1s_1}\mathscr{L} & \ddots & \vdots & 0 \\ \vdots & \ddots & \ddots & 0 & 0 \\ 0 & \cdots & 0 & \nabla^2_{s_{M-1}s_{M-1}}\mathscr{L} & 0 \\ \nabla^2_{s_Ms_0}\mathscr{L} & 0 & 0 & 0 & \nabla^2_{s_Ms_M}\mathscr{L} \end{pmatrix},$$

with

$$\begin{aligned}
\nabla^2_{s_0s_0}\mathscr{L} =& -\sum_{l=1}^{n_x} \nabla^2_{s_0s_0} x_l\left(\tau_1; s_0, q_0; \rho^j, v_0^j\right)\lambda^{\mathrm{MC}}_{0,l} \\
& -\sum_{l=1}^{n_h} \nabla^2_{s_0s_0} h_l\left(s_0, q_0; \rho^j, v_0^j\right)\mu^h_{0,l} \\
& -\sum_{l=1}^{n_{r^\mathrm{e}}} \nabla^2_{s_0s_0} r^\mathrm{e}_l\left(s_0, s_M; \rho^j, v_0^j, v_M^j\right)\lambda^r_l - \sum_{l=1}^{n_{r^\mathrm{i}}} \nabla^2_{s_0s_0} r^\mathrm{i}_l\left(s_0, s_M; \rho^j, v_0^j, v_M^j\right)\mu^r_l, \\
\nabla^2_{s_ms_m}\mathscr{L} =& -\sum_{l=1}^{n_x} \nabla^2_{s_ms_m} x_l\left(\tau_{m+1}; s_m, q_m; \rho^j, v_m^j\right)\lambda^{\mathrm{MC}}_{m,l} \\
& -\sum_{l=1}^{n_h} \nabla^2_{s_ms_m} h_l\left(s_m, q_m; \rho^j, v_m^j\right)\mu^h_{m,l}, \quad m = 1, \ldots, m-1, \\
\nabla^2_{s_Ms_M}\mathscr{L} =& \nabla^2_{s_Ms_M}\Phi\left(s_M; \rho, v_M\right) \\
& -\sum_{l=1}^{n_{r^\mathrm{e}}} \nabla^2_{s_Ms_M} r^\mathrm{e}_l\left(s_0, s_M; \rho^j, v_0^j, v_M^j\right)\lambda^r_l - \sum_{l=1}^{n_{r^\mathrm{i}}} \nabla^2_{s_Ms_M} r^\mathrm{i}_l\left(s_0, s_M; \rho^j, v_0^j, v_M^j\right)\mu^r_l, \\
\nabla^2_{s_0s_M}\mathscr{L} =& -\sum_{l=1}^{n_{r^\mathrm{e}}} \nabla^2_{s_0s_M} r^\mathrm{e}_l\left(s_0, s_M; \rho^j, v_0^j, v_M^j\right)\lambda^r_l - \sum_{l=1}^{n_{r^\mathrm{i}}} \nabla^2_{s_0s_M} r^\mathrm{i}_l\left(s_0, s_M; \rho^j, v_0^j, v_M^j\right)\mu^r_l, \\
\nabla^2_{s_Ms_0}\mathscr{L} =& -\sum_{l=1}^{n_{r^\mathrm{e}}} \nabla^2_{s_Ms_0} r^\mathrm{e}_l\left(s_0, s_M; \rho^j, v_0^j, v_M^j\right)\lambda^r_l - \sum_{l=1}^{n_{r^\mathrm{i}}} \nabla^2_{s_Ms_0} r^\mathrm{i}_l\left(s_0, s_M; \rho^j, v_0^j, v_M^j\right)\mu^r_l.
\end{aligned}$$

$\nabla^2_{sq}\mathscr{L}$

The Hessian block $\nabla^2_{sq}\mathscr{L}$ is given by

$$\nabla^2_{sq}\mathscr{L} = \begin{pmatrix} \nabla^2_{s_0q_0}\mathscr{L} & 0 & \cdots & 0 \\ 0 & \nabla^2_{s_1q_1}\mathscr{L} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \nabla^2_{s_{M-1}q_{M-1}}\mathscr{L} \\ 0 & \cdots & 0 & 0 \end{pmatrix},$$

with

$$\begin{aligned}
\nabla^2_{s_mq_m}\mathscr{L} =& -\sum_{l=1}^{n_x} \nabla^2_{s_mq_m} x_l\left(\tau_{m+1}; s_m, q_m; \rho^j, v_m^j\right)\lambda^{\mathrm{MC}}_{m,l} \\
& -\sum_{l=1}^{n_h} \nabla^2_{s_mq_m} h_l\left(s_m, q_m; \rho^j, v_m^j\right)\mu^h_{m,l}, \quad m = 0, \ldots, M-1.
\end{aligned}$$

$\nabla^2_{s\rho}\mathscr{L}$

The Hessian block $\nabla^2_{s\rho}\mathscr{L}$ is given by

$$\nabla^2_{s\rho}\mathscr{L} = \begin{pmatrix} \nabla^2_{s_0\rho}\mathscr{L} \\ \vdots \\ \nabla^2_{s_M\rho}\mathscr{L} \end{pmatrix},$$

with

$$\begin{aligned}
\nabla^2_{s_0\rho}\mathscr{L} =& -\sum_{l=1}^{n_x} \nabla^2_{s_0\rho} x_l\left(\tau_0; s_0, q_0; \rho^j, v_0^j\right)\lambda^{\mathrm{MC}}_{0,l} \\
&- \sum_{l=1}^{n_h} \nabla^2_{s_0\rho} h_l\left(s_0, q_0; \rho^j, v_0^j\right)\mu^h_{0,l} \\
&- \sum_{l=1}^{n_{r^e}} \nabla^2_{s_0\rho} r^e_l\left(s_0, s_M; \rho^j, v_0^j, v_M^j\right)\lambda^r_l - \sum_{l=1}^{n_{r^i}} \nabla^2_{s_0\rho} r^i_l\left(s_0, s_M; \rho^j, v_0^j, v_M^j\right)\mu^r_l, \\
\nabla^2_{s_m\rho}\mathscr{L} =& -\sum_{l=1}^{n_x} \nabla^2_{s_m\rho} x_l\left(\tau_{m+1}; s_m, q_m; \rho^j, v_m^j\right)\lambda^{\mathrm{MC}}_{m,l} \\
&- \sum_{l=1}^{n_h} \nabla^2_{s_m\rho} h_l\left(s_m, q_m; \rho^j, v_m^j\right)\mu^h_{m,l}, \quad m = 1, \ldots, M-1, \\
\nabla^2_{s_M\rho}\mathscr{L} =& \nabla^2_{s_M\rho}\Phi\left(s_M; \rho, v_M\right) \\
&- \sum_{l=1}^{n_{r^e}} \nabla^2_{s_M\rho} r^e_l\left(s_0, s_M; \rho^j, v_0^j, v_M^j\right)\lambda^r_l - \sum_{l=1}^{n_{r^i}} \nabla^2_{s_M\rho} r^i_l\left(s_0, s_M; \rho^j, v_0^j, v_M^j\right)\mu^r_l.
\end{aligned}$$

$\nabla^2_{sv}\mathscr{L}$

The Hessian block $\nabla^2_{sv}\mathscr{L}$ is given by

$$\nabla^2_{sv}\mathscr{L} = \begin{pmatrix}
\nabla^2_{s_0v_0}\mathscr{L} & 0 & \cdots & 0 & \nabla^2_{s_0v_M}\mathscr{L} \\
0 & \nabla^2_{s_1v_1}\mathscr{L} & \ddots & \vdots & 0 \\
\vdots & \ddots & \ddots & 0 & 0 \\
0 & \cdots & 0 & \nabla^2_{s_{M-1}v_{M-1}}\mathscr{L} & 0 \\
\nabla^2_{s_Mv_0}\mathscr{L} & 0 & 0 & 0 & \nabla^2_{s_Mv_M}\mathscr{L}
\end{pmatrix},$$

with

$$\begin{aligned}
\nabla^2_{s_0v_0}\mathscr{L} =& -\sum_{l=1}^{n_x} \nabla^2_{s_0v_0} x_l\left(\tau_0; s_0, q_0; \rho^j, v_0^j\right)\lambda^{\mathrm{MC}}_{0,l} \\
&- \sum_{l=1}^{n_h} \nabla^2_{s_0v_0} h_l\left(s_0, q_0; \rho^j, v_0^j\right)\mu^h_{0,l} \\
&- \sum_{l=1}^{n_{r^e}} \nabla^2_{s_0v_0} r^e_l\left(s_0, s_M; \rho^j, v_0^j, v_M^j\right)\lambda^r_l - \sum_{l=1}^{n_{r^i}} \nabla^2_{s_0v_0} r^i_l\left(s_0, s_M; \rho^j, v_0^j, v_M^j\right)\mu^r_l, \\
\nabla^2_{s_mv_m}\mathscr{L} =& -\sum_{l=1}^{n_x} \nabla^2_{s_mv_m} x_l\left(\tau_{m+1}; s_m, q_m; \rho^j, v_m^j\right)\lambda^{\mathrm{MC}}_{m,l} \\
&- \sum_{l=1}^{n_h} \nabla^2_{s_mv_m} h_l\left(s_m, q_m; \rho^j, v_m^j\right)\mu^h_{m,l}, \quad m = 1, \ldots, M-1,
\end{aligned}$$

$$
\begin{aligned}
\nabla^2_{s_M v_M}\mathscr{L} =& \nabla^2_{s_M v_M}\Phi(s_M;\rho,v_M)\\
&- \sum_{l=1}^{n_r^{\mathrm{e}}} \nabla^2_{s_M v_M} r_l^{\mathrm{e}}\left(s_0,s_M;\rho^j,v_0^j,v_M^j\right)\lambda_l^r - \sum_{l=1}^{n_r^{\mathrm{i}}} \nabla^2_{s_M v_M} r_l^{\mathrm{i}}\left(s_0,s_M;\rho^j,v_0^j,v_M^j\right)\mu_l^r,\\
\nabla^2_{s_0 v_M}\mathscr{L} =& -\sum_{l=1}^{n_r^{\mathrm{e}}} \nabla^2_{s_0 v_M} r_l^{\mathrm{e}}\left(s_0,s_M;\rho^j,v_0^j,v_M^j\right)\lambda_l^r - \sum_{l=1}^{n_r^{\mathrm{i}}} \nabla^2_{s_0 v_M} r_l^{\mathrm{i}}\left(s_0,s_M;\rho^j,v_0^j,v_M^j\right)\mu_l^r,\\
\nabla^2_{s_M v_0}\mathscr{L} =& -\sum_{l=1}^{n_r^{\mathrm{e}}} \nabla^2_{s_M v_0} r_l^{\mathrm{e}}\left(s_0,s_M;\rho^j,v_0^j,v_M^j\right)\lambda_l^r - \sum_{l=1}^{n_r^{\mathrm{i}}} \nabla^2_{s_M v_0} r_l^{\mathrm{i}}\left(s_0,s_M;\rho^j,v_0^j,v_M^j\right)\mu_l^r.
\end{aligned}
$$

$\nabla^2_{qq}\mathscr{L}$

The Hessian block $\nabla^2_{qq}\mathscr{L}$ is given by

$$
\nabla^2_{qq}\mathscr{L} = \begin{pmatrix}
\nabla^2_{q_0 q_0}\mathscr{L} & 0 & \cdots & 0\\
0 & \nabla^2_{q_1 q_1}\mathscr{L} & \ddots & \vdots\\
\vdots & \ddots & \ddots & 0\\
0 & \cdots & 0 & \nabla^2_{q_{M-1} q_{M-1}}\mathscr{L}
\end{pmatrix},
$$

with

$$
\begin{aligned}
\nabla^2_{q_m q_m}\mathscr{L} =& -\sum_{l=1}^{n_x} \nabla^2_{q_m q_m} x_l\left(\tau_{m+1};s_m,q_m;\rho^j,v_m^j\right)\lambda_{m,l}^{\mathrm{MC}}\\
&- \sum_{l=1}^{n_h} \nabla^2_{q_m q_m} h_l\left(s_m,q_m;\rho^j,v_m^j\right)\mu_{m,l}^h, \quad m=0,\ldots,M-1.
\end{aligned}
$$

$\nabla^2_{q\rho}\mathscr{L}$

The Hessian block $\nabla^2_{q\rho}\mathscr{L}$ is given by

$$
\nabla^2_{q\rho}\mathscr{L} = \begin{pmatrix}
\nabla^2_{q_0\rho}\mathscr{L}\\
\vdots\\
\nabla^2_{q_{M-1}\rho}\mathscr{L}
\end{pmatrix},
$$

with

$$
\begin{aligned}
\nabla^2_{q_m\rho}\mathscr{L} =& -\sum_{l=1}^{n_x} \nabla^2_{q_m\rho} x_l\left(\tau_{m+1};s_m,q_m;\rho^j,v_m^j\right)\lambda_{m,l}^{\mathrm{MC}}\\
&- \sum_{l=1}^{n_h} \nabla^2_{q_m\rho} h_l\left(s_m,q_m;\rho^j,v_m^j\right)\mu_{m,l}^h, \quad m=0,\ldots,M-1.
\end{aligned}
$$

$\nabla^2_{qv}\mathscr{L}$

The Hessian block $\nabla^2_{qv}\mathscr{L}$ is given by

$$\nabla^2_{qv}\mathscr{L} = \begin{pmatrix} \nabla^2_{q_0 v_0}\mathscr{L} & 0 & \cdots & 0 & 0 \\ 0 & \nabla^2_{q_1 v_1}\mathscr{L} & \ddots & \vdots & 0 \\ \vdots & \ddots & \ddots & 0 & 0 \\ 0 & \cdots & 0 & \nabla^2_{q_{M-1} v_{M-1}}\mathscr{L} & 0 \end{pmatrix},$$

with

$$\begin{aligned} \nabla^2_{q_m v_0}\mathscr{L} = &- \sum_{l=1}^{n_x} \nabla^2_{q_m v_0} x_l\left(\tau_{m+1}; s_m, q_m; \rho^j, v_m^j\right)\lambda^{\mathrm{MC}}_{m,l} \\ &- \sum_{l=1}^{n_h} \nabla^2_{q_m v_0} h_l\left(s_m, q_m; \rho^j, v_m^j\right)\mu^h_{m,l}, \quad m = 0, \ldots, M-1. \end{aligned}$$

$\nabla^2_{\rho\rho}\mathscr{L}$

The Hessian block $\nabla^2_{\rho\rho}\mathscr{L}$ is given by

$$\begin{aligned} \nabla^2_{\rho\rho}\mathscr{L} = &\nabla^2_{\rho\rho}\Phi\left(s_M; \rho, v_M\right) \\ &- \sum_{m=0}^{M-1}\sum_{l=1}^{n_x} \nabla^2_{\rho\rho} x_l\left(\tau_{m+1}; s_m, q_m; \rho^j, v_m^j\right)\lambda^{\mathrm{MC}}_{m,l} \\ &- \sum_{l=1}^{n_h} \nabla^2_{\rho\rho} h_l\left(s_m, q_m; \rho^j, v_m^j\right)\mu^h_{m,l} \\ &- \sum_{l=1}^{n_{r^{\mathrm{e}}}} \nabla^2_{\rho\rho} r^{\mathrm{e}}_l\left(s_0, s_M; \rho^j, v_0^j, v_M^j\right)\lambda^r_l - \sum_{l=1}^{n_{r^{\mathrm{i}}}} \nabla^2_{\rho\rho} r^{\mathrm{i}}_l\left(s_0, s_M; \rho^j, v_0^j, v_M^j\right)\mu^r_l. \end{aligned}$$

$\nabla^2_{\rho v}\mathscr{L}$

The Hessian block $\nabla^2_{\rho v}\mathscr{L}$ is given by $\nabla^2_{\rho v}\mathscr{L} = \begin{pmatrix} \nabla^2_{\rho v_0}\mathscr{L} & \cdots & \nabla^2_{\rho v_M}\mathscr{L} \end{pmatrix}$ with

$$\begin{aligned} \nabla^2_{\rho v_0}\mathscr{L} = &- \sum_{m=0}^{M-1}\sum_{l=1}^{n_x} \nabla^2_{\rho v_0} x_l\left(\tau_{m+1}; s_m, q_m; \rho^j, v_m^j\right)\lambda^{\mathrm{MC}}_{m,l} \\ &- \sum_{m=0}^{M-1}\sum_{l=1}^{n_h} \nabla^2_{\rho v_0} h_l\left(s_m, q_m; \rho^j, v_m^j\right)\mu^h_{m,l}, \\ \nabla^2_{\rho v_m}\mathscr{L} = &- \sum_{l=1}^{n_x} \nabla^2_{\rho v_m} x_l\left(\tau_{m+1}; s_m, q_m; \rho^j, v_m^j\right)\lambda^{\mathrm{MC}}_{m,l} \\ &- \sum_{l=1}^{n_h} \nabla^2_{\rho v_m} h_l\left(s_m, q_m; \rho^j, v_m^j\right)\mu^h_{m,l}, \quad m = 1, \ldots, M-1, \\ \nabla^2_{\rho v_M}\mathscr{L} = &\nabla^2_{\rho v_M}\Phi\left(s_M; \rho, v_M\right) \\ &- \sum_{l=1}^{n_{r^{\mathrm{e}}}} \nabla^2_{\rho v_M} r^{\mathrm{e}}_l\left(s_0, s_M; \rho^j, v_0^j, v_M^j\right)\lambda^r_l - \sum_{l=1}^{n_{r^{\mathrm{i}}}} \nabla^2_{\rho v_M} r^{\mathrm{i}}_l\left(s_0, s_M; \rho^j, v_0^j, v_M^j\right)\mu^r_l. \end{aligned}$$

$\nabla^2_{vv}\mathscr{L}$

The Hessian block $\nabla^2_{vv}\mathscr{L}$ is given by

$$\nabla^2_{vv}\mathscr{L} = \begin{pmatrix} \nabla^2_{v_0v_0}\mathscr{L} & 0 & \cdots & 0 & \nabla^2_{v_0v_M}\mathscr{L} \\ 0 & \nabla^2_{v_1v_1}\mathscr{L} & \ddots & \vdots & 0 \\ \vdots & \ddots & \ddots & 0 & 0 \\ 0 & \cdots & 0 & \nabla^2_{v_{M-1}v_{M-1}}\mathscr{L} & 0 \\ \nabla^2_{v_Mv_0}\mathscr{L} & 0 & 0 & 0 & \nabla^2_{v_Mv_M}\mathscr{L} \end{pmatrix},$$

with

$$\begin{aligned}
\nabla^2_{v_0v_0}\mathscr{L} =& -\sum_{m=0}^{M-1}\sum_{l=1}^{n_x}\nabla^2_{v_0v_0}x_l\left(\tau_{m+1};s_m,q_m;\rho^j,v_m^j\right)\lambda^{\mathrm{MC}}_{m,l} \\
&-\sum_{m=0}^{M-1}\sum_{l=1}^{n_h}\nabla^2_{v_0v_0}h_l\left(s_m,q_m;\rho^j,v_m^j\right)\mu^h_{m,l}, \\
&-\sum_{l=1}^{n_{r^\mathrm{e}}}\nabla^2_{v_0v_0}r^\mathrm{e}_l\left(s_0,s_M;\rho^j,v_0^j,v_M^j\right)\lambda^r_l-\sum_{l=1}^{n_{r^\mathrm{i}}}\nabla^2_{v_0v_0}r^\mathrm{i}_l\left(s_0,s_M;\rho^j,v_0^j,v_M^j\right)\mu^r_l, \\
\nabla^2_{v_mv_m}\mathscr{L} =& -\sum_{l=1}^{n_x}\nabla^2_{v_mv_m}x_l\left(\tau_{m+1};s_m,q_m;\rho^j,v_m^j\right)\lambda^{\mathrm{MC}}_{m,l} \\
&-\sum_{l=1}^{n_h}\nabla^2_{v_mv_m}h_l\left(s_m,q_m;\rho^j,v_m^j\right)\mu^h_{m,l}, \quad m=1,\dots,M-1, \\
\nabla^2_{v_Mv_M}\mathscr{L} =& \nabla^2_{v_Mv_M}\Phi\left(s_M;\rho,v_M\right) \\
&-\sum_{l=1}^{n_{r^\mathrm{e}}}\nabla^2_{v_Mv_M}r^\mathrm{e}_l\left(s_0,s_M;\rho^j,v_0^j,v_M^j\right)\lambda^r_l-\sum_{l=1}^{n_{r^\mathrm{i}}}\nabla^2_{v_Mv_M}r^\mathrm{i}_l\left(s_0,s_M;\rho^j,v_0^j,v_M^j\right)\mu^r_l, \\
\nabla^2_{v_0v_M}\mathscr{L} =& -\sum_{l=1}^{n_{r^\mathrm{e}}}\nabla^2_{v_0v_M}r^\mathrm{e}_l\left(s_0,s_M;\rho^j,v_0^j,v_M^j\right)\lambda^r_l-\sum_{l=1}^{n_{r^\mathrm{i}}}\nabla^2_{v_0v_M}r^\mathrm{i}_l\left(s_0,s_M;\rho^j,v_0^j,v_M^j\right)\mu^r_l, \\
\nabla^2_{v_Mv_0}\mathscr{L} =& -\sum_{l=1}^{n_{r^\mathrm{e}}}\nabla^2_{v_Mv_0}r^\mathrm{e}_l\left(s_0,s_M;\rho^j,v_0^j,v_M^j\right)\lambda^r_l-\sum_{l=1}^{n_{r^\mathrm{i}}}\nabla^2_{v_Mv_0}r^\mathrm{i}_l\left(s_0,s_M;\rho^j,v_0^j,v_M^j\right)\mu^r_l.
\end{aligned}$$

D. Condensing with External Inputs

In this appendix, we present the condensing procedure for QP (6.10).

In analogy to Equation (3.29) in Subsection 3.3.2 we use

$$\begin{aligned}
\delta_m &:= x\left(\tau_{m+1}; s_m, q_m; \rho^j, v_m^j\right) - s_{m+1}, && m = 0, \dots, M-1,\\
h_m &:= h\left(s_m, q_m; \rho^j, v_m^j\right), && m = 0, \dots, M-1,\\
r^{\mathrm{e}} &:= r^{\mathrm{e}}\left(s_0, s_M; \rho^j, v_0^j, v_M^j\right),\\
r^{\mathrm{i}} &:= r^{\mathrm{i}}\left(s_0, s_M; \rho^j, v_0^j, v_M^j\right).
\end{aligned}$$

Similarly to the condensing matrices $E_m^{s_0}$ and $E_m^{q_l}$, cf. Equations (3.31) and (3.32), we define E_m^{ρ} and $E_m^{v_l}$ for $m = 1, \dots, M$ and $l = 0, \dots, M-1$ by

$$\begin{aligned}
E_m^{\rho} &= S_{m-1}^{s} E_{m-1}^{\rho} + S_{m-1}^{\rho}, \ m = 2, \dots, M, && \text{with} && E_1^{\rho} = S_0^{\rho},\\
E_m^{v_l} &= S_{m-1}^{s} E_{m-1}^{v_l}, \ m = l+2, \dots, M, && \text{with} && E_{l+1}^{v_l} = S_l^{v}.
\end{aligned}$$

We can then proceed as for the condensing of QP (3.24), cf. Equation (3.30), to compute Δs_{m+1} for $m = 0, \dots, M-1$ by

$$\Delta s_{m+1} = E_{m+1}^{s_0} \Delta s_0 + \sum_{l=0}^{m} \left(E_{m+1}^{q_l} \Delta q_l + E_{m+1}^{v_l} \Delta v_l \right) + E_{m+1}^{\rho} \Delta \rho + \hat{\delta}_m,$$

where the condensed matching conditions residuals $\hat{\delta}_m$ are as before defined by the recursion

$$\hat{\delta}_m = S_{m-1}^{s} \hat{\delta}_{m-1} + \delta_m, \ m = 1, \dots, M-1, \ \text{with } \hat{\delta}_0 = \delta_0. \tag{3.33}$$

To reformulate the objective function in terms of the remaining optimization variables, we need to adjust the condensed Hessian $\hat{B}$. The condensed Hessian has the block structure

$$\hat{B} = \begin{pmatrix} \hat{B}^{s_0 s_0} & \hat{B}^{s_0 q} & \hat{B}^{s_0 \rho} & \hat{B}^{s_0 v} \\ \hat{B}^{q s_0} & \hat{B}^{q q} & \hat{B}^{q \rho} & \hat{B}^{q v} \\ \hat{B}^{\rho s_0} & \hat{B}^{\rho q} & \hat{B}^{\rho \rho} & \hat{B}^{\rho v} \\ \hat{B}^{v s_0} & \hat{B}^{v q} & \hat{B}^{v \rho} & \hat{B}^{v v} \end{pmatrix}.$$

In the following, we give the expressions for the respective blocks.

First, we consider the blocks in the first column and accordingly the first row. The blocks $\hat{B}^{q s_0}$ and $\hat{B}^{v s_0}$ are themselves partitioned into blocks by

$$\hat{B}^{q s_0} = \begin{pmatrix} \hat{B}^{q_0 s_0} \\ \vdots \\ \hat{B}^{q_{M-1} s_0} \end{pmatrix}, \quad \hat{B}^{v s_0} = \begin{pmatrix} \hat{B}^{v_0 s_0} \\ \vdots \\ \hat{B}^{v_M s_0} \end{pmatrix}.$$

The first column and row are then made up from the blocks

$$\begin{aligned}
\hat{B}^{s_0 s_0} &:= B^{s_0 s_0} + \sum_{l=1}^{M} \left(E_l^{s_0}\right)^T B^{s_l s_l} E_l^{s_0} + B^{s_0 s_M} E_M^{s_0} + \left(E_M^{s_0}\right)^T B^{s_M s_0}, \\
\left(\hat{B}^{q_0 s_0}\right)^T = \hat{B}^{s_0 q_0} &:= B^{s_0 q_0} + B^{s_0 s_M} E_M^{q_0} + \sum_{l=1}^{M} \left(E_l^{s_0}\right)^T B^{s_l s_l} E_l^{q_0}, \\
\left(\hat{B}^{q_m s_0}\right)^T = \hat{B}^{s_0 q_m} &:= \left(E_m^{s_0}\right)^T B^{s_m q_m} + B^{s_0 s_M} E_M^{q_m} + \sum_{l=m+1}^{M} \left(E_l^{s_0}\right)^T B^{s_l s_l} E_l^{q_m}, \ m = 1, \dots, M-1, \\
\left(\hat{B}^{\rho s_0}\right)^T = \hat{B}^{s_0 \rho} &:= B^{s_0 \rho} + B^{s_0 s_M} E_M^{\rho} + \sum_{l=1}^{M} \left(E_l^{s_0}\right)^T B^{s_l s_l} E_l^{\rho} + \left(E_l^{s_0}\right)^T B^{s_l \rho}, \\
\left(\hat{B}^{v_0 s_0}\right)^T = \hat{B}^{s_0 v_0} &:= B^{s_0 v_0} + B^{s_0 s_M} E_M^{v_0} + \left(E_M^{s_0}\right)^T B^{s_M v_0} + \sum_{l=1}^{M} \left(E_l^{s_0}\right)^T B^{s_l s_l} E_l^{v_0}, \\
\left(\hat{B}^{v_m s_0}\right)^T = \hat{B}^{s_0 v_m} &:= \left(E_m^{s_0}\right)^T B^{s_m v_m} + B^{s_0 s_M} E_M^{v_m} + \sum_{l=m+1}^{M} \left(E_l^{s_0}\right)^T B^{s_l s_l} E_l^{v_m}, \ m = 1, \dots, M-1, \\
\left(\hat{B}^{v_M s_0}\right)^T = \hat{B}^{s_0 v_M} &:= B^{s_0 v_M} + \left(E_M^{s_0}\right)^T B^{s_M v_M}.
\end{aligned}$$

Next, we consider the blocks in the second column and accordingly the second row that have not been considered already. We partition the block $\hat{B}^{qq}$ into the blocks

$$\hat{B}^{qq} = \begin{pmatrix} \hat{B}^{q_0 q_0} & \dots & \hat{B}^{q_0 q_{M-1}} \\ \vdots & & \vdots \\ \hat{B}^{q_{M-1} q_0} & \dots & \hat{B}^{q_{M-1} q_{M-1}} \end{pmatrix},$$

which are given by

$$\begin{aligned}
\hat{B}^{q_m q_m} &:= B^{q_m q_m} + \sum_{l=m+1}^{M} \left(E_l^{q_m}\right)^T B^{s_l s_l} E_l^{q_m}, \ m = 0, \dots, M-1, \\
\left(\hat{B}^{q_p q_m}\right)^T = \hat{B}^{q_m q_p} &:= \left(E_p^{q_m}\right)^T B^{s_p q_p} + \sum_{l=p+1}^{M} \left(E_l^{q_m}\right)^T B^{s_l s_l} E_l^{q_p}, \ m = 0, \dots, M-2, \ p = m+1, \dots, M-1.
\end{aligned}$$

The block $\hat{B}^{\rho q}$ is given by

$$\hat{B}^{\rho q} = \begin{pmatrix} \hat{B}^{\rho q_0} \\ \vdots \\ \hat{B}^{\rho q_{M-1}} \end{pmatrix},$$

where

$$\begin{aligned}
\left(\hat{B}^{q_0 \rho}\right)^T = \hat{B}^{\rho q_0} &:= B^{\rho q_0} + \sum_{l=1}^{M} \left(E_l^{\rho}\right)^T B^{s_l s_l} E_l^{q_0} + B^{\rho s_l} E_l^{q_0}, \\
\left(\hat{B}^{q_m \rho}\right)^T = \hat{B}^{\rho q_m} &:= B^{\rho q_m} + \left(E_m^{\rho}\right)^T B^{s_m q_m} + \sum_{l=m+1}^{M} \left(E_l^{\rho}\right)^T B^{s_l s_l} E_l^{q_m} + B^{\rho s_l} E_l^{q_m}, \ m = 1, \dots, M-1.
\end{aligned}$$

The block $\hat{B}^{vq}$ is given by

$$\hat{B}^{vq} = \begin{pmatrix} \hat{B}^{v_0 q_0} & \dots & \hat{B}^{v_0 q_{M-1}} \\ \vdots & & \vdots \\ \hat{B}^{v_M q_0} & \dots & \hat{B}^{v_M q_{M-1}} \end{pmatrix},$$

where

$$\left(\hat{B}^{q_0 v_0}\right)^T = \hat{B}^{v_0 q_0} := B^{v_0 q_0} + B^{v_0 s_M} E_M^{q_0} + \sum_{l=1}^{M} \left(E_l^{v_0}\right) B^{s_l s_l} E_l^{q_0},$$

$$\left(\hat{B}^{q_m v_m}\right)^T = \hat{B}^{v_m q_m} := B^{v_m q_m} + \sum_{l=m+1}^{M} \left(E_l^{v_m}\right) B^{s_l s_l} E_l^{q_m},\ m = 1, \ldots, M-1,$$

$$\left(\hat{B}^{q_p v_m}\right)^T = \hat{B}^{v_m q_p} := \left(E_p^{v_m}\right)^T B^{s_p q_p} + \sum_{l=m+1}^{M} \left(E_l^{v_m}\right) B^{s_l s_l} E_l^{q_p},\ m = 0, \ldots, M-2,\ p = m+1, \ldots, M-1,$$

$$\left(\hat{B}^{q_m v_p}\right)^T = \hat{B}^{v_p q_m} := B^{v_p s_p} E_p^{q_m} + \sum_{l=m+1}^{M} \left(E_l^{v_p}\right) B^{s_l s_l} E_l^{q_m},\ m = 0, \ldots, M-2,\ p = m+1, \ldots, M-1,$$

$$\left(\hat{B}^{q_m v_M}\right)^T = \hat{B}^{v_M q_m} := B^{v_M s_M} E_M^{q_m},\ m = 0, \ldots, M-1.$$

In the third column and the third row the blocks $\hat{B}^{\rho\rho}$ and $\hat{B}^{v\rho}$ remain. The block $\hat{B}^{\rho\rho}$ is given by

$$\hat{B}^{\rho\rho} = B^{\rho\rho} + \sum_{l=1}^{M} \left(E_l^{\rho}\right)^T B^{s_l s_l} E_l^{\rho} + \left(E_l^{\rho}\right) B^{\rho s_l} + B^{s_l \rho} E_l^{\rho}.$$

The block $\hat{B}^{v\rho}$ is given by

$$\hat{B}^{v\rho} = \begin{pmatrix} \hat{B}^{v_0 \rho} \\ \vdots \\ \hat{B}^{v_M \rho} \end{pmatrix},$$

where

$$\left(\hat{B}^{\rho v_0}\right)^T = \hat{B}^{v_0 \rho} := B^{v_0 \rho} + B^{v_0 s_M} E_M^{\rho},$$

$$\left(\hat{B}^{\rho v_m}\right)^T = \hat{B}^{v_m \rho} := B^{v_m \rho} + B^{v_m s_m} E_M^{\rho} + \sum_{l=m+1}^{M} \left(E_l^{v_m}\right)^T B^{s_l s_l} E_l^{\rho} + \left(E_l^{v_m}\right)^T B^{s_l \rho},\ m = 1, \ldots, M-1,$$

$$\left(\hat{B}^{\rho v_M}\right)^T = \hat{B}^{v_M \rho} := B^{v_M \rho} + B^{v_M s_M} E_M^{\rho}.$$

The final block $\hat{B}^{vv}$ has the structure

$$\hat{B}^{vv} = \begin{pmatrix} \hat{B}^{v_0 v_0} & \ldots & \hat{B}^{v_0 v_M} \\ \vdots & & \vdots \\ \hat{B}^{v_M v_0} & \ldots & \hat{B}^{v_M v_M} \end{pmatrix},$$

where

$$\hat{B}^{v_0 v_0} := B^{v_0 v_0} + B^{v_0 s_M} E_M^{v_0} + \left(E_M^{v_0}\right)^T B^{s_M v_0} + \sum_{l=1}^{M} \left(E_l^{v_0}\right)^T B^{s_l s_l} E_l^{v_0},$$

$$\hat{B}^{v_m v_m} := B^{v_m v_m} + \sum_{l=m+1}^{M} \left(E_l^{v_m}\right)^T B^{s_l s_l} E_l^{v_m},\ m = 1, \ldots, M-1,$$

$$\hat{B}^{v_M v_M} := B^{v_M v_M},$$

$$\left(\hat{B}^{v_p v_m}\right)^T = \hat{B}^{v_m v_p} := \left(E_p^{v_m}\right)^T B^{s_p v_p} + \sum_{l=p+1}^{M} \left(E_l^{v_m}\right)^T B^{s_l s_l} E_l^{v_p},\ m = 0, \dots, M-2,\ p = m+1, \dots, M-1,$$

$$\left(\hat{B}^{v_M v_0}\right)^T = \hat{B}^{v_0 v_M} := B^{v_0 v_M} + E_M^{v_0} B^{s_M v_M},$$

$$\left(\hat{B}^{v_M v_m}\right)^T = \hat{B}^{v_m v_M} := E_M^{v_m} B^{s_M v_M},\ m = 1, \dots, M-1.$$

Next, we turn our attention to the condensed gradients $\hat{b}^{s_0}$, $\hat{b}^q$, $\hat{b}^\rho$, and $\hat{b}^v$. The condensed gradients $\hat{b}^{s_0}$ and $\hat{b}^q$ are as given in Equations (3.37) and (3.38). The condensed gradient $\hat{b}^v$ is subdivided according to

$$\hat{b}^v = \begin{pmatrix} \hat{b}^{v_0} \\ \vdots \\ \hat{b}^{v_M} \end{pmatrix}.$$

The condensed gradients $\hat{b}^\rho$ and $\hat{b}^v$ are then given by

$$\begin{aligned}
\hat{b}^\rho &:= b^\rho + \sum_{l=1}^{M} B^{\rho s_l} \hat{\delta}_{l-1} + \left(E_l^\rho\right)^T \left(B^{s_l s_l} \hat{\delta}_{l-1} + b^{s_l}\right), \\
\hat{b}^{v_0} &:= b^{v_0} + B^{v_0 s_M} \hat{\delta}_{M-1} + \sum_{l=1}^{M} \left(E_l^{v_0}\right)^T \left(B^{s_l s_l} \hat{\delta}_{l-1} + b^{s_l}\right), \\
\hat{b}^{v_m} &:= b^{v_m} + B^{v_m s_m} \hat{\delta}_{m-1} + \sum_{l=m+1}^{M} \left(E_l^{v_m}\right)^T \left(B^{s_l s_l} \hat{\delta}_{l-1} + b^{s_l}\right),\ m = 1, \dots, M-1, \\
\hat{b}^{v_M} &:= b^{v_M} + B^{v_M s_M} \hat{\delta}_{M-1}.
\end{aligned}$$

Finally, we reformulate the constraints in terms of the remaining optimization variables. To that end, we introduce for the mixed state-control constraints in analogy to Equation (3.39)

$$\begin{aligned}
\hat{H}_m^\rho &:= H_m^s E_m^\rho + H_m^\rho, && m = 1, \dots, M-1, \\
\hat{H}_m^{v_l} &:= \begin{cases} H_m^s E_m^{v_l}, & \text{if } l = 0, \dots, m-1, \\ H_m^v, & \text{if } l = m \end{cases} && m = 1, \dots, M-1, \\
\hat{h}_m &:= h_m + H_m^s \hat{\delta}_{m-1}, && m = 1, \dots, M-1,
\end{aligned}$$

and for the boundary constraints in analogy to Equation (3.40)

$$\begin{aligned}
\hat{R}_\rho^{\mathrm{e}} &:= R_{s_M}^{\mathrm{e}} E_m^\rho + R_\rho^{\mathrm{e}}, \\
\hat{R}_{v_l}^{\mathrm{e}} &:= \begin{cases} R_{s_M}^{\mathrm{e}} E_M^{v_0} + R_{v_0}^{\mathrm{e}}, & \text{if } l = 0, \\ R_{s_M}^{\mathrm{e}} E_M^{v_l}, & \text{if } l = 1, \dots, M-1, \end{cases} \\
\hat{R}_\rho^{\mathrm{i}} &:= R_{s_M}^{\mathrm{i}} E_m^\rho + R_\rho^{\mathrm{i}}, \\
\hat{R}_{v_l}^{\mathrm{i}} &:= \begin{cases} R_{s_M}^{\mathrm{i}} E_M^{v_0} + R_{v_0}^{\mathrm{i}}, & \text{if } l = 0, \\ R_{s_M}^{\mathrm{i}} E_M^{v_l}, & \text{if } l = 1, \dots, M-1, \end{cases} \\
\hat{r}^{\mathrm{e}} &:= r^{\mathrm{e}} + R_{s_M}^{\mathrm{e}} \hat{\delta}_{M-1}, \\
\hat{r}^{\mathrm{i}} &:= r^{\mathrm{i}} + R_{s_M}^{\mathrm{i}} \hat{\delta}_{M-1}.
\end{aligned}$$

With that, we arrive at QP (6.16) as the condensed form of QP (6.10).

E. Condensing for SensEIS Feedback

In this appendix, we present the condensing procedure for QP (7.23).

In contrast to Subsection 3.3.2, we have to add the disturbances caused by the changing parameters and external inputs to the residuals of the constraints. In analogy to Equation (3.29) we thus define

$$
\begin{aligned}
\delta_m &:= S_m^\rho \Delta\rho^j + S_m^v \Delta v_m^j, && m = 0, \dots, M-1,\\
h_m &:= H_m^\rho \Delta\rho^j + H_m^v \Delta v_m^j + \bar{h}_m, && m = 0, \dots, M-1,\\
r^{\mathrm{e}} &:= R_\rho^{\mathrm{e}} \Delta\rho^j + R_{v_0}^{\mathrm{e}} \Delta v_0^j + R_{v_M}^{\mathrm{e}} \Delta v_M^j,\\
r^{\mathrm{i}} &:= R_\rho^{\mathrm{i}} \Delta\rho^j + R_{v_0}^{\mathrm{i}} \Delta v_0^j + R_{v_M}^{\mathrm{i}} \Delta v_M^j + \bar{r}^{\mathrm{i}}.
\end{aligned}
$$

As all residuals are evaluated at the primal solution $\bar{s}, \bar{q}$ of NLP (7.10) with the reference parameters $\hat{\bar{x}}$, $\bar{\rho}$, and $\bar{v}$, it is $x\left(\tau_{m+1}; \bar{s}_m, \bar{q}_m; \bar{\rho}, \bar{v}_m\right) - \bar{s}_{m+1} = 0$ for all $m = 0, \dots, M-1$ and $r^{\mathrm{e}} = 0$.

With that we can write (7.23) as

$$
\begin{aligned}
\min_{\substack{\Delta s = \left(\Delta s_0^T, \dots, \Delta s_M^T\right)^T \in \mathbb{R}^{n_s} \\ \Delta q = \left(\Delta q_0^T, \dots, \Delta q_{M-1}^T\right)^T \in \mathbb{R}^{n_q}}} \quad & \frac{1}{2} \begin{pmatrix} \Delta s \\ \Delta q \end{pmatrix}^T \begin{pmatrix} B^{ss} & B^{sq} \\ B^{qs} & B^{qq} \end{pmatrix} \begin{pmatrix} \Delta s \\ \Delta q \end{pmatrix} + \begin{pmatrix} \Delta x^j \\ \Delta\rho^j \\ \Delta v^j \end{pmatrix}^T \begin{pmatrix} B^{\hat{x}s_m} & B^{\hat{x}q} \\ B^{\rho s} & B^{\rho q} \\ B^{vs} & B^{vq} \end{pmatrix} \begin{pmatrix} \Delta s \\ \Delta q \end{pmatrix} + \begin{pmatrix} b^s \\ b^q \end{pmatrix}^T \begin{pmatrix} \Delta s \\ \Delta q \end{pmatrix} \\
\text{s.t.} \quad & 0 = S_m^s \Delta s_m + S_m^q \Delta q_m - \Delta s_{m+1} + \delta_m, && m = 0, \dots, M-1,\\
& 0 \le H_m^s \Delta s_m + H_m^q \Delta q_m + h_m, && m = 0, \dots, M-1,\\
& 0 = R_{s_0}^{\mathrm{e}} \Delta s_0 + R_{s_M}^{\mathrm{e}} \Delta s_M + r^{\mathrm{e}},\\
& 0 \le R_{s_0}^{\mathrm{i}} \Delta s_0 + R_{s_M}^{\mathrm{i}} \Delta s_M + r^{\mathrm{i}},\\
& 0 = \Delta x^j - \Delta s_0,
\end{aligned}
$$

where the constraints are now in a form that is analogous to the one used in QP (3.24) for which the traditional condensing is formulated.

Similarly to the condensing matrices $E_m^{s_0}$ and $E_m^{q_l}$, cf. Equations (3.31) and (3.32), we define E_m^ρ and $E_m^{v_l}$ for $m = 1, \dots, M$ and $l = 0, \dots, M-1$ by

$$
\begin{aligned}
E_m^\rho &= S_{m-1}^s E_{m-1}^\rho + S_{m-1}^\rho, \ m = 2, \dots, M, && \text{with} \quad E_1^\rho = S_0^\rho,\\
E_m^{v_l} &= S_{m-1}^s E_{m-1}^{v_l}, \ m = l+2, \dots, M, && \text{with} \quad E_{l+1}^{v_l} = S_l^v.
\end{aligned}
$$

We can then proceed as for the condensing of QP (3.24), cf. Equation (3.30), to compute Δs_{m+1} for $m = 0, \dots, M-1$ by

$$
\Delta s_{m+1} = E_{m+1}^{s_0} \Delta s_0 + \sum_{l=0}^{m} E_{m+1}^{q_l} \Delta q_l + \hat{\delta}_m,
$$

where the condensed matching conditions residuals $\hat{\delta}_m$ are as before defined by the recursion

$$\hat{\delta}_m = S^s_{m-1}\hat{\delta}_{m-1} + \delta_m,\ m = 1,\dots,M-1,\ \text{with}\ \hat{\delta}_0 = \delta_0. \tag{3.33}$$

The explicit form of $\hat{\delta}_m$ for QP (7.23) given by

$$\hat{\delta}_m = E^{\rho}_{m+1}\Delta\rho^j + \sum_{l=0}^{m} E^{v_l}_{m+1}\Delta v^j_l. \tag{E.1}$$

The condensed Hessian remains given by Equations (3.35) and (3.36).

The general form of the condensed gradients $\hat{b}^{s_0}$ and $\hat{b}^{q}$ given by Equations (3.37) and (3.38) also remains the same. For convenience however, we give the full expressions for them where we have incorporated the new definition of $\hat{\delta}_m$. The condensed gradient $\hat{b}^{s_0}$ is given by

It is $B^{\hat{x}s_m} = 0$ for $m = 1,\dots,M$, cf. Appendix C. Therefore, the terms $\left(B^{\hat{x}s_m}\right)^T\Delta x^j$ do not occur for $m = 1,\dots,M$ in the following expressions for $\hat{b}^{s_0}$ and $\hat{b}^{q_m}$.

$$\begin{aligned}
\hat{b}^{s_0} &= b^{s_0} + \sum_{m=1}^{M}\left(E^{s_0}_m\right)^T B^{s_m s_m} b^{s_m} \\
&\quad + \left((B^{\rho s_0} + B^{\rho s_M})^T + B^{s_0 s_M}E^{\rho}_M\right)\Delta\rho^j \\
&\quad + \sum_{m=1}^{M}\left(E^{s_0}_m\right)^T B^{s_m s_m}\left(E^{\rho}_m + (B^{\rho s_m})^T\right)\Delta\rho^j \\
&\quad + \sum_{m=1}^{M}\left(E^{s_0}_m\right)^T B^{s_m s_m}\left((B^{v s})^T\Delta v^j_m + \sum_{l=0}^{m-1} B^{s_0 s_M}E^{v_l}_m\Delta v^j_l\right) \\
&\quad + \sum_{l=0}^{M} B^{s_0 s_M}E^{v_l}_M\Delta v^j_l \\
&\quad + (B^{v_0 s_0} + B^{v_0 s_M})^T\Delta v^j_0 + (B^{v_M s_0} + B^{v_M s_M})^T\Delta v^j_M \\
&\quad + \left(B^{\hat{x}s_0}\right)^T\Delta x^j.
\end{aligned}$$

The condensed gradient $\hat{b}^{q_m}$ for $m = 0,\dots,M-1$ is given by

$$\begin{aligned}
\hat{b}^{q_m} &= b^{q_m} + \sum_{l=m+1}^{M}\left(E^{q_m}_l\right)^T b^{s_l} \\
&\quad + \sum_{l=1}^{m-1} B^{q_m s_m}E^{v_l}_m\Delta v^j_l \\
&\quad + \left(B^{q_m s_m}E^{\rho}_m + \sum_{l=m+1}^{M}\left(E^{q_m}_l\right)^T\left((B^{\rho s_l})^T + B^{s_l s_l}E^{\rho}_l\right)\right)\Delta\rho^j \\
&\quad + \sum_{l=m+1}^{M}\left(E^{q_m}_l\right)^T(B^{v_l s_l})^T\Delta v^j_l \\
&\quad + \sum_{l=m+1}^{M}\sum_{k=0}^{l-1}\left(E^{q_m}_l\right)^T B^{s_l s_l}E^{v_k}_l\Delta v^j_k.
\end{aligned}$$

In general, Equation (3.39) continues to describe the condensed residuals of the mixed state-control constraints. The same holds for Equation (3.40c) and the residuals of the boundary constraints.

To explicitly state the form that $\hat{h}_m$ takes with the condensed residuals $\hat{\delta}_m$ of the matching conditions as given in Equation (E.1), we introduce in analogy to Equation (3.39)

$$\begin{aligned}
\hat{H}_m^{\rho} &:= H_m^s E_m^{\rho} + H_m^{\rho}, && m = 1, \dots, M-1, \\
\hat{H}_m^{v_l} &:= \begin{cases} H_m^s E_m^{v_l}, & \text{if } l = 0, \dots, m-1, \\ H_m^v, & \text{if } l = m \end{cases} && m = 1, \dots, M-1.
\end{aligned}$$

The explicit form of $\hat{h}_m$ for $m = 1, \dots, M-1$ is then

$$\hat{h}_m = \bar{h}_m + \hat{H}_m^{\rho} \Delta\rho^j + \sum_{l=0}^{m} \hat{H}_m^{v_l} \Delta v_l^j.$$

We proceed similarly for the boundary constraints (7.23d) - (7.23e). I.e., we define for $m = 1, \dots, M-1$ in analogy to Equation (3.40)

$$\begin{aligned}
\hat{R}_{\rho}^{\mathrm{e}} &:= R_{s_M}^{\mathrm{e}} E_m^{\rho} + R_{\rho}^{\mathrm{e}}, \\
\hat{R}_{v_l}^{\mathrm{e}} &:= \begin{cases} R_{s_M}^{\mathrm{e}} E_M^{v_0} + R_{v_0}^{\mathrm{e}}, & \text{if } l = 0, \\ R_{s_M}^{\mathrm{e}} E_M^{v_l}, & \text{if } l = 1, \dots, M-1, \end{cases} \\
\hat{R}_{\rho}^{\mathrm{i}} &:= R_{s_M}^{\mathrm{i}} E_m^{\rho} + R_{\rho}^{\mathrm{i}}, \\
\hat{R}_{v_l}^{\mathrm{i}} &:= \begin{cases} R_{s_M}^{\mathrm{i}} E_M^{v_0} + R_{v_0}^{\mathrm{i}}, & \text{if } l = 0, \\ R_{s_M}^{\mathrm{i}} E_M^{v_l}, & \text{if } l = 1, \dots, M-1. \end{cases}
\end{aligned}$$

The explicit form of $\hat{r}^{\mathrm{e}}$ and $\hat{r}^{\mathrm{i}}$ is then

$$\begin{aligned}
\hat{r}^{\mathrm{e}} &= \hat{R}_{\rho}^{\mathrm{e}} \Delta\rho^j + R_{v_M}^{\mathrm{e}} \Delta v_M^j \sum_{l=0}^{M-1} \hat{R}_{\rho}^{\mathrm{e}} \Delta v_l^j, \\
\hat{r}^{\mathrm{i}} &= \bar{r}^{\mathrm{i}} + \hat{R}_{\rho}^{\mathrm{i}} \Delta\rho^j + R_{v_M}^{\mathrm{i}} \Delta v_M^j \sum_{l=0}^{M-1} \hat{R}_{\rho}^{\mathrm{i}} \Delta v_l^j.
\end{aligned}$$

Finally, we define

$$\begin{aligned}
h_0(\rho^j, v^j) &:= h_0, \\
\hat{h}_m(\rho^j, v^j) &:= \hat{h}_m, && m = 1, \dots, M-1, \\
\hat{r}^{\mathrm{e}}(\rho^j, v^j) &:= \hat{r}^{\mathrm{e}}, \\
\hat{r}^{\mathrm{i}}(\rho^j, v^j) &:= \hat{r}^{\mathrm{i}}
\end{aligned}$$

to arrive at QP (7.24) which is the condensed version of QP (7.23).

Bibliography

[1] R. A. Adams and J. J. Fournier. *Sobolev spaces*. 2nd ed. Vol. 140. Pure and Applied Mathematics. Amsterdam: Elsevier, 2003 (cited on pages 15, 16, 56, 58, 80).

[2] H. Akima. "A New Method of Interpolation and Smooth Curve Fitting Based on Local Procedures". In: *J. ACM* 17.4 (1970), pp. 589–602 (cited on page 84).

[3] H. Akima. "A Method of Bivariate Interpolation and Smooth Surface Fitting Based on Local Procedures". In: *Commun. ACM* 17.1 (1974), pp. 18–20 (cited on page 84).

[4] J. Albersmeyer. "Adjoint based algorithms and numerical methods for sensitivity generation and optimization of large scale dynamic systems". PhD thesis. Heidelberg University, 2010 (cited on pages 25, 47).

[5] J. Albersmeyer, D. Beigel, C. Kirches, L. Wirsching, H. G. Bock, and J. P. Schlöder. "Fast Nonlinear Model Predictive Control with an Application in Automotive Engineering". In: *Nonlinear Model Predictive Control: Towards New Challenging Applications*. Ed. by L. Magni, D. M. Raimondo, and F. Allgöwer. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 471–480 (cited on page 45).

[6] V. M. Alekseev, V. M. Tikhomirov, and S. V. Fomin. *Optimal Control*. Boston, MA: Springer US, 1987 (cited on page 11).

[7] H. W. Alt. *Linear Functional Analysis*. London: Springer London, 2016 (cited on page 56).

[8] J. Andersson. "A general-purpose software framework for dynamic optimization". PhD thesis. KU Leuven, 2013 (cited on page 38).

[9] K. Åström and T. Hägglund. *PID Controllers: Theory, Design, and Tuning*. ISA - The Instrumentation, Systems and Automation Society, 1995 (cited on page 11).

[10] K. E. Atkinson. *An introduction to numerical analysis*. 2nd ed. Milton Keynes: John Wiley & Sons, 1991 (cited on page 77).

[11] D. Augustin and H. Maurer. "Sensitivity Analysis and Real-Time Control of a Container Crane under State Constraints". In: *Online Optimization of Large Scale Systems*. Ed. by M. Grötschel, S. O. Krumke, and J. Rambau. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 69–82 (cited on page 119).

[12] R. E. Barnhill. "Representation and Approximation of Surfaces". In: *Mathematical Software*. Ed. by J. R. Rice. Academic Press, 1977, pp. 69–120 (cited on page 87).

[13] R. E. Barnhill. "Coon's patches". In: *Computers in Industry* 3.1 (1982). Double Issue- In Memory of Steven Anson Coons, pp. 37–43 (cited on page 87).

[14] R. E. Barnhill. "Computer aided surface representation and design". In: *Surfaces in Computer Aided Geometric Design*. Ed. by R. E. Barnhill and W. Boehm. Amsterdam: North-Holland, 1983, pp. 1–24 (cited on page 84).

[15] R. E. Barnhill. "Surfaces in computer aided geometric design: a survey with new results". In: *Computer Aided Geometric Design* 2.1 (1985), pp. 1–17 (cited on page 85).

[16] R. E. Barnhill, B. R. Piper, and S. E. Stead. "A multidimensional surface problem: pressure on a wing". In: *Computer Aided Geometric Design* 2.1 (1985), pp. 185–187 (cited on page 85).

[17] R. E. Barnhill and A. J. Worsey. "Smooth interpolation over hypercubes". In: *Computer Aided Geometric Design* 1.2 (1984), pp. 101–113 (cited on page 85).

[18] R. Bellman. "The theory of dynamic programming". In: *Bulletin of the American Mathematical Society* 60.6 (1954), pp. 503–515 (cited on page 25).

[19] R. Bellman. *Dynamic Programming*. Princeton, NJ: Princeton University Press, 1957 (cited on page 25).

[20] R. Bellman. "Dynamic Programming". In: *Science* 153.3731 (1966), pp. 34–37 (cited on page 25).

[21] L. D. Berkovitz. *Optimal Control Theory*. Vol. 12. New York, NY: Springer New York, 1974 (cited on page 11).

[22] D. P. Bertsekas. *Dynamic programming and optimal control*. 4th ed. Vol. 1. Athena scientific optimization and computation series. Belmont, Massachusetts: Athena Scientific, 2020 (cited on page 25).

[23] M. J. Best. "An Algorithm for the Solution of the Parametric Quadratic Programming Problem". In: *Applied Mathematics and Parallel Computing: Festschrift for Klaus Ritter*. Ed. by H. Fischer, B. Riedmüller, and S. Schäffler. Heidelberg: Physica-Verlag HD, 1996, pp. 57–76 (cited on page 43).

[24] L. T. Biegler. "A Survey on Sensitivity-based Nonlinear Model Predictive Control". In: *IFAC Proceedings Volumes* 46.32 (2013). 10th IFAC International Symposium on Dynamics and Control of Process Systems, pp. 499–510 (cited on page 119).

[25] L. T. Biegler, X. Yang, and G. A. G. Fischer. "Advances in sensitivity-based nonlinear model predictive control and dynamic real-time optimization". In: *Journal of Process Control* 30 (2015). CAB/DYCOPS 2013, pp. 104–116 (cited on page 119).

[26] F. Biral, E. Bertolazzi, and P. Bosetti. "Notes on Numerical Methods for Solving Optimal Control Problems". In: *IEEJ Journal of Industry Applications* 5.2 (2016), pp. 154–166 (cited on page 23).

[27] H. G. Bock. "Numerische Berechnung zustandsbeschränkter optimaler Steuerungen mit der Mehrzielmethode". In: *Carl-Cranz-Gesellschaft* (1978) (cited on pages 24, 25).

[28] H. G. Bock, M. Diehl, E. Kostina, and J. P. Schlöder. "Constrained Optimal Feedback Control of Systems Governed by Large Differential Algebraic Equations". In: *Real-Time PDE-Constrained Optimization*. 2007. Chap. 1, pp. 3–24 (cited on pages 3, 45, 48).

[29] H. G. Bock and K. J. Plitt. "A Multiple Shooting Algorithm for Direct Solution of Optimal Control Problems". In: *IFAC Proceedings Volumes* 17.2 (1984). 9th IFAC World Congress: A Bridge Between Control Science and Technology, Budapest, Hungary, 2-6 July 1984, pp. 1603–1608 (cited on pages 3, 25, 36).

[30] A. Britzelmeier and M. Gerdts. "Non-linear Model Predictive Control of Connected, Automatic Cars in a Road Network Using Optimal Control Methods". In: *IFAC-PapersOnLine* 51.2 (2018). 9th Vienna International Conference on Mathematical Modelling, pp. 168–173 (cited on page 11).

[31] A. Britzelmeier, M. Gerdts, and T. Rottmann. "Control of interacting vehicles using model-predictive control, generalized Nash equilibrium problems, and dynamic inversion". In: *IFAC-PapersOnLine* 53.2 (2020). 21st IFAC World Congress, pp. 15146–15153 (cited on page 11).

[32] K. Brodlie and S. Butt. "Preserving convexity using piecewise cubic interpolation". In: *Computers & Graphics* 15.1 (1991), pp. 15–23 (cited on page 81).

[33] R. Bulirsch. "Die Mehrzielmethode zur numerischen Lösung von nichtlinearen Randwertproblemen und Aufgaben der optimalen Steuerung". In: *Report der Carl-Cranz-Gesellschaft* 251 (1971) (cited on pages 24, 25).

[34] Bundesministerium für Bildung und Forschung (BMBF). *Forschung für Nachhaltigkeit. Eine Strategie des Bundesministeriums für Bildung und Forschung*. Accessed: 2025-01-13. 2020 (cited on page 1).

[35] C. Büskens. "Optimierungsmethoden und Sensitivitätsanalyse für optimale Steuerprozesse mit Steuer- und Zustands-Beschränkungen". PhD thesis. Institut für Numerische Mathematik, Universität Münster, 1998 (cited on page 119).

[36] C. Büskens and H. Maurer. "Sensitivity Analysis and Real-Time Control of Nonlinear Optimal Control Systems via Nonlinear Programming Methods". In: *Variational Calculus, Optimal Control and Applications*. Ed. by W. H. Schmidt, K. Heier, L. Bittner, and R. Bulirsch. Basel: Birkhäuser Basel, 1998, pp. 185–196 (cited on page 119).

[37] C. Büskens and H. Maurer. "Realtime control of robots with initial value perturbations via nonlinear programming methods". In: *Optimization* 47.3-4 (2000), pp. 383–405 (cited on page 119).

[38] C. Büskens and H. Maurer. “Sensitivity Analysis and Real-Time Control of Parametric Optimal Control Problems Using Nonlinear Programming Methods”. In: *Online Optimization of Large Scale Systems*. Ed. by M. Grötschel, S. O. Krumke, and J. Rambau. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 57–68 (cited on page 119).

[39] C. Büskens and H. Maurer. “Sensitivity Analysis and Real-Time Optimization of Parametric Nonlinear Programming Problems”. In: *Online Optimization of Large Scale Systems*. Ed. by M. Grötschel, S. O. Krumke, and J. Rambau. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 3–16 (cited on pages 118, 119).

[40] E. F. Camacho and C. Bordons. *Model Predictive Control*. Grimble, Michael J. and Johnson, Michael A. London: Springer London, 2007 (cited on page 11).

[41] R. E. Carlson and F. N. Fritsch. “Monotone Piecewise Bicubic Interpolation”. In: *SIAM Journal on Numerical Analysis* 22.2 (1985), pp. 386–400 (cited on page 84).

[42] R. E. Carlson and F. N. Fritsch. “An Algorithm for Monotone Piecewise Bicubic Interpolation”. In: *SIAM Journal on Numerical Analysis* 26.1 (1989), pp. 230–238 (cited on page 84).

[43] R. E. Carlson and F. N. Fritsch. “A Bivariate Interpolation Algorithm for Data that are Monotone in One Variable”. In: *SIAM Journal on Scientific and Statistical Computing* 12.4 (1991), pp. 859–866 (cited on page 84).

[44] E. Casas. “Pontryagin’s Principle for State-Constrained Boundary Control Problems of Semilinear Parabolic Equations”. In: *SIAM Journal on Control and Optimization* 35.4 (1997), pp. 1297–1327 (cited on pages 4, 56).

[45] E. Casas and F. Tröltzsch. “Stability for Semilinear Parabolic Optimal Control Problems with Respect to Initial Data”. In: *Applied Mathematics and Optimization* 86.2 (2022), p. 16 (cited on page 62).

[46] S. A. Coons. *Surfaces for computer-aided design*. Tech. rep. Was available as AD 663 504 from the National Technical Information service, Springfield, VA 22161, but has been replaced by [47]. USA: Design Division. Mech. Eng. Dept. MIT, 1964 (cited on pages 84, 87, 89).

[47] S. A. Coons. *Surfaces for computer-aided design of space forms*. Tech. rep. Project MAC-TR 41. USA: MIT, 1967 (cited on page 203).

[48] Copernicus Climate Change Service. *Global Climate Highlights 2024*. Accessed: 2025-01-10. 2024 (cited on page 1).

[49] P. Costantini. “An algorithm for computing shape-preserving interpolating splines of arbitrary degree”. In: *Journal of Computational and Applied Mathematics* 22.1 (1988), pp. 89–136 (cited on pages 78, 84, 86, 98, 100, 103).

[50] P. Costantini. “Boundary-Valued Shape-Preserving Interpolating Splines”. In: *ACM Trans. Math. Softw.* 23.2 (1997), pp. 229–251 (cited on page 84).

[51] P. Costantini and F. Fontanella. “Shape-Preserving Bivariate Interpolation”. In: *SIAM Journal on Numerical Analysis* 27.2 (1990), pp. 488–506 (cited on page 84).

[52] P. Costantini and C. Manni. “A local scheme for bivariate co-monotone interpolation”. In: *Computer Aided Geometric Design* 8.5 (1991), pp. 371–391 (cited on pages 81, 84).

[53] P. Costantini and C. Manni. “A bicubic shape-preserving blending scheme”. In: *Computer Aided Geometric Design* 13.4 (1996), pp. 307–331 (cited on page 84).

[54] P. Costantini and C. Manni. “Monotonicity-preserving interpolation of nongridded data”. In: *Computer Aided Geometric Design* 13.5 (1996), pp. 467–495 (cited on page 84).

[55] C. De Boor. *A practical guide to splines*. Applied mathematical sciences. New York: Springer, 1978 (cited on page 84).

[56] J. W. Demmel. *Applied Numerical Linear Algebra*. Society for Industrial and Applied Mathematics, 1997 (cited on page 127).

[57] M. Diehl. “Real-time optimization for large scale nonlinear processes”. PhD thesis. Heidelberg University, 2001 (cited on pages 3, 40, 43, 44, 115, 119).

[58] M. Diehl, H. G. Bock, H. Diedam, and P.-B. Wieber. “Fast Direct Multiple Shooting Algorithms for Optimal Robot Control”. In: *Fast Motions in Biomechanics and Robotics: Optimization and Feedback Control*. Ed. by M. Diehl and K. Mombaur. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 65–93 (cited on page 41).

[59] M. Diehl, H. G. Bock, and J. P. Schlöder. “Newton-type methods for the approximate solution of nonlinear programming problems in real-time”. In: *High Performance Algorithms and Software for Nonlinear Optimization*. Springer, 2003, pp. 177–200 (cited on pages 3, 40).

[60] M. Diehl, H. G. Bock, and J. P. Schlöder. “A Real-Time Iteration Scheme for Nonlinear Optimization in Optimal Feedback Control”. In: *SIAM Journal on Control and Optimization* 43.5 (2005), pp. 1714–1736 (cited on page 44).

[61] M. Diehl, H. G. Bock, J. P. Schlöder, R. Findeisen, Z. Nagy, and F. Allgöwer. “Real-time optimization and nonlinear model predictive control of processes governed by differential-algebraic equations”. In: *Journal of Process Control* 12.4 (2002), pp. 577–585 (cited on pages 3, 40).

[62] M. Diehl, H. J. Ferreau, and N. Haverbeke. “Efficient Numerical Methods for Nonlinear MPC and Moving Horizon Estimation”. In: *Nonlinear Model Predictive Control: Towards New Challenging Applications*. Ed. by L. Magni, D. M. Raimondo, and F. Allgöwer. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 391–417 (cited on page 43).

[63] M. Diehl, R. Findeisen, and F. Allgöwer. “A Stabilizing Real-Time Implementation of Nonlinear Model Predictive Control”. In: *Real-Time PDE-Constrained Optimization*. 2007. Chap. 2, pp. 25–52 (cited on page 44).

[64] M. Diehl, R. Findeisen, F. Allgöwer, H. G. Bock, and J. P. Schlöder. “Nominal stability of real-time iteration scheme for nonlinear model predictive control”. In: *IEE Proceedings-Control Theory and Applications* 152.3 (2005), pp. 296–308 (cited on pages 3, 40, 44).

[65] M. Diehl, R. Findeisen, S. Schwarzkopf, I. Uslu, F. Allgöwer, H. G. Bock, E. D. Gilles, and J. P. Schlöder. “An Efficient Algorithm for Nonlinear Model Predictive Control of Large-Scale Systems Part I: Description of the Method (Ein effizienter Algorithmus für die nichtlineare prädiktive Regelung großer Systeme Teil I: Methodenbeschreibung)”. In: *at - Automatisierungstechnik* 50.12 (2002), p. 557 (cited on page 42).

[66] M. Diehl, L. Magni, and G. De Nicolao. “Efficient NMPC of unstable periodic systems using approximate infinite horizon closed loop costing”. In: *Annual Reviews in Control* 28.1 (2004), pp. 37–45 (cited on page 41).

[67] M. Diehl, I. Uslu, R. Findeisen, S. Schwarzkopf, F. Allgöwer, H. G. Bock, T. Bürner, E. D. Gilles, A. Kienle, J. P. Schlöder, et al. “Real-time optimization for large scale processes: Nonlinear model predictive control of a high purity distillation column”. In: *Online optimization of large scale systems*. Springer, 2001, pp. 363–383 (cited on pages 3, 40, 41, 119).

[68] A. Domahidi, A. U. Zgraggen, M. N. Zeilinger, M. Morari, and C. N. Jones. “Efficient interior point methods for multistage problems arising in receding horizon control”. In: *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*. 2012, pp. 668–674 (cited on page 144).

[69] L. F. Domínguez and E. N. Pistikopoulos. “Recent Advances in Explicit Multiparametric Nonlinear Model Predictive Control”. In: *Industrial & Engineering Chemistry Research* 50.2 (2011), pp. 609–619 (cited on pages 119, 141).

[70] R. L. Dougherty, A. S. Edelman, and J. M. Hyman. “Nonnegativity-, monotonicity-, or convexity-preserving cubic and quintic Hermite interpolation”. In: *Mathematics of Computation* 52.186 (1989), pp. 471–494 (cited on page 81).

[71] G. Farin. *Curves and Surfaces for CAGD. A Practical Guide*. Fifth Edition. The Morgan Kaufmann Series in Computer Graphics. San Francisco: Morgan Kaufmann, 2002 (cited on pages 87, 90).

[72] H. J. Ferreau, H. G. Bock, and M. Diehl. “An online active set strategy to overcome the limitations of explicit MPC”. In: *International Journal of Robust and Nonlinear Control* 18.8 (2008), pp. 816–830 (cited on page 43).

[73] H. J. Ferreau, B. Houska, K. Geebelen, and M. Diehl. “Real-time control of a kite-model using an auto-generated nonlinear MPC algorithm”. In: *IFAC Proceedings Volumes* 44.1 (2011), pp. 2488–2493 (cited on page 41).

[74] H. J. Ferreau, C. Kirches, A. Potschka, H. G. Bock, and M. Diehl. “qpOASES: a parametric active-set algorithm for quadratic programming”. In: *Mathematical Programming Computation* 6.4 (2014), pp. 327–363 (cited on pages 43, 137).

[75] A. V. Fiacco. “Sensitivity analysis for nonlinear programming using penalty methods”. In: *Mathematical Programming* 10.1 (1976), pp. 287–311 (cited on pages 118, 120).

[76] A. V. Fiacco. *Introduction to Sensitivity and Stability Analysis in Nonlinear Programming*. Vol. 165. Mathematics in Science and Engineering. Academic Press, New York, 1983 (cited on page 118).

[77] A. V. Fiacco and G. P. McCormick. *Nonlinear Programming: Sequential Unconstrained Minimization Techniques*. This book was reprinted as [78]. Wiley, New York, 1968 (cited on page 118).

[78] A. V. Fiacco and G. P. McCormick. *Nonlinear Programming: Sequential Unconstrained Minimization Techniques*. Society for Industrial and Applied Mathematics, 1990 (cited on page 205).

[79] V. Fors and J. C. Gerdes. “Long-Horizon Vehicle Planning and Control Through Real-Time Iterations”. In: *2023 IEEE Intelligent Vehicles Symposium (IV)*. 2023, pp. 1–6 (cited on page 144).

[80] 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”. In: *2013 European Control Conference (ECC)*. 2013, pp. 4136–4141 (cited on pages 41, 43, 144).

[81] J. V. Frasch, L. Wirsching, S. Sager, and H. G. Bock. “Mixed-Level Iteration Schemes for Nonlinear Model Predictive Control”. In: *IFAC Proceedings Volumes* 45.17 (2012). 4th IFAC Conference on Nonlinear Model Predictive Control, pp. 138–144 (cited on pages 45, 144).

[82] J. Frey, A. Nurkanović, and M. Diehl. “Advanced-Step Real-Time Iterations With Four Levels – New Error Bounds and Fast Implementation in acados”. In: *IEEE Control Systems Letters* 8 (2024), pp. 1703–1708 (cited on page 45).

[83] F. N. Fritsch and J. Butland. “A Method for Constructing Local Monotone Piecewise Cubic Interpolants”. In: *SIAM Journal on Scientific and Statistical Computing* 5.2 (1984), pp. 300–304 (cited on page 84).

[84] F. N. Fritsch and R. E. Carlson. “Monotone Piecewise Cubic Interpolation”. In: *SIAM Journal on Numerical Analysis* 17.2 (1980), pp. 238–246 (cited on page 84).

[85] F. N. Fritsch and R. E. Carlson. “Monotonicity preserving bicubic interpolation: A progress report”. In: *Computer Aided Geometric Design* 2.1 (1985), pp. 117–121 (cited on page 84).

[86] M. Gasca and T. Sauer. “Polynomial interpolation in several variables”. In: *Advances in Computational Mathematics* 12.4 (2000), pp. 377–410 (cited on page 85).

[87] C. Geiger and C. Kanzow. *Theorie und Numerik restringierter Optimierungsaufgaben*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002 (cited on page 31).

[88] M. Gerdts. *Optimal Control of ODEs and DAEs*. Berlin, Boston: De Gruyter Oldenbourg, 2024 (cited on pages 11, 16, 24, 120–124).

[89] T. N. T. Goodman. “Shape preserving interpolation by curves”. In: *Algorithms For Approximation IV*. Ed. by J. Levesley, I. J. Anderson, and J. C. Maso. University of Huddersfield Proceedings Published, 2002, pp. 24–35 (cited on pages 81, 84, 85).

[90] W. J. Gordon. “Distributive lattices and the approximation of multivariate functions”. In: *Approximation with Special Emphasis on Splines*. Ed. by I. Schoenberg. Madison: University of Wisconsin Press, 1969 (cited on page 84).

[91] W. J. Gordon. *Free-form surface interpolation through curve networks*. Tech. rep. GMR-921. General Motors Research Laboratories, 1969 (cited on page 84).

[92] W. J. Gordon. “Spline-blended surface interpolation through curve networks”. In: *Journal of Mathematics and Mechanics* 18.10 (1969), pp. 931–952 (cited on page 84).

[93] W. J. Gordon. “Blending-function methods of bivariate and multivariate interpolation and approximation”. In: *SIAM Journal on Numerical Analysis* 8.1 (1971), pp. 158–177 (cited on pages 84, 85).

[94] J. A. Gregory. “Smooth interpolation without twist constraints”. In: *Computer Aided Geometric Design*. Ed. by R. E. Barnhill and R. F. Riesenfeld. Academic Press, 1974, pp. 71–87 (cited on page 84).

[95] A. Griewank and A. Walther. *Evaluating Derivatives*. 2nd. Philadelphia: Society for Industrial and Applied Mathematics, 2008 (cited on page 47).

[96] T. H. Gronwall. “Note on the Derivatives with Respect to a Parameter of the Solutions of a System of Differential Equations”. In: *Annals of Mathematics* 20.4 (1919), pp. 292–296 (cited on page 167).

[97] S. Gros, R. Quirynen, and M. Diehl. “Aircraft control based on fast non-linear MPC & multiple-shooting”. In: *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*. 2012, pp. 1142–1147 (cited on page 41).

[98] S. Gros, R. Quirynen, and M. Diehl. “An improved real-time economic NMPC scheme for Wind Turbine control using spline-interpolated aerodynamic coefficients”. In: *53rd IEEE Conference on Decision and Control*. 2014, pp. 935–940 (cited on page 41).

[99] S. Gros, M. Zanon, and M. Diehl. “Control of Airborne Wind Energy systems based on Nonlinear Model Predictive Control & Moving Horizon Estimation”. In: *2013 European Control Conference (ECC)*. 2013, pp. 1017–1022 (cited on page 41).

[100] M. Grötschel, S. O. Krumke, and J. Rambau. *Online Optimization of Large Scale Systems*. Berlin and Heidelberg: Springer, 2001 (cited on page 119).

[101] L. Grüne and J. Pannek. *Nonlinear Model Predictive Control*. Cham: Springer International Publishing, 2017 (cited on pages 2, 11–15, 19–21, 55, 73).

[102] J. Guanetti, Y. Kim, and F. Borrelli. “Control of connected and automated vehicles: State of the art and future challenges”. In: *Annual Reviews in Control* 45 (2018), pp. 18–40 (cited on page 144).

[103] J. Guddat, F. G. Vazquez, and H. T. Jongen. *Parametric Optimization: Singularities, Pathfollowing and Jumps*. Stuttgart: Teubner, 1990 (cited on pages 43, 115).

[104] J. Gutekunst, R. Scholz, A. Nurkanović, A. Mešanović, H. G. Bock, and E. Kostina. “Fast moving horizon estimation using multi-level iterations for microgrid control”. In: *at - Automatisierungstechnik* 68.12 (2020), pp. 1059–1076 (cited on pages 45, 106).

[105] L. Guzzella and A. Sciarretta. *Vehicle Propulsion Systems. Introduction to Modeling and Optimization*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013 (cited on pages 145–148).

[106] W. Hahn. *Stability of Motion*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1967 (cited on page 19).

[107] D. Haßkerl, A. Meyer, N. Azadfallah, S. Engell, A. Potschka, L. Wirsching, and H. G. Bock. “Study of the performance of the multi-level iteration scheme for dynamic online optimization for a fed-batch reactor example”. In: *2016 European Control Conference (ECC)*. 2016, pp. 459–464 (cited on page 51).

[108] W. Heß and J. W. Schmidt. “Positive quartic, monotone quintic C^2-spline interpolation in one and two dimensions”. In: *Journal of Computational and Applied Mathematics* 55.1 (1994), pp. 51–67 (cited on page 84).

[109] B. Houska, H. J. Ferreau, and M. Diehl. “ACADO toolkit—An open-source framework for automatic control and dynamic optimization”. In: *Optimal Control Applications and Methods* 32.3 (2011), pp. 298–312 (cited on pages 25, 144).

[110] B. Houska, H. J. Ferreau, and M. Diehl. “An auto-generated real-time iteration algorithm for non-linear MPC in the microsecond range”. In: *Automatica* 47.10 (2011), pp. 2279–2285 (cited on page 41).

[111] D. Hrovat, S. Di Cairano, H. Tseng, and I. Kolmanovsky. “The development of Model Predictive Control in automotive industry: A survey”. In: *2012 IEEE International Conference on Control Applications*. 2012, pp. 295–302 (cited on page 144).

[112] A. Huber and M. Gerdts. “A dynamic programming MPC approach for automatic driving along tracks and its realization with online steering controllers”. In: *IFAC-PapersOnLine* 50.1 (2017). This material is partly based upon work supported by the Air Force Office of Scientific Research, Air Force Materiel Command, USAF, under Award No, FA9550-14-11-0298., pp. 8686–8691 (cited on page 11).

[113] M. Z. Hussain and M. Sarfraz. “Positivity-preserving interpolation of positive data by rational cubics”. In: *Journal of Computational and Applied Mathematics* 218.2 (2008). The Proceedings of the Twelfth International Congress on Computational and Applied Mathematics, pp. 446–458 (cited on pages 81, 84).

[114] J. M. Hyman. “Accurate Monotonicity Preserving Cubic Interpolation”. In: *SIAM Journal on Scientific and Statistical Computing* 4.4 (1983), pp. 645–654 (cited on page 81).

[115] A. Ilzhöfer, B. Houska, and M. Diehl. “Nonlinear MPC of kites under varying wind conditions for a new class of large-scale wind power generators”. In: *International Journal of Robust and Nonlinear Control: IFAC-Affiliated Journal* 17.17 (2007), pp. 1590–1599 (cited on page 41).

[116] T. Kaczorek. “The Choice of the Forms of Lyapunov Functions for a Positive 2D Roesser Model”. In: *International Journal of Applied Mathematics and Computer Science* 17.4 (2008), pp. 471–475 (cited on page 71).

[117] C. Kirches, H. G. Bock, J. P. Schlöder, and S. Sager. “Block-structured quadratic programming for the direct multiple shooting method for optimal control”. In: *Optimization Methods and Software* 26.2 (2011), pp. 239–257 (cited on page 38).

[118] C. Kirches, H. G. Bock, J. P. Schlöder, and S. Sager. “Mixed-integer NMPC for predictive cruise control of heavy-duty trucks”. In: *2013 European Control Conference (ECC)*. 2013, pp. 4118–4123 (cited on pages 11, 144).

[119] C. Kirches, L. Wirsching, H. G. Bock, and J. P. Schlöder. “Efficient direct multiple shooting for nonlinear model predictive control on long horizons”. In: *Journal of Process Control* 22.3 (2012), pp. 540–550 (cited on page 45).

[120] C. Kirches, L. Wirsching, S. Sager, and H. G. Bock. “Efficient Numerics for Nonlinear Model Predictive Control”. In: *Recent Advances in Optimization and its Applications in Engineering*. Ed. by M. Diehl, F. Glineur, E. Jarlebring, and W. Michiels. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 339–357 (cited on pages 45, 119, 127).

[121] P. Kühl, M. Diehl, T. Kraus, J. P. Schlöder, and H. G. Bock. “A real-time algorithm for moving horizon state and parameter estimation”. In: *Computers & Chemical Engineering* 35.1 (2011), pp. 71–83 (cited on page 106).

[122] P. Kühl, H. J. Ferreau, J. Albersmeyer, C. Kirches, L. Wirsching, S. Sager, A. Potschka, G. Schulz, M. Diehl, D. B. Leineweber, and A. A. S. Schäfer. *MUSCOD-II Users’ Manual*. Users’ Manual. Heidelberg, Germany: Universität Heidelberg, 2016 (cited on page 25).

[123] F. Kuijt. “Convexity preserving interpolation: stationary nonlinear subdivision and splines”. PhD thesis. University of Twente, 1998 (cited on page 81).

[124] W. H. Kwon and S. Han. *Receding Horizon Control: Model Predictive Control for State Models*. London: Scholars Portal, 2005 (cited on page 11).

[125] J. E. Lavery. “Shape-preserving, multiscale interpolation by bi- and multivariate cubic L1 splines”. In: *Computer Aided Geometric Design* 18.4 (2001), pp. 321–343 (cited on pages 81, 85).

[126] J. H. Lee. “Model predictive control: Review of the three decades of development”. In: *International Journal of Control, Automation and Systems* 9.3 (2011), pp. 415–424 (cited on page 12).

[127] D. B. Leineweber. “Analyse und Restrukturierung eines Verfahrens zur direkten Lösung von Optimal-Steuerungsproblemen. The Theory of MUSCOD in a Nutshell”. Diploma thesis. Heidelberg University, 1995 (cited on page 38).

[128] D. B. Leineweber. *Efficient reduced SQP methods for the optimization of chemical processes described by large sparse DAE models*. Tech. rep. 613. Düsseldorf: VDI Verlag, 1999 (cited on pages 25, 38).

[129] 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 1: theoretical aspects”. In: *Computers & Chemical Engineering* 27.2 (2003), pp. 157–166 (cited on page 25).

[130] D. B. Leineweber, A. Schäfer, H. G. Bock, and J. P. Schlöder. “An efficient multiple shooting based reduced SQP strategy for large-scale dynamic process optimization: Part II: Software aspects and applications”. In: *Computers & Chemical Engineering* 27.2 (2003), pp. 167–174 (cited on page 25).

[131] F. L. Lewis, D. L. Vrabie, and V. L. Syrmos. *Optimal Control*. Wiley, 2012 (cited on page 11).

[132] D. Liao-McPherson, M. M. Nicotra, and I. Kolmanovsky. “Time-distributed optimization for real-time model predictive control: Stability, robustness, and constraint satisfaction”. In: *Automatica* 117 (2020), p. 108973 (cited on page 41).

[133] C. Lindscheid, D. Haßkerl, A. Meyer, A. Potschka, H. G. Bock, and S. Engell. “Parallelization of modes of the multi-level iteration scheme for nonlinear model-predictive control of an industrial process”. In: *2016 IEEE Conference on Control Applications (CCA)*. 2016, pp. 1506–1512 (cited on page 45).

[134] H. Maurer and D. Augustin. “Sensitivity Analysis and Real-Time Control of Parametric Optimal Control Problems Using Boundary Value Methods”. In: *Online Optimization of Large Scale Systems*. Ed. by M. Grötschel, S. O. Krumke, and J. Rambau. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 17–55 (cited on page 119).

[135] H. Maurer and H. J. Pesch. “Solution differentiability for parametric nonlinear control problems with control-state constraints”. In: *Journal of Optimization Theory and Applications* 86.2 (1995), pp. 285–309 (cited on page 119).

[136] E. G. Merino. “Real-time optimization for estimation and control: Application to waste heat recovery for heavy duty trucks”. PhD thesis. Heidelberg University, 2018 (cited on pages 11, 106, 144).

[137] E. G. Merino, J. P. Schloder, and C. Kirches. “A Nonlinear Model-Predictive Control Scheme for a Heavy Duty Truck’s Waste Heat Recovery System Featuring Moving Horizon Estimation”. In: *2018 Annual American Control Conference (ACC)*. 2018, pp. 6329–6334 (cited on pages 11, 106, 144).

[138] A. Meyer. “Numerical solution of optimal control problems with explicit and implicit switches”. PhD thesis. Heidelberg University, 2020 (cited on pages 16, 45, 185).

[139] A. Michel. “Stability: the common thread in the evolution of feedback control”. In: *IEEE Control Systems Magazine* 16.3 (1996), pp. 50–60 (cited on page 20).

[140] A. Mironchenko and F. Wirth. “Characterizations of Input-to-State Stability for Infinite-Dimensional Systems”. In: *IEEE Transactions on Automatic Control* 63.6 (2018), pp. 1692–1707 (cited on page 62).

[141] A. Musa, M. Pipicelli, M. Spano, F. Tufano, F. De Nola, G. Di Blasio, A. Gimelli, D. A. Misul, and G. Toscano. “A Review of Model Predictive Controls Applied to Advanced Driver-Assistance Systems”. In: *Energies* 14.23 (2021) (cited on page 144).

[142] G. M. Nielson. “Some piecewise polynomial alternatives to splines under tension”. In: *Computer Aided Geometric Design*. Ed. by R. E. Barnhill and R. F. Riesenfeld. Academic Press, 1974, pp. 209–235 (cited on page 84).

[143] J. Nocedal and S. J. Wright. *Numerical Optimization*. 2nd ed. Springer New York, 2006 (cited on pages 24, 31–33, 134).

[144] A. Nurkanović, S. Albrecht, and M. Diehl. “Multi-level Iterations for Economic Nonlinear Model Predictive Control”. In: *Recent Advances in Model Predictive Control: Theory, Algorithms, and Applications*. Ed. by T. Faulwasser, M. A. Müller, and K. Worthmann. Cham: Springer International Publishing, 2021, pp. 65–105 (cited on page 45).

[145] A. Nurkanović, A. Zanelli, S. Albrecht, and M. Diehl. “The Advanced Step Real Time Iteration for NMPC”. In: *2019 IEEE 58th Conference on Decision and Control (CDC)*. 2019, pp. 5298–5305 (cited on pages 40, 45, 141).

[146] A. Nurkanović, A. Zanelli, S. Albrecht, G. Frison, and M. Diehl. “Contraction Properties of the Advanced Step Real-Time Iteration for NMPC”. In: *IFAC-PapersOnLine* 53.2 (2020). 21st IFAC World Congress, pp. 7041–7048 (cited on page 40).

[147] P. J. Olver. “On Multivariate Interpolation”. In: *Studies in Applied Mathematics* 116.2 (2006), pp. 201–240 (cited on page 85).

[148] C. Pan, A. Huang, L. Chen, Y. Cai, L. Chen, J. Liang, and W. Zhou. “A review of the development trend of adaptive cruise control for ecological driving”. In: *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering* 236.9 (2022), pp. 1931–1948 (cited on pages 2, 144).

[149] S. D. Pendleton, H. Andersen, X. Du, X. Shen, M. Meghjani, Y. H. Eng, D. Rus, and M. H. Ang. “Perception, Planning, Control, and Coordination for Autonomous Vehicles”. In: *Machines* 5.1 (2017) (cited on page 144).

[150] K. J. Plitt. “Ein superlinear konvergentes Mehrzielverfahren zur direkten Berechnung beschränkter optimaler Steuerungen”. Diploma thesis. Rheinische Friedrich-Wilhelms-Universität Bonn, 1981 (cited on page 25).

[151] L. S. Pontryagin, V. G. Boltyanskii, R. V. Gamkrelidze, and E. F. Mishchenko. *The mathematical theory of optimal processes*. Vol. volume 4. Classics of Soviet mathematics. Boca Raton: CRC Press, Taylor & Francis Group, 1986 (cited on page 24).

[152] A. Potschka. “Handling path constraints in a direct multiple shooting method for optimal control problems”. Diploma thesis. Heidelberg University, 2006 (cited on page 29).

[153] A. Potschka. “A direct method for the numerical solution of optimization problems with time-periodic PDE constraints”. PhD thesis. Heidelberg University, 2011 (cited on page 25).

[154] A. Potschka, H. G. Bock, and J. P. Schlöder. “A minima tracking variant of semi-infinite programming for the treatment of path constraints within direct solution of optimal control problems”. In: *Optimization Methods and Software* 24.2 (2009), pp. 237–252 (cited on page 29).

[155] M. J. D. Powell. “A fast algorithm for nonlinearly constrained optimization calculations”. In: *Numerical Analysis*. Ed. by G. A. Watson. Berlin, Heidelberg: Springer Berlin Heidelberg, 1977, pp. 144–157 (cited on page 31).

[156] S. Qin and T. A. Badgwell. “A survey of industrial model predictive control technology”. In: *Control Engineering Practice* 11.7 (2003), pp. 733–764 (cited on page 11).

[157] A. V. Rao. “A Survey of Numerical Methods for Optimal Control”. In: *Advances in the Astronautical Sciences* 135 (2010), pp. 497–528 (cited on pages 11, 23).

[158] A. V. Rao. “Trajectory Optimization: A Survey”. In: *Optimization and Optimal Control in Automotive Systems*. Ed. by H. Waschl, I. Kolmanovsky, M. Steinbuch, and L. del Re. Cham: Springer International Publishing, 2014, pp. 3–21 (cited on page 23).

[159] J. B. Rawlings, D. Q. Mayne, and M. Diehl. *Model predictive control: Theory, computation, and design*. 2nd ed., 4th printing. Santa Barbara (California): Nob Hill Publishing, LLC, 2022 (cited on pages 11, 12, 15, 62).

[160] J. P. Raymond and H. Zidani. “Hamiltonian Pontryagin’s Principles for Control Problems Governed by Semilinear Parabolic Equations”. In: *Applied Mathematics and Optimization* 39.2 (1999), pp. 143–177 (cited on pages 4, 56).

[161] A. Rezaei and J. B. Burl. “Prediction of Vehicle Velocity for Model Predictive Control”. In: *IFAC-PapersOnLine* 48.15 (2015). 4th IFAC Workshop on Engine and Powertrain Control, Simulation and Modeling E-COSM 2015, pp. 257–262 (cited on page 151).

[162] M. Rick, J. Clemens, L. Sommer, A. Folkers, K. Schill, and C. Büskens. “Autonomous Driving Based on Nonlinear Model Predictive Control and Multi-Sensor Fusion”. In: *IFAC-PapersOnLine* 52.8 (2019). 10th IFAC Symposium on Intelligent Autonomous Vehicles IAV 2019, pp. 182–187 (cited on page 11).

[163] S. M. Robinson. “Perturbed Kuhn-Tucker points and rates of convergence for a class of nonlinear-programming algorithms”. In: *Mathematical Programming* 7.1 (1974), pp. 1–16 (cited on page 118).

[164] J. A. Rossiter. *Model-based predictive control: A practical approach*. Control series. Boca Raton: CRC Press, 2004 (cited on page 11).

[165] A. A. S. Schäfer. “Efficient reduced Newton-type methods for solution of large-scale structured optimization problems with application to biological and chemical processes”. PhD thesis. Heidelberg University, 2005 (cited on page 25).

[166] J. P. Schlöder. *Numerische Methoden zur Behandlung hochdimensionaler Aufgaben der Parameteridentifizierung*. Ed. by E. Brieskorn. Bonner Mathematische Schriften. 1988 (cited on page 25).

[167] J. W. Schmidt and W. Heß. “S-convex, monotone, and positive interpolation with rational bicubic splines of C^2-continuity”. In: *BIT* 33.3 (1993), pp. 496–511 (cited on page 84).

[168] R. Schmied, H. Waschl, R. Quirynen, M. Diehl, and L. del Re. “Nonlinear MPC for Emission Efficient Cooperative Adaptive Cruise Control”. In: *IFAC-PapersOnLine* 48.23 (2015). 5th IFAC Conference on Nonlinear Model Predictive Control NMPC 2015, pp. 160–165 (cited on page 144).

[169] W. Schnabel and D. Lohse. *Grundlagen der Straßenverkehrstechnik und der Verkehrsplanung: Band 1-Straßenverkehrstechnik*. 3., vollst. überarb. Aufl. Studium. Berlin et al.: Verl. für Bauwesen, Beuth, and Kirschbaum, 2011 (cited on page 77).

[170] R. Scholz. “Stabiles Condensing für Optimale Steuerung”. Master thesis. Heidelberg University, 2016 (cited on page 38).

[171] R. Scholz. “Model-based optimal feedback control For microgrids”. PhD thesis. Heidelberg University, 2022 (cited on pages 41, 43–45, 48, 49, 51).

[172] R. Scholz, A. Nurkanović, A. Mešanović, J. Gutekunst, A. Potschka, H. G. Bock, and E. Kostina. “Model-based optimal feedback control for microgrids with multi-level iterations”. In: *Operations Research Proceedings 2019*. Springer, 2020, pp. 73–79 (cited on pages 43–45, 50).

[173] R. Scholz, A. Nurkanović, A. Mešanović, J. Gutekunst, A. Potschka, H. G. Bock, and E. Kostina. “Multi-level iterations for microgrid control with automatic level choice”. In: *Scientific Computing in Electrical Engineering*. Springer, 2021, pp. 293–301 (cited on pages 43–45).

[174] D. G. Schweikert. “An interpolation curve using a spline in tension”. In: *J. Math. Phys* 45.3 (1966), pp. 312–317 (cited on page 84).

[175] M. Schwenzer, M. Ay, T. Bergs, and D. Abel. “Review on model predictive control: an engineering perspective”. In: *The International Journal of Advanced Manufacturing Technology* 117.5 (2021), pp. 1327–1349 (cited on pages 2, 11).

[176] E. Siampis, E. Velenis, S. Gariuolo, and S. Longo. “A Real-Time Nonlinear Model Predictive Control Strategy for Stabilization of an Electric Vehicle at the Limits of Handling”. In: *IEEE Transactions on Control Systems Technology* 26.6 (2018), pp. 1982–1994 (cited on page 144).

[177] E. D. Sontag. “Smooth stabilization implies coprime factorization”. In: *IEEE Transactions on Automatic Control* 34.4 (1989), pp. 435–443 (cited on page 19).

[178] E. D. Sontag. *Mathematical Control Theory*. Ed. by J. E. Marsden, L. Sirovich, M. Golubitsky, and W. Jäger. Vol. 6. New York, NY: Springer New York, 1998 (cited on pages 14, 15).

[179] H. Späth. *Spline algorithms for curves and surfaces*. Winnipeg: Utilitas Mathematica Pub., 1974 (cited on page 84).

[180] E. Süli and D. F. Mayers. *An Introduction to Numerical Analysis*. Cambridge University Press, 2003 (cited on page 77).

[181] K. L. Teo, B. Li, C. Yu, and V. Rehbock. *Applied and Computational Optimal Control*. Vol. 171. Cham: Springer International Publishing, 2021 (cited on page 23).

[182] L. N. Trefethen and D. Bau III. *Numerical Linear Algebra*. The book has been reissued as [183]. Philadelphia, PA: Society for Industrial and Applied Mathematics, 1997 (cited on page 127).

[183] L. N. Trefethen and D. Bau III. *Numerical Linear Algebra, Twenty-fifth Anniversary Edition*. Philadelphia, PA: Society for Industrial and Applied Mathematics, 2022 (cited on page 210).

[184] F. Tröltzsch. *Optimale Steuerung partieller Differentialgleichungen: Theorie, Verfahren und Anwendungen*. 2., überarb. Aufl. Studium. Wiesbaden: Vieweg + Teubner, 2009 (cited on pages 11, 56–59).

[185] L. H. Tsoukalas, R. E. Uhrig, and L. A. Zadeh. *Fuzzy and neural approaches in engineering*. Adaptive and Learning Systems for Signal Processing, Communications, and Control. New York: A Wiley-Interscience Publication, John Wiley & Sons, 1997 (cited on page 11).

[186] V. Turri, B. Besselink, and K. H. Johansson. "Cooperative Look-Ahead Control for Fuel-Efficient and Safe Heavy-Duty Vehicle Platooning". In: *IEEE Transactions on Control Systems Technology* 25.1 (2017), pp. 12–28 (cited on page 144).

[187] V. Turri, Y. Kim, J. Guanetti, K. H. Johansson, and F. Borrelli. "A model predictive controller for non-cooperative eco-platooning". In: *2017 American Control Conference (ACC)*. 2017, pp. 2309–2314 (cited on page 144).

[188] U.S. Energy Information Administration (EIA). *Energy Conversion Calculators*. Accessed: 2025-01-16. 2025 (cited on page 164).

[189] M. Ulbrich. "Optimization Methods in Banach Spaces". In: *Optimization with PDE Constraints*. Dordrecht: Springer Netherlands, 2009, pp. 97–156 (cited on pages 24, 63).

[190] M. Ulbrich and S. Ulbrich. *Nichtlineare Optimierung*. Basel: Springer Basel, 2012 (cited on page 31).

[191] Umweltbundesamt. *Energieverbrauch nach Energieträgern und Sektoren*. Accessed: 2025-01-13. 2024 (cited on pages 1, 164).

[192] United Nations Framework Convention on Climate Change (UNFCCC). *Paris Agreement*. Adopted December 12, 2015. Accessed: 2025-01-10. 2015 (cited on page 1).

[193] M. Vajedi and N. L. Azad. "Ecological Adaptive Cruise Controller for Plug-In Hybrid Electric Vehicles Using Nonlinear Model Predictive Control". In: *IEEE Transactions on Intelligent Transportation Systems* 17.1 (2016), pp. 113–122 (cited on page 144).

[194] R. Verschueren, G. Frison, D. Kouzoupis, J. Frey, N. van Duijkeren, A. Zanelli, B. Novoselnik, T. Albin, R. Quirynen, and M. Diehl. "acados – a modular open-source framework for fast embedded optimal control". In: *Mathematical Programming Computation* (2021) (cited on page 25).

[195] R. Verschueren, M. Zanon, R. Quirynen, and M. Diehl. "Time-optimal race car driving using an online exact hessian based nonlinear MPC algorithm". In: *2016 European Control Conference (ECC)*. 2016, pp. 141–147 (cited on page 43).

[196] R. Vinter. *Optimal Control*. Boston: Birkhäuser Boston, 2010 (cited on page 11).

[197] L. Wang. *Model Predictive Control System Design and Implementation Using MATLAB®*. London: Springer London, 2009 (cited on page 11).

[198] A. Weißmann, D. Görges, and X. Lin. "Energy-Optimal Adaptive Cruise Control based on Model Predictive Control". In: *IFAC-PapersOnLine* 50.1 (2017). 20th IFAC World Congress, pp. 12563–12568 (cited on page 144).

[199] A. Winkler, J. Frey, T. Fahrbach, G. Frison, R. Scheer, M. Diehl, and J. Andert. "Embedded Real-Time Nonlinear Model Predictive Control for the Thermal Torque Derating of an Electric Vehicle". In: *IFAC-PapersOnLine* 54.6 (2021). 7th IFAC Conference on Nonlinear Model Predictive Control NMPC 2021, pp. 359–364 (cited on page 11).

[200] L. Wirsching. "Multi-level iteration schemes with adaptive level choice for nonlinear model predictive control". PhD thesis. Heidelberg University, 2018 (cited on pages 3, 34, 37, 42–51).

[201] L. Wirsching, J. Albersmeyer, P. Kühl, M. Diehl, and H. G. Bock. "An Adjoint-based Numerical Method for Fast Nonlinear Model Predictive Control". In: *IFAC Proceedings Volumes* 41.2 (2008). 17th IFAC World Congress, pp. 1934–1939 (cited on page 45).

[202] L. Wirsching, H. G. Bock, and M. Diehl. "Fast NMPC of a chain of masses connected by springs". In: *2006 IEEE Conference on Computer Aided Control System Design, 2006 IEEE International Conference on Control Applications, 2006 IEEE International Symposium on Intelligent Control*. 2006, pp. 591–596 (cited on page 45).

[203] L. Wirsching, H. J. Ferreau, H. G. Bock, and M. Diehl. "An online active set strategy for fast adjoint based nonlinear model predictive control". In: *IFAC Proceedings Volumes* 40.12 (2007). 7th IFAC Symposium on Nonlinear Control Systems, pp. 234–239 (cited on pages 43, 45).

[204] I. J. Wolf and W. Marquardt. “Fast NMPC schemes for regulatory and economic NMPC – A review”. In: *Journal of Process Control* 44 (2016), pp. 162–183 (cited on page 119).

[205] A. J. Worsey. “A modified C^2 Coons’ patch”. In: *Computer Aided Geometric Design* 1.4 (1984), pp. 357–360 (cited on page 84).

[206] K. Yu, J. Yang, and D. Yamaguchi. “Model predictive control for hybrid vehicle ecological driving using traffic signal and road slope information”. In: *Control Theory and Technology* 13.1 (2015), pp. 17–28 (cited on page 11).

[207] A. Zanelli, A. Domahidi, J. Jerez, and M. Morari. “FORCES NLP: an efficient implementation of interior-point methods for multistage nonlinear nonconvex programs”. In: *International Journal of Control* 93.1 (2017), pp. 13–29 (cited on page 144).

[208] A. Zanelli, Q. Tran-Dinh, and M. Diehl. “A Lyapunov function for the combined system-optimizer dynamics in inexact model predictive control”. In: *Automatica* 134 (2021), p. 109901 (cited on pages xi, 44, 55, 62, 63, 71, 75).

[209] M. Zanon, J. V. Frasch, M. Vukov, S. Sager, and M. Diehl. “Model Predictive Control of Autonomous Vehicles”. In: *Optimization and Optimal Control in Automotive Systems*. Ed. by H. Waschl, I. Kolmanovsky, M. Steinbuch, and L. del Re. Cham: Springer International Publishing, 2014, pp. 41–57 (cited on pages 11, 41, 144).

[210] M. Zanon, G. Horn, S. Gros, and M. Diehl. “Control of Dual-Airfoil Airborne Wind Energy systems based on nonlinear MPC and MHE”. In: *2014 European Control Conference (ECC)*. 2014, pp. 1801–1806 (cited on page 41).

[211] V. M. Zavala and L. T. Biegler. “The advanced-step NMPC controller: Optimality, stability and robustness”. In: *Automatica* 45.1 (2009), pp. 86–93 (cited on pages 40, 141).

[212] Y. Zhu. “C^2 positivity-preserving rational interpolation splines in one and two dimensions”. In: *Applied Mathematics and Computation* 316 (2018), pp. 186–204 (cited on pages 81, 84).

Acronyms

A
ACC Adaptive Cruise Control. vii, xv, 2, 11, 53, 77, 117, 143, 144, 146, 148, 150, 152, 154, 156, 158, 160, 163, 213
AD Automatic Differentiation. 47
ADAS Advanced Driver-Assistance System. 2, 143, 144, 151

B
BMBF Bundesministerium für Bildung und Forschung. ix, 1

C
CAGD Computer Aided Geometric Design. 83–85
CC Cruise Control. vii, xv, 2, 11, 53, 77, 117, 143, 144, 146, 148, 150, 152, 154, 156, 158, 160, 163, 213
CPU Central Processing Unit. 31, 117

D
DAE Differential Algebraic Equation. 14, 25, 124
DMS Direct Multiple Shooting. v, vii, xiii–xv, 3–9, 23, 25–31, 34–37, 40, 53, 105–112, 114, 116, 119, 120, 125–127, 130, 131, 139, 147, 150, 151, 163–166, 185, 186, 188, 190, 192, 215, 217
DP Dynamic Programming. 24, 25

E
EACC Ecological Adaptive Cruise Control. v, vii, ix, xv, 2–8, 11, 14, 53, 77, 106, 117, 124, 127, 135, 136, 138–141, 143, 144, 146, 148–150, 152, 154, 156, 158–160, 163–165, 215

H
HJB Hamilton-Jacobi-Bellman. 24, 25
HVAC heating, ventilation, air conditioning. 145, 147

I
IFT Implicit Function Theorem. 118, 123
IND Internal Numerical Differentiation. 47
IVE Initial Value Embedding. xiii, 41–43, 112
IVP Initial Value Problem. 14, 15, 21, 27, 28, 42, 48, 56, 57, 63, 106, 109, 115, 145

K
KKT Karush-Kuhn-Tucker. 31, 32, 47, 48, 122, 125, 129, 130

L
LICQ Linear Independence Constraint Qualification. 32, 121, 122, 124
LMPC Linear Model Predictive Controller. 49
LUT Lookup Table. v, vii, 3–5, 7, 8, 77, 85, 143, 147–150, 152, 153, 156, 157, 160, 163
LUTs Lookup Tables. 149

M
MHE Moving Horizon Estimation. 45, 106
MLI Multi-Level Iterations. v, vii, ix, xiii–xv, 3–9, 23, 43–47, 49–53, 105–108, 110–119, 125, 127, 131, 132, 134, 135, 138, 139, 141, 144, 145, 147, 150–160, 163–165, 215, 217
MPBVP Multipoint Boundary Value Problem. 24, 25
MPC Model Predictive Control. 2, 11, 12, 43, 144
MSO Mathematical Modeling, Simulation and Optimization. 1, 2

N

List of Figures

List of Tables

List of Algorithms