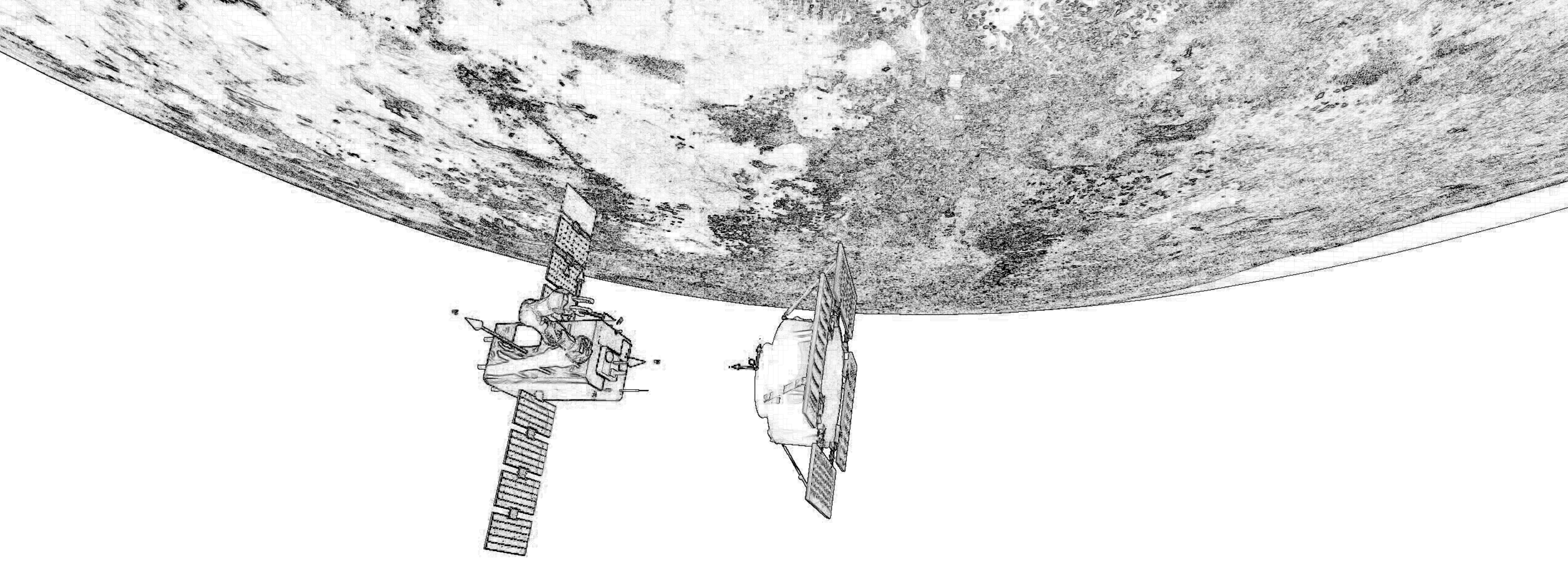


Optimal Control Parametrization Strategies for MPC with Application to On-Orbit Servicing

Fabio Schneider

Master Thesis

2025

Supervisor:	Peter Kötting	DLR
	Renato Loureiro	iFR
Examiner:	Dr. Torbjørn Cunis	iFR

University of Stuttgart
Institute of Flight Dynamics and Control

German Aerospace Center
Institute of Robotics and Mechatronics

Erklärung

Hiermit versichere ich, dass ich diese Master Thesis selbständig mit Unterstützung der Betreuer angefertigt und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Die Arbeit oder wesentliche Bestandteile davon sind weder an dieser noch an einer anderen Bildungseinrichtung bereits zur Erlangung eines Abschlusses eingereicht worden.

Ich erkläre weiterhin, bei der Erstellung der Arbeit die einschlägigen Bestimmungen zum Urheberschutz fremder Beiträge entsprechend den Regeln guter wissenschaftlicher Praxis[1] eingehalten zu haben. Soweit meine Arbeit fremde Beiträge (z.B. in der Form von Bilder, Zeichnungen, Textpassagen etc.) enthält, habe ich diese Beiträge als solche gekennzeichnet (Zitat, Quellenangabe) und eventuell erforderlich gewordene Zustimmungen der Urheber zur Nutzung dieser Beiträge in meiner Arbeit eingeholt. Mir ist bekannt, dass ich im Fall einer schuldhaften Verletzung dieser Pflichten die daraus entstehenden Konsequenzen zu tragen habe.

Ort, Datum	Fabio Schneider

Betreuer: Dr. Torbjørn Cunis

[1]Nachzulesen in den DFG-Empfehlungen zur „Sicherung guter wissenschaftlicher Praxis“ bzw. in der Satzung der Universität Stuttgart zur “Sicherung der Integrität wissenschaftlicher Praxis und zum Umgang mit Fehlverhalten in der Wissenschaft“

Acknowledgements

I would like to thank my supervisors, Peter Kötting and Renato Loureiro, for their guidance, feedback, and continuous support throughout the duration of this thesis. Their regular and insightful discussions greatly contributed to the progress of this work and helped shape the research direction as a whole.

My thanks also extend to Dr. Cunis for always having an open ear and for providing valuable assistance with theoretical questions.

Finally, I would like to acknowledge all family and friends who supported me during this process, whether through constructive discussions or encouragement along the way.

Kurzfassung

Ziel dieser Thesis ist es, die performanteste Optimalsteuerungsmethode für Model Predictive Control (MPC) Formulierungen zu ermitteln. Zunächst werden verschiedene explizite und implizite Methoden der Optimalsteuerung anhand eines repräsentativen Beispiels für Mehrkörpersysteme untersucht. Der Fokus liegt dabei auf der Echtzeitfähigkeit der Methoden, ihrer Regelgüte und dem Verhalten im offenen und geschlossenen Regelkreis. Die Optimalsteuerungsmethoden werden zusammen mit den verschiedenen nichtlinearen Solvern FATROP, IPOPT und SQP bewertet. Als effektivsten Ansatz erweist sich die Kombination von FATROP entweder mit einer Multiple-Shooting (MS) Methode oder einer angepassten lokalen Legendre-Gauss-Radau (LGR-LOC*) Pseudospektralmethode.

Anschließend werden MS und LGR-LOC* verglichen und für den Tracking- und Escape-Fall im On-Orbit Servicing (OOS) bewertet. Die MS Methode erweist sich als die rechnerisch effizienteste Lösung, während die LGR-LOC* Methode eine geringfügig bessere Regelgüte erzielt.

Im letzten Kapitel wird ein universelles Bayes'sches Optimierungsverfahren (BO) zur Identifizierung der performantesten Optimalsteuerungsmethode für MPC vorgeschlagen. Besonderheit der entwickelten LGR-LOC* Methode ist das Beinhalten expliziter MS Methoden und impliziter Kollokationsmethoden geringer und hoher Ordnung in einer Formulierung. Dies nutzt der BO-Algorithmus, um die optimale Parametrisierung zu bestimmen, wodurch die Notwendigkeit entfällt, jede Methode separat zu implementieren und zu evaluieren.

Mit diesem Ansatz wird die Rechenzeit der MPC für OOS um 45 % reduziert, während gleichzeitig die Leistung des geschlossenen Regelkreises um 8-43 %, je nach Manöver, verbessert wird.

Abstract

This thesis aims to identify the most performant optimal control parametrization for Model Predictive Control (MPC) setups. The work begins with a systematic investigation of various explicit and implicit optimal control methods based on a generic multibody benchmark example. A detailed validation is done on the real-time performance capabilities of the methods, their closed-loop performance, and open-loop versus closed-loop behavior. The optimal control strategies are evaluated along the nonlinear program solvers FATROP, IPOPT and SQP. Among the approaches considered, combining FATROP with a MS method or an adapted local LGR-LOC* pseudospectral method proves in simulations to be the most effective.

Further, MS and LGR-LOC* are compared and evaluated in the context of a tracking and escape maneuver for On-Orbit Servicing (OOS). The MS method is identified to be the computationally most efficient solution, while the LGR-LOC* method achieved minimally improved closed-loop performance.

The final section of the thesis proposes a universal Bayesian Optimization (BO) framework for identifying the most suitable optimal control method and parametrization for a respective MPC control problem. The developed LGR-LOC* incorporates explicit multiple shooting and implicit low- and high-order collocation methods in one formulation. This leverages the BO algorithm to determine the optimal parametrization, eliminating the need to implement each control method individually.

This approach decreases the computational cost of MPC for OOS by 45 %, while simultaneously improving the closed-loop performance by 8-43 %, depending on the maneuver.

Contents

Nomenclature

Abbreviations

ADR	Active Debris Removal
BO	Bayesian Optimization
CL	Closed-loop
DLR	German Aerospace Center (Deutsches Zentrum für Luft- und Raumfahrt)
DOF	Degrees of Freedom
FoV	Field-of-View
HS	Hermite-Simpson collocation method
IVP	Initial Value Problem
LGL	Legendre-Gauss-Lobatto pseudospectral method
LGL-LOC	Local Legendre-Gauss-Lobatto pseudospectral method
LGR	Legendre-Gauss-Radau pseudospectral method
LGR-LOC	Local Legendre-Gauss-Radau pseudospectral method
LGR-LOC*	Local Legendre-Gauss-Radau method optimized for FATROP
MPC	Model Predictive Control
MS	Multiple shooting method
NLP	Nonlinear Program
NMPC	Nonlinear Model Predictive Control
OCP	Optimal Control Problem
ODE	Ordinary Differential Equation
OOS	On-Orbit Servicing
PS	Pseudospectral method
RK	Runge-Kutta integration scheme
SS	Single shooting method
TC	Trapezoidal collocation method
TPE	Tree-Structured Parzen Estimator

Latin symbols

$\boldsymbol{f}$	Forces vector
$\boldsymbol{g}$	Gravity vector in multi-body systems
$\boldsymbol{p}$	Position vector
$\boldsymbol{q}$	Quaternion vector
$\boldsymbol{u}$	Control input vector
$\boldsymbol{v}$	Translational velocity vector
$\boldsymbol{x}$	State vector
Δt	Step-size or controller frequency in simulation
$\mathbf{C}$	Coriolis-Centrifugal matrix
$\mathbf{M}$	Mass matrix
$\mathbf{P}$	Penality matrix for the terminal region
$\mathbf{Q}$	Weighting matrix for the states
$\mathbf{R}$	Weighting matrix for the control inputs
a	First point of an interval.
b	Last point of an interval.
e	Cumulated error to reference
h	Step-size between two nodes
J	Cost functional
L_i^N	i-th Lagrange polynomial with total N nodes
N	Prediction Horizon (steps)
N_{int}	Number of intervals for local methods
n_{u}	Dimension of the control input vector $u \in \mathbb{R}^{n_{\text{u}}}$
n_{x}	Dimension of the state vector $x \in \mathbb{R}^{n_{\text{x}}}$
N_l	Number of collocation nodes per interval
T	Prediction Horizon (time)
t	Quadrature point in the domain $t \in [a, b]$
t_{adp}	Interval borders for adaptive grid
t_{comp}	Computation time
V_{f}	Terminal cost term
w	Weights in quadrature

Greek symbols

α	Scaling parameter for the terminal region
$\boldsymbol{\nu}$	Velocity vector containing the angular ω and linear velocities v
$\boldsymbol{\omega}$	Angular velocities in each axis
$\boldsymbol{\tau}$	Momentum
$\boldsymbol{\theta}$	Angles of an n-joint inverted pendulum
$\mathcal{L}$	Lagrange term in cost functional (Stage costs)
$\mathcal{T}$	Kinetic energy term in Lagrangian formulation
$\mathcal{U}$	Potential energy term in Lagrangian formulation
$\mathcal{X}$	Terminal Region
$\tilde{\mathcal{M}}$	Set of all methods used in the survey excluding shooting methods
$\mathcal{M}$	Set of all methods used in the survey
Π	Closed-loop cost
τ	Quadrature point in the domain $\tau \in [-1, 1]$
Θ	Explicit numerical scheme for integration of IVPs

Indices

$(\cdot)^l$	Specifies the interval $l = 1, \dots, N_{\text{int}}$ in a local pseudospectral method
$(\cdot)_{\text{CL}}$	Closed-loop
$(\cdot)_{\text{f}}$	Final point in an optimal control problem
$(\cdot)_{\text{H}}$	Hybrid maneuver simulation in On-Orbit Servicing
$(\cdot)_{\text{OL}}$	Open-loop
$(\cdot)_{\text{ref}}$	Reference solution
$(\cdot)_{\text{T}}$	Tracking maneuver simulation in On-Orbit Servicing

Scalars, vectors and matrices

$\boldsymbol{v}$	Vector, bold in italics
$\mathbf{M}$	Matrix, bold, Latin font with capital letters
s	Scalar, not bold

1 Introduction

"*Satellite collisions would produce orbiting fragments, each of which would increase the probability of further collisions, leading to the growth of a belt of debris around earth.*" — Kessler, 1978 [29]

This is a quote by Donald J. Kessler in 1978 identifying the Kessler Syndrome for the first time. Over 40 years later, this problem is undoubtedly an increasing threat for human's access to space [39]. The greater the population of spacecraft orbiting the earth, the greater the impact on the space environment. Methods to prevent uncooperative spacecrafts that avoid potential collisions have to be established.

Reducing the number of launches while oppositely having a growing space economy with increased interest in space applications is globally infeasible. Therefore, approaches that reliably extend the lifetime of satellites until performing a deorbit maneuver are required to prevent further debris. This leads to a reduced number of non-maneuverable objects in space that can cause collisions.

For already existing non-cooperative spacecraft, one promising solution to remove space debris is the Active Debris Removal (ADR) [33]. The idea is to capture passive satellites with a servicing spacecraft and then perform a controlled deorbit maneuver. This procedure reduces the number of collision-causing debris in orbit.

More generally, ADR is a special case of On-Orbit Servicing (OOS). It describes the term of performing various types of servicing tasks in orbit, such as inspection, refueling or repair with a robotic manipulator mounted on the spacecraft [33]. The key functionality for the OOS to be successful is the capability to grasp the target satellite and stabilize it. Tailored technology for this capturing approach has to be developed.

One crucial part of this technology development is to find a suitable controller strategy. More precisely, control inputs for the control of the spacecraft's servicer (position and attitude) as well as the robotic manipulator are sought. This is particularly challenging due to the space environment, real-time performance requirements, autonomy and safety requirements, and the complexity of the dynamics of a free-flying spacecraft base with a coupled robotic manipulator. One potential controller strategy is Model Predictive Control (MPC) that is optimized for real-time performance within this thesis.

1.1. On-Orbit-Servicing at the DLR

At the Institute of Robotics and Mechatronics at the German Aerospace Center (Deutsches Zentrum für Luft- und Raumfahrt) (DLR) in Oberpfaffenhofen, technology for the purpose of OOS is being developed. The idea is to approach a target satellite with a servicing satellite (Servicer or Chaser). In the most challenging case, the target is non-cooperative, meaning that it has no active communication or attitude and orbit control system. Consequently, no navigation data can be shared, and the target may be tumbling. This requires detumbling before performing the actual servicing task. From a technological point of view, this is the most complex scenario.

Figure 1.1 shows the hardware simulation of an OOS-scenario. The left satellite represents the Chaser with the 7 DOF robotic manipulator mounted while the right satellite represents a non-cooperative target satellite. To test and validate control and pose-estimations algorithms on earth, the two satellite are mounted on robotic arms to simulate relative motion.

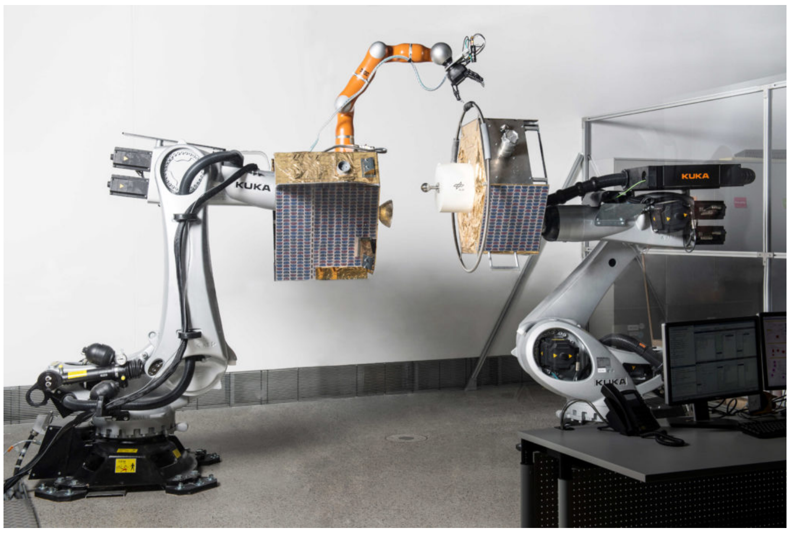


Figure 1.1.: OOS simulator at the DLR Institute of Robotics and Mechatronics [21].

Combined or coordinated control describes the terminology of simultaneously controlling the spacecrafts base position and attitude as well as the pose of the robotic manipulator. This approach involves a high-dimensional state and control input. Assuming a 7 Degrees of Freedom (DOF) robotic manipulator as a free-flying robot on a spacecraft based leads to $\boldsymbol{x} \in \mathbb{R}^{27}$ and $\boldsymbol{u} \in \mathbb{R}^{13}$.

1.2. Motivation

One approach to face the challenge of controller development tailored for OOS is Model Predictive Control (MPC). It is a model based control strategy, that optimizes, based on the incorporated nonlinear dynamics, for the optimal control inputs to minimize a defined metric [47]. The advantage to conventional control strategies is the capability to deal with nonlinear models and to guarantee constraints. In the case of OOS, such constraints can be actuator limits, the Field-of-View (FoV) or the collision avoidance with obstacles [32]. On the negative side, MPC approaches rely on solving an Optimal Control Problem (OCP) each control iteration. This has to be computationally tractable within the real-time requirements of the system. Consequently, the computation time of the optimization problem limits the application of MPC.

Most MPC formulations use direct optimal control formulations to transcribe the continuous problem into a nonlinear optimization problem [47]. The method and discretization strategy impact the computation time of MPC drastically. Therefore, the question arises, which optimal control method fits the best to the MPC formulation? General recommendations are restricted due to the dependency on the system dynamics, the incorporated constraints, the cost function design and real-time requirements.

This thesis discusses general properties of various optimal control methods and provides a universal framework to find a tailored optimal control method for the MPC. With this, the application of MPC in complex, high-dimensional problems is facilitated since a low computation time is a pre-condition for its utilization. The capability of the framework is used to enhance the performance of the MPC for OOS. However, it is expected that the strategy can be applied to various fields of applications that involve MPC.

1.3. Literature Review

Solving an OCP is a substantial part of MPC formulations. For solving a continuous optimal control problem indirect and direct methods exist.

Indirect methods use the calculus of variations to derive optimality conditions resulting in a two-point boundary value problem [30, 46]. This is also known as solving Pontryagin's Minimum principle [10]. Due to the analytical derivation of indirect methods the solution is always exact, although it suffers from a reduced region of convergence [7]. A good initial guess for the adjoint variables is required, which can be hard in practice due to sensitive behavior when varying the adjoint variables [10].

Direct methods discretize the continuous OCP into a Nonlinear Program (NLP) that can be solved with nonlinear optimization solvers. Instead of deriving necessary conditions that requires the differentiation of the Hamiltonian of the system, a sequence of control inputs is sought that minimizes a given cost functional [8]. Direct methods rely on a discretization of the continuous-time OCP, yielding an approximate solution. This discretization-based approach typically enhances by making the method less sensitive to poor initial guesses [8]. Hence, the preferable choice for MPC applications are direct methods [23].

1.3.1. Overview of direct methods

Direct methods use a discretization to find a suitable solution. Different methods to solve the Nonlinear Program (NLP) exist [46]. As shown in Figure 1.2, one separates between control parameterization and state & control parameterization.

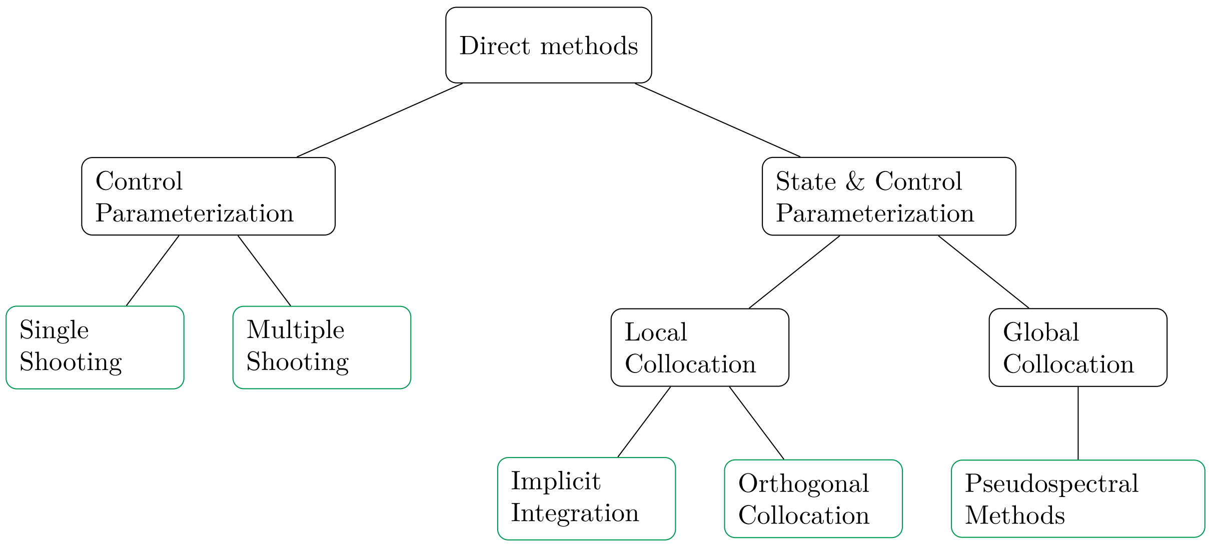


Figure 1.2.: Overview of direct methods to solve OCPs (adapted from [46]).

Specific methods for control parameterization are shooting methods. Even though multiple shooting involves a discretization of the state for the multiple shooting intervals, the discretization of the intervals does not affect the accuracy and is therefore categorized as only control input parameterization.

State and control parameterization can be achieved using collocation methods which are subdivided into local and global collocation. For local collocation, the finite problem is separated into sub-intervals on which the collocation is solved. The solutions of each subinterval are then merged by introducing continuity constraints. Examples for implicit integration are the Trapezoidal collocation and the Hermite-Simpson collocation [28].

Global collocation methods are commonly referred to as pseudospectral methods. The fundamental concept behind these methods is the approximation of the state and control input trajectories using a global function. While the terms “pseudospectral” and "orthogonal methods" are often used interchangeably in the literature, they describe the same underlying technique [46]. In contrast, local orthogonal collocation involves the application of pseudospectral methods on multiple sub-intervals, thereby partitioning the problem into several segments.

1.3.2. Existing surveys comparing optimal control methods

Table 1.1 gives a comparative summary on existing surveys of optimal control methods. Additionally, it is indicated which direct methods, namely shooting, uniform collocation or pseudospectral collocation methods are included in the comparison. Apart from the content itself, it is also stated if the survey includes the application in an MPC context.

Table 1.1.: Existing surveys that compare different direct methods for discretizing OCPs.

Source		**Shooting**	**Uniform colloc.**	**PS colloc.**	**MPC setup**	**Content**
Betts	[7]	×	×			Comparison in the formulation of indirect and direct OCPs.
Rao	[46]	×	×	×		Thorough comparison of direct methods for optimal control.
Kelly	[28]	(×)	×	(×)		Tutorial on how to formulate equidistant collocation methods. Shooting and pseudospectral methods briefly mentioned.
Diehl et al.	[13]	×	×		×	Comparison of single and multiple shooting with collocation for embedded NMPC applications. Focus on shooting methods.
Sanchez et al.	[49]	×		×	×	Comparison of shooting and pseudospectral methods on two examples.
Garg et al.	[19]			×		Overview of LG, LGR and LGL methods.
Fahroo et al.	[16]			×		Overview in their mathematical properties.
Huntington et al.	[27]			×		Comparison of global and local pseudospectral methods.

One of the earliest surveys in optimal control addressing various discretization approaches is published in [7]. A more recent and comprehensive overview, incorporating advanced pseudospectral collocation techniques, is presented in [46]. Both works provide a comparative discussion of the advantages and disadvantages of different methods and include remarks on computational effort. In [28], a detailed derivation of the formulation of equidistant collocation optimal control problems is presented. The work primarily focuses on the formulation itself rather than on a comparative analysis that also includes pseudospectral or shooting methods.

None of the aforementioned studies provide concrete numerical examples contrasting the approaches, particularly with respect to computational time. Moreover, no specific recommendations are given regarding their applicability within a MPC framework.

In [13], the focus lies on the applicability of direct methods for MPC. Nevertheless, given its publication date in 2006, the study does not consider more recent solver developments or advanced methods such as pseudospectral collocation. A subsequent comparison by [49] includes these methods alongside shooting techniques within an MPC framework, evaluated on two benchmark examples. Instead of comparing the methods towards accuracy, the methods are evaluated assuming an identical number of discretization nodes. This might bias the comparison since different methods require a different number of nodes for achieving similar accuracy.

The works presented in [16, 19, 27] primarily investigate pseudospectral methods tailored for offline trajectory optimization problems. Consequently, a comprehensive comparison of these advanced techniques with uniform collocation and shooting methods in the context of MPC would represent a valuable contribution to the field.

1.3.3. Pros and Cons of OCP methods

The performance of optimal control strategies strongly depends on the underlying system dynamics, constraint setup, and cost functional, which complicates the formulation of general guidelines for method selection [28]. The following section discusses the specific properties of each direct method in more detail.

The Single shooting method (SS) only uses control input discretization resulting in small, but dense Nonlinear Program (NLP) with a few degrees of freedom [13]. For highly nonlinear systems this makes it hard to potentially find feasible solutions [13].

Multiple shooting overcomes this issue by simultaneously shooting in parallel within each interval [9]. Even though having an increased size of the NLP, the sensitivity towards poor initial guesses is reduced due to integration over shorter time domains in each subinterval [46]. This makes multiple shooting the prioritized method compared to single shooting [28].

In contrast to single shooting, multiple shooting provides a sparse NLP that can efficiently be solved with modern solvers [13]. Generally, shooting methods are a suitable option for OCPs assuming a relatively simple control input parameterization and only a few optimization variables (smaller prediction horizons) included in the NLP.

Collocation methods generally produce a very sparse NLP [8, 13, 46]. Additionally due to their implicit formulation they are also capable of stabilizing systems with stiff dynamics [47]. In the context of offline trajectory optimization, collocation methods are the method of choice with increasing problem complexity [46]. One disadvantage, specifically for higher-order collocation methods, are oscillations that might require grid adaption. Different strategies are explained in [12, 45, 36]. Applying such a scheme to MPC is time-consuming and not always feasible since the NLP dimension changes over each iteration [13].

Direct transcriptions (collocation) methods of higher order are referred to as pseudospectral collocation. Assuming smooth state and control inputs, the solution trajectories are highly accurate [47]. If smoothness is not sufficiently given, local pseudospectral combined with grid adaption might be a viable alternative [36]. In [27], a comparison between global and local strategies is pursued. Specifically the structure of local PS method can be exploited with modern solvers, as shown within the thesis.

As a summary one can conclude, that shooting methods are mostly used for online applications. Collocation methods are rooted more in the trajectory optimization background that involve more complex NLPs with more optimization variables included.

1.3.4. Application of collocation methods for OCPs

Recent developments reveal an increased popularity with collocation methods applied to MPC. Exemplary use-cases are the application of the pseudospectral collocation method to motion planning tasks for vehicles [17], the autonomous driving of vehicles [3, 14] or the guidance for hypersonic reentry [53]. Specifically [3] and [17] focus on embedded applications comparing the computation time between shooting and the pseudospectral methods both indicating a performance boost when applying pseudospectral methods. In [5], a thorough insight in the difficulties arising with pseudospectral methods applied to MPC schemes is given, and their suitability is demonstrated on several benchmark examples.

General applications of collocation strategies to trajectory optimization tasks are given in [18, 37, 58, 59]. In [18] a comparison of equidistant collocation with pseudospectral methods indicates the pseudospectral method to be more performant. Another analysis in [59] indicated equidistant collocation to be more performant. However, the latter analysis assumed an identical number of nodes for each method and not comparable accuracy biasing the results.

1.4. Scope and Contribution

The contribution of the thesis divides into three different topics:

1. A comparative survey of optimal control methods for MPC.
2. The application and comparison of a pseudospectral method to multiple shooting for MPC in OOS.
3. A universal framework to find the best optimal control method for a given control problem.

Comparative Survey A comparative overview of optimal control methods and their properties with respect to MPC implementations is provided. Special focus is set on the real-time performance capabilities. The characteristics of shooting and collocation methods are evaluated on an academic benchmark example representative for multi-body dynamics. The compatibility of each method with various nonlinear optimization solvers is analyzed to identify efficient method-solver combinations. As evaluated in Table 1.1, no comprehensive survey has systematically examined optimal control methods including different solver combinations in the MPC context.

Pseudospectral methods in MPC A pseudospectral formulation is identified to be a promising alternative to an already existing Multiple-shooting baseline version. Both methods are applied and compared on OOS scenarios.

Framework for finding the best optimal control method Furthermore, this thesis presents a universal framework for finding the most performant optimal control method for MPC tailored to domain-specific design-objectives, such as computation time or closed-loop performance. A novel Bayesian Optimization (BO) strategy derives the most suitable combination of optimal control method, its number of intervals and its non-uniform interval spacing.

An adapted multiple-segement Legendre-Gauss-Radau method (LGR-LOC*) is core of the BO. It incorporates explicit Multiple Shooting and implicit low and high-order collocation methods in one formulation. Tunable hyperparameters of the LGR-LOC* change the formulation from one to another. This enables the BO to find the best parameterization obviating the need to implement each optimal control method individually.

Moreover, the designed LGR-LOC* preserves a special structure of the OCP making it for all configurations compatible with the sparsity pattern required by the nonlinear solver FATROP [55].

The developed framework facilitates the design process of MPC by the combination of

- incorporating a single method that contains explicit and implicit formulations,
- ensuring the sparsity pattern of FATROP and
- finding the best hyperparameter setup for the optimal control method via BO.

To the best knowledge of the author, this combined approach is a novelty.

1.5. Thesis Structure

In Chapter 2, the mathematical foundations for numerical integration and the solution of initial value problems are presented. With these concepts, different explicit and implicit optimal control methods are derived. Lastly, the concept of MPC is introduced and relevant components for stability are explained.

In the following Chapter 3, the introduced optimal control methods are evaluated along different metrics. Such are computation time, closed-loop performance, method-solver combinations and the influence of open-loop oscillations on the trajectory. The results are systematically organized and summarized in the form of structured remarks. As a benchmark example a multijoint inverted pendulum is utilized.

The previously evaluated most performant optimal control methods are implemented on OOS scenarios and compared towards computation time and tracking accuracy in Chapter 4. The aforementioned methodologies comprised a local pseudospectral method and the Multiple-shooting approach.

The last Chapter 5 introduces the framework for finding the most performant optimal control method for a specific problem. The mathematical formulation of the LGR-LOC* is further addressed and the workflow for the hyperparameter optimization is explained. In a consequent step the strategy is applied to the OOS controllers and the results are presented.

2

Theoretical Background

This chapter provides a comprehensive background on the methodologies employed within this thesis. Section 2.1 focuses on numerical integration, with particular emphasis on the Newton-Cotes integration formulas and the Gauss quadrature rule.

The Newton-Cotes formulas form the foundation for standard collocation methods. In particular, the trapezoidal rule and the Hermite-Simpson method are based on this integration scheme. The Gauss quadrature rule underpins the more advanced pseudospectral collocation technique.

In Section 2.2 numerical methods for solving initial value problems are explained. They serve as a basis principle for the shooting methods that are, next to various collocation methods, explained in Section 2.3. All optimal control formulations are embedded within the context of Model Predictive Control (MPC), with a detailed discussion provided in Section 2.4.

2.1. Numerical Integration

The concept of numerical integration is to approximate an analytical integral using a quadrature rule, i.e.,

$$\int_a^b f(\tau)\mathrm{d}\tau \approx \sum_{i=1}^{N} w_i f(\tau_i) \tag{2.1}$$

where the integral is estimated by a weighted sum of function evaluations at selected points τ_i. In this approach, the function $f(\cdot)$ is effectively approximated through its values at these discrete points.

As seen in Eq. (2.1), there are two degrees of freedom, namely the choice of the quadrature weights w_i and the quadrature points τ_i. Newton-Cotes formulas assume equidistant quadrature points, while Gauss quadrature employs non-equidistant points, exploiting both degrees of freedom to achieve higher accuracy.

Table 2.1.: The Newton–Cotes formulas for numerical integration, adapted from [43]. The order of the polynomial is denoted with p, the number of quadrature points with N and the order of the method with m.

	p	N	m	$\sum_{i=1}^{N} w_i f_i$
Rectangle rule	0	1	1	hf_i
Trapezoidal rule	1	2	2	$\frac{h}{2}(f_i + f_{i+1})$
Simpson rule	2	3	4	$\frac{h}{6}(f_i + 4f_{i+\frac{1}{2}} + f_{i+1})$

2.1.1. Newton-Cotes formulas

The Newton-Cotes formulas belong to the class of interpolating integration schemes aiming to approximate the integrand $f(t)$ by a polynomial $p(t)$ of order p. This results in the general quadrature form with $N = p + 1$ equidistant quadrature points

$$\int_a^b f(t)\mathrm{d}t \approx \int_a^b p(t)\mathrm{d}t = \int_a^b \underbrace{\sum_{i=1}^{N} L_i^N(t) f(t_i)}_{p(t)} \mathrm{d}t = \sum_{i=1}^{N} w_i f(t_i), \qquad \text{with } t_{i+1} = t_i + h,$$

where L_i^N refers to the i-th Lagrange polynomial

$$L_i^N(t) = \prod_{j=1,\, j\neq i}^{N} \frac{t - t_j}{t_i - t_j}, \qquad i = 1, \dots, N. \tag{2.2}$$

It is zero at all quadrature points except from the i-th. The step size between two quadrature points is denoted as h. In Newton-Cotes formulas the only degree of freedom are the weights w_i, that are obtained by

$$w_i = \int_a^b L_i^N(t)\mathrm{d}t. \tag{2.3}$$

The accuracy of Newton-Cotes formulas is influenced by the order p of the polynomial. Common Newton-Cotes formulas and their order are listed in Table 2.1. The order m of a method indicates that the formula integrates polynomials of degree $p = m - 1$ exact. For all Newton-Cotes quadrature rules it is at least $m \geq p + 1$, this property is discussed in [48].

2.1.2. Gauss quadrature

The Gauss quadrature is an improved method to the Newton-Cotes formulas assuming non-equidistant quadrature points. This results in an order of the Gauss method of $m = 2p + 1$, as noted in [48].

The quadrature points τ are chosen as the roots from orthogonal polynomials. For classical orthogonal polynomials, the roots are in the interval $\tau \in [-1, 1]$. Applying them to arbitrary intervals $t \in [-1, 1]$ can be achieved with the transformation

$$T : [-1, 1] \to [a, b], \quad \tau \mapsto t = \frac{b-a}{2}\tau + \frac{b+a}{2}. \tag{2.4}$$

The most common families of orthogonal polynomials, specifically for pseudospectral collocation (see Section 2.3.4), are Legendre or Chebyshev polynomials [18]. In this thesis, the Legendre polynomials [25, 50]

$$P_N(\tau) = \frac{1}{2^n n!}\frac{\mathrm{d}^n}{\mathrm{d}\tau^n}\left[\left(\tau^2 - 1\right)^n\right] \tag{2.5}$$

are used. Independent of the chosen group of orthogonal polynomials, the location of the roots $\tau_1, \ldots, \tau_N$ can either include one (Radau methods), two (Lobatto methods) or no endpoint of the interval (Gauss methods).

For the case of Legendre polynomials, the quadrature points are derived in the following way:

- Legendre Gauss (LG), $\tau \in (-1, 1)$: Roots of the P_N-th Legendre polynomial. Neither of the endpoints is included.
- Legendre Gauss Radau (LGR), $\tau \in [-1, 1)$: Roots of the $P_N(\tau) + P_{N-1}(\tau)$ polynomial. Only the start point is included.[1]
- Legendre Gauss Lobatto (LGL), $\tau \in [-1, 1]$: Roots of the $(1 - \tau^2)\dot{P}_{N-1}(\tau)$ polynomial. Both, start and end point are included.

As a summary, the Gauss quadrature rule approximates an arbitrary interval $t \in [a, b]$ with

$$\begin{aligned}\int_a^b f(t)\mathrm{d}t &= \int_{-1}^1 f(T(\tau))\frac{\mathrm{d}T(\tau)}{\mathrm{d}\tau}\mathrm{d}\tau = \frac{b-a}{2}\int_{-1}^1 \underbrace{f(T(\tau))}_{:=\bar{f}(\tau)}\mathrm{d}\tau = \frac{b-a}{2}\int_{-1}^1 \bar{f}(\tau)\mathrm{d}\tau \\ &\approx \frac{b-a}{2}\int_{-1}^1 \sum_{i=1}^N L_i^N(\tau)\bar{f}(\tau_i)\mathrm{d}\tau = \frac{b-a}{2}\sum_{i=1}^N w_i\bar{f}(\tau_i) = \frac{b-a}{2}\sum_{i=1}^N w_i f(t_i).\end{aligned} \tag{2.6}$$

The weights w_i in (2.6) are computed with $w_i = \int_{-1}^1 L_i^N(\tau)\mathrm{d}\tau$. The transformation (2.4) is used to shift the roots τ to any general interval $t \in [a, b]$. An illustration of LGR and LGL roots τ and weights w is found in Figure 2.1.

The refinement over the Newton-Cotes formulas presented in Section 2.1.1 lies in the choice of quadrature points τ_i. Instead of equidistant nodes, they are selected as the

[1]LGR can also contain only the right endpoint. In this thesis, only LGR with the left-endpoint is used.

roots of an orthogonal polynomial, most commonly Legendre polynomials, leading to a substantial increase in the accuracy and order of the method.

2.2. Numerical methods for solving Initial Value Problems

Initial Value Problems (IVPs) for non-autonomous time-invariant systems have the form

$$\dot{x}(t) = f(x(t), u(t)), \quad x(t_0) = x_0.$$

Solving this problem for complex dynamics is not always feasible analytically. Instead, different numerical methods exist to approximate the solution. A good overview about the different types of time-marching methods for IVPs is given in [46]. Generally, explicit and implicit methods are distinguished.

A multistep explicit method computes the next state with information only from the current and previous states $x_{i+1} = g(x_i, u_i, x_{i-1}, u_{i-1}, \ldots)$. An explicit single step method uses information only from the current state $x_{i+1} = g(x_i, u_i)$.

Implicit methods for IVPs involve information from unknown states in the future: $x_{i+1} = g(x_i, u_i, x_{i+1}, u_{i+1}, \ldots)$. Consequently, a nonlinear system of equations must be solved. This is discussed in detail for OCP solved with collocation in Section 2.3.2 and 2.3.3.

One state-of-the-art explicit method is the Runge-Kutta (RK) method. The most trivial is the RK method of order 1 (RK1), i.e.,

$$x_{i+1} = x_i + hf(x_i, u_i), \quad \text{with } h = t_{i+1} - t_i, \tag{2.7}$$

which is also known as the Euler-Cauchy rule [43] (in the following only referred as Euler rule).

A widely used higher-order method is the fourth-order Runge-Kutta (RK4) scheme:

$$\begin{aligned}
k_1 &= f(x_i, u_i), \\
k_2 &= f(x_i + \frac{h}{2}k_1, u_i), \\
k_3 &= f(x_i + \frac{h}{2}k_2, u_i), \\
k_4 &= f(x_i + hk_3, u_i), \\
x_{i+1} &= x_i + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4).
\end{aligned} \tag{2.8}$$

Here, the control input u is assumed to remain constant over the interval length.

Both the Euler method RK1 (2.7) and the RK4 method (2.8) are used for shooting methods.

2.3. Optimal Control

The continuous optimal control problem (OCP) with the initial time assumed to be zero, $t_0 = 0$, is given as

$$\begin{aligned}
\min_{u(\cdot)} \quad & J = V_\mathrm{f}(\boldsymbol{x}(T)) + \int_0^T \mathcal{L}(\boldsymbol{x}(t), \boldsymbol{u}(t), t)\,\mathrm{d}t && (2.9)\\
\text{subject to} \quad & \dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t), t), \quad t \in [0, T] && (2.10)\\
& \boldsymbol{x}(0) = \boldsymbol{x}_\mathrm{start} && (2.11)\\
& \boldsymbol{g}(\boldsymbol{x}(t), \boldsymbol{u}(t), t) \leq 0, \quad t \in [0, T] && (2.12)\\
& \boldsymbol{x}(T) = \boldsymbol{x}_\mathrm{final} \in \mathcal{X}_\mathrm{f} && (2.13)
\end{aligned}$$

with $\boldsymbol{f}(\cdot)$ denoting the (nonlinear) system dynamics, $\boldsymbol{x}(0)$ the initial condition, $\boldsymbol{g}(\cdot)$ state or control input constraints and $\boldsymbol{x}(T)$ the terminal state within the terminal region $\mathcal{X}_\mathrm{f}$. The cost functional J consists of the terminal cost term $V_\mathrm{f}(\boldsymbol{x}(T))$, which penalizes deviation from the desired final state, and the stage cost term expressed as the integral with the Lagrangian $\mathcal{L}(\boldsymbol{x}(t), \boldsymbol{u}(t), t)$ over the time horizon. A common choice for the Lagrangian is a quadratic form in the states and inputs, as presented in (2.56).

Indirect and direct methods exist for discretizing the continuous OCP (2.9). A comparative overview with focus on direct methods can be found in Section 1.3.

In the following sections the derivation of a discrete nonlinear program (NLP) based on the continuous optimal control problem (2.9) is presented for different methods.

2.3.1. Direct Shooting methods

Single Shooting

The single shooting method (SS) transcribes the continuous OCP (2.9) into a NLP by discretizing control input $\boldsymbol{u}(t)$. The number of optimization variables for the control input is referred to as N_u. The step size between two discretized control inputs $\boldsymbol{u}_k, \boldsymbol{u}_{k+1}$ is denoted as h^k. The formulation is shown in NLP 2.1.

The Lagrangian $\mathcal{L}$ in the cost functional does not take state variables as an argument, which is an important property of the SS. The state approximation is only given passively by incorporating the Ordinary Differential Equation (ODE) by an explicit numerical scheme $\boldsymbol{\Phi}$, discussed in Section 2.2. Such integrator can be an Euler (2.7) or RK4 method (2.8). However, no optimization variables for the state discretization exist.

The major drawback of single shooting methods is the poor convergence radius [46]. Due to the explicit integration from start point to end point, small changes in the initial guess produce large deviations in the terminal state $\boldsymbol{x}_\mathrm{end}$ [8, 46].

NLP 2.1: Single Shooting

$$\min_{\boldsymbol{u}_1,\dots,\boldsymbol{u}_{N_\mathrm{u}}} \quad J = V_\mathrm{f}(\boldsymbol{x}_\mathrm{end}) + \sum_{k=1}^{N_\mathrm{u}} h^k \mathcal{L}(\boldsymbol{u}^1,\dots,\boldsymbol{u}^{N_\mathrm{u}}) \tag{2.14}$$

$$\text{subject to} \quad \boldsymbol{x}_\mathrm{end} = \boldsymbol{\Phi}(\boldsymbol{f}(\boldsymbol{x},\boldsymbol{u}), \boldsymbol{x}_\mathrm{start}, \boldsymbol{u}^1,\dots,\boldsymbol{u}^{N_\mathrm{u}}, h^1,\dots,h^{N_\mathrm{u}}), \tag{2.15}$$

$$\boldsymbol{g}(\boldsymbol{u}^1,\dots,\boldsymbol{u}^{N_\mathrm{u}}) \leq 0 \tag{2.16}$$

$$\boldsymbol{x}_\mathrm{end} \in \mathcal{X}_\mathrm{f}. \tag{2.17}$$

Multiple Shooting

The multiple shooting (MS) method overcomes the poor convergence properties [9]. Instead of integrating from start to the terminal state, multiple segments are introduced, on which a single shooting method is applied. This results in a discretization in state and control inputs, where $\boldsymbol{x}(t)$ is approximated by N_x shooting intervals $\boldsymbol{x}_i$. The control inputs are approximated by discrete variables $\boldsymbol{u}_i^k$ with $k = 1,\dots,N_{\mathrm{u},i}$ and $i = 1,\dots,N_\mathrm{x}$. The total number of control inputs is given by

$$N_\mathrm{u} = \sum_{i=1}^{N_\mathrm{x}} N_{\mathrm{u},i}.$$

To ease the notation, all control inputs in the interval i are grouped to the matrix $\mathbf{U}_i = (\boldsymbol{u}_i^1, \boldsymbol{u}_i^2, \dots, \boldsymbol{u}_i^{N_{\mathrm{u},i}})$. Respectively, different-step size are grouped per interval to the vector $\boldsymbol{h}_i = (h_i^1, h_i^2, \dots, h_i^{N_{\mathrm{u},i}})$. The most general form of formulation for MS is shwon in NLP 2.2.

NLP 2.2: Multiple Shooting

$$\min_{\boldsymbol{x}_1,\dots,\boldsymbol{x}_{N_\mathrm{x}},\mathbf{U}_1,\dots,\mathbf{U}_{N_\mathrm{x}}} \quad J = V_\mathrm{f}(\boldsymbol{x}_{N_\mathrm{x}}) + \sum_{i=1}^{N_\mathrm{x}-1} \sum_{k=1}^{N_{\mathrm{u},i}} h_i^k \mathcal{L}(\boldsymbol{x}_i, \boldsymbol{u}_i^k) \tag{2.18}$$

$$\text{subject to} \quad \boldsymbol{x}_{i+1} = \boldsymbol{\Phi}(\boldsymbol{f}(\boldsymbol{x},\boldsymbol{u}), \boldsymbol{x}_i, \mathbf{U}_i, \boldsymbol{h}_i),\ i = 1,\dots,N_\mathrm{x}-1 \tag{2.19}$$

$$\boldsymbol{x}_1 = \boldsymbol{x}_\mathrm{start}$$

$$\boldsymbol{g}(\boldsymbol{x}_1,\dots,\boldsymbol{x}_{N_\mathrm{x}}, \mathbf{U}_1,\dots,\mathbf{U}_{N_\mathrm{x}}) \leq 0$$

$$\boldsymbol{x}_{N_\mathrm{x}} \in \mathcal{X}_\mathrm{f}$$

The discretization of the control inputs should be finer than that of the state variables, meaning $N_\mathrm{u} + 1 \geq N_\mathrm{x}$. The additional "+1" accounts for the fact that no discretization of the control input is required at the terminal point $\boldsymbol{x}_N$. In practice, a common choice is to set the number of control inputs equal to the number of shooting intervals, such that $N_\mathrm{x} = N$ and $N_\mathrm{u} = N - 1$. The NLP 2.2 reduces to

$$\begin{aligned} \min_{\boldsymbol{x}_1,\dots,\boldsymbol{x}_N,\boldsymbol{u}_1,\dots\boldsymbol{u}_{N-1}} \quad & J = V_\mathrm{f}(\boldsymbol{x}_N) + \sum_{i=1}^{N-1} h_i \mathcal{L}(\boldsymbol{x}_i, \boldsymbol{u}_i) \\ \text{subject to} \quad & \boldsymbol{x}_{i+1} = \boldsymbol{\Phi}(\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}), \boldsymbol{x}_i, \boldsymbol{u}_i, h_i), \quad i = 1, \dots, N-1 \\ & \boldsymbol{x}_1 = \boldsymbol{x}_\mathrm{start} \\ & \boldsymbol{g}(\boldsymbol{x}_1, \dots, \boldsymbol{x}_N, \boldsymbol{u}_1, \dots, \boldsymbol{u}_{N-1}) \leq 0 \\ & \boldsymbol{x}_N \in \mathcal{X}_\mathrm{f}. \end{aligned} \tag{2.20}$$

2.3.2. Direct Trapezoidal collocation

In comparison to shooting methods, collocation methods enforce the system dynamics constraints implicitly. The coefficients of an n-order polynomial are fitted to match the system dynamics at given collocation points. In the case of trapezoidal collocation, the system dynamics are approximated by piecewise linear polynomials

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t)) \approx \boldsymbol{a}(t - t_i) + \boldsymbol{b} \tag{2.21}$$

in each interval $t \in [t_i, t_{i+1}]$. The resulting state trajectory in a given interval i is a quadratic polynomial

$$\begin{aligned} \boldsymbol{x}(t) &= \boldsymbol{x}(t_i) + \int_{t_i}^{t} \boldsymbol{f}(\boldsymbol{x}(\tau), \boldsymbol{u}(\tau))\,\mathrm{d}\tau \quad \text{with } t_i \leq t \leq t_{i+1} \\ &\approx \boldsymbol{x}(t_i) + \int_{t_i}^{t} \boldsymbol{a}\tau + \boldsymbol{b}\,\mathrm{d}\tau \\ &= \boldsymbol{x}_i + \frac{\boldsymbol{a}}{2}(t - t_i)^2 + \boldsymbol{b}(t - t_i). \end{aligned} \tag{2.22}$$

The value of the state at $\boldsymbol{x}(t_i)$ is abbreviated with $\boldsymbol{x}_i$. The polynomial (2.22) has to be aligned with the system dynamics by choosing the coefficients $\boldsymbol{a}, \boldsymbol{b}$ properly. For the determination of the two unknowns $\boldsymbol{a}, \boldsymbol{b}$, the system dynamics have to be equivalent to the state approximation at the start time t_i and the end time t_{i+1} of the interval:

$$\dot{\boldsymbol{x}}(t = t_i) = \boldsymbol{f}(\boldsymbol{x}_i, \boldsymbol{u}_i) = \boldsymbol{f}_i \overset{!(2.21)}{=} \boldsymbol{a}(t_i - t_i) + \boldsymbol{b} \quad \Longrightarrow \boldsymbol{b} = \boldsymbol{f}_i = \boldsymbol{f}(\boldsymbol{x}_i, \boldsymbol{u}_i) \tag{2.23}$$

$$\dot{\boldsymbol{x}}(t = t_{i+1}) = \boldsymbol{f}(\boldsymbol{x}_{i+1}, \boldsymbol{u}_{i+1}) = \boldsymbol{f}_{i+1} \overset{!(2.21)}{=} \boldsymbol{a}(t_{i+1} - t_i) + \boldsymbol{b} \quad \Longrightarrow \boldsymbol{a} = \frac{\boldsymbol{f}_{i+1} - \boldsymbol{f}_i}{t_{i+1} - t_i}. \tag{2.24}$$

Inserting the above constraints (2.23) and (2.24) into the formulation (2.22) leads to a function of the state in each interval

$$\boldsymbol{x}(t) = \boldsymbol{x}_i + \frac{1}{2}\frac{\boldsymbol{f}_{i+1} - \boldsymbol{f}_i}{t_{i+1} - t_i}(t - t_i)^2 + \boldsymbol{f}_i(t - t_i), \quad t \in [t_i, t_{i+1}]. \tag{2.25}$$

NLP transcription for trapezoidal collocation

The variables $\boldsymbol{x}_{i+1}, \boldsymbol{u}_{i+1}$ and consequently $\boldsymbol{f}_{i+1} = \boldsymbol{f}(\boldsymbol{x}_{i+1}, \boldsymbol{u}_{i+1})$ are unknown at a given point t_i. Computing equation (2.25) at any time $t \in [t_i, t_{i+1}]$ is only possible after having the full state and control input solution. To account for this, continuity constraints are incorporated in the NLP transcription. To derive them, the state trajectory (2.25) is evaluated at the end of the interval $t = t_{i+1}$ leading to the formulation

$$\begin{aligned}\boldsymbol{x}(t = t_{i+1}) = \boldsymbol{x}_{i+1} &= \boldsymbol{x}_i + \frac{1}{2}\frac{\boldsymbol{f}_{i+1} - \boldsymbol{f}_i}{t_{i+1} - t_i}(t_{i+1} - t_i)^2 + \boldsymbol{f}_i(t_{i+1} - t_i) \\ &= \boldsymbol{x}_i + \frac{1}{2}(\boldsymbol{f}_{i+1} - \boldsymbol{f}_i)(t_{i+1} - t_i) + \boldsymbol{f}_i(t_{i+1} - t_i) \\ &= \boldsymbol{x}_i + \frac{1}{2}(\boldsymbol{f}_{i+1} - \boldsymbol{f}_i)h_i + \boldsymbol{f}_i h_i = \boldsymbol{x}_i + \frac{h_i}{2}(\boldsymbol{f}_{i+1} + \boldsymbol{f}_i)\end{aligned}$$

which can be compactly written as the classical trapezoidal update:

$$\boldsymbol{x}_{i+1} - \boldsymbol{x}_i = \frac{h_i}{2}(\boldsymbol{f}_{i+1} + \boldsymbol{f}_i). \tag{2.26}$$

with $h_i = t_{i+1} - t_i$ being the step size of the interval.

Equation (2.26) is the trapezoidal rule for approximating an integral which gives the method its name (see Table 2.1). This leads to the formulation in NLP 2.3.

The control inputs are chosen equivalently as the state discretization and are interpolated linearly in between interval borders. The integral in the cost function (2.9) is approximated with the trapezoidal rule.

2.3.3. Direct Hermite Simpson collocation

The direct Hermite Simpson follows a similar approach than the TC, but uses a quadratic polynomial instead of a linear polynomial for the state derivative approximation

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t)) \approx \boldsymbol{a}(t - t_i)^2 + \boldsymbol{b}(t - t_i) + \boldsymbol{c}. \tag{2.29}$$

This leads to a cubic representation for the state

$$\boldsymbol{x}(t) \approx \boldsymbol{x}_i + \frac{\boldsymbol{a}}{3}(t - t_i)^3 + \frac{\boldsymbol{b}}{2}(t - t_i)^2 + \boldsymbol{c}t, \quad \text{with } t_i \le t \le t_{i+1}. \tag{2.30}$$

NLP 2.3: Trapezoidal Collocation (TC)

$$\min_{\boldsymbol{x}_1,\dots,\boldsymbol{x}_N,\boldsymbol{u}_1,\dots\boldsymbol{u}_{N-1}} \quad J = V_\mathrm{f}(\boldsymbol{x}_N) + \sum_{i=1}^{N-1} \frac{h_i}{2}(\mathcal{L}(\boldsymbol{x}_i,\boldsymbol{u}_i) + \mathcal{L}(\boldsymbol{x}_{i+1},\boldsymbol{u}_{i+1})) \tag{2.27}$$

$$\text{subject to} \quad \boldsymbol{x}_{i+1} - \boldsymbol{x}_i = \frac{h}{2}(\boldsymbol{f}_{i+1} + \boldsymbol{f}_i) \quad i = 1,\dots,N-1 \tag{2.28}$$

$$\boldsymbol{x}_1 = \boldsymbol{x}_\mathrm{start}$$

$$\boldsymbol{g}(\boldsymbol{x}_1,\dots,\boldsymbol{x}_N,\boldsymbol{u}_1,\dots,\boldsymbol{u}_{N-1}) \leq 0$$

$$\boldsymbol{x}_N \in \mathcal{X}_\mathrm{f}$$

The control input $\boldsymbol{u}_N$ at the final point is required for the computation of $\boldsymbol{f}_N$ in the trapezoidal rule. It is assumed to be constant to the previous control input $\boldsymbol{u}_{N-1} = \boldsymbol{u}_N$ and consequently not considered as optimization variable.

To define the coefficients $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}$, three constraints are required. The derivatives of the polynomial are enforced to be equivalent to the system dynamics at the start (2.31) and end point of the interval (2.32). The third constraint is to apply the dynamics equal to the derivative of the polynomial at the mid-point of the interval (2.33):

$$\dot{\boldsymbol{x}}(t = t_i) = \boldsymbol{f}(\boldsymbol{x}_i,\boldsymbol{u}_i) \overset{!(2.29)}{=} \boldsymbol{a}(t_i - t_i)^2 + \boldsymbol{b}(t_i - t_i) + \boldsymbol{c} \qquad \Longrightarrow \boldsymbol{c} = \boldsymbol{f}_i = \boldsymbol{f}(\boldsymbol{x}_i,\boldsymbol{u}_i) \tag{2.31}$$

$$\dot{\boldsymbol{x}}(t = t_{i+1}) = \boldsymbol{f}(\boldsymbol{x}_{i+1},\boldsymbol{u}_{i+1}) = \boldsymbol{f}_{i+1} \overset{!(2.29)}{=} \boldsymbol{a}(t_{i+1} - t_i)^2 + \boldsymbol{b}(t_{i+1} - t_i) + \boldsymbol{c} \tag{2.32}$$

$$\dot{\boldsymbol{x}}(t = t_{i+\frac{1}{2}}) = \boldsymbol{f}(\boldsymbol{x}_{i+\frac{1}{2}},\boldsymbol{u}_{i+\frac{1}{2}}) = \boldsymbol{f}_{i+\frac{1}{2}} \overset{!(2.29)}{=} \boldsymbol{a}(t_{i+\frac{1}{2}} - t_i)^2 + \boldsymbol{b}(t_{i+\frac{1}{2}} - t_i) + \boldsymbol{c}. \tag{2.33}$$

The first constraint (2.31) can be solved directly, while for the constraints (2.32) and (2.32) the solution of a linear equation system leads to the coefficients

$$\boldsymbol{a} = \frac{2\boldsymbol{f}_i - 4\boldsymbol{f}_{i+\frac{1}{2}} + 2\boldsymbol{f}_{i+1}}{h^2}, \qquad \boldsymbol{b} = \frac{-3\boldsymbol{f}_i + 4\boldsymbol{f}_{i+\frac{1}{2}} - \boldsymbol{f}_{i+1}}{h}. \tag{2.34}$$

The state trajectory (2.30) with the coefficients then is

$$\boldsymbol{x}(t) = \boldsymbol{x}_0 + \frac{2\boldsymbol{f}_i - 4\boldsymbol{f}_{i+\frac{1}{2}} + 2\boldsymbol{f}_{i+1}}{3h^2}(t - t_i)^3 + \frac{-3\boldsymbol{f}_i + 4\boldsymbol{f}_{i+\frac{1}{2}} - \boldsymbol{f}_{i+1}}{2h}(t - t_i)^2 + \boldsymbol{f}_i(t - t_i). \tag{2.35}$$

NLP transcription for Hermite-Simpson collocation

The implicit formulation for the Hermite-Simpson collocation method follows by evaluating (2.35) at $t = t_{i+1}$ leading to

$$\boldsymbol{x}(t = t_{i+1}) \approx \boldsymbol{x}_i + \frac{h}{6}(\boldsymbol{f}_i + 4\boldsymbol{f}_{i+\frac{1}{2}} + \boldsymbol{f}_{i+1}) = \boldsymbol{x}_{i+1}. \tag{2.36}$$

In this formulation, in addition to the unknown variables $\boldsymbol{x}_{i+1}, \boldsymbol{u}_{i+1}$ as in the case of trapezoidal collocation, the state at the mid-point $\boldsymbol{x}_{i+\frac{1}{2}}, \boldsymbol{u}_{i+\frac{1}{2}}$ to compute $\boldsymbol{f}_{i+\frac{1}{2}}$ is unknown. To embed the mid-point within the NLP, one differs between the compressed and separated form [8]. In the compressed form, an approximation for the mid-point state is used (interpolating polynomial) while for the separated form mid-point optimization variables are included in the NLP. This increases the dimension, but might have some advantages for specific problem and solver combinations [8]. In the frame of this thesis, the compressed form is used.

As an approximation for the state $\boldsymbol{x}_{i+\frac{1}{2}}$ a cubic Hermite interpolation method is used that predicts the mid-point state based on x_i and x_{i+1}:

$$\boldsymbol{x}_{i+\frac{1}{2}} = \frac{1}{2}(\boldsymbol{x}_i + \boldsymbol{x}_{i+1}) + \frac{h}{8}(\boldsymbol{f}_i - \boldsymbol{f}_{i+1}). \tag{2.37}$$

The Equations (2.36) and (2.37) give the method its name. The first rule is identical to the Simpson rule for integration (refer to Table 2.1), while the interpolation rule for the estimate at the midpoint is called cubic Hermite interpolation. This results in the NLP 2.4.

The Hermite-Simpson rule discretizes the continuous integral in the cost function (2.9) with the Simpson quadrature to avoid inconsistencies. The control inputs are piecewise constants within each interval $[t_i, t_{i+1}]$.

2.3.4. Pseudospectral methods

A comparison between pseudospectral methods and local implicit collocation schemes highlights two fundamental distinctions in their design principles. The first one is to use global approximation instead of dividing the interval into subsegments. The second one is the application of higher order quadrature rules, namely the Gauss quadrature (Section 2.1.2), to approximate the system dynamics and the Lagrange term in the cost functional. This results in spectral (exponential) convergence properties for pseudospectral methods assuming smooth problems [15, 54].

Three different methods for the quadrature can be used, namely Legendre Gauss (LG), Legendre Gauss Radau (LGR) and Legendre Gauss Lobatto (LGL) methods. In this thesis, only LGR and LGL quadrature is used. LG methods are more complex to embed

NLP 2.4: Hermite-Simpson Collocation (HS)

Compressed form, using an approximation for the midpoint state $\boldsymbol{x}_{i+\frac{1}{2}}$ [8].

$$\min_{\boldsymbol{x}_1,\dots,\boldsymbol{x}_N,\boldsymbol{u}_1,\dots\boldsymbol{u}_{N-1}} \quad J = V_\mathrm{f}(\boldsymbol{x}_N) + \sum_{i=1}^{N-1} \frac{h_i}{6}\Big(\mathcal{L}(\boldsymbol{x}_i,\boldsymbol{u}_i) + 4\mathcal{L}(\boldsymbol{x}_{i+\frac{1}{2}},\boldsymbol{u}_{i+\frac{1}{2}}) + \mathcal{L}(\boldsymbol{x}_{i+1},\boldsymbol{u}_{i+1})\Big) \tag{2.38}$$

$$\text{subject to} \quad \boldsymbol{x}_{i+1} - \boldsymbol{x}_i = \frac{h}{6}(\boldsymbol{f}_i + 4\boldsymbol{f}_{i+\frac{1}{2}} + \boldsymbol{f}_{i+1}), \quad i = 1,\dots,N-1 \tag{2.39}$$

$$\boldsymbol{x}_1 = \boldsymbol{x}_\mathrm{start}$$

$$\boldsymbol{g}(\boldsymbol{x}_1,\dots,\boldsymbol{x}_N,\boldsymbol{u}_1,\dots,\boldsymbol{u}_{N-1}) \leq 0$$

$$\boldsymbol{x}_N \in \mathcal{X}_\mathrm{f}$$

The control input $\boldsymbol{u}_N$ at the final point is required for the computation of $\boldsymbol{f}_N$ in the Hermite-Simpson rule. It is assumed to be constant to the previous control input $\boldsymbol{u}_{N-1} = \boldsymbol{u}_N$ and consequently not considered as optimization variable.

in MPC applications, due to the non-existence of a collocation point at $\tau = -1$ and $\tau = 1$, which is required for the initial and the terminal state in the receding horizon. The specific nodes and corresponding weights for LGL and LGR are shown in Figure 2.1. The LGR method that includes the left endpoint is used throughout the thesis.

NLP transcription for Legendre Gauss Lobatto (LGL) method

The LGL method is firstly implemented by [15]. The state and control input trajectories are approximated according to the Gauss quadrature with Lagrange polynomials L_i (2.2)

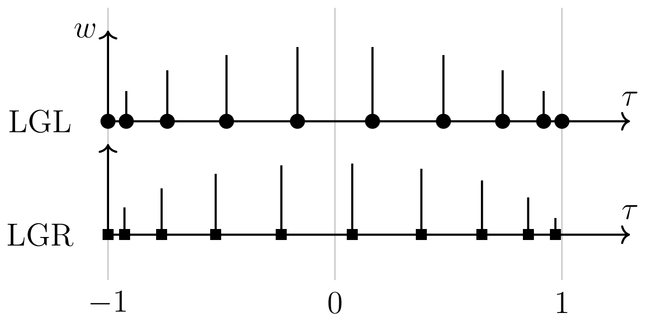


Figure 2.1.: Roots τ and weights w of the Legendre polynomial with $N = 10$ for the LGL and LGR method.

as basis functions

$$\boldsymbol{x}(\tau) = \sum_{i=1}^{N} \boldsymbol{x}_i L_i^N(\tau), \qquad \boldsymbol{u}(\tau) = \sum_{i=1}^{N} \boldsymbol{u}_i L_i^N(\tau). \tag{2.40}$$

An important property of the LGL method is, that within the N collocation points, the initial point $\boldsymbol{x}_1$, but also the final point $\boldsymbol{x}_N = \boldsymbol{x}_{\text{final}}$ is included.

To determine the unknown parameters $\boldsymbol{x}_i, \boldsymbol{u}_i$ in (2.40), collocation constraints are enforced at the orthogonal collocation points $\tau_1, \dots, \tau_N$ derived by the quadrature rule

$$\begin{aligned} \dot{\boldsymbol{x}}(t_j) &= \frac{\mathrm{d}}{\mathrm{d}t}\left(\sum_{i=1}^{N} \boldsymbol{x}_i L_i^N(\tau_j)\right) = \frac{\mathrm{d}}{\mathrm{d}\tau}\frac{\mathrm{d}\tau}{\mathrm{d}t}\left(\sum_{i=1}^{N} \boldsymbol{x}_i L_i^N(\tau_j)\right) \\ &= \frac{2}{b-a}\sum_{i=1}^{N} \boldsymbol{x}_i \dot{L}_i^N(\tau_j) = \frac{2}{b-a}\sum_{i=1}^{N} \boldsymbol{x}_i D_{ji} \overset{!}{=} \boldsymbol{f}(\boldsymbol{x}_j, \boldsymbol{u}_j). \end{aligned} \tag{2.41}$$

This formulation must be incorporated for each collocation point $j = 1, \dots, N$. The derivative of the Lagrange polynomials is described as

$$\frac{\mathrm{d}}{\mathrm{d}\tau} L_i^N(\tau_j) = \dot{L}_i^N(\tau_j).$$

The derivation of the transformation (2.4) equals to

$$\frac{\mathrm{d}\tau}{\mathrm{d}t} = \frac{2}{b-a}.$$

With the differentiation matrix

$$\mathbf{D} = \begin{pmatrix} \dot{L}_1(\tau_1) & \cdots & \dot{L}_N(\tau_1) \\ \dot{L}_1(\tau_2) & \cdots & \dot{L}_N(\tau_2) \\ & \vdots & \\ \dot{L}_1(\tau_N) & \cdots & \dot{L}_N(\tau_N) \end{pmatrix} \in \mathbb{R}^{N \times N}$$

the formulation (2.41) is written compact as

$$\frac{2}{b-a}\mathbf{D}\mathbf{X} \overset{!}{=} \mathbf{F}(\mathbf{X}, \mathbf{U}). \tag{2.42}$$

The matrices $\mathbf{X} \in \mathbb{R}^{N \times n_{\mathrm{x}}}$ and $\mathbf{U} \in \mathbb{R}^{N \times n_{\mathrm{u}}}$ are denoted as the state and control input matrix, storing the state information at each collocation point in a matrix

$$\mathbf{X} = \begin{pmatrix} \boldsymbol{x}_1^{\mathrm{T}} \\ \vdots \\ \boldsymbol{x}_N^{\mathrm{T}} \end{pmatrix} = \begin{pmatrix} x_{11} & \dots & x_{1n_{\mathrm{x}}} \\ \vdots & \ddots & \vdots \\ x_{N1} & \dots & x_{Nn_{\mathrm{x}}} \end{pmatrix}, \quad \mathbf{U} = \begin{pmatrix} \boldsymbol{u}_1^{\mathrm{T}} \\ \vdots \\ \boldsymbol{u}_N^{\mathrm{T}} \end{pmatrix} = \begin{pmatrix} u_{11} & \dots & u_{1n_{\mathrm{u}}} \\ \vdots & \ddots & \vdots \\ u_{N1} & \dots & u_{Nn_{\mathrm{u}}} \end{pmatrix}. \tag{2.43}$$

The matrix for the system dynamics is defined as

$$\mathbf{F} = \begin{pmatrix} \boldsymbol{f}(\boldsymbol{x}_1, \boldsymbol{u}_1)^{\mathrm{T}} \\ \vdots \\ \boldsymbol{f}(\boldsymbol{x}_N, \boldsymbol{u}_N)^{\mathrm{T}} \end{pmatrix} = \begin{pmatrix} f_1(x_1, u_1) & \dots & f_{n_{\mathrm{x}}}(x_1, u_1) \\ \vdots & \ddots & \vdots \\ f_1(x_N, u_N) & \dots & f_{n_{\mathrm{x}}}(x_N, u_N) \end{pmatrix}.$$

Fahroo et al. give more details on the mathematical properties on the differentiation matrix [16]. This results in the NLP 2.5.

NLP 2.5: Legendre Gauss Lobatto method (LGL)

$$\min_{\boldsymbol{x}_1,\dots,\boldsymbol{x}_N,\boldsymbol{u}_1,\dots,\boldsymbol{u}_N} \quad J = V_{\mathrm{f}}(\boldsymbol{x}_N) + \frac{b-a}{2}\sum_{i=1}^{N} w_i \mathcal{L}(\boldsymbol{x}_i, \boldsymbol{u}_i) \tag{2.44}$$

$$\begin{aligned} \text{subject to} \quad & \frac{2}{b-a}\mathbf{D}\mathbf{X} \overset{!}{=} \mathbf{F}(\mathbf{X}, \mathbf{U}) \\ & \boldsymbol{x}_1 = \boldsymbol{x}_{\mathrm{start}} \\ & \boldsymbol{g}(\boldsymbol{x}_1, \dots, \boldsymbol{x}_N, \boldsymbol{u}_1, \dots, \boldsymbol{u}_N) \leq 0 \\ & \boldsymbol{x}_N \in \mathcal{X}_{\mathrm{f}} \end{aligned}$$

All parameters $\boldsymbol{x}_1, \dots, \boldsymbol{x}_N$ and $\boldsymbol{u}_1, \dots, \boldsymbol{u}_N$ refer to the values at the collocation points t_i. A similar discretization of the state and control inputs is assumed.

NLP transcription for Legendre Gauss Radau (LGR) method

The LGR method does not contain the end-point of the interval $\tau = 1, t = a$ (see Figure 2.1). In addition to the N collocation points $\tau_1, \dots, \tau_N$, another node τ_{N+1} exists, that is not collocated, but used for interpolation to the terminal state

$$\boldsymbol{x}(\tau) = \sum_{i=1}^{N+1} \boldsymbol{x}_i L_i^{N+1}(\tau), \qquad \boldsymbol{u}(\tau) = \sum_{i=1}^{N+1} \boldsymbol{u}_i L_i^{N+1}(\tau) \tag{2.45}$$

with

$$L_i^{N+1}(\tau) = \prod_{j=1,\, j\neq i}^{N+1} \frac{\tau - \tau_j}{\tau_i - \tau_j}, \qquad \forall i = 1, \dots, N+1.$$

This formulation is similar to the one used in [20]. Deriving the differentiation matrix with a non-equal number of interpolation points and collocation points results in a non-square shape

$$\mathbf{D} = \begin{pmatrix} \dot{L}_1(\tau_1) & \cdots & \dot{L}_N(\tau_1) & \dot{L}_{N+1}(\tau_1) \\ \dot{L}_1(\tau_2) & \cdots & \dot{L}_N(\tau_2) & \dot{L}_{N+1}(\tau_2) \\ & \vdots & & \\ \dot{L}_1(\tau_N) & \cdots & \dot{L}_N(\tau_N) & \dot{L}_{N+1}(\tau_N) \end{pmatrix} \in \mathbb{R}^{N\times N+1}. \tag{2.46}$$

The dimension for the state matrix, with the structure shown in (2.43), is $\mathbf{X} \in \mathbb{R}^{(N+1)\times n_\mathrm{x}}$, for the control input matrix it is $\mathbf{U} \in \mathbb{R}^{(N+1)\times n_\mathrm{u}}$ and for the system dynamics matrix it is $\mathbf{F} \in \mathbb{R}^{N\times n_\mathrm{x}}$. This leads to the collocation condition

$$\frac{2}{b-a}\mathbf{D}\mathbf{X} \overset{!}{=} \mathbf{F}(\mathbf{X},\mathbf{U}). \tag{2.47}$$

The difference between (2.42) and (2.47) is the different collocation points used within the differentiation matrix and the corresponding dimensions of **D** and **X**. Fundamentally different mathematical properties result from the different methods, that are discussed in detail in [19]. This results in the NLP 2.6.

NLP 2.6: Legendre Gauss Radau method (LGR)

$$\begin{aligned} \min_{\boldsymbol{x}_1,\ldots,\boldsymbol{x}_{N+1},\boldsymbol{u}_1,\ldots,\boldsymbol{u}_N} \quad & J = V_\mathrm{f}(\boldsymbol{x}_{N+1}) + \frac{b-a}{2}\sum_{i=1}^{N} w_i \mathcal{L}(\boldsymbol{x}_i,\boldsymbol{u}_i) \\ \text{subject to} \quad & \frac{2}{b-a}\mathbf{D}\mathbf{X} \overset{!}{=} \mathbf{F}(\mathbf{X},\mathbf{U}) \\ & \boldsymbol{x}_1 = \boldsymbol{x}_\mathrm{start} \\ & \boldsymbol{g}(\boldsymbol{x}_1,\ldots,\boldsymbol{x}_{N+1},\boldsymbol{u}_1,\ldots,\boldsymbol{u}_{N+1}) \le 0 \\ & \boldsymbol{x}_{N+1} \in \mathcal{X}_\mathrm{f} \end{aligned} \tag{2.48}$$

All parameters $\boldsymbol{x}_1,\ldots,\boldsymbol{x}_N$ and $\boldsymbol{u}_1,\ldots,\boldsymbol{u}_N$ refer to the values at the collocation points t_i, the variable $\boldsymbol{x}_{N+1}$ refers to the non-collocated state variables at the final point.

2.3.5. Local orthogonal methods

Local orthogonal methods are a subgroup of pseudospectral methods using not only one global polynomial, but dividing the domain $[a,b]$ into N_int intervals:

$$a = a_1 < b_1 = a_2 < b_2 = a_3 < \ldots b_{N_\mathrm{int}} = b.$$

Within each subdomain $[a_l, b_l]$ with $l \in N_{\text{int}}$, a classical pseudospectral method is applied. In local orthogonal methods, the scalar N extends to a vector $\boldsymbol{N} = (N_1, \dots, N_{N_{\text{int}}})$ specifying the number of collocation nodes in each interval $l = 1, \dots, N_{\text{int}}$. In index notation each element of the vector $\boldsymbol{N}$ is denoted as N_l, $l = 1, \dots, N_{\text{int}}$. For simplicity, the number of collocation nodes is assumed to be constant in each interval for this work:

$$N_l = \text{const.}, \forall l = 1, \dots, N_{\text{int}}.$$

As a result, the NLP formulation of a local pseudospectral method is parameterized by the two parameters N_{int} for the number of intervals and N_l for the number of collocation nodes in each subinterval.

An illustration of a local orthogonal method is shown in Figure 2.2. The domain is divided into two equidistant intervals ($N_{\text{int}} = 2$). In comparison to Figure 2.1 with $N = 10$ collocation points, the local orthogonal method has five collocation points ($N_l = 5$) in each interval summing up to a total number of $\sum_{l=1}^{N_{\text{int}}} N_l = 10$.

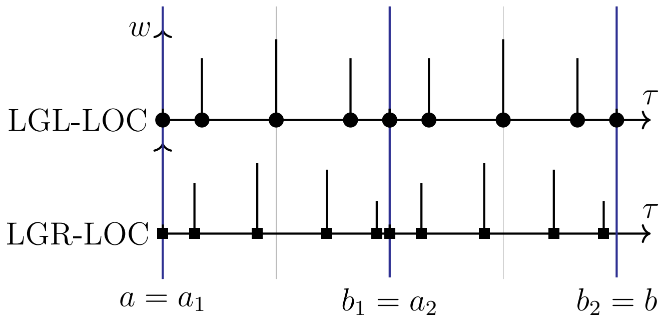


Figure 2.2.: Roots τ for a local LGL and LGR method with two intervals $N_{\text{int}} = 2$. The domain $[a, b]$ is divided into equidistant intervals. Both methods use $N_l = 5$ collocation points in each interval. On the y-axis the weights w corresponding to the quadrature rule are shown.

The advantage of local orthogonal methods is the increased sparsity in the NLP Jacobian [12]. This can be seen in Figure 2.3, in which a global pseudospectral method with $N = 10$ (collocation points from Figure 2.1) and a local orthogonal pseudospectral method with $N_{\text{int}} = 2, N_l = 5$ (collocation points from Figure 2.2) are compared. A sparser problem is handled more efficiently with modern nonlinear solvers resulting in reduced computational costs.

It should be noted, that the sparsity can be further enhanced by increasing the number of intervals N_{int}. However, a lower bound exists, as each interval must contain a minimum number of collocation points to ensure the validity of the discretization.

The spectral convergence properties of global pseudospectral methods (Section 2.3.4) cannot be guaranteed for the local scheme, although the local methods might be capable for

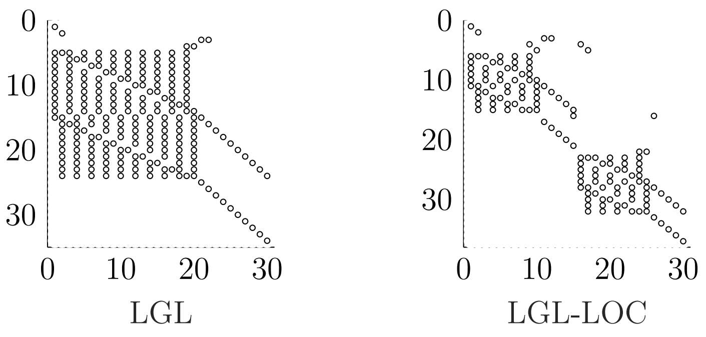


Figure 2.3.: Sparsity comparison of the constraint Jacobian $\nabla \boldsymbol{g}$ for the global LGL method ($N = 10$) and the local method LGL-LOC ($N = 10$).

handling non-smooth problems due to the possibility to divide the interval into multiple segments [12].

NLP transcription for the Local Legendre Gauss Lobatto (LGL-LOC) method

For the LGL-LOC method, the state and control input within each subinterval $\boldsymbol{x}^l(t), \boldsymbol{x}^l(t)$ is approximated via Eq. (2.40). This leads to N_{int} piecewise polynomial definitions. The state and control input matrix, according to Eq. (2.43), are denoted for each interval as $\mathbf{X}^l \in \mathbb{R}^{N_l \times n_{\text{x}}}$ and $\mathbf{U}^l \in \mathbb{R}^{N_l \times n_{\text{u}}}$. The formulation for LGL-LOC is given in NLP 2.7. Continuity for the LGL-LOC method is ensured by the constraint (2.51) and (2.52). The last row of $\mathbf{X}^l$, namely $\boldsymbol{x}^l_{N_l}$, is equivalent to the first row of $\mathbf{X}^{l+1}$, that is $\boldsymbol{x}^{l+1}_1$. This can be seen in Figure 2.2, since two collocation points coincide on the interval borders. The last point of the premier interval is identical with the first point of the second interval for LGL.

NLP transcription for the Local Legendre Gauss Radau (LGR-LOC) method

The LGR-LOC state approximation in each segment is identical to Eq. (2.45). The state and control input matrix have the shape $\mathbf{X}^l \in \mathbb{R}^{(N_l+1) \times n_{\text{x}}}$ and $\mathbf{U}^l \in \mathbb{R}^{(N_l+1) \times n_{\text{u}}}$. The non-collocated endpoint of each interval is not included as an additional optimization variable. Instead, the first point of the following interval is used. The state matrix in each interval is given as

$$\mathbf{X}^l = \begin{pmatrix} (\boldsymbol{x}^l_1)^{\text{T}} \\ \vdots \\ (\boldsymbol{x}_{N_l})^{\text{T}} \\ (\boldsymbol{x}^{l+1}_1)^{\text{T}} \end{pmatrix} = \begin{pmatrix} x^l_{11} & \dots & x^l_{1n_{\text{x}}} \\ \vdots & \ddots & \vdots \\ x^l_{N_l 1} & \dots & x^l_{N_l n_{\text{x}}} \\ x^{l+1}_{11} & \dots & x^{l+1}_{1n_{\text{x}}} \end{pmatrix} \tag{2.53}$$

NLP 2.7: Local Legendre Gauss Lobatto method (LGL-LOC)

$$\min_{\mathbf{X}^1,\dots,\mathbf{X}^{N_\text{int}},\mathbf{U}^1,\dots,\mathbf{U}^{N_\text{int}}} \quad J = V_\text{f}(\boldsymbol{x}_{N_l}^{N_\text{int}}) + \sum_{l=1}^{N_\text{int}} \frac{b_l - a_l}{2} \sum_{i=1}^{N_l} w_i^l \mathcal{L}(\boldsymbol{x}_i^l, \boldsymbol{u}_i^l) \tag{2.49}$$

$$\text{subject to} \quad \frac{2}{b_l - a_l}\mathbf{D}^l\mathbf{X}^l \overset{!}{=} \mathbf{F}^l(\mathbf{X}^l, \mathbf{U}^l), \quad l = 1, \dots, N_\text{int} \tag{2.50}$$

$$\boldsymbol{x}_1 = \boldsymbol{x}_\text{start}$$

$$\boldsymbol{x}_1^{l+1} = \boldsymbol{x}_{N_l}^l, \quad l = 1, \dots, N_\text{int} - 1 \tag{2.51}$$

$$\boldsymbol{u}_1^{l+1} = \boldsymbol{u}_{N_l}^l, \quad l = 1, \dots, N_\text{int} - 1 \tag{2.52}$$

$$\boldsymbol{g}(\mathbf{X}^1, \dots, \mathbf{X}^{N_\text{int}}, \mathbf{U}^1, \dots, \mathbf{U}^{N_\text{int}}) \leq 0$$

$$\boldsymbol{x}_{N_l}^{N_\text{int}} \in \mathcal{X}_\text{f}$$

The parameters $w_i^l, \boldsymbol{x}_i^l, \boldsymbol{u}_i^l$ refer to the value at the collocation point $i = 1, \dots, N_l$ in the interval $l = 1, \dots, N_\text{int}$.

and the control input matrix as:

$$\mathbf{U}^l = \begin{pmatrix} (\boldsymbol{u}_1^l)^\text{T} \\ \vdots \\ (\boldsymbol{u}_N^l)^\text{T} \end{pmatrix} = \begin{pmatrix} u_{11}^l & \dots & u_{1n_\text{u}}^l \\ \vdots & \ddots & \vdots \\ u_{N_l 1}^l & \dots & u_{N_l n_\text{u}}^l \end{pmatrix}.$$

For the final state another optimization variable $\boldsymbol{x}^\text{final}$ is introduced. It is incorporated in the last state matrix $\mathbf{X}^{N_\text{int}}$ in the last row. This results in the NLP formulation 2.8.

The incorporation of the state value of the next interval $l + 1$ in the state matrix $\mathbf{X}^l$ in (2.53) makes continuity constraints for the state redundant. For the control inputs no continuity between intervals are considered to reduce computation time.

2.4. Model Predictive Control

Model Predictive Control (MPC) is an optimization-based control strategy in which, at each sampling instant, an open-loop optimal control problem (OCP) is solved online and only the first element of the resulting control sequence is applied to the plant. The optimization is then repeated at the next time step, which yields closed-loop control setup.

A schematic representation of an MPC scheme is shown in Figure 2.4. At each time step t_n, an open-loop OCP of the form (2.9) is solved, subject to system dynamics (2.10),

NLP 2.8: Local Legendre Gauss Radau method (LGR-LOC)

$$\min_{\mathbf{X}^1,\ldots,\mathbf{X}^{N_{\text{int}}},\mathbf{U}^1,\ldots,\mathbf{U}^{N_{\text{int}}}} \quad J = V_{\text{f}}(\boldsymbol{x}^{\text{final}}) + \sum_{l=1}^{N_{\text{int}}} \frac{b_l - a_l}{2} \sum_{i=1}^{N_l} w_i^l \mathcal{L}(\boldsymbol{x}_i^l, \boldsymbol{u}_i^l) \tag{2.54}$$

$$\text{subject to} \quad \frac{2}{b_l - a_l}\mathbf{D}^l \mathbf{X}^l \overset{!}{=} \mathbf{F}^l(\mathbf{X}^l, \mathbf{U}^l), \quad l = 1, \ldots, N_{\text{int}} \tag{2.55}$$

$$\boldsymbol{x}_1 = \boldsymbol{x}_{\text{start}}$$

$$\boldsymbol{g}(\mathbf{X}^1, \ldots, \mathbf{X}^{N_{\text{int}}}, \mathbf{U}^1, \ldots, \mathbf{U}^{N_{\text{int}}}) \leq 0$$

$$\boldsymbol{x}^{\text{final}} \in \mathcal{X}_{\text{f}}$$

The parameters $w_i^l, \boldsymbol{x}_i^l, \boldsymbol{u}_i^l$ refer to the value at the collocation point $i = 1, \ldots, N_l$ in the interval $l = 1, \ldots, N_{\text{int}}$.

which describe the evolution of the plant over time. This iterative reformulation of the optimization over a shifting prediction horizon is the reason MPC is often referred to as “moving” or “receding” horizon control [23].

If the model is nonlinear, the methodology is referred as Nonlinear Model Predictive Control (NMPC). The continuous prediction horizon is denoted as T (in seconds), while the discrete prediction horizon is denoted with N (dimensionless). The corresponding resolution in the OCP is given as $\Delta t = t_{n+1} - t_n = \frac{T}{N-1}$.

In standard MPC, only the first control input $u(0)$ from the optimized sequence is applied to the plant. At the next sampling instance t_{n+1}, the MPC optimization is recomputed, allowing the controller to compensate for disturbances and model mismatches. This approach relies on a sufficiently accurate system model f. The closed-loop trajectory shown in Figure 2.4 illustrates nine successive MPC iterations, each producing a single control input that is applied to the plant.

Another requirement for solving the OCP is accurate information about the initial state of the problem (2.11) at each time step. This involves precise state estimation based on state measurements of past states [47]. This is assumed to be exact within this thesis.

Generally, one differs among tracking and stabilizing MPC [23]. Tracking MPC describes the control of the state to follow a predefined reference $\boldsymbol{x}_{\text{ref}}$. Stabilizing MPC describes the control of the states to an equilibrium point, that is the subcase of a tracking MPC with $\boldsymbol{x}_{\text{ref}} = 0$.

For the cost functional (2.9) quadratic costs are a common choice. Specifically, the Lagrangian $\mathcal{L}$ for the case of a tracking MPC has the form

$$\mathcal{L}(\boldsymbol{x}(t), \boldsymbol{u}(t)) = (\boldsymbol{x}(t) - \boldsymbol{x}_{\text{ref}}(t))^{\text{T}}\, \mathbf{Q}\, (\boldsymbol{x}(t) - \boldsymbol{x}_{\text{ref}}(t)) + (\boldsymbol{u}(t))^{\text{T}}\, \mathbf{R}\, (\boldsymbol{u}(t)) \tag{2.56}$$

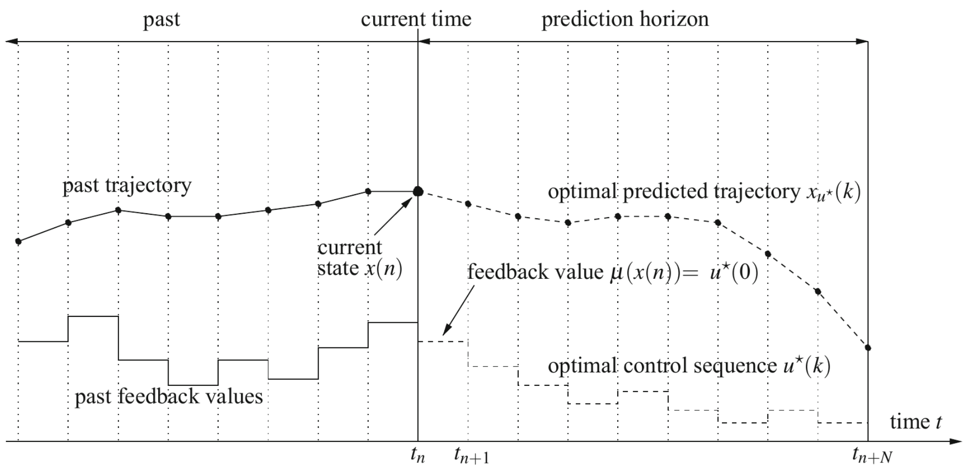

past
current time
prediction horizon
past trajectory
optimal predicted trajectory $x_{u^\star}(k)$
current state $x(n)$
feedback value $\mu(x(n)) = u^\star(0)$
optimal control sequence $u^\star(k)$
past feedback values
time t
t_n
t_{n+1}
t_{n+N}

Figure 2.4.: Moving horizon MPC scheme [23].

with the weighting matrices $\mathbf{Q}, \mathbf{R}$.

2.4.1. Terminal Region

A desirable property of the controller is to guarantee asymptotic stability. The integration of a terminal region and a corresponding terminal cost is a necessary condition to show closed-loop stability in a formal proof. Chen et al. provide a detailed analysis and framework for the determination of the terminal region with a fixed setpoint [11]. Köhler et al. extend the scheme to trajectory tracking [31]. The general form of a terminal region $\mathcal{X}_\mathrm{f}$ with reference tracking is given as

$$\mathcal{X}_\mathrm{f} = \{\boldsymbol{x}(t) \in \mathbb{R}^{n_\mathrm{x}} | (\boldsymbol{x}(t) - \boldsymbol{x}_\mathrm{ref}(t))^\mathrm{T} \, \mathbf{P} \, (\boldsymbol{x}(t) - \boldsymbol{x}_\mathrm{ref}(t)) \leq \alpha\} \tag{2.57}$$

with $\mathbf{P}$ denoting the penalty matrix for the terminal region. With the scaling parameter α it defines an n_x-dimensional ellipsoid as terminal region.

The terminal cost term uses the same metric

$$V_\mathrm{f} = (\boldsymbol{x}(t) - \boldsymbol{x}_\mathrm{ref}(t))^\mathrm{T} \, \mathbf{P} \, (\boldsymbol{x}(t) - \boldsymbol{x}_\mathrm{final}). \tag{2.58}$$

For more information about the stability proof and properties of terminal conditions the reader is referred to the literature, i.e. in [11, 23, 47].

3

Survey of optimal control methods for MPC on a benchmark example

One of the key components of MPC is the underlying OCP. The choice of the direct method to discretize the OCP influences the computational runtime and the control performance significantly.

This chapter compares different methods, namely Multiple shooting (MS), Trapezoidal (TC) and Hermite-Simpson (HS) collocation and the pseudospectral methods Legendre-Gauss-Lobatto (LGL) and Legendre-Gauss-Radau (LGR) on a benchmark example.

The setup of the comparison is described in Section 3.1, the methodology for identifying comparable parameters for each method is explained in Section 3.2 and the closed-loop results are discussed in Section 3.3.

3.1. Setup

3.1.1. Benchmark example: Multi-body system

The survey of collocation methods should be representative for multi-body systems with the general form

$$\mathbf{M}(\boldsymbol{\theta})\ddot{\boldsymbol{\theta}} + \mathbf{C}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}})\dot{\boldsymbol{\theta}} + \boldsymbol{g}(\boldsymbol{\theta}) = \boldsymbol{\tau}, \tag{3.1}$$

with $\mathbf{M}(\theta)$ denoting the mass matrix, $\mathbf{C}(\theta, \dot{\theta})$ denoting the Coriolis-centrifugal matrix and $\boldsymbol{g}(\theta)$ denoting the gravity vector [51]. Externally applied momentum are referred to as $\boldsymbol{\tau}$. Multi-body system dynamics have broad applicability across a wide range of robotic platforms. A particularly relevant domain for this thesis is space robotics, where robotic manipulators are mounted on a spacecraft to execute complex tasks in orbit. Typical operations include on-orbit assembly of large structures, servicing of satellites and active debris removal maneuvers [2, 32, 33, 38]. The unique challenges of these scenarios, such as the absence of a fixed base and the coupling between manipulator motion and spacecraft dynamics, require precise modeling of the underlying multi-body dynamics. These systems can be rigorously described by the joint-space formulation of robot dynamics, as expressed in Eq. (3.1).

For a representative benchmark of general multi-body applications, an inverted pendulum with a varying number of joints is used. Specifically, pendulums with one and three joints in the plane are employed to evaluate different methods. The three-joint pendulum, in particular, is an effective simplification of a robotic manipulator. Although real-world robotics applications mostly incorporate all three dimensions, this example demonstrates the scalability and compatibility of the optimal control methods for general multi-body systems.

In the following, the Euler-Lagrange derivation of equations of motions are derived for the one-joint and three-joint inverted pendulum. Each arm is assumed to have a mass of $m_i = 2.5\,\text{kg}$ and a length of $l_i = 0.2\,\text{m}$.

1-joint inverted pendulum

The inverted pendulum with one joint is shown in Figure 3.1a. The system dynamics are derived with the Euler-Lagrange formulation as in [22]. This leads to the IVP for the single inverted pendulum

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, u) = \begin{pmatrix} \dot{\theta}_1 \\ -\frac{2g}{l}\sin(\theta_1) + \frac{4}{ml^2}\tau_1 \end{pmatrix}. \tag{3.2}$$

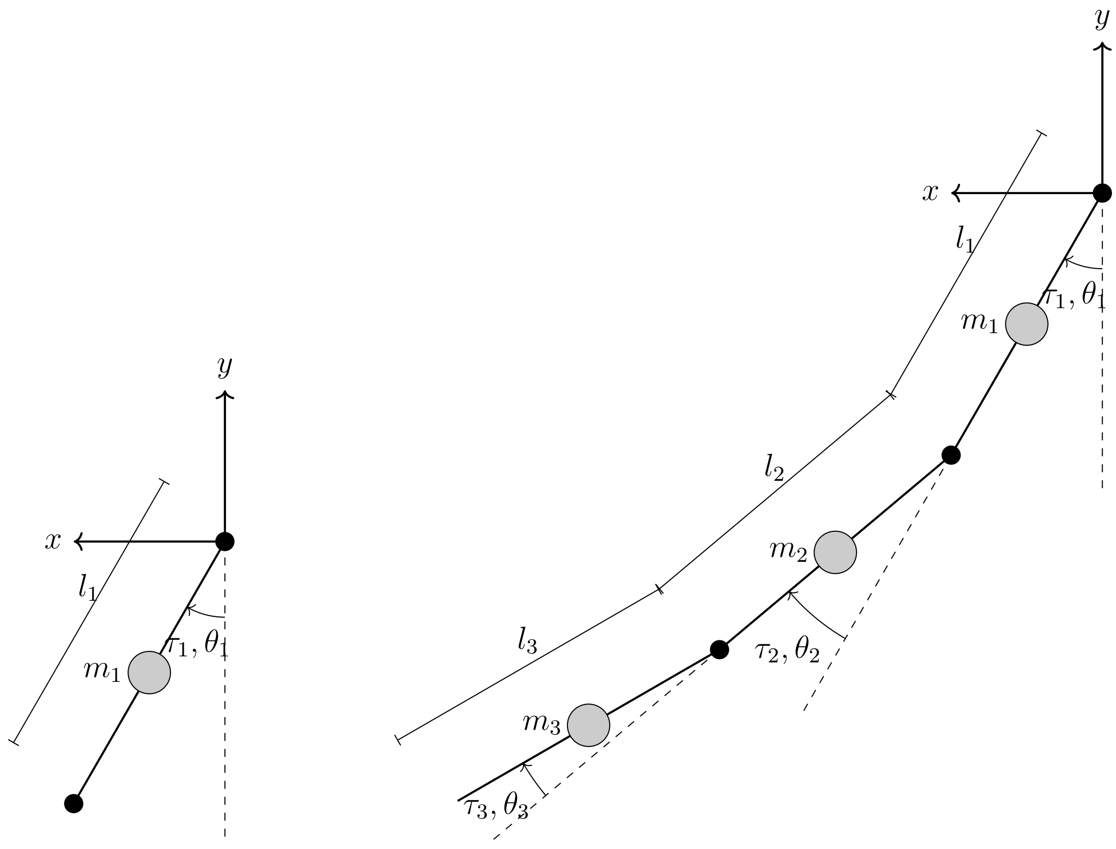
The state and control input is defined as $\boldsymbol{x} = \left(\theta_1, \dot{\theta}_1\right)^{\mathrm{T}} \in \mathbb{R}^2$ and $u = (\tau_1) \in \mathbb{R}$, respectively.

3-joint inverted pendulum

The three-joint inverted pendulum extends the single-joint inverted pendulum model by incorporating two additional arms within the plane. The state space is increased to $\boldsymbol{x} \in \mathbb{R}^6$ and the control input dimension to $\boldsymbol{u} \in \mathbb{R}^3$. Thus, $n_{\mathrm{x}} = 6$ and $n_{\mathrm{u}} = 3$. Due to coupling effects of the different arms the non-linearity is increased.

To derive the equations of motion of a pendulum with 3 joints the Euler-Lagrange formulation is used. A schematic is shown in Figure 3.1b.

The angles θ_i are referenced relative to each other, which is common in robotics. However, they can also be referenced to the vertical plane, resulting in a simplified formulation of the dynamics [34]. Since this example should represent a reduced version of a space manipulator, the more complex, relative referencing of the angles is used.



(a) 1 joint (b) 3 joints

Figure 3.1.: Schematic of the benchmark inverted pendulum.

Following the angle definitions from Figure 3.1b, the kinematic constraints for the x-coordinate for each point mass are given as

$$\begin{aligned}
x_1 &= \sin(\theta_1)\frac{l_1}{2}, \\
x_2 &= \sin(\theta_1)l_1 + \sin(\theta_1+\theta_2)\frac{l_2}{2}, \\
x_3 &= \sin(\theta_1)l_1 + \sin(\theta_1+\theta_2)l_2 + \sin(\theta_1+\theta_2+\theta_3)\frac{l_3}{2}.
\end{aligned}$$

The kinematics for the y-coordinate yield

$$\begin{aligned}
y_1 &= -\cos(\theta_1)\frac{l_1}{2}, \\
y_2 &= -\cos(\theta_1)l_1 - \cos(\theta_1+\theta_2)\frac{l_2}{2}, \\
y_3 &= -\cos(\theta_1)l_1 - \cos(\theta_1+\theta_2)l_2 - \cos(\theta_1+\theta_2+\theta_3)\frac{l_3}{2}.
\end{aligned}$$

The kinetic energy term

$$\mathcal{T} = \sum_{i=1}^{3} \frac{1}{2} m_i (\dot{x}_i^2 + \dot{y}_i^2)$$

and the potential energy

$$\mathcal{U} = \sum_{i=1}^{3} m_i g y_i$$

form the Lagrangian $\mathcal{L} = \mathcal{T} - \mathcal{U}$. The Lagrangian expression for the model derivation should not be confused with the Lagrangian in the cost functional (2.9), since both refer to different expressions. Solving the Euler-Lagrange equation [22]

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial \mathcal{L}}{\partial \dot{\theta}_i}\right) - \frac{\partial \mathcal{L}}{\partial \theta_i} = \tau_i, \quad i = 1, \ldots, 3, \tag{3.3}$$

leads to an ODE of the form $\boldsymbol{f}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}, \ddot{\boldsymbol{\theta}}) = \boldsymbol{\tau}$, which corresponds to the standard manipulator dynamics commonly used in robotics, as shown in Eq. (3.1).

All angles are grouped to the vector $\boldsymbol{\theta} = (\theta_1, \theta_2, \theta_3)^{\mathrm{T}}$. Accordingly, the angular rates and accelerations are grouped to $\dot{\boldsymbol{\theta}}$ and $\ddot{\boldsymbol{\theta}}$, respectively.

The system dynamics of the OCP (2.10) should be of the form of an IVP $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u})$. Rearranging (3.1) leads to

$$\ddot{\boldsymbol{\theta}} = \boldsymbol{f}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}) = \mathbf{M}(\boldsymbol{\theta})^{-1}(\boldsymbol{\tau} - \boldsymbol{g}(\boldsymbol{\theta}) - \mathbf{C}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}})\dot{\boldsymbol{\theta}}) \in \mathbb{R}^3.$$

The matrix $\mathbf{M}$ is extracted from the Euler-Lagrange (3.3) equation by taking the derivative with respect to $\ddot{\boldsymbol{\theta}}$

$$\begin{aligned}
\mathbf{M} &= \frac{\partial}{\partial \ddot{\boldsymbol{\theta}}}\left(\left[\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial \mathcal{L}}{\partial \dot{\boldsymbol{\theta}}}\right)\right] - \frac{\partial \mathcal{L}}{\partial \boldsymbol{\theta}}\right) \\
&= \frac{\partial}{\partial \ddot{\boldsymbol{\theta}}}\left(\left[\frac{\partial}{\partial \boldsymbol{\theta}}\left(\frac{\partial \mathcal{L}}{\partial \dot{\boldsymbol{\theta}}}\right)\dot{\boldsymbol{\theta}} + \frac{\partial}{\partial \dot{\boldsymbol{\theta}}}\left(\frac{\partial \mathcal{L}}{\partial \dot{\boldsymbol{\theta}}}\right)\ddot{\boldsymbol{\theta}}\right] - \frac{\partial \mathcal{L}}{\partial \boldsymbol{\theta}}\right) \\
&= \frac{\partial}{\partial \ddot{\boldsymbol{\theta}}}\left[\frac{\partial}{\partial \boldsymbol{\theta}}\left(\frac{\partial \mathcal{L}}{\partial \dot{\boldsymbol{\theta}}}\right)\dot{\boldsymbol{\theta}} + \frac{\partial}{\partial \dot{\boldsymbol{\theta}}}\left(\frac{\partial \mathcal{L}}{\partial \dot{\boldsymbol{\theta}}}\right)\ddot{\boldsymbol{\theta}}\right].
\end{aligned} \tag{3.4}$$

The Coriolis matrix $\mathbf{C}$ is computed with Christoffel symbols [51]. The gravity vector $\boldsymbol{g}$ is derived by taking the derivative of the potential energy $\boldsymbol{g} = {}^{\partial \mathcal{U}}/_{\partial \boldsymbol{\theta}}$.

The aforementioned relations are combined to obtain a closed-form representation of the system dynamics. The state vector is defined as $\boldsymbol{x} = \left(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}\right)^{\mathrm{T}} \in \mathbb{R}^6$. Note that $\boldsymbol{x}$ represents the state vector and is not to be mistaken for the Cartesian coordinate x. The control input vector is similarly defined as $\boldsymbol{u} = \left(\tau_1, \tau_2, \tau_3\right)^{\mathrm{T}} \in \mathbb{R}^3$. This leads to the IVP

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}) = \begin{pmatrix} \dot{\boldsymbol{\theta}} \\ \mathbf{M}(\boldsymbol{\theta})^{-1}(\boldsymbol{\tau} - \boldsymbol{g}(\boldsymbol{\theta}) - \mathbf{C}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}})\dot{\boldsymbol{\theta}}) \end{pmatrix}. \tag{3.5}$$

3.1.2. Optimal Control Methods: A survey

An overview of the different methods taken into account in the survey can be found in Table 3.1. Multiple Shooting (MS) with Euler integration and identical discretization for the state and control inputs is used as a baseline configuration. An identical multiple shooting method with RK4 integration (MS-RK4) is applied to evaluate the increasing complexity in the NLP with higher-order integration. The Trapezoidal (TC) and Hermite-Simpson (HS) collocation methods are used with equidistant spacing. Pseudospectral methods are applied both globally (LGL, LGR) and on a local domain (LGL-LOC, LGR-LOC). For the local method the interval is divided at one half of the prediction horizon T. Other interval borders or number of intervals are considerable, but are not further analyzed. The accuracy of all collocation methods is aligned with the methodology derived in Section 3.2.1. Consequently, each method has a different number of discretization points N.

Table 3.1.: Overview of the methods used for the survey.

Method	NLP	Description
Multiple Shooting (MS)	2.2	Uniform spacing with Euler-Integration $\boldsymbol{\Phi}^{\mathrm{Euler}}$.
Multiple Shooting RK4 (MS-RK4)	2.2	Uniform spacing with Runge-Kutta order 4 Integration $\boldsymbol{\Phi}^{\mathrm{RK4}}$.
Trapezoidal Collocation (TC)	2.3	Uniform spacing.
Hermite-Simpson Collocation (HS)	2.4	Uniform spacing.
Legendre-Gauss-Lobatto (LGL)	2.5	Spacing according to roots of LGL polynomial.
Local Legendre-Gauss-Lobatto (LGL-LOC)	2.7	Spacing according to roots of LGL polynomial. Two intervals are used: $[0, \frac{1}{2}T]$, $[\frac{1}{2}T, T]$.
Legendre-Gauss-Radau (LGR)	2.6	Spacing according to roots of LGR polynomial.
Local Legendre-Gauss-Radau (LGR-LOC)	2.8	Spacing according to roots of LGR polynomial. Two intervals are used: $[0, \frac{1}{2}T]$, $[\frac{1}{2}T, T]$.

For all methods, control input constraints over the whole prediction horizon are included

$$\boldsymbol{u}_{\min} \leq \boldsymbol{u}_i \leq \boldsymbol{u}_{\max}, \quad i = 1, \dots, N. \tag{3.6}$$

A terminal region $\mathcal{X}_{\mathrm{f}}$ and a terminal cost term V_{f} is identified. Based on the procedure from [31], the terminal region and terminal cost is given in the form of (2.57) and (2.58) and are incorporated in all OCP methods.

3.1.3. Implementation and nonlinear solvers

The survey is implemented in Matlab [41] with the nonlinear optimization and optimal-control framework CasADi [4]. Three different solvers are evaluated in the survey: FATROP [55], IPOPT [57] and CasADi's built-in SQP solver (sqpmethod) [4].

FATROP has the capability to automatically detect and exploit the block sparsity structure of multiple-shooting OCPs [55] resulting in more efficient computations. For MS and MS-RK4 this feature is natively supported. For all collocation methods, FATROP does not natively support the structure detection. FATROP requires an explicit functional mapping from the current state to the next: $x_{i+1} = f(x_i, u_i)$ to exploit the sparsity. This is inherently given with shooting approaches, as seen in Eq. (2.19). Collocation approaches natively implement an implicit strategy resulting in a different pattern. However, for specific collocation methods, the explicit sparsity pattern required by FATROP might be achieved by incorporating additional optimization variables. Adapting the LGR-LOC to FATROP's required block-sparsity is shown in Section 3.4.

For the SQP solver, the quadratic solver OSQP [52] with exact Hessian approximation is used.

To analyze the computational performance, mex-files are generated with CasADi's code generation capability. This significantly improves the computational runtime. The *expand* option is disabled for all methods as tests have shown that enabling it increases runtime. This is likely due to the conditional statements.

Unless otherwise stated, default values are used.

3.2. Methodology and assumptions for the comparative analysis

This chapter explains the methodology for the survey. At first, in an open-loop manner the cumulated error to a reference solution of each method is aligned. As an output, each method is parameterized by the number of discretization points (prediction horizon) N. The identified methods are then compared on different closed-loop trajectories to assess performance and computation time.

Nomenclature

Throughout the chapter, the set $\mathcal{M}$ refers to all OCP methods

$$\mathcal{M} = \{\text{MS, MS-RK4, TC, HS, LGL, LGR, LGL-LOC, LGR-LOC}\}.$$

The set $\tilde{\mathcal{M}}$ denotes all collocation methods, so excluding MS and MS-RK4.

Four different indices exist to distinguish between the i-th discretization point in the j-th open-loop trajectory in the overall k-th closed-loop experiment

$$
{}^{j}_{k}\boldsymbol{x}^{m}_{i} \quad \text{with} \begin{cases} m \in \mathcal{M} \text{ or } \tilde{m} \in \tilde{\mathcal{M}} & \text{OCP-method or collocation-cethod,} \\ i \in \{1, \ldots, N^{m}\} & i\text{-th discretization point,} \\ j \in \{1, \ldots, N_{\mathrm{OL}}\} & j\text{-th open-loop iteration,} \\ k \in \{1, \ldots, N_{\mathrm{CL}}\} & k\text{-th closed-loop experiment.} \end{cases}
$$

The number of open-loop or closed-loop experiments is denoted with N_{OL} and N_{CL}, respectively.

To give illustration of the above nomenclature, one can think of the indices i, j, k have increasing order when going from inner to outer loops of the pipeline. Assume having 5 closed-loop experiments $k_{\max} = 5$ consisting out of $j_{\max} = 100$ MPC-iterations each and a number of nodes in the MPC of $i_{\max} = 10$. The state ${}^{1}_{7}\boldsymbol{x}^{m}_{10}$ would then describe the state value of the final point in the prediction horizon in the first open-loop iteration in the experiment 7.

3.2.1. Open-loop error estimation and parameter finding

A difficulty when comparing different optimal control methods is to find a comparable setup. Using a similar number of nodes for the different methods does not produce similar results, since each method is mathematically constructed in a fundamental different way. The idea in this analysis is to find a balanced configuration that facilitates comparable performance of the methods.

Strategy for alignment of collocation methods to MS

The number of discretization variables $N^{\tilde{m}}$ of each method $\tilde{m} \in \tilde{\mathcal{M}}$ is adapted to achieve comparable cumulative errors to MS with Euler integration in different open-loop simulations.

The above criteria is evaluated on different open-loop trajectories by sampling $N_{\mathrm{OL,init}}$ initial conditions. The detailed pipeline is given in the following:

1. **Sample initial condition**: An initial condition is sampled from the terminal region $\boldsymbol{x}_{\mathrm{init}} \sim \mathrm{uniform}(\mathcal{X}_{\mathrm{f}})$ assuming uniform distribution. Sampling from within the terminal region ensures a feasible optimization problem. However, by using this approach, different convergence properties when starting from outside the terminal region are neglected. To account for different trajectories, the following steps are repeated for $N_{\mathrm{OL,init}}$ different initial conditions.

2. **Generate reference solution**: A reference solution $\boldsymbol{x}^{\mathrm{ref}}, \boldsymbol{u}^{\mathrm{ref}}$ is generated by solving the open-loop OCP with a finer discretized MS-RK4 method with $N^{\mathrm{ref}} = 10N^{\mathrm{MS}}$ uniformly spaced shooting points.

3. **Error estimation of methods to reference**: The squared error is calculated by measuring the weighted difference between the method's results and the reference solution at each discretization point of the latter

$$e^m_{i_{\mathrm{ref}}} = (\boldsymbol{x}^m(t^{\mathrm{ref}}_{i_{\mathrm{ref}}}) - \boldsymbol{x}^{\mathrm{ref}}_{i_{\mathrm{ref}}})^{\mathrm{T}}\mathbf{Q}(\boldsymbol{x}^m(t^{\mathrm{ref}}_{i_{\mathrm{ref}}}) - \boldsymbol{x}^{\mathrm{ref}}_{i_{\mathrm{ref}}}), \quad i_{\mathrm{ref}} = 1, \dots, N^{\mathrm{ref}}. \tag{3.7}$$

   The weighting matrix $\mathbf{Q}$ is chosen equivalently to the weighting matrix in the cost functional (2.56). The deviation of the control input is not considered for the open-loop parameter fitting, since it is sensitive to the discretization of the different methods. However, it is taken into account in the closed-loop cost measure Π (see Section 3.2.2).

   For MS and MS-RK4 the evaluation of $\boldsymbol{x}^m(t^{\mathrm{ref}}_{i_{\mathrm{ref}}})$ requires linear interpolation between the optimization variables $\boldsymbol{x}^{\mathrm{MS}}_i$ and $\boldsymbol{x}^{\mathrm{MS}}_{i+1}$. This is illustrated in Figure 3.2. For all collocation methods $\tilde{m} \in \tilde{\mathcal{M}}$, a continuous state trajectory is inherently

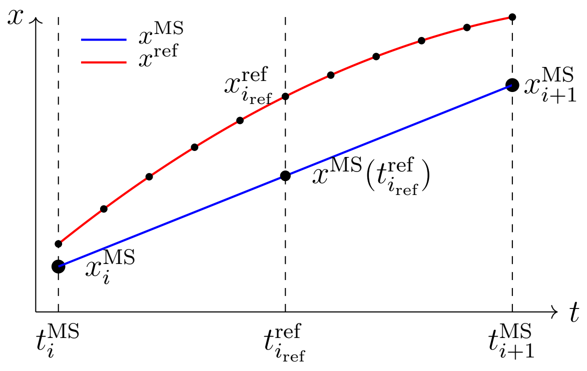

x
t
x^{MS}
x^{ref}
$x^{\mathrm{ref}}_{i_{\mathrm{ref}}}$
x^{MS}_{i+1}
$x^{\mathrm{MS}}(t^{\mathrm{ref}}_{i_{\mathrm{ref}}})$
x^{MS}_i
t^{MS}_i
$t^{\mathrm{ref}}_{i_{\mathrm{ref}}}$
t^{MS}_{i+1}

Figure 3.2.: Illustration of the error estimation of the MS method to the reference.

   provided, so the functions can directly be evaluated.

   The cumulated error is then computed by

$$e^m = \sum_{i_{\mathrm{ref}}=1}^{N^{\mathrm{ref}}} e^m_{i_{\mathrm{ref}}}.$$

4. **Parameter finding for the collocation methods**: To align with the accuracy of the shooting method, the cumulated error for each collocation method is required to be less than or equal to 10 % of the cumulated shooting error:

$$\frac{|e^{\tilde{m}} - e^{\mathrm{MS}}|}{e^{\mathrm{MS}}} \leq 0.1, \quad \forall \tilde{m} \in \tilde{\mathcal{M}}. \tag{3.8}$$

Note here, that the terms $e^{\tilde{m}}$ refer to the cumulated error of the collocation method $\tilde{m} \in \tilde{\mathcal{M}}$ to the reference. The term e^{MS} denotes the cumulated error of the MS method to the reference. Consequently, Eq. (3.8) defines a measure requiring the relative error of each collocation method to be in the range of 10 % around the cumulated shooting error to the reference.

The reasoning for defining a threshold and not strictly better open-loop accuracy than the MS method is the fact that for small cumulative errors, all collocation methods would require at least as many points as MS. Specifically, the first control inputs have high influence on the trajectory. If shooting methods have a relative large N^{MS}, a small Δt is the consequence. To reproduce the same qualitative behavior with collocation methods, a similar Δt is required, which leads to a similar number of nodes.

Starting with a minimal number of discretization points $N^{\tilde{m}}$, the grid is refined step-wise by $N^{\tilde{m}} \leftarrow N^{\tilde{m}} + 1$ until the criteria (3.8) is fulfilled. For TC and HS, equidistant spacing is assumed when inserting new nodes. For the pseudospectral methods LGL and LGR the collocation points for the increased number of nodes are derived by finding the roots of a higher order Legendre polynomial. The local LGL-LOC and LGR-LOC (that also retrieve the new points by the higher order orthogonal polynomial) increase the number of points simultaneously within each subinterval. This way an identical number of points within each segment is ensured. The refinement is repeated until the criteria (3.8) is fulfilled.

5. **Result of parameter finding**: All collocation methods have a fixed OCP with a given discretization $N^{\tilde{m}}$. This parametrization is used for the closed-loop analysis in the next section.

As described earlier, the MS method serves as the baseline, with its time step Δt determining the frequency of each closed-loop MPC iteration. All collocation methods operate at identical sampling frequency. However, when a method employs a finer discretization of the control input than the MS method, numerical instabilities may arise. This is because control inputs defined before t_2^{MS} cannot be considered within the closed-loop framework (t_1 is equivalent by constraint for all methods). Therefore, for methods requiring a finer temporal resolution at the beginning, the control input is held constant at the initial value. In the following, this is referred to as the

$$\text{MPC-constraint} \qquad \boldsymbol{u}_i^{\tilde{m}} = \begin{cases} \boldsymbol{u}_1^{\tilde{m}}, & \text{if } t_i^{\tilde{m}} < t_2^{\text{MS}} \\ \boldsymbol{u}_i^{\tilde{m}} & \text{else} \end{cases} \forall i = 2, \dots, N^{\tilde{m}}. \tag{3.9}$$

This constraint is particularly likely to be required by pseudospectral methods. Due to their spacing of points according to the roots of a Legendre-polynomial, a dense distribution of nodes is at the beginning of the interval (see Figure 2.1). This leverages a finer control input discretization compared to MS making the constraint (3.9) active for the first nodes. A detailed discussion about the choice of control inputs for multi-interval pseudospectral methods is found in [5].

3.2.2. Closed-loop comparison and performance metrics

The parameterized methods are now evaluated for the computation time and cost on closed-loop simulations. The following scheme is applied:

1. **Sample initial condition**: Sample N_{CL} initial conditions uniformly distributed within the terminal region
$$ {}^{1}_{k}\boldsymbol{x}^{m}_{1} = \boldsymbol{x}(t=0) \sim \text{uniform-}(\mathcal{X}_{\mathrm{f}}), \qquad \forall k = 1, \ldots, N_{\mathrm{CL}}. $$
2. **Compute the closed-loop trajectory**: Compute the closed-loop trajectories for all methods $m \in \mathcal{M}$ for a fixed simulation time T_{sim}. The sampling frequency for the MPC is given for all methods by Δt^{MS}, that is the time discretization of the shooting method.

   For the closed-loop simulation a RK4 method is used to propagate in time. The time-step increment is refined by a factor of ten relative to the MPC sampling period, i.e., $\Delta t_{\mathrm{sim}} = 1/10 \cdot \Delta t^{\mathrm{MS}}$.

   No disturbances are considered for the closed-loop simulation. In each closed-loop trajectory, the solution from the previous iteration is used as a warm-start guess for all methods and solvers. As a cold start for a new closed-loop experiment k, the state is set constant to ${}^{1}_{k}\boldsymbol{x}^{m}_{1}$ and the control input to zero.
3. **Compute the metrics**: The following measures are used to compare the different methods:
   - For each open-loop iteration: Computation time ${}^{j}_{k}t^{m}_{\mathrm{comp}}$ for the solver to generate a solution. This is measured with the Matlab command tic-toc. In total, there are $N_{\mathrm{CL}}N_{\mathrm{OL}}$ samples for the computational time.
   - For each closed-loop trajectory: The closed-loop cost Π is evaluated by
$$ {}_{k}\Pi^{m} = \sum_{j=1}^{N_{\mathrm{OL}}} ||{}^{j}_{k}\boldsymbol{x}^{m}_{1} - \boldsymbol{x}_{\mathrm{final}}||^{2}_{\mathbf{Q}} + ||{}^{j}_{k}\boldsymbol{u}^{m}_{1}||^{2}_{\mathbf{R}}, \qquad \forall k = 1, \ldots, N_{\mathrm{CL}}, \tag{3.10} $$
     assuming equivalent weighting matrices as for the cost functional. The notation $|| \cdot ||^{2}_{\mathbf{M}}$ refers to the quadratic norm similarly to (2.56) with weighting matrix $\mathbf{M}$. The index measures the closed-loop cost and only takes the first state and control input of each open-loop trajectory into account. This results in N_{CL} closed-loop cost samples.
4. **Repeat for all initial conditions** Repeat above steps for all sampled initial conditions $k = 1, \ldots, N_{\mathrm{CL}}$.

3.3. Results

3.3.1. Open-loop parameters

The simulations on the three-joint inverted pendulum are performed with different configurations for the baseline method MS. Table 3.2 summarizes the combinations. For each configuration an acronym is introduced. The first two digits indicate the prediction horizon T and the last two digits the prediction horizon steps N, i.e.

$$\text{Configuration} \quad \underbrace{01}_{T=0.1\,\mathrm{s}} \; \underbrace{10}_{N=10} .$$

The focus is placed on maintaining a constant sampling interval Δt. As the prediction horizon N is extended, the resulting NLP becomes more complex. By increasing N while keeping Δt fixed, the total prediction time into the future, T, is adjusted accordingly. The weighting matrices $\mathbf{Q}$ and $\mathbf{R}$ are kept unchanged throughout. Greater emphasis is placed on deviations in angular positions compared to angular velocities. Meanwhile, the control input weights in $\mathbf{R}$ are reduced to ensure satisfactory closed-loop performance. Actuator constraints of the pendulum are considered and a terminal region in the form of (2.57) is included in the NLP.

Table 3.2.: Prediction horizon and sampling frequency for three-joint inverted pendulum.

Config.	T	N	Δt	Weighting matrices	Constraints
0110	0.1 s	10	0.011 s	$\mathbf{Q} = \mathbf{I}^{6\times6}$ with $Q_{1,1} = Q_{3,3} = Q_{5,5} = 10$	$\lvert u_i\rvert \leq 30\,\mathrm{Nm}$,
0330	0.3 s	30	0.011 s	$\mathbf{R} = \mathbf{I}^{3\times3}/2500$	$\forall i = 1, \dots, 3$
0550	0.5 s	50	0.011 s		

With the procedure from Section 3.2.1, open-loop parameters for all collocation methods are estimated. For this, five different initial conditions are tested, $N_{\mathrm{OL,\,init}} = 5$.

Table 3.3 provides a comprehensive summary of all estimated parameters. For configuration 0110, the collocation methods HS, LGL, and LGR require fewer discretization points compared to the MS method. The TC method is of lower order than HS, resulting in 11 points. The two local orthogonal collocation methods, LGL-LOC and LGR-LOC, demand a higher number of collocation points relative to MS. It is important to note that within each subinterval, a minimum number of points is necessary to ensure an adequately precise and smooth trajectory, particularly to maintain continuity between intervals.

An examination of the configurations 0330 and 0550 demonstrate the efficiency of the collocation methods. In particular, pseudospectral methods (PS) capitalize on their spectral convergence properties [15]. While MS employs 50 discretization points, the global methods LGL and LGR require at most 20 points. Equidistant collocation methods such as TC and HS also utilize fewer points than MS, although their scaling performance is inferior to that of pseudospectral methods.

Table 3.3.: Parameter overview derived from the open-loop schematic.

Config.	N^{MS}	$N^{\text{MS-RK4}}$	N^{TC}	N^{HS}	N^{LGL}	$N^{\text{LGL-LOC}}$	N^{LGR}	$N^{\text{LGR-LOC}}$
0110	10	10	11	6	9	14	8	12
0330	30	30	23	13	10	22	16	22
0550	50	50	39	21	13	28	20	30

3.3.2. Computation time

The computation time for the methods summarized in Table 3.1 are shown in Figure 3.3 for $N_{\text{CL}} = 10$ closed-loop simulations. Each closed-loop simulation is propagated for two seconds, $T_{\text{sim}} = 2\,\text{s}$, which is sufficient to achieve convergence. An MPC iteration is executed every $\Delta t = 0.011\,\text{s}$, which results in $N_{\text{OL}} = 181$ open-loop iterations. The sampling time for the simulation is refined by the factor ten, resulting in $\Delta t_{\text{sim}} = 0.0011\,\text{s}$.

The boxes show the median, 5 % and 95 % quantiles, as well as the minimum and maximum values. The gray crosses indicate the computation time of the first solution in each closed-loop trajectory, without warm start information from the previous iteration. They are not considered within the statistical evaluation. The top row shows the number of parameters used within each method, that is evaluated with the methodology from Section 3.2.1. The analysis is grouped by remarks followed by a more detailed explanation.

Remark 3.3.1

FATROP with MS is computationally most performant for prediction horizons $N \leq 50$ for the analyzed benchmark example.

For all solver and method combinations, MS with FATROP achieves the lowest maximum value for the computation time. Specifically for a relatively short prediction horizon, FATROP is most performant. With increasing prediction horizon, the LGL method performs similar.

Comparing MS-FATROP with MS-IPOPT, one can observe a performance boost with FATROP, that is mainly rooted in the structure detection algorithm to exploit the block sparsity [55]. Considering only the IPOPT solver, collocation methods perform similar as shooting with the tendency of becoming computational more feasible the longer the prediction horizon.

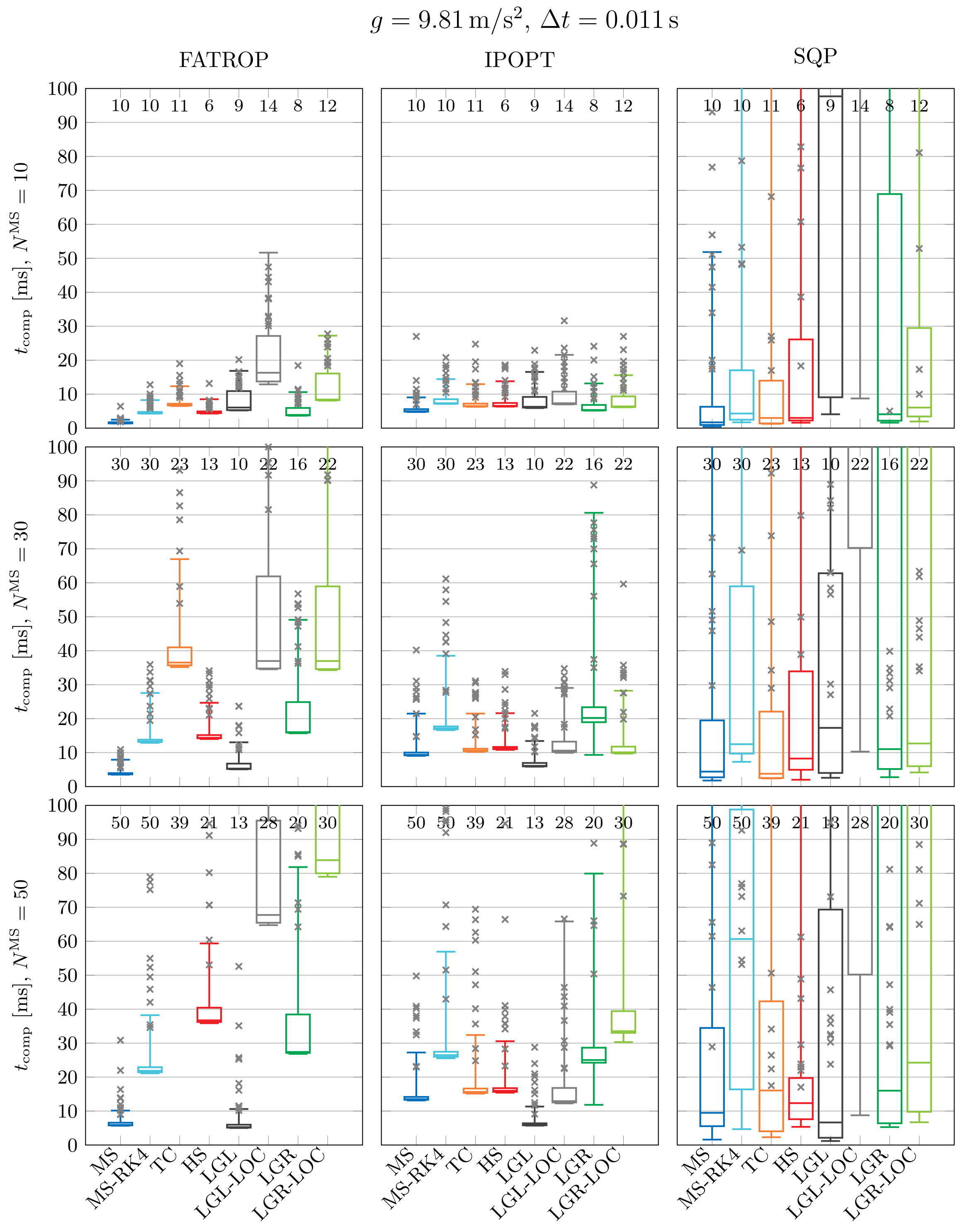


Figure 3.3.: Computation time for the three-joint inverted pendulum. The gray crosses indicate the first solution of each closed-loop trajectory. The box plot shows the median, 5 % and 95 % quantiles and the corresponding minimum and maximum values. The top-row of each plot indicates the number of discretization parameters for each method.

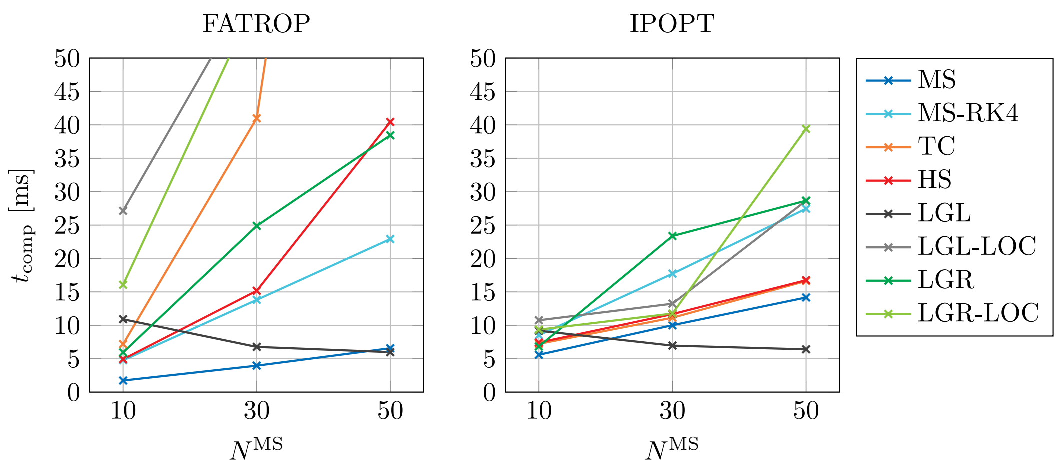


Figure 3.4.: Computation time scaling (95 %-quantile) for the different methods over increased N^{MS} on the three joint inverted pendulum.

The SQP solver achieves large variances in the computation time. The nonlinear system dynamics require a larger number of iterations for to achieve convergence. A more detailed discussion is found in Remark 3.3.7.

Remark 3.3.2

FATROP's structure detection has positive effect on shooting methods outperforming other solver combinations.
The preferred choice for collocation methods with sparse problem structure is IPOPT (assuming natively non-compatibility with FATROP's structure detection).

Figure 3.4 shows the 95 %-quantiles of each method over increasing prediction horizon, both for FATROP and IPOPT. Because of large variances of the SQP solver, it is excluded from this analysis. Comparing the plot for FATROP with IPOPT one can clearly see the three main effects.

The first effect is the performance boost when using shooting methods with FATROP. Specifically the low order integration scheme RK1 significantly improves the computation time. For the different combinations of N, the MS-RK4 method, solved with FATROP, is slower by the factor between 2.75 ($N = 10$) to 3.5 ($N = 30, 50$) compared to MS. If the relative difference in computation time stagnates at 3.5 for increasing N has to be further evaluated for longer prediction horizons. Nevertheless, a higher-order integration scheme significantly increases the computation time.

The second effect is the loss of performance when using FATROP with sparse NLPs that are not supported for structure detection. Comparing the gradient in computation times of TC, HS, LGL-LOC and LGR-LOC for FATROP and IPOPT, the latter achieves significantly improved scaling properties. All those methods have a sparse constraint Jacobian, as shown in Figure 2.3 for the local pseudospectral methods, that is not exploited by FATROP. In contrast, the computation time even increases. FATROP should consequently only be used for problems that are compatible to its structure detection algorithm. Under specific assumptions, implicit OCP formulation can be adapted to align with the sparsity pattern, what is shown in Section 3.4. Without adapting the OCP to this required structure, IPOPT is the more performant solver that can universally be applied.

The third effect is the similarity of computation time for non-sparse NLPs. LGL or LGR, with a relatively dense constraint Jacobian (see Figure 2.3), qualitatively scale comparable for both solvers.

Remark 3.3.3

Considering the benchmark example, global PS methods scale for longer prediction horizons better in terms of computation times than shooting methods

A good property of global pseudospectral methods is their spectral convergence properties [15, 54]. The computation time for LGL methods with increasing prediction horizon remain on a comparable level. Reason for this is, that for $N^{\mathrm{MS}} = 10$ and $N^{\mathrm{MS}} = 50$, the collocation points for LGL only increase from $N^{\mathrm{LGL}} = 9$ to $N^{\mathrm{LGL}} = 13$.

Apart from this high-level observation, a contradiction in the computation time is identified. With increasing number of optimization variables, the computation time for the LGL method decreases. This is exhibited in Figure 3.4. One reason for this odd behavior is the MPC-constraint (3.9) that ensures that no control input is applied in between two sampling points of the closed-loop MPC. For $N = 10$, this constraint is active, what artificially increases the complexity of the NLP resulting in degraded computation time. For $N = 30$ and $N = 50$ this constraint is not active.

To some extent, pseudospectral methods might outperform shooting methods with increasing prediction horizon steps $N \geq 50$. In the context of MPC, only a limited number of prediction horizon steps can be considered due to computational constraints. Consequently, it depends on the specific parametrization, model and accuracy requirements, if pseudospectral methods outperform the standard shooting approach. As a general trend, the more accurate the solution trajectory and the longer the prediction horizon, the better collocation performs.

Remark 3.3.4

Collocation points by LGL and LGR yield different accuracies on the benchmark example.

If one compares the mathematical formulation of LGL (2.44) and LGR (2.48), the two methods mainly differ in the choice of the collocation points, that are derived from the roots of a different polynomial. This can effectively result in differing accuracies and computation times. Where LGL only required $9, 10, 13$ points, LGR required $8, 16, 20$ points to achieve comparable accuracy with the methodology described in Section 3.2.1. Consequently, the finer the discretization of each method, the larger the computation time. Assuming the same number of discretization points for each method, the computation times are comparable [26]. Which method performs better for given system dynamics might be problem dependent.

Remark 3.3.5

HS and TC achieve similar computation times for similar accuracy on the benchmark example.

Considering the computation times of the collocation methods TC and HS (see Figure 3.3 and 3.4) the computation time and its corresponding variances are comparable. While TC requires more discretization points which introduces more complexity in the NLP, HS suffices fewer discretization points but has inherently an overall more complex NLP due to the higher order polynomial. However, as discussed in the next Section 3.3.3, HS provides better closed-loop performance making it the preferable choice among the two.

3.3.3. Closed-loop performance

Figure 3.5 shows the control cost for each collocation method and solver. More precisely, the normalized deviation from the corresponding method $m \in \mathcal{M}$ to the MS method $(\Pi^m - \Pi^{\mathrm{MS}})/\Pi^{\mathrm{MS}}$ is shown. Consequently, positive numbers represent a worse closed-loop behavior, while negative numbers indicate an improved trajectory. A detailed explanation for the closed-loop cost index is given in Section 3.2.2. All examined solvers—FATROP, IPOPT, and SQP—converged to comparable minima, differing only within the bounds of numerical rounding errors, thereby yielding essentially equivalent closed-loop cost.

Remark 3.3.6

The smaller N, the more variances occur in the closed-loop costs of the different methods.

For a discretization level of $N = 10$, the worst-case closed-loop performance of the collocation methods matched that of the MS method. The median performance outperformed approximately 10 to 20 % relative to the baseline MS approach. For increasing horizon, the closed-loop cost of the different methods align with each other.

This observation suggests that the discretization levels determined using the methodology outlined in Section 3.2.1 are conservative. In fact, a reduced number of discretization points would suffice for the collocation methods to achieve parity with the MS method in terms of closed-loop cost. Nevertheless, this would also incorporate sample trajectories to perform significantly worse than the MS method due to the variances. As a summary, collocation method do not generally perform better than shooting, but are rather pa-

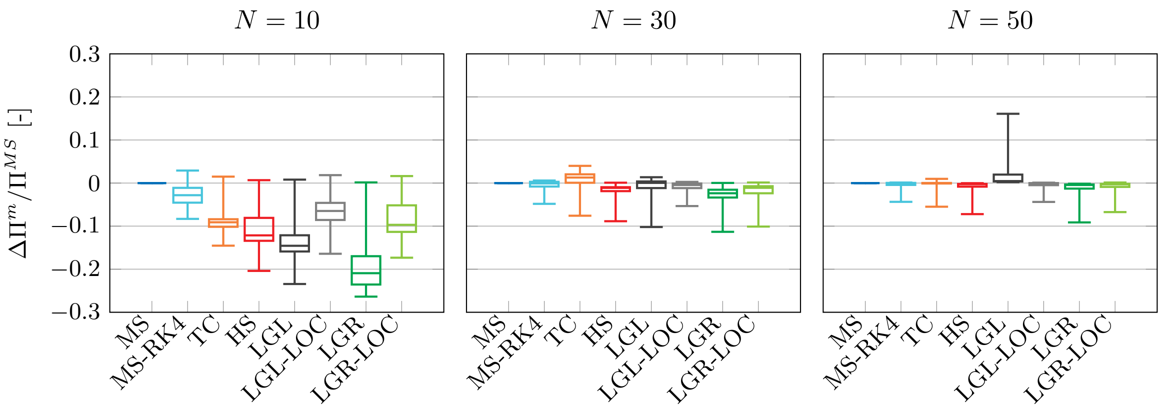


Figure 3.5.: Normalized closed-loop cost of each method to the baseline multiple shooting method. The median, 25- and 75% quantiles and the min. and max. values are shown for $N_{\text{CL}} = 10$ closed-loop trajectories.

rameterized more conservatively by having more optimization variables than potentially required for achieving similar accuracy as MS.

An exemplary closed-loop trajectory for one initial condition is given in Figure 3.6. For the angular positions, the deviation from the final point $\Delta\boldsymbol{\theta} = \boldsymbol{\theta} - \boldsymbol{\theta}_{\text{final}}$ is used. This plot compares the MS method with the global pseudospectral LGL method. The position states θ_1 and θ_2 outline minimal differences between the two methods. However, for the third joint θ_3, the collocation method demonstrates a faster convergence towards zero compared to the MS method. At the end of the simulation, $t_{\text{sim}} = 2\,\text{s}$, the LGL method achieves a position approximately $1.8\,^\circ$ closer to the vertical than the MS method. Given that the angular positions are assigned greater importance through the weighting matrix $\mathbf{Q}$ (see Table 3.2), this improvement primarily accounts for the enhanced closed-loop performance observed with the LGL method.

The angular velocities $\dot{\theta}_1, \dot{\theta}_2, \dot{\theta}_3$ mostly align with each other. Only at the beginning of the trajectory the collocation methods tend to oscillate, which is further discussed in Remark 3.3.8. According to the observed behavior for the angle θ_3 with LGL, the rate $\dot{\theta}_3$ is constantly on a higher level than for MS.

With increasing number prediction horizon steps ($N = 30, N = 50$), the closed-loop cost in Figure 3.5 of the different methods have reduced variances. Since more optimization variables are involved in the OCP, less variance is achieved for the different methods.

Remark 3.3.7

Assumption: Casadi's built-in SQP with the quadratic solver OSQP.
The SQP computation times are sensitive to the magnitude of change in the states per iteration. FATROP's computation times remain on a constant level.

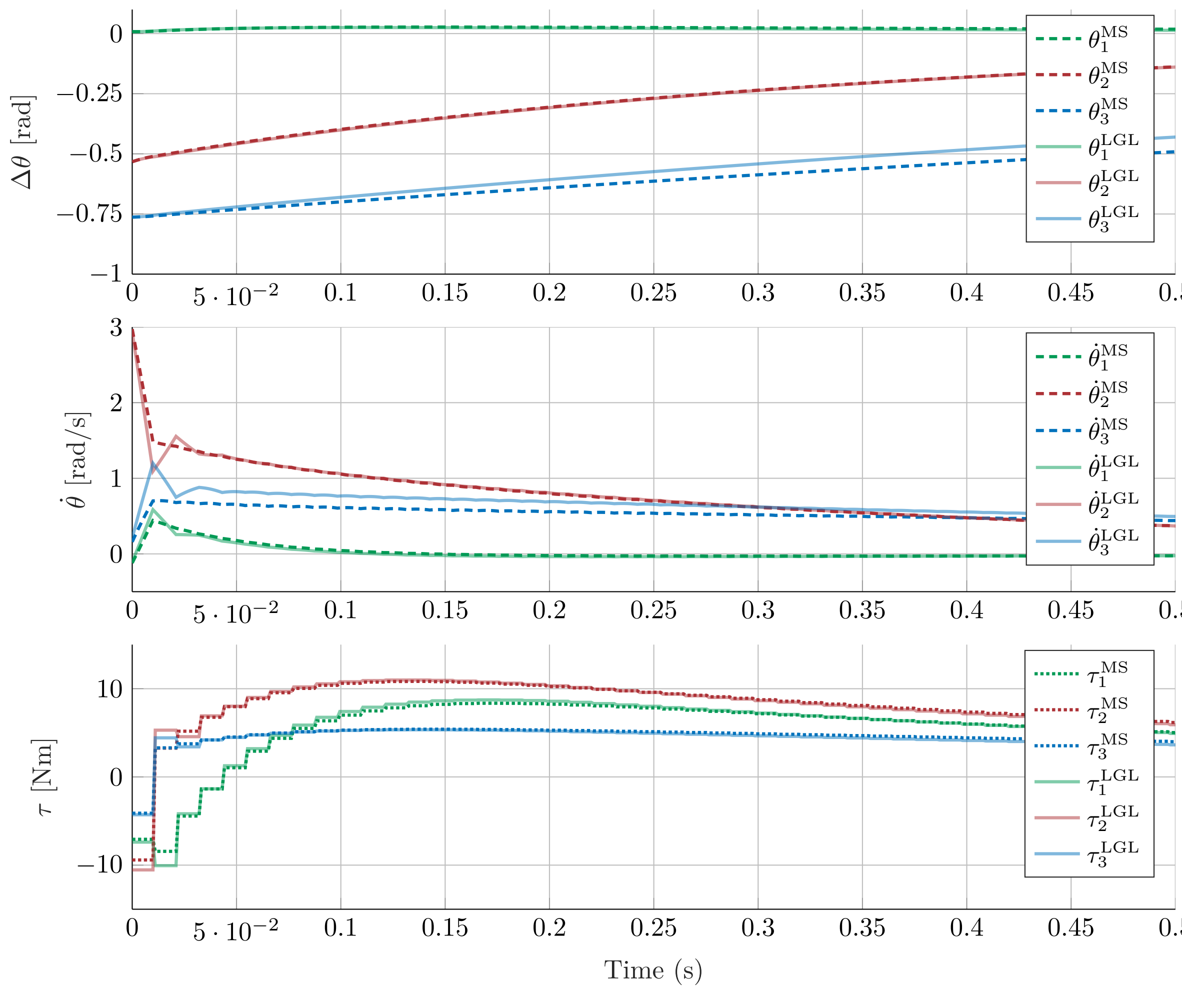


Figure 3.6.: Closed-loop state and control input trajectory for the MS and the LGL method for $N^{\mathrm{MS}} = 10$. For θ, the deviation $\Delta\theta$ to the final point is plotted.

Figure 3.7 compares the computation times for each solver over a closed-loop iteration, consisting out of $N_{\mathrm{OL}} = 187$ open-loop computations. Apart from the computation times, the state trajectory is plotted for all six different states.

The computation time of SQP clearly correlates with the magnitude of state changes per iteration. This is particularly evident at the beginning of the closed-loop simulation, where frequent control updates and significant state variations result in noticeably longer solve times. Toward the end, as the system approaches the final setpoint and state changes diminish, the computation time decreases accordingly and remains mostly constant.

In contrast, FATROP exhibits a more stable computational profile, largely unaffected by the extent of state variation. Although increased dynamics can lead to slightly higher computation times in some closed-loop scenarios, the overall trend remains comparatively

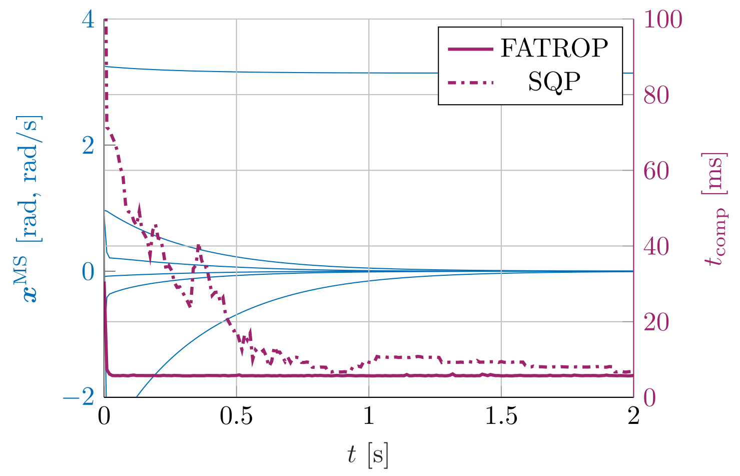


Figure 3.7.: Computation times of FATROP and SQP for a single closed-loop trajectory for the MS method. The corresponding state evolutions for the six states are shown on the left y-axis, while the computation time is shown on the right y-axis.

consistent.

This observation aligns with Figure 3.3. The column for SQP has significantly larger variances in the computation time than FATROP or IPOPT. Reason for this behavior are the nonlinear dynamics, that require SQP to have a large number of iterations.

3.3.4. Influence of the trajectories' gradient on the open-loop prediction

Within the following section, the gradient of the solution trajectory is referred to as the time-derivative of the solution trajectory $\frac{\partial \boldsymbol{x}}{\partial t} = \nabla_t \boldsymbol{x}$.

Remark 3.3.8

The smaller the gradient ${}^{\partial \boldsymbol{x}}/_{\partial t}$ of the solution trajectory, the better collocation methods outperform shooting methods. Rapidly varying dynamics lead to oscillations with collocation methods in the open-loop trajectories.

A computed open-loop trajectory for the first iteration of the closed-loop simulation (Figure 3.6) is illustrated in Figure 3.8. For better visibility, only a subset of the methods (MS, TC, HS and LGL) and of the states (joint three: $\theta_3, \dot{\theta}_3$) is shown. A reference solution, generated by MS with RK4 and $N^{\mathrm{ref}} = 10 \cdot N^{\mathrm{MS}}$, is plotted as a dashed line.

All collocation methods can accurately follow the angular trajectory θ_3 without causing oscillations. The gradient ${}^{\partial \boldsymbol{x}}/_{\partial t}$ is moderate and the solution trajectories' gradient is

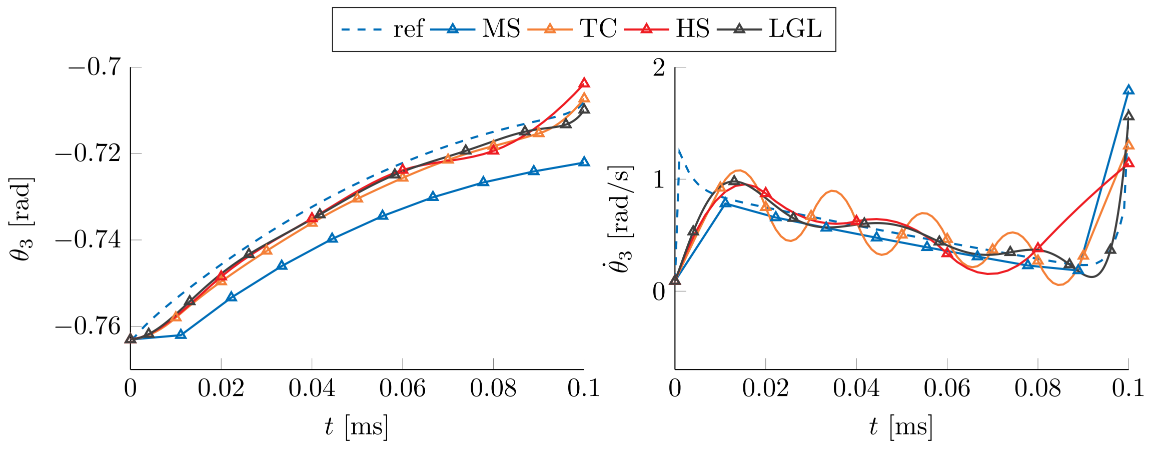


Figure 3.8.: Open-loop trajectory for the joint three of the inverted pendulum. On the left the angular evolution and on the right the angular right evolution is shown for the config 0110.

moderate and overall on a comparable level.

The angular rate trajectory $\dot{\theta}_3$ is shown on the right in Figure 3.8. A rapidly changing angular velocity, so a high gradient $\partial\dot{\theta}_3/\partial t$ is identified at the beginning of the open-loop interval. This results in oscillating solutions for all collocation methods while the shooting method does not reveal this characteristic.

The gradient for the angular position $\partial\theta_3/\partial t$ is mostly on a constant level and changes only gradually over the prediction horizon. The angular velocities' gradient $\partial\dot{\theta}_3/\partial t$ is rugged and non-constant, specifically at the very beginning and the end of the interval. Reason for this effect is rooted in the discretization of the control inputs. As identified in Eq. (3.5), the control inputs directly affect the state derivative $\dot{\boldsymbol{x}}$.

When working on discrete systems, the control inputs are often assumed to be constant over one control interval (see Section 2.4). Thus, the control input changes rapidly from one iteration to another leading to an abrupt change in the state derivative, according to Eq. (3.5).

The problem for collocation methods with rapidly changing state derivatives is the fact, that the solution trajectory is generated by constraining the polynomials' derivative to be equivalent to the system dynamics at collocation points. Assuming a rugged state derivative function as a result of applying piecewise-constant control inputs, makes the polynomial approximation inaccurate. As a result, oscillations occur.

Another caveat comes along with the oscillating behavior. Constraints can only be assured at discretization points. For MS, that assumes linear interpolation between the states, the constraint cannot be violated in between, if it is fulfilled at the nodes. In contrast, since collocation uses a polynomial, the oscillating behavior might result in the non-fulfillment of the constraints along the predicted trajectory. A workaround for polynomials to estimate its safety bounds is given in [3].

Adaptive methods offer an alternative approach to mitigate oscillations [3, 12, 17, 35, 36, 45]. These methods estimate the error using a predefined metric, compare it against a threshold, and then adjust the mesh for the next iteration — either by modifying the polynomial order or the step size. Local pseudospectral methods are one way to incorporate rapidly changing system dynamics within a pseudospectral framework. Their interval borders can be chosen according to the highly changing state derivates. However, for the specific example of an inverted pendulum, different parameter combinations for local pseudospectral methods could not outperform global PS methods. This conclusion aligns with the analysis performed in [27].

Remark 3.3.9

Collocation methods: The closed-loop trajectory is mainly affected by the oscillations at points with large change in magnitude of the control inputs for the benchmark exapmle.

The open-loop plot in Figure 3.8 is the first iteration for the closed-loop simulation illustrated in Figure 3.6. The oscillatory open-loop behavior for collocation methods, explained in Remark 3.3.8, is also visible in the closed-loop simulation.

For the LGL method, the large change in the control input at the beginning of the simulation facilitates oscillations in the angular rates. This phenomenon is observed for all joints accordingly. However, when the first two piecewise-constant control input signals are applied to the plant, the inputs only change gradually from iteration to another. This enables relatively smooth trajectories of the angular rates without the occurring oscillations and proves the correlation of the oscillations with large changes in the control input.

3.3.5. Influence of coupled dynamics and model-dimension

To assess the influence of the coupling of different joints and the model-dimension on the computation time, simulations with the single inverted pendulum in Figure 3.1a are performed. For comparison, the same number of nodes as evaluated in Table 3.3 for the pendulum with three joints is used. The MPC parametrization is listed in Table 3.4. The control input constraints and the weighting matrices $\mathbf{R}$ are scaled down to the reduced inertia of the pendulum. The computation times are illustrated in Figure 3.9. Similarly, as in the previous plot for the three-joint pendulum, the methods with different solver and prediction horizon parametrization are shown.

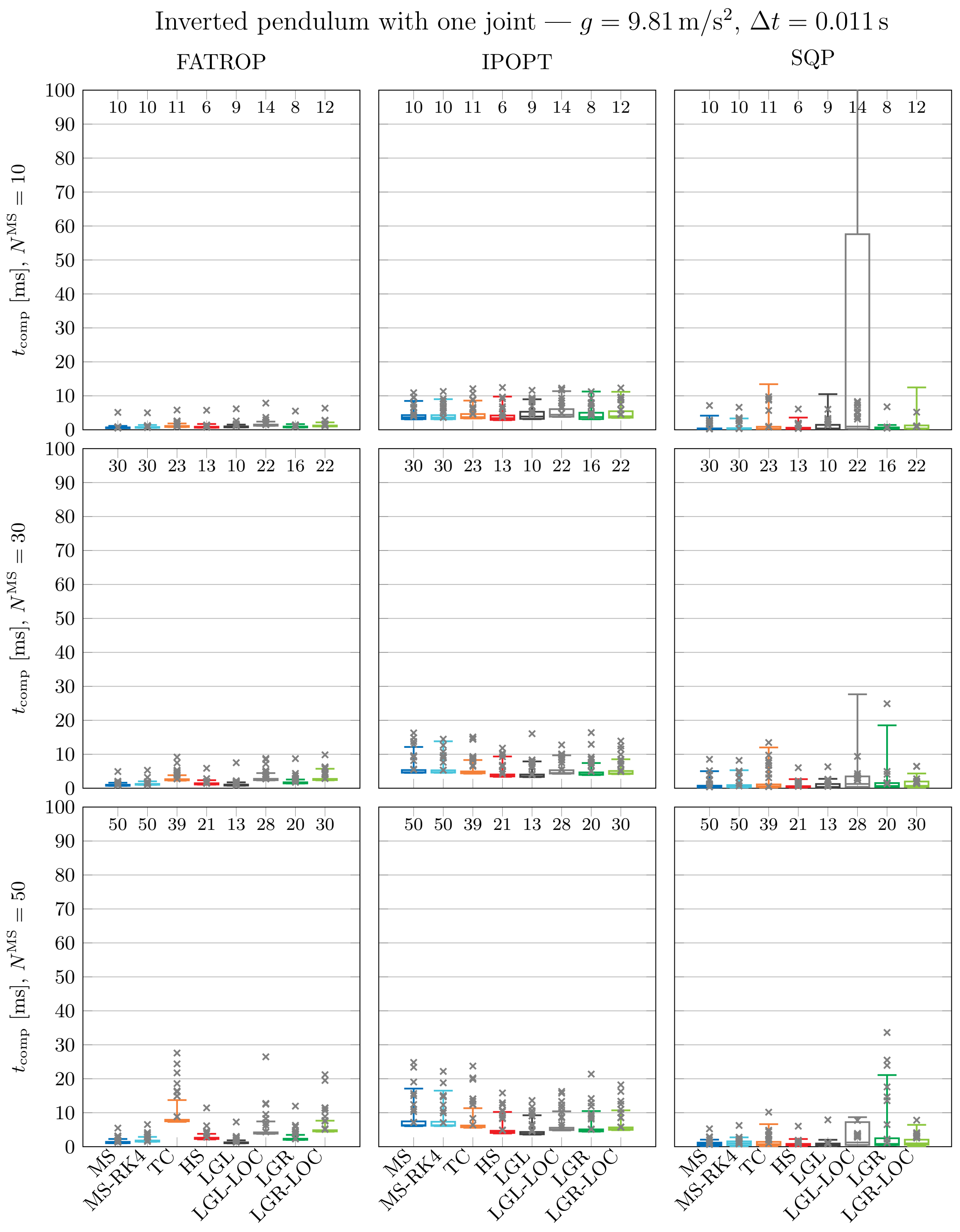


Figure 3.9.: Computation time for the one-joint inverted pendulum. The gray crosses indicate the first solution of each closed-loop trajectory. The box plot shows the median, 5 and 95% quantiles and the corresponding min. and max. values. The top-row of each plot indicates the number of discretization parameters for each method.

Table 3.4.: Prediction horizon and sampling frequency for one-joint inverted pendulum.

Config.	T	N	Δt	Weighting matrices	Constraints
0110	0.1 s	10	0.011 s	$\mathbf{Q} = \mathbf{I}^{2\times 2}$ with $Q_{1,1} = 10$	
0330	0.3 s	30	0.011 s	$\mathbf{R} = \frac{30}{2500}\mathbf{I}^{1\times 1}$	$\lvert u\rvert \leq 1\,\mathrm{Nm}$
0550	0.5 s	50	0.011 s		

Remark 3.3.10

SQP is a comparable alternative to FATROP and IPOPT with reduced model complexity.

For the simplified model, the SQP solver demonstrates superior performance when considering the computation time median. For the shooting methods and all collocation methods — except for LGL-LOC — the upper quantiles of computation times with SQP are comparable to or lower than those of FATROP and IPOPT for $N^{\mathrm{MS}} = 10$. The worst-case computation times of SQP generally fall between those of FATROP and IPOPT. Consequently, for models with reduced complexity, SQP presents a viable alternative, although its robustness, particularly in terms of worst-case execution times, warrants further investigation.

Remark 3.3.11

MS with FATROP achieves the best scalability in computation time with increasing model complexity.

In Figure 3.10 the absolute computation times for the one-joint and the three-joint inverted pendulum are illustrated. The solving times (upper-quantiles) for the configuration 0330 are used. Other configurations with different prediction horizon are assumed to behave qualitatively similar. A detailed analysis of the influence of prediction horizon on computation time is stated in Remark 3.3.2.

FATROP with MS achieves the fewest increase in computation time over all methods and is the most performant solver for both model complexities. Applying IPOPT with MS consumes significantly more time, since IPOPT is not capable of using structure detection for shooting methods.

For MS-RK4 both solvers achieve a comparable increase in time, but the overall computation time for FATROP is below IPOPT. The difference in computation time between the one-joint and three-joint inverted pendulum is significantly higher for MS-RK4 than MS. With further added model complexity, a greater difference is expected. As a result, a good trade-off between potentially enhanced closed-loop performance with higher-order integration schemes and computation time needs to be established.

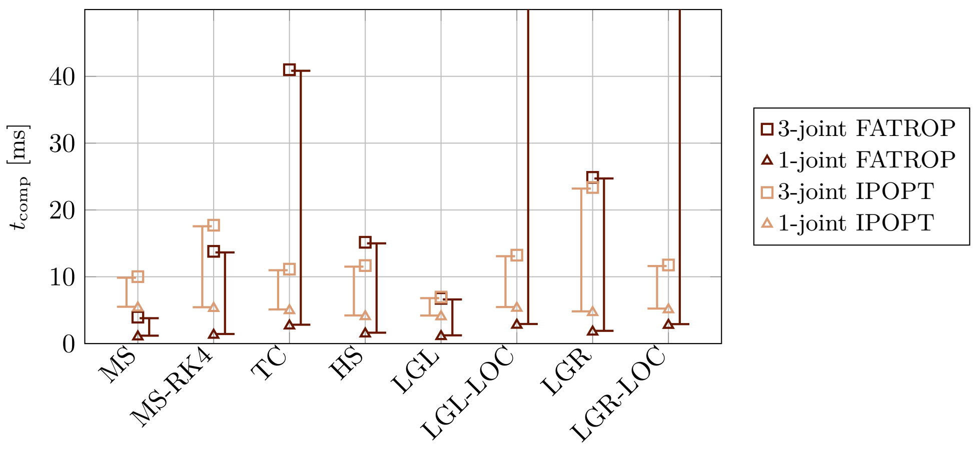


Figure 3.10.: Scaling of computation times of the different methods for different model complexity (dimension and coupling effects). The upper quantile for each method is illustrated for configuration $N^{\mathrm{MS}} = 30$.

Remark 3.3.12

Collocation methods should be solved with FATROP for reduced model complexity. For large and complex NLPs, IPOPT is the most performant option for collocation if they are not compatible with FATROP's structure detection.

Figure 3.10 highlights that for low model complexity the best solver for collocation methods is FATROP. All computation times for the one-joint pendulum are below the ones of IPOPT. However, as the NLP complexity increases by adding dimensionality and state couplings, FATROP is not capable of efficiently handling the sparse structure of the collocation NLPs. Thus, IPOPT is the preferable choice for collocation.

Specifically the methods TC, LGL-LOC and LGR-LOC, that produce a relatively large, sparse NLP, cannot be efficiently solved with FATROP. The NLPs with dense structure, LGL and LGR, perform comparable for both solvers and model complexities.

3.4. LGR-LOC*: A pseudospectral method compatible with FATROP's sparsity pattern

The results in Section 3.3.2 outline the performance boost when combining the FATROP solver with multiple shooting methods. This result is highlighted and explained in Remark 3.3.1. The collocation methods are not adapted to the structure detection feature because of their inherent implicit formulation which is not natively compatible with FATROP's sparsity pattern.

However, by arranging the optimization variables in a special structure and introducing additional optimization variables to preserve the expected pattern, certain collocation methods can be adapted to FATROP with structure detection. For the global collocation methods LGL and LGR this is not possible, because of their dense constraint Jacobian, as shown in Figure 2.3. A detailed analysis of this aspect lies beyond the scope of this work.

One important characteristic to make the pseudospectral method adapt to FATROP is to have explicit continuity constraints

$$\boldsymbol{x}_{i+1} = \boldsymbol{\Theta}(\boldsymbol{x}_i, \boldsymbol{u}_i), \tag{3.11}$$

with $\boldsymbol{\Theta}$ denoting a mapping from the current state x_i to x_{i+1}. Comparing the above requirement with the equation ensuring continuity for TC (2.28) and for HS (2.39), one identifies different structure. For both methods, the continuity constraints have the form $\boldsymbol{x}_{i+1} = \boldsymbol{\Theta}(\boldsymbol{x}_i, \boldsymbol{u}_i, \boldsymbol{x}_{i+1}, \boldsymbol{u}_{i+1})$. However, by introducing more optimization variables, the required explicit structure might be achieved. This is not further analyzed within the frame of this work.

Instead, the LGR-LOC method is chosen to adapt to the sparsity pattern. The LGR-LOC method explained in Section 2.3.5 does not require continuity constraints, since the interpolated final state in each interval is not considered as an additional optimization variable. Instead, the variable of the next interval is used, as shown in (2.53).

However, by introducing additional optimization variables for the interpolated final state in each interval, two optimization variable exist that represent an identical state in time. As a result, continuity constraints have to be included to ensure the two variables to have an identical state. This is desired to align with the required pattern in Eq. (3.11). The state matrix then yields

$$\mathbf{X}^l = \begin{pmatrix} (\boldsymbol{x}_1^l)^{\mathrm{T}} \\ \vdots \\ (\boldsymbol{x}_{N_l}^l)^{\mathrm{T}} \\ (\boldsymbol{x}_{N_l+1}^l)^{\mathrm{T}} \end{pmatrix} = \begin{pmatrix} x_{11}^l & \dots & x_{1n_{\mathrm{x}}}^l \\ \vdots & \ddots & \vdots \\ x_{N_l1}^l & \dots & x_{N_ln_{\mathrm{x}}}^l \\ x_{(N_l+1)1}^l & \dots & x_{(N_l+1)n_{\mathrm{x}}}^l \end{pmatrix} \tag{3.12}$$

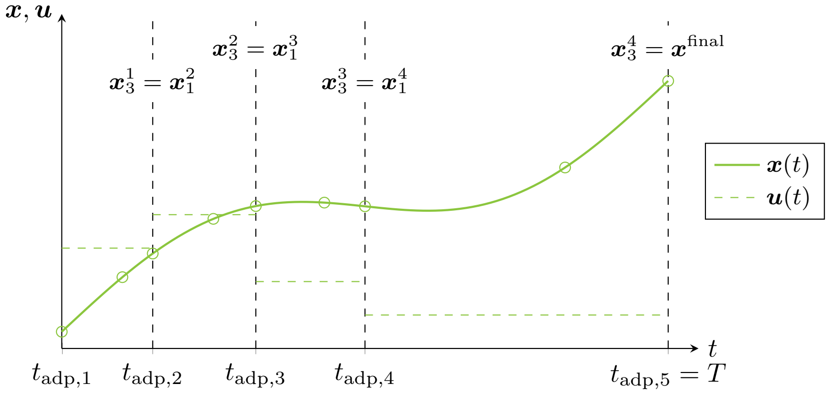
with N_l denoting the number of collocation points in each interval (constant for all intervals) and the index l generally referring to the current interval $l = 1, \dots, N_{\mathrm{int}}$. If one compares Eq. (3.12) with Eq. (2.53) it is evident, that the variable $(\boldsymbol{x}_1^{l+1})^{\mathrm{T}}$ is replaced by the extra optimization variable$(\boldsymbol{x}_{N_l+1}^l)^{\mathrm{T}}$. To compensate for the loss of continuity, the constraints

$$\boldsymbol{x}_{N_l+1}^l = \boldsymbol{x}_1^{l+1} \tag{3.13}$$

are introduced. To further reduce the couplings between intervals, the control inputs are assumed to be constant over each interval. The control input matrix reduces to a vector

$$\mathbf{U}^l = \left((\boldsymbol{u}_1^l)^{\mathrm{T}}\right) = \begin{pmatrix} u_{11}^l & \dots & u_{1n_{\mathrm{u}}}^l \end{pmatrix}. \tag{3.14}$$

Figure 3.11 shows an example of the state and control input discretization. The continuity constraints are shown, too. In this example, the state is approximated with $N_l = 2$



$\boldsymbol{x}, \boldsymbol{u}$
$\boldsymbol{x}_3^1 = \boldsymbol{x}_1^2$
$\boldsymbol{x}_3^2 = \boldsymbol{x}_1^3$
$\boldsymbol{x}_3^3 = \boldsymbol{x}_1^4$
$\boldsymbol{x}_3^4 = \boldsymbol{x}^{\text{final}}$
$\boldsymbol{x}(t)$
$\boldsymbol{u}(t)$
t
$t_{\text{adp},1}$
$t_{\text{adp},2}$
$t_{\text{adp},3}$
$t_{\text{adp},4}$
$t_{\text{adp},5} = T$

Figure 3.11.: Schematic for the state and control input approximation of the LGR-LOC* for $N_l = 2$ and $N_{\text{int}} = 4$.

collocation nodes in all four intervals. The interval border of the first interval $t_{\text{adp},2}$ is always assumed to be greater than Δt^{MS}, that is the closed-loop frequency of the MPC. With this assumption, the MPC-constraint (3.9) can be omitted in the NLP.

The method adapted to the structure detection algorithm is denoted as LGR-LOC*, to distinguish from the baseline LGR-LOC. The NLP formulation is given in NLP 3.1.

NLP 3.1: Local Legendre Gauss Radau with sparsity (LGR-LOC*)

$$\min_{\mathbf{X}^1,\ldots,\mathbf{X}^{N_{\text{int}}},\boldsymbol{u}^1,\ldots,\boldsymbol{u}^{N_{\text{int}}}} \quad J = V_{\text{f}}(\boldsymbol{x}^{\text{final}}) + \sum_{l=1}^{N_{\text{int}}} \frac{t_{\text{adp},l+1} - t_{\text{adp},l}}{2} \sum_{i=1}^{N_l} w_i^l \mathcal{L}(\boldsymbol{x}_i^l, \boldsymbol{u}^l) \tag{3.15}$$

$$\text{subject to} \quad \frac{2}{t_{\text{adp},l+1} - t_{\text{adp},l}} \mathbf{D}^l \mathbf{X}^l \overset{!}{=} \mathbf{F}^l(\mathbf{X}^l, \boldsymbol{u}^l), \quad l = 1, \ldots, N_{\text{int}} \tag{3.16}$$

$$\boldsymbol{x}_1 = \boldsymbol{x}_{\text{start}}$$

$$\boldsymbol{x}_1^{l+1} = \boldsymbol{x}_{N_l+1}^l, \quad l = 1, \ldots, N_{\text{int}} - 1 \tag{3.17}$$

$$\boldsymbol{g}(\mathbf{X}^1, \ldots, \mathbf{X}^{N_{\text{int}}}, \boldsymbol{u}^1, \ldots, \boldsymbol{u}^{N_{\text{int}}}) \leq 0$$

$$\boldsymbol{x}^{\text{final}} \in \mathcal{X}_{\text{f}}$$

The parameters $w_i^l, \boldsymbol{x}_i^l$ refer to the value at the collocation point $i = 1, \ldots, N_l$ in the interval $l = 1, \ldots, N_{\text{int}}$. The control input value $\boldsymbol{u}^l$ denotes the constant input for the given interval.

In Figure 3.12 the sparsity pattern compatible with FATROP's exploitation algorithm is

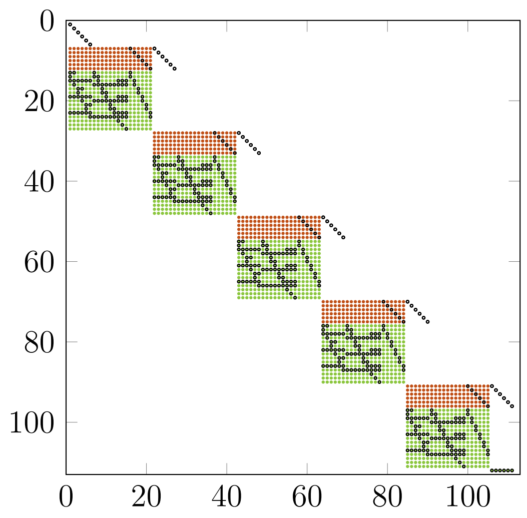


Figure 3.12.: Sparsity pattern of the constraint Jacobian for the LGR-LOC* method adapted to structure detection in FATROP for $N_{\text{int}} = 5$. The highlighted elements in red are continuity constraints, while the green elements are general NLP constraints. The black circles indicate a non-zero element.

shown. The first six rows of the matrix represent the initial constraints for the state. The continuity constraints (3.17) to align the state from one interval to another are highlighted in red. The collocation constraints (3.16) together with control input constraints in each interval are represented in green. The last row of the Figure shows the terminal constraints for the final state. As can be seen, another optimization variable is introduced for the terminal state to align with the sparsity pattern.

In theory, the LGL-LOC method could also be adapted to the sparsity pattern. However, first evaluations have shown the OCP to become infeasible by not having enough degrees of freedom. A potential reason for this behavior might be the necessity to include an additional optimization variable for the final state. This might increase the dimension of the NLP without introducing more DOF. Further analysis is required for assessing applicability of LGL-LOC to FATROP's sparsity pattern.

Methodology to align MS with LGR-LOC*

In contrast to the described methodology to align all the methods in an open-loop setting, that is explained in Section 3.2.1, the LGR-LOC* is aligned to MS with the criteria of achieving comparable closed-loop cost. The resulting parameterization is shown in Table 3.5. The weighting matrices and the baseline configuration is chosen equivalently to as stated in Table 3.2.

More degrees of freedom for finding a suitable parametrization exist for the LGR-LOC*,

Table 3.5.: Interval borders and parameters for the LGR-LOC*.

	MS	**MS-RK4**	**LGR-LOC***			
Config.	N^{MS}	$N^{\text{MS-RK4}}$	N_{int}	$\boldsymbol{t}_{\mathrm{adp}}$	$\boldsymbol{N}^{\text{LGR-LOC*}}$	$\sum \boldsymbol{N}$
0110	10	10	2	$(0.011, T)$	(2,2)	4
0330	30	30	4	$(0.011, 0.05, 0.1, T)$	(2,2,2,2)	8
0550	50	50	5	$(0.011, 0.05, 0.1, 0.3, T)$	(2,2,2,2,2)	10

due to the flexibility of changing not only the number of intervals, but also the number of collocation nodes within. Apart from the constraint of choosing $t_{\mathrm{adp},1} = \Delta t^{\mathrm{MS}}$, the interval borders are chosen with increasing spacing the longer the prediction horizon. The number of nodes is set constant to $N_l = 2$ in each interval. Simulations have shown to have enhanced computational performance with increased number of intervals and fewer optimization variables in each leading to a sparser NLP. Importantly, the number of intervals also defines the control input discretization, as previously illustrated in Figure 3.11.

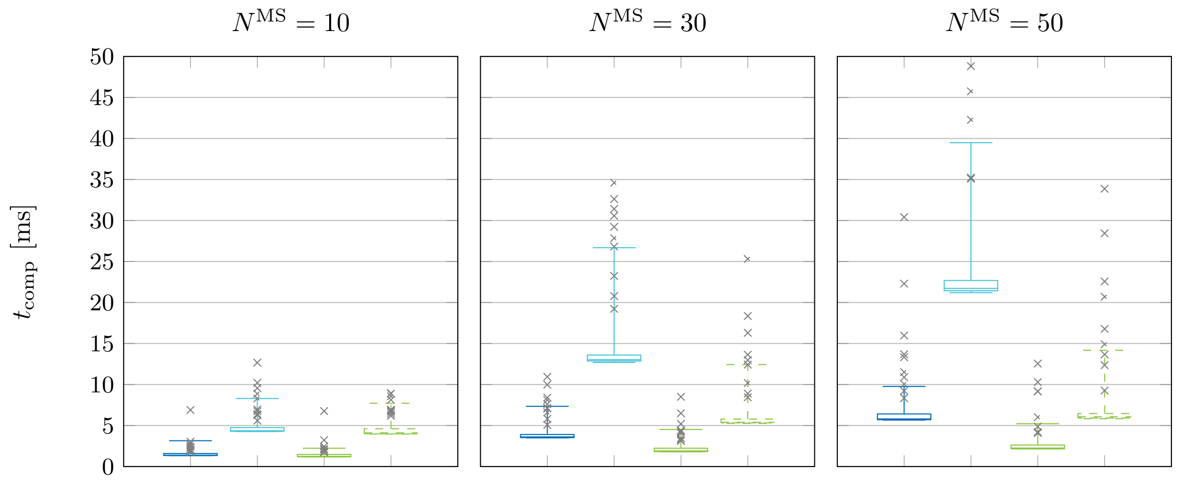
Results of LGR-LOC*

The LGR-LOC* and MS methods are compared in an identical scenario as applied in Section 3.3.2. The results for the computation time and closed-loop cost are shown in Figure 3.13. All solutions are solved with the solver FATROP with structure detection enabled. The adapted method LGR-LOC* is additionally solved with IPOPT to assess the improvement.

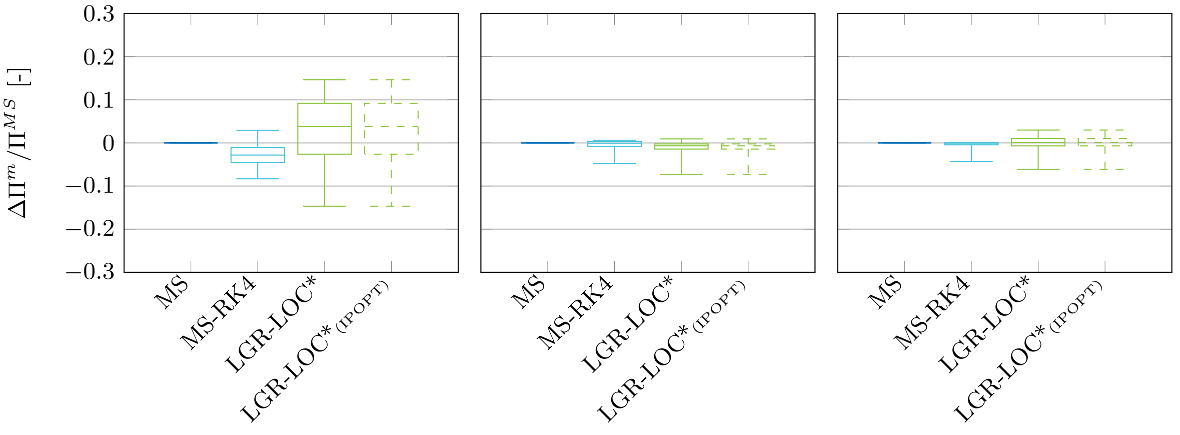
Remark 3.4.1

MS and LGR-LOC* both solved with FATROP exhibit comparable performance for short prediction horizon $N^{\mathrm{MS}} \approx 10$.

For a low number of prediction horizon steps, the LGR-LOC* and MS perform comparable in terms of computation time. Specifically, for the given configuration, the worst-case execution time for LGR-LOC* is slightly improved while the closed-loop performance is slightly degraded compared to the baseline MS. However, if adding another interval in configuration 0110 to the LGR-LOC* method, the computation time is slightly worse than MS but the closed-loop performance is enhanced. Generally, as already observed in Remark 3.3.6, the variance is high for low N concluding that both methods behave relatively similar for a low number of nodes.



(a) Computation time



(b) Closed-loop cost

Figure 3.13.: Comparison of LGR-LOC*, MS and MS-RK4 solved with FATROP and structure detection.

Remark 3.4.2

LGR-LOC* outperforms for $N^{\text{MS}} > 10$ and is the most-performant solver-method combination over all evaluated parametrization.

For increasing prediction horizon, the closed-loop behavior for both methods is comparable. The computation of LGR-LOC* is significantly improved compared to MS and MS-RK4. For $N^{\text{MS}} = 30$, the upper quantile of the computation time for LGR-LOC* is reduced by factor of 1.75, for $N^{\text{MS}} = 50$ of 2.45.

Comparing the LGR-LOC* solved with FATROP and with IPOPT (dashed), an improvement in computation time is observed. While in Remark 3.3.12 the solver of choice for all collocation methods is identified as IPOPT, this result could be outperformed with the above designed LGR-LOC* method. Specifically, even if the number of nodes from configuration 0110 (4 nodes) increased to configuration 0550 (10 nodes), the computation time is mainly not affected because of efficient sparsity exploitation in the solver.

3.5. Discussion of the results

The results of the previous section outline the best method-solver combinations. To give a throughout summary, Table 3.6 gives an overview about the properties of each shooting and collocation method.

Table 3.6.: Property matrix for optimal control methods over different criteria evaluated on the benchmark example.

		MS	MS RK4	TC	HS	LGL	LGL LOC	LGR	LGR LOC	LGR LOC*	Ref.
t_{comp}	$N^{\text{MS}} = 10$	$+$								$+$	3.3.1
	$N^{\text{MS}} \in [30, 50]$	$+$								$+$	3.4.1
	$N^{\text{MS}} > 50$	$(+)^1$				$+$				$+$	3.4.2
Scalability over increased model complexity and dimension		$+$								$(+)^2$	3.3.11
Closed-loop oscillations		No	No	With rapidly changing dynamics (large gradient $\partial \boldsymbol{x}/\partial t$)							3.3.8
Constraints fulfillment		$+$	$+$	Oscillating behavior does not ensure constraint fulfillment in between nodes.							3.3.8

1 A suitable option when collocation methods tend to oscillate.
2 Not specifically analysed, but assumed to scale similar to MS due to FATROP's structure detection.

The computational performance for small prediction horizons $N = 10$ is superior when using multiple shooting (MS) or adapted LGR-LOC* method, both solved with FATROP. For increasing prediction horizons, the LGR-LOC* method is computationally more performant on the benchmark example while simultaneously achieving comparable closed-loop performance. Shooting methods might be applied when incorporating rapidly changing dynamics that produce oscillations for collocation methods. For longer prediction horizons ($N > 50$), also other collocation methods such as LGL are anticipated to become more efficient. In the MPC context, global pseudospectral methods are hindered by the MPC constraint (3.9), which enforces piecewise-constant control inputs within each sampling interval.

Regarding scalability, MS combined with FATROP exhibits the most favorable scaling behavior as the model complexity increases — from a single inverted pendulum to a three-link inverted pendulum system. This advantage can be attributed to FATROP's efficient structure detection algorithm, which effectively exploits problem sparsity [55]. Due to the ability to apply the sparsity exploitation algorithm to LGR-LOC*, the method is expected to behave similarly. However, no specific analysis has been performed regarding scalability.

In both open-loop and closed-loop simulations, collocation methods tend to produce oscillatory state trajectories when predicting rapidly varying state derivatives. This behavior is particularly influenced by the discretized control inputs. Oscillations are largely mitigated during iterations with minimal variation in control inputs. In practical applications such as On-Orbit Servicing — especially during rapid escape maneuvers — rapid state changes may induce oscillations that render collocation methods less suitable. Conversely, for scenarios characterized by smoother state transitions, collocation methods may represent a viable alternative.

The survey also contained the comparison of different solvers, namely FATROP, IPOPT and SQP. Table 3.7 gives overview about the key aspects.

Table 3.7.: Property matrix for nonlinear solvers of different criteria.

		FATROP	IPOPT	SQP	Ref.
Shooting	simple model	✓		✓	3.3.10
	complex model	✓			3.3.2
Collocation[1]	simple model	✓		✓	3.3.10
	complex model	(✓)[1]	✓		3.3.12
Sensitivity to non-linearity		low	low	high	3.3.7

[1] FATROP is the solver of choice for collocation when the method can be aligned with the required sparsity pattern, as shown for LGR-LOC*.

For shooting methods (MS and MS-RK4), FATROP demonstrates the highest computational performance. In particular, its structure detection algorithm, which exploits sparsity, significantly reduces computation time compared to IPOPT. Inherently, collocation methods cannot leverage the structure detection capability. However, with specific assumptions and introduction of additional optimization variables some collocation methods might be aligned with the structure detection algorithm, as shown for the LGR-LOC* method. If this is applicable, FATROP is rendered as the most performant option for collocation. Otherwise, IPOPT is the preferred solver for complex models. For models characterized by low coupling effects and limited state dimensionality, FATROP may still represent a viable alternative even without active sparsity exploitation.

Sequential Quadratic Programming (SQP) generally struggles to find solutions for coupled nonlinear systems. Specifically, for the three-joint system, obtaining a feasible trajectory requires a substantial number of iterations. SQP is most effective for smaller systems with reduced dimensionality and couplings. Moreover, the maximum execution times for SQP may exceed those of interior-point solvers. It is also worth noting the application of CASADI's built-in SQP solver with OSQP. Alternative SQP implementations, such as ACADOS [56] or MOSEK [42], might offer improved performance.

Further work The method for determining a comparable parameter setup for each optimal control strategy is critical for conducting a thorough analysis. The open-loop approach described in Section 3.2.1 has been shown to yield comparable closed-loop costs across the different methods. However, collocation methods demonstrated superior results for shorter prediction horizons. Consequently, aligning each method based on equivalent closed-loop cost may represent a viable alternative.

To further assess the influence of model dimensionality and complexity, an inverted pendulum with more than three joints could be examined. This would provide deeper insights into the effects of coupling and, in particular, state dimension. Such an analysis could improve the extrapolation of performance across various method-solver combinations.

In this survey, multi-body dynamics are assumed. The physical parametrization of the model incorporated relatively fast-changing dynamics, which are potentially better suited to MS methods. Applying the same methodology to different benchmark examples, characterized by inherently slow and gradually changing dynamics, could yield additional insights regarding the applicability of collocation methods.

4 The pseudospectral method for MPC in On-Orbit Servicing

The survey in the previous chapter revealed a promising alternative to the standard multiple shooting version: the LGR-LOC* method. It is adapted to the required sparsity pattern of FATROP to enhance computation time. A consequent step is to evaluate the performance of the newly developed method on a more complex model and simulation framework for On-Orbit Servicing.

In Section 4.1 the specific scenario and the two implemented MPCs are explained. The following two Sections 4.2 and 4.3 then compare the two optimal control methods on a tracking and a hybrid maneuver.

4.1. On-Orbit Servicing maneuvers and controller design

The general purpose of OOS is introduced in Section 1.1. A distinction is made between spacecraft base control and combined control. The former focuses on determining control inputs for the translation and rotation of the spacecraft servicer. In contrast, the latter extends both the state and input dimensions to include the 7 DOF of the robotic manipulator. This work is primarily concerned with spacecraft base control.

4.1.1. On-Orbit Servicing maneuver

As an initial condition for the OOS one assumes the servicing satellite in a close-range corridor around the tumbling target. Figure 4.1 gives an overview about the different tasks that are involved in OOS.

The pose estimation module computes the relative position and orientation of the servicer with respect to the target. Measurements from Lidar and Cameras for relative pose

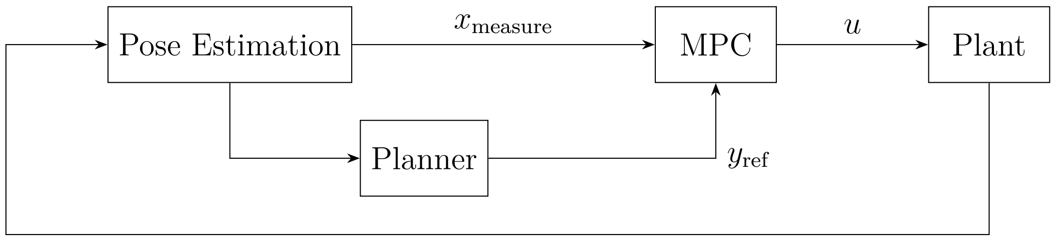


Figure 4.1.: Schematic of OOS phases.

estimation in combination with star trackers and GPS for global estimation are processed. The state vector is given in an orbital reference system.

The planner computes a reference for the MPC. It can either be used as a trajectory planner or as an operational point planner. The first computes a time-varying reference while the latter outputs a time-constant pose. Depending on the mode the servicer is in, either trajectory planning or operational point planning is used.

Both signals, so the measurement state vector provided by the pose estimation and the reference generated by the planner are inputs to the MPC. The controller computes forces and torques that are in a consequent step applied to the plant. In the combined control case additional torques for the robotic manipulator are computed.

A special mode for OOS is a so-called escape maneuver. If anomalies during the approach phase to the target are detected, such a maneuver is performed. The purpose is to increase the distance to the target as fast as possible to prevent potential collisions.

4.1.2. MPC design

The MPC for spacecraft base control involves the following state and control input vector

$$\boldsymbol{x} = \begin{pmatrix} \boldsymbol{p} \in \mathbb{R}^3 \\ \boldsymbol{q} \in \mathbb{R}^4 \\ \boldsymbol{\nu} \in \mathbb{R}^6 \end{pmatrix} \in \mathbb{R}^{13}, \qquad \boldsymbol{u} = \begin{pmatrix} \boldsymbol{f} \in \mathbb{R}^3 \\ \boldsymbol{\tau} \in \mathbb{R}^3 \end{pmatrix} \in \mathbb{R}^6.$$

The spacecraft position is denoted with $\boldsymbol{p}$ and the orientation in unit quaternions as $\boldsymbol{q}$. The velocity vector $\boldsymbol{\nu} = (\boldsymbol{v}, \boldsymbol{\omega})$ consists of translational velocities $\boldsymbol{v}$ and angular velocities $\boldsymbol{\omega}$. As control inputs forces $\boldsymbol{f}$ and torques $\boldsymbol{\tau}$ are considered for each axis.

A multi-body dynamics formulation in the form of (3.1) is used as a dynamical model for the spacecraft servicer. For orientation propagation, the quaternion representation is used. Further details can be found in [40].

Actuator limits by box constraints are considered in the formulation throughout the prediction horizon. As an additional OOS specific requirement, Field-of-View (FoV) constraints are modeled as polytopes. They ensure the target satellite to remain in the

FoV corridor of the servicer. This is relevant for enabling pose-estimation which is camera or laser based and consequently relies on visual contact to the target.

A terminal region in the form of Eq. (2.57) is included in the MPC formulation ensuring recursive feasibility and stability. Quadratic cost according to Eq. (2.56) paired with a terminal cost term (2.58) form the cost functional to minimize.

Baseline: Multiple Shooting The baseline MPC method uses a Multiple shooting approach to discretize the optimal control problem with $N = 10$ uniformly spaced shooting intervals and a prediction horizon of $T = 1\,\mathrm{s}$. This results in an interval length of $\Delta t = 0.1\,\mathrm{s}$ that is also the control loop frequency.

LGR-LOC*: Pseudospectral method As evaluated in the survey of optimal control methods, a viable alternative to multiple-shooting is the LGR-LOC* method described in Section 3.4. Specifically as demonstrated in Table 3.5, for $N = 10$ prediction horizon steps, the LGR-LOC* with two intervals is supposed to perform similarly to MS (compare Figure 3.13). Consequently, for this purpose an identical parametrization with two intervals $N_{\text{int}} = 2$ and two collocated nodes in each interval $N^l = 2$ is used. The prediction horizon is set equivalently to the baseline MS method to $T = 1\,\mathrm{s}$. All constraints are embedded on both methods. To ensure better comparability, the FoV constraints are implemented on 5 nodes for the baseline and for the LGR-LOC* version.

4.1.3. Simulation environment

The OOS simulation framework is implemented in Python. A more complex and detailed model is embedded via Functional Mockup Units for the closed-loop simulation. This further analyses the robustness of the MPC due to the different complexity of prediction model and closed-loop model. The MPC is integrated by compiled executables generated from C-source code from CasADi [4].

The computational measurements are performed on an 11th Generation Intel Core i7-1185G7 on Windows Operating system with 16 GB of RAM. The process priority for the simulation is set to high to reduce operating system based delays. For both methods the maximum number of solver iterations is limited to 15.

4.2. Tracking maneuver

The purpose of the tracking maneuver is to synchronize position and orientation over time with the tumbling target. The spacecraft base should remain at constant distance from the target while continuously synchronizing orientation. In this mode, the actual

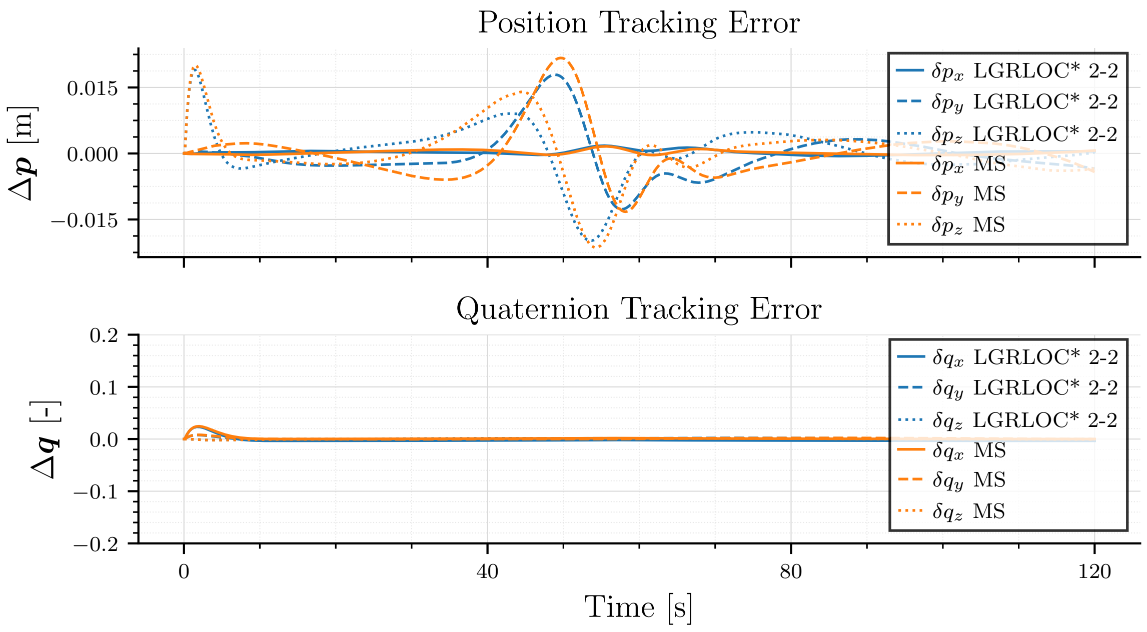


Figure 4.2.: Tracking error between the LGR-LOC* and the MS method.

grasping maneuver with the robotic manipulator would be performed. However, since in this thesis only the spacecraft base is analysed, this maneuver can be seen as a preliminary for the actual OOS task. For tracking, the trajectory planner is used.

Figure 4.3 shows closed-loop plots for a tracking maneuver of the LGR-LOC* method. The left column represents the states and control inputs for the translational DOF and the right column for angular DOF. In the bottom left sub-figure the closed-loop cost Π is shown. It is identically evaluated as in described in (3.10) and measures the deviation from the current state to the fed-in reference as well as the magnitude of control inputs at each control update of the MPC. The closed-loop results for the MS method are attached in the Appendix A.1. The deviation in closed-loop cost of the two methods is below $\leq 0.5\,\%$ resulting in a quasi identical closed-loop behavior.

At the beginning of the simulation, a peak in the force and torque control inputs is observed, primarily due to the unsynchronized angular velocities and the associated linear motion required to orbit the target. Once the rotational speed is identical to the target after about $t \approx 10\,\mathrm{s}$, the torques are converging to zero. This is expected, since the target is a tumbling object without external forces or torques being applied. The MPC commands non-zero forces due to the translational motion required to orbit the target.

The quaternion to represent the orientation is enforced to have a positive scalar value. Consequently, the sign of the quaternion switches after rotation over 180°. This is the case at around $t = 50\,\mathrm{s}$. A full tumbling period is around $t = 100\,\mathrm{s}$, which can be observed

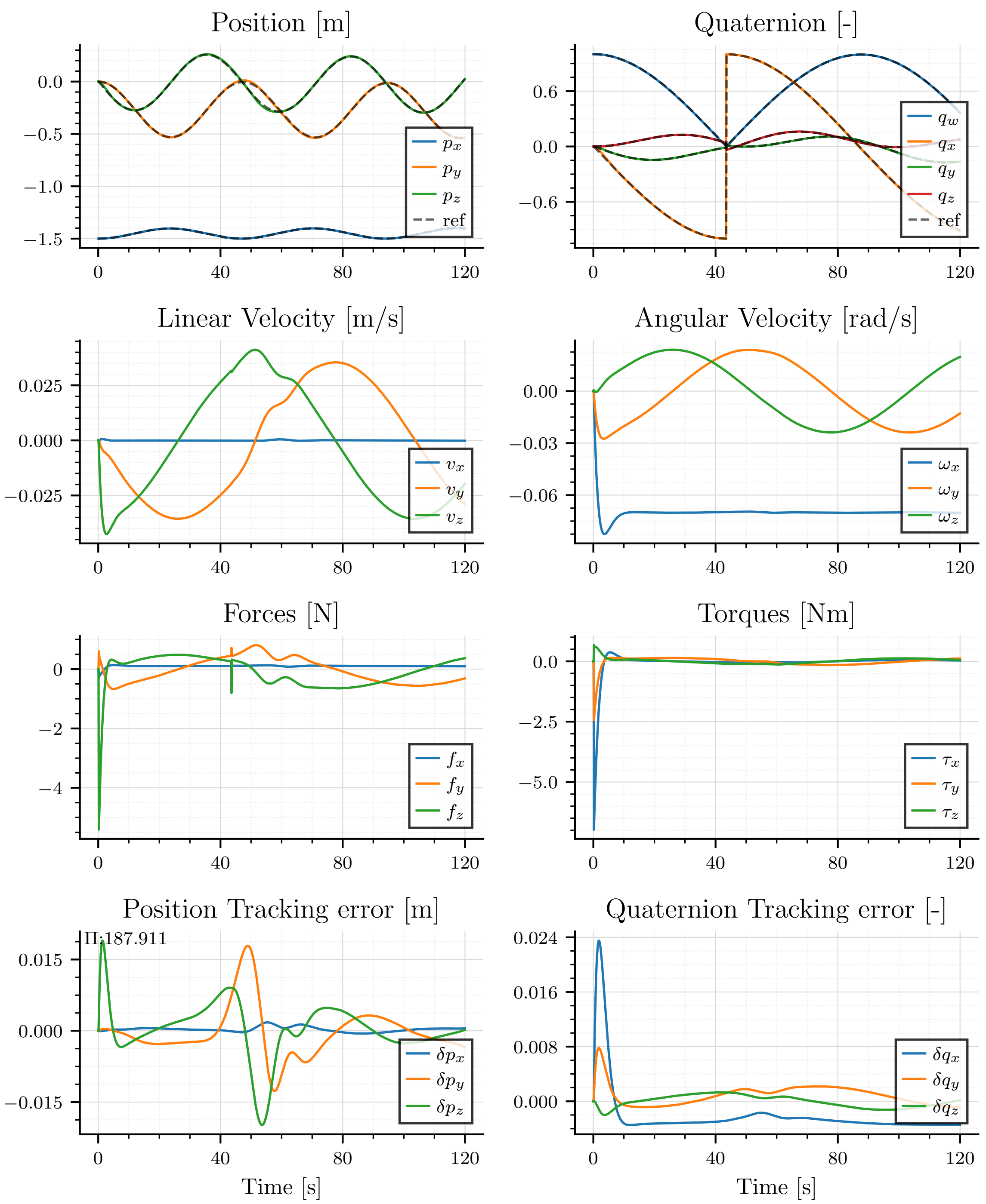


Figure 4.3.: Closed-loop plot for the LGR-LOC* method in a tracking maneuver.

in the position and quaternion plot as periodic curves.

The position and quaternion tracking error for the LGR-LOC* and the MS is revealed in Figure 4.2. It underlines the quasi identical behavior of the two methods. The orientation tracking error converges almost to zero, while the position error is bounded at ±2 cm.

The computation time of the two methods is shown in Figure 4.4. The dotted lines represent the 95 % quantiles and the dashed lines the median. Computational advantages

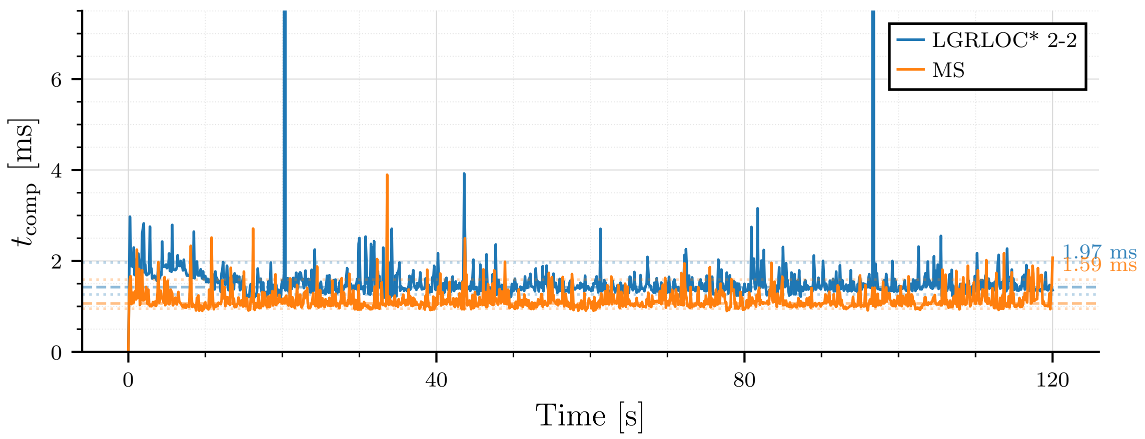


Figure 4.4.: Computation time for the two methods on a tracking maneuver. The dotted lines represent the 5 % and 95 % quantile, the dashed lines represent the median.

of approximately 24 % are observed for the MS method compared to the LGR-LOC*. The second method shows occasional outliers in computation time. This may be due to the testing environment, which runs on a desktop PC not designed for real-time processing. Further analysis is needed to better understand the cause of this behavior.

4.3. Hybrid scenario with escape maneuver

A more challenging task for the MPC than tracking is a maneuver that does not involve a pre-computed reference trajectory. The effect of only assuming an operational pose as reference is analyzed in this Section. Specifically, the sequence of maneuvers shown in Figure 4.5 and its visualization in Figure 4.6.

In a first step, the target is approached from a distance of 4 m, with the objective of achieving identical orientation as the target while maintaining a constant relative distance of 1.5 m (see Fig. 4.6a and 4.6b). The second sequence represents the situation of detecting anomalies during the approach, triggering an escape maneuver (Fig. 4.6c). Once the

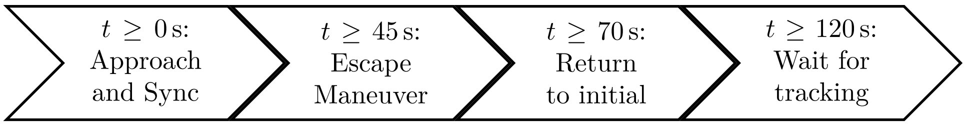
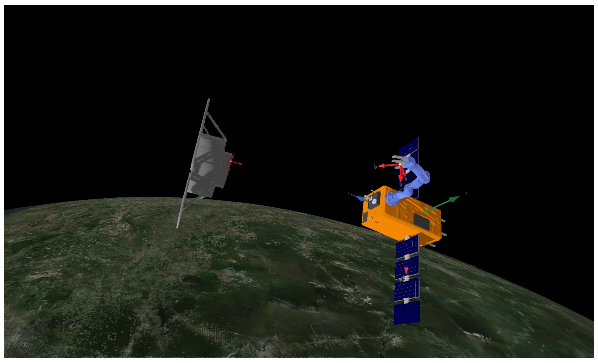
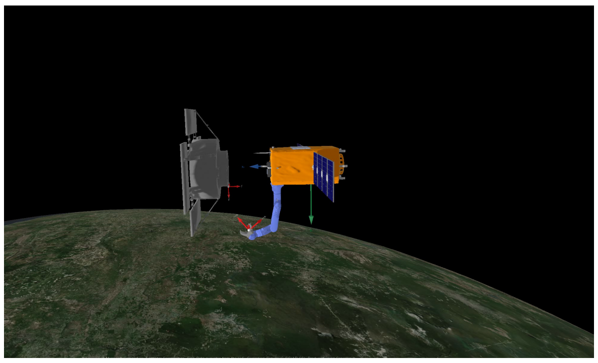
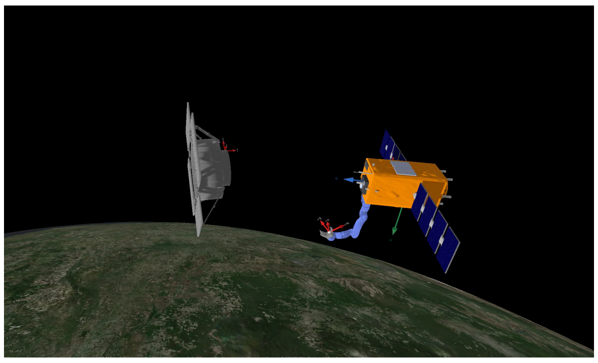

$t \geq 0\,\mathrm{s}$: Approach and Sync
$t \geq 45\,\mathrm{s}$: Escape Maneuver
$t \geq 70\,\mathrm{s}$: Return to initial
$t \geq 120\,\mathrm{s}$: Wait for tracking

Figure 4.5.: Sequence of maneuvers in a hybrid scenario.


(a) Approach to target


(b) Tracking the target


(c) Escape maneuver

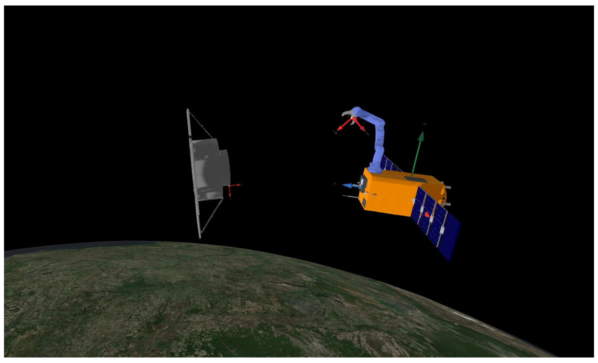

(d) Back in initial position

Figure 4.6.: Sequence of maneuvers involved in OOS.

distance to the target has been sufficiently increased in this escape scenario, the target returns to its initial position (Fig. 4.6d). At this point, the servicer is positioned to initiate a subsequent approach, provided that the orientation of the target is within the servicer's FoV corridor.

The closed-loop plots for LGR-LOC* and MS are shown in Figure 4.7 and 4.8. An improvement of about $\approx 8\,\%$ is achieved with the collocation method when comparing the closed-loop cost.

Moreover, the two methods converge to different local minima. While the LGR-LOC* methods flies a trajectory that is above the reference in p_y, the MS method yields a trajectory below the reference for the y-position. This is evident in Figure 4.9, too, that shows the tracking error for the position and the orientation.

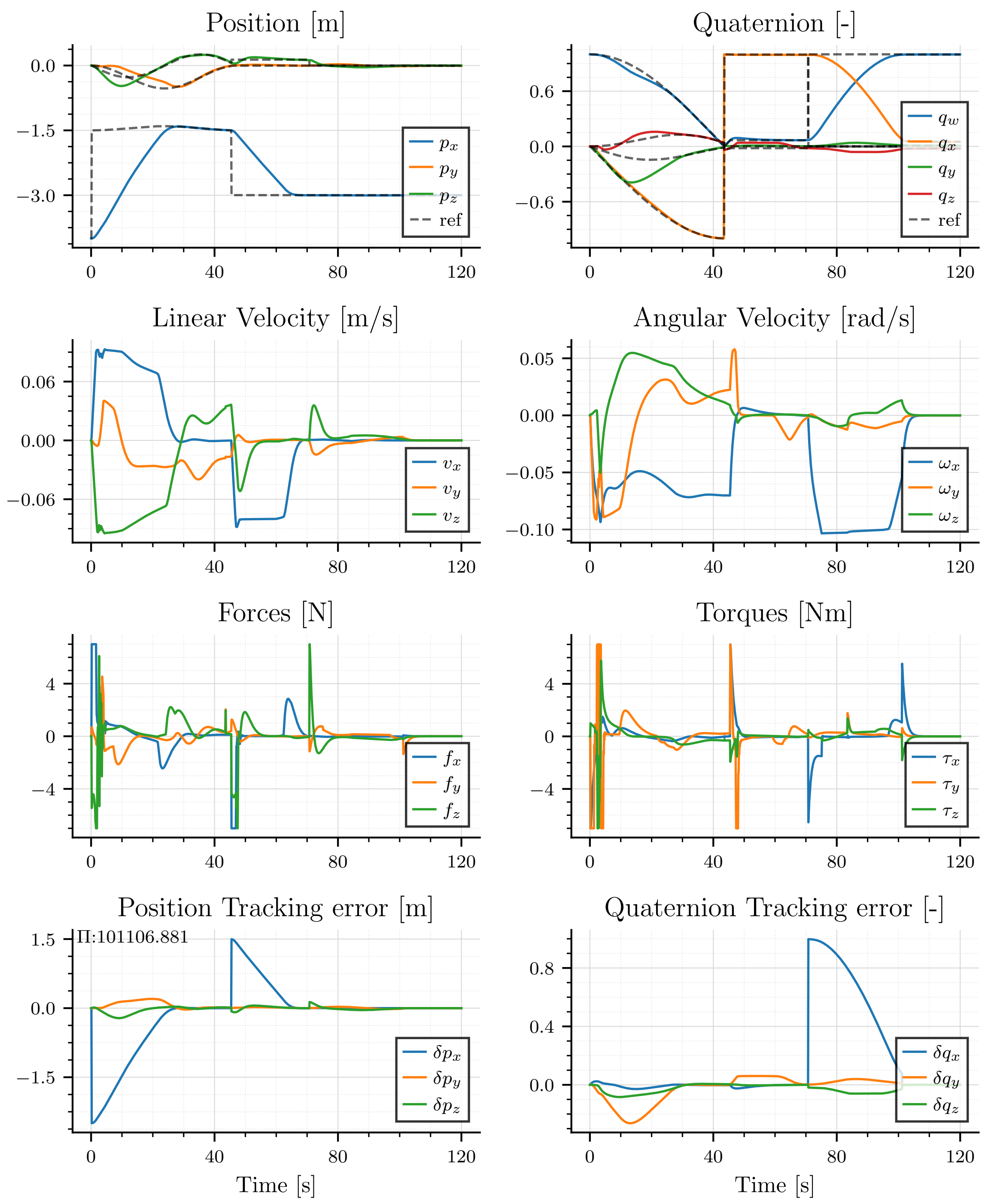


Figure 4.7.: Closed-Loop plot for the LGR-LOC* method in a hybrid maneuver with escape scenario and return to initial position.

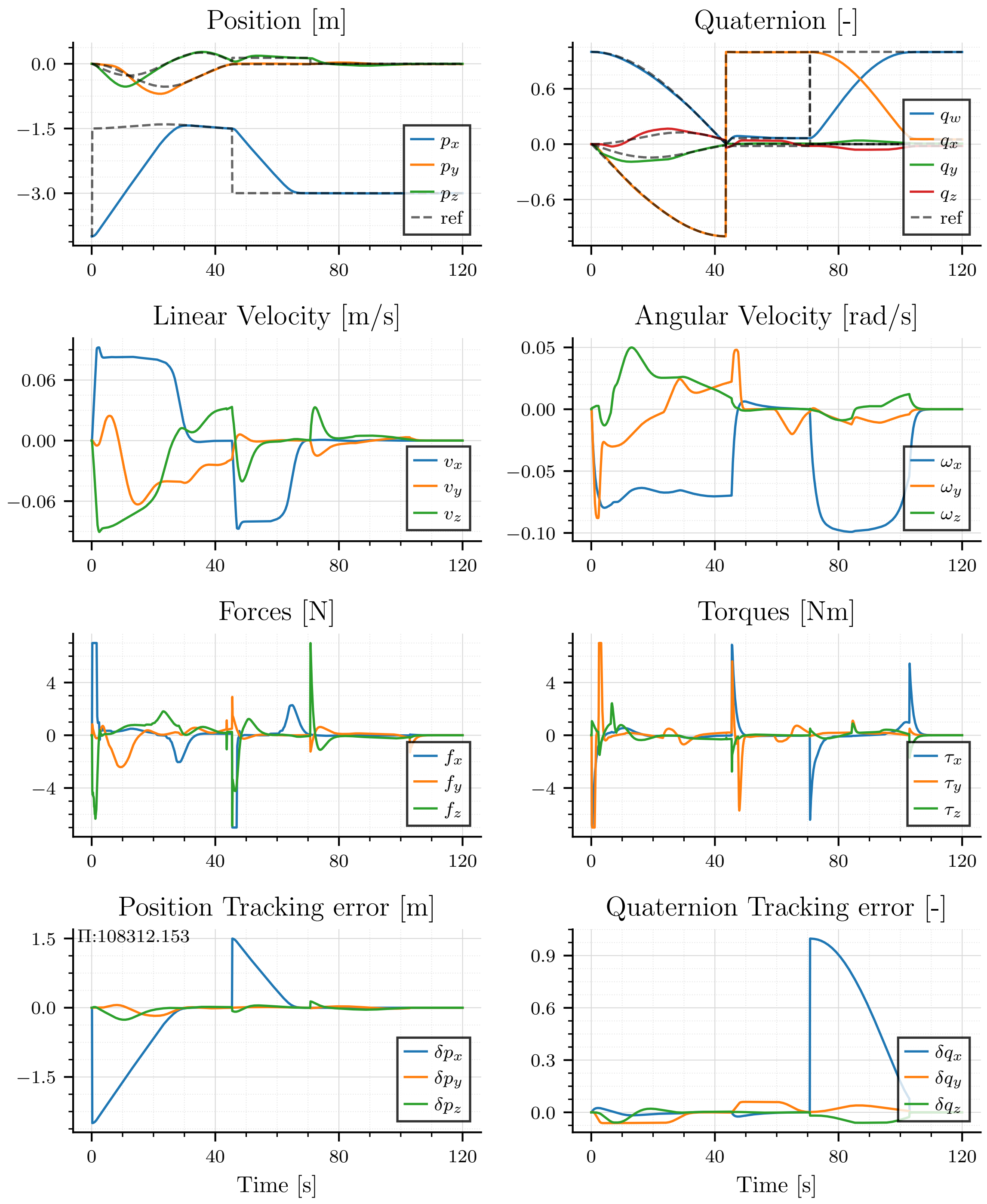


Figure 4.8.: Closed-Loop plot for the MS method in a hybrid maneuver with escape scenario and return to initial position.

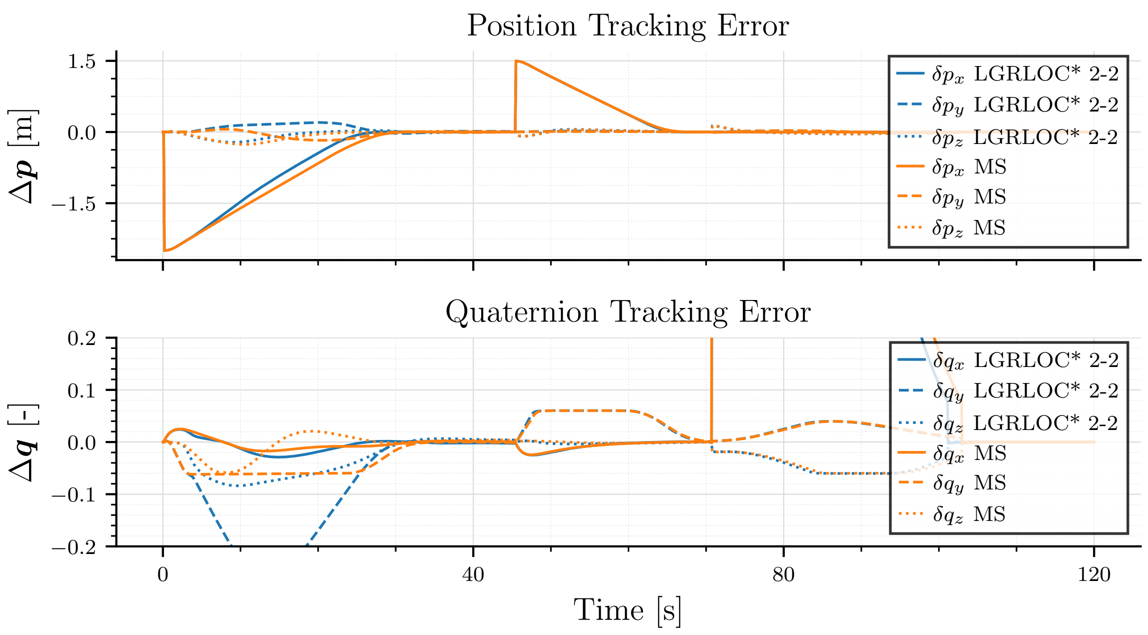


Figure 4.9.: Closed-Loop plot for the LGR-LOC* method.

The control inputs on the LGR-LOC* method are identified to be more rugged and aggressive in contrast to MS. This can be seen specifically at the beginning of the simulation in the forces f_y, f_z, but also in torques τ_z. As a result, the velocities of the Servicer are smoother for the MS-case.

As a main reason for the collocation method to outperform shooting is the enhanced behavior when returning from the escape pose back to the initial pose. This can be seen in the tracking error plot in Figure 4.9 at $t \geq 100\,\mathrm{s}$: The MS quaternion error is delayed by approximately $2\,\mathrm{s}$ to the LGR-LOC* curve indicating an increased time to converge back to the nominal pose.

Comparing the computation time of the two methods in Figure 4.10 for the hybrid maneuver reveals further differences. As expected for the more complex hybrid maneuver, the computation times are generally higher than for the tracking case. The LGR-LOC* method requires approximately 22 % more computation time.

4.4. Summary

The pseudospectral method is successfully implemented for MPC in OOS scenarios. The result is demonstrated on a tracking and on a hybrid maneuver.

An almost identical closed-loop behavior is observed for identical parameters such as pre-

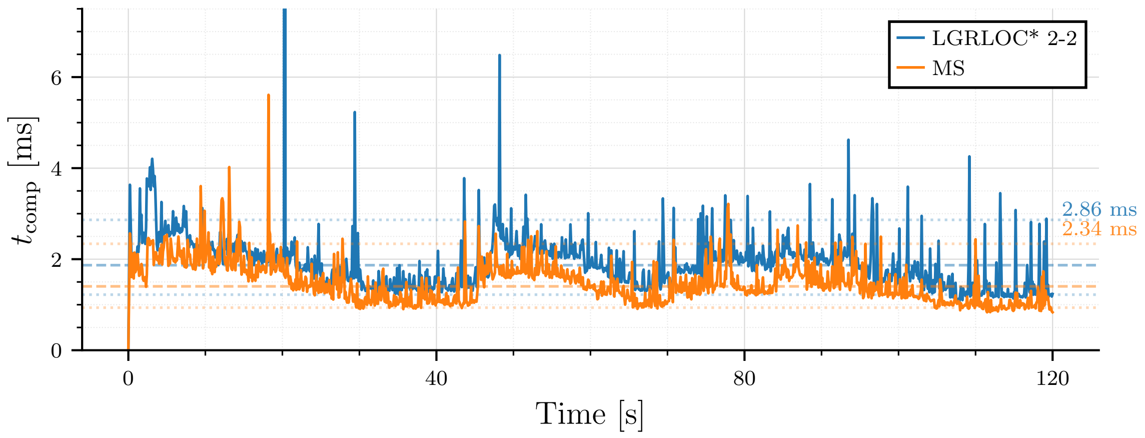


Figure 4.10.: Computation time for the two methods on a hybrid maneuver. The dotted lines represent the 5 % and 95 % quantile, the dashed lines the median.

diction horizon and number of constraints. The computational efficiency is identified to be more performant for the MS method making it the desirable choice for the application in OOS.

Nevertheless, by incorporating only two control inputs in the LGR-LOC* method and still yielding identical closed-loop performance, might enable some potential to also improve the interval spacing and length for the MS method.

5 A universal approach to find the best optimal control method

One of the key aspects of MPC defining the performance is the choice of the optimal control method. It specifies the computation time, the accuracy of the predicted trajectory and as a result the overall performance of the controller. This raises the question:

> *Is it possible to systematically identify the most performant optimal control method and its parameterization for a given MPC control problem?*

In this Chapter, an answer to the above question is given by presenting a Bayesian Optimization (BO) framework to find the best parameterization of the OCP.

In Section 5.1 the theoretical background for the BO is discussed. The framework and the optimization scheme is presented in Section 5.2 and its application to the scenario of OOS is discussed in Section 5.3.

5.1. LGR-LOC*: A universal method to incorporate collocation and shooting methods

The method introduced in Section 3.4 combines a pseudospectral method on subintervals with the sparsity pattern of FATROP. The performance on the benchmark example and on the OOS framework is demonstrated. The LGR-LOC* method can be parameterized by three different parameters:

N_{int}	defines the number of intervals,
N^l	defines the number of collocation nodes in each interval and
$\boldsymbol{t}_{\text{adp}}$	specifies the interval borders.

It can be shown, that by choosing the above three parameters in a certain way, each of the following evaluated optimal control methods can be derived as a subclass of the

LGR-LOC*:

$$\left.\begin{array}{l}\text{MS}\\ \text{adapted TC}\\ \text{adapted HS}\\ \text{LGR-LOC}\\ \text{LGR}\end{array}\right\} \subseteq \text{LGR-LOC*}.$$

A detailed derivation for the MS, the adapted TC and HS is given in the appendix in Section B.1. Table 5.1 summarizes the results. It shows for which combination of N_l and N_{int} the LGR-LOC* equals which specific method.

Table 5.1.: Overview of the methods that can be derived out of LGR-LOC*.

N_l	N_{int}	Method	Constraint	Ref
1	≥ 2	MS	$x_{i+1} = x_i + hf(x_i, u_i)$	B.1.1
2	≥ 2	Adapted TC	$x_{i+1} - x_i = \frac{h}{4}(3f_{i1} + f_i)$	B.1.2
3	≥ 2	Adapted HS	$x_{i+1} - x_i = \frac{h}{36}\left(4f_i + (16+\sqrt{6})f_{i1} + (16-\sqrt{6})f_{i2}\right)$	B.1.3
> 3	≥ 2	Pseudospectral LGR-LOC with higher order polynomial		
> 3	$= 1$	Global Pseudospectral LGR with higher order polynomial[1]		

[1] Sparsity exploitation with FATROP is lost.

The derived Multiple-Shooting is mathematically fully equivalent to the formulation in Eq. (2.20) using Euler-Integration. However, since the classical LGR-LOC* is implicit, additional optimization variables are introduced to align with the sparsity pattern in FATROP. Consequently, if the optimization algorithm converged, the classical MS method and the LGR-LOC* with $N_l = 1$ return the identical solution, even though the latter incorporated more optimization variables. A comparative discussion is found in Section 5.3.1.

For $N_l = 2$ and $N_l = 3$, adapted versions of the Trapezoidal and Hermite-Simpson methods, respectively, are generated. The methods are adapted in the sense of using non-uniform nodes for evaluating the dynamics. This increases the order of the quadrature by one compared to the standard Newton-Cotes TC and HS. Thus, the adapted TC and HS obtained by the LGR-LOC* are a more accurate approximation. A detailed derivation of the equations is shown in the Appendix B.1.2 and B.1.3.

If more than three collocation nodes are considered in each interval a classical LGR pseudospectral method defined on multiple segments is derived. Conversely, if a high number of nodes are used on a single interval, so $N_{\text{int}} = 1$, the classical global pseudospectral method is extracted. However, as shown already in Figure 2.3, global methods result in a dense constraint Jacobian. Therefore, they are not further considered in this chapter since they cannot be combined with the performant sparsity exploitation of FATROP.

To conclude, it could be proven that the developed LGR-LOC* method contains a variety of different optimal control methods. The parameter N_l defines the optimal control methods itself and the parameter N_{int} specifies whether to use the methods on a global or a local scheme. The more intervals are used, the finer the discretization. The third parameter $\boldsymbol{t}_{\text{adp}}$ is a configuration parameter to adapt the characteristic of a given method to the incorporated dynamics and constraints. The so gained results induce, that if one is capable of identifying the optimal parameter setup consisting of N_l, N_{int} and $\boldsymbol{t}_{\text{adp}}$, evaluated against a design-specific objective, one is able to find the best optimal control method for a given setup. This is demonstrated in the following sections for On-Orbit Servicing.

5.2. Bayesian Optimization framework

For finding an optimal set of parameters for the LGR-LOC* method a multi-objective Bayesian Optimization (BO) is performed using the Optuna framework [1]. As a sampling method the multi-objective Tree-Structured Parzen Estimator (TPE) is used [6, 44]. This sampler is chosen for the specific problem of simultaneously incorporating a multi-objective optimization with conditional search space.

The workflow for the BO routine is shown in Figure 5.1. Each step is discussed individually in the following paragraphs.

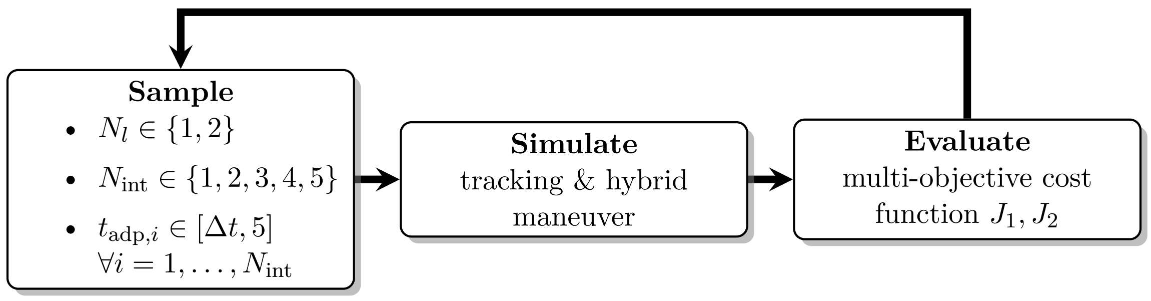

Sample
$N_l \in \{1, 2\}$
$N_{\text{int}} \in \{1, 2, 3, 4, 5\}$
$t_{\text{adp},i} \in [\Delta t, 5]$ $\forall i = 1, \ldots, N_{\text{int}}$
Simulate
tracking & hybrid maneuver
Evaluate
multi-objective cost function J_1, J_2

Figure 5.1.: Schematic for the pipeline of the Bayesian Optimization.

Sampling The three parameters that describe the LGR-LOC* method are sampled. The parameter N_l defines the optimal control method. It is assumed to be one or two resulting in only incorporating the MS and the adapted Trapezoidal method in the formulation (see Table 5.1). A main criterion for the OCP in MPC is the computational performance making LGR-LOC* methods of higher order computational infeasible for the given scenario.

The number of intervals is sampled between one and five intervals giving the optimizer the flexibility to search for more accurate solutions while maintaining low order approximating polynomials. The more intervals are sampled, the more complex the NLP. Sparsity in FATROP is ensured independently of the sampled number of intervals.

A conditional sampling parameter is the interval spacing $\boldsymbol{t}_{\text{adp}}$. It relies on the previously sampled number of intervals. Thus, for each interval the borders are sampled according to the sampling strategy of the TPE method.

By the sampling strategy explained above, it can not be ensured to always generate feasible MPC configurations. Therefore, a simulation can also be pruned by the BO framework if the optimizer does not find a solution.

Simulation The hyperparameter optimization is performed for the specific use-case of OOS. The specific maneuver on which the sampled method is evaluated is crucial for ensuring feasible results. In Section 4.2 and Section 4.3 the case of target tracking and a hybrid maneuver is compared. While the tracking scenario defines the nominal operations in OOS, the hybrid maneuver sets the worst-case execution time which is crucial for the fulfillment of real-time requirements. Hence, the combination of both scenarios is used in the BO optimization. Consequently, one BO iteration consists out of two simulations, one for the tracking case and one for the hybrid maneuver.

Evaluation The performance obtained in the simulation is measured against two objectives: The 95- % quantile of the computation time and the closed-loop cost for the tracking (index T) and the hybrid maneuver (index H)

$$J_1 = \Delta q_{95} = \frac{1}{2}\left(\frac{q_{95,\text{T}}}{q_{95,\text{T-ref}}} + \frac{q_{95,\text{H}}}{q_{95,\text{H-ref}}}\right), \qquad J_2 = \Delta\Pi = \frac{1}{2}\left(\frac{\Pi_{\text{T}}}{\Pi_{\text{T-ref}}} + \frac{\Pi_{\text{H}}}{\Pi_{\text{H-ref}}}\right). \tag{5.1}$$

The mean of each metric is used as the objective for the BO. As a reference, the baseline MS method described in Section 4.1.2 is used that is simulated once on a tracking and on a hybrid maneuver to generate the reference values $(\cdot)_\text{T}$ and $(\cdot)_\text{H}$. By this, the objective J_1 gives information about the relative improvement in computation time and the objective J_2 about the relative improvement of closed-loop behavior.

A multi-objective optimization is chosen to have the flexibility of evaluating manually whether to put more priority to computation time or to closed-loop cost.

The sequence of sampling, simulation and evaluation is repeated iteratively until a specified number of loops is performed.

5.3. Finding the best optimal control method for On-Orbit Servicing

A BO is performed with over 600 samples evaluating different methods, number of intervals and spacing of the interval borders. The results and the best performing optimal control methods are discussed in this Section.

A workflow on how to analyze the result from the BO is given in Table 5.2. Starting from a high-level perspective by evaluating Pareto-optimal solution candidates the scheme gives a strategy to parameterize the best optimal control method for a given scenario. This workflow divides into High-level and Low-level analysis. This scheme is applied to the problem and each step is sequentially executed for OOS.

Table 5.2.: Workflow for finding the best optimal control method.

	Step		Description	Fig.
1	Method evaluation	N_l	Decide on potential Pareto-optimal solution candidates evaluated along the multi-objective.	5.2
2	Number of intervals evaluation	N_{int}	Decide on a local scheme that best fits the multi-objective for the previously evaluated method.	5.2
3	High-level prediction horizon evaluation	T, Π	Evaluate the impact of the prediction horizon on the overall CL cost.	5.3
			High-level analysis	
			Low-level analysis	
4	Low-level prediction horizon evaluation	T, Π	Evaluate the impact of the prediction horizon for the specifically chosen method-interval combination.	5.4 5.6
5	Interval spacing evaluation for CL performance	$\boldsymbol{t}_{\text{adp}}, \Pi$	Evaluate the impact of different interval spacing on the closed-loop cost for the previously evaluated prediction horizon.	5.5 5.7
6	Interval spacing evaluation for computation time	$\boldsymbol{t}_{\text{adp}}, q_{95}$	Evaluate suitable spacing of interval borders to optimize computation time.	5.8

5.3.1. A comment about the computation time of Multiple Shooting methods derived out of LGR-LOC*

The MS approach is incorporated in the LGR-LOC* method. However, LGR-LOC* is by construction an implicit method and the derivation of MS can be seen as a special case. As required for implicit formulations to align with the sparsity pattern in FATROP, additional optimization variables must be introduced. This is explained in detail in Section 3.4.

As a result, when deriving an MS formulation out of LGR-LOC*, those additional optimization variables can theoretically be omitted since MS is inherently an explicit scheme. Mathematically the NLP describes the same problem and produces identical solutions if the solver fully converges. The only impact of this property is a negatively influenced computation time. Consequently, the computation time of shooting methods in the hyperparameter framework can be seen as a worst-case computation time.

If a shooting method is supposed to be the most suitable choice for a given problem, a pure MS formulation that is not derived out of LGR-LOC* can reduce the number of optimization variables by $N_{\text{int}} \cdot n$ with n denoting the dimensionality of the problem.

5.3.2. High-level analysis

Figure 5.2 provides an overview of the evaluated methods measured against the multi-objective. Throughout the plots, the tuple (N_l-N_{int}) denoted in the legend gives information about the method and its number of intervals. All methods with (1-$\cdot$) denote multiple shooting methods, while all methods with (2-$\cdot$) refer to implicit collocation schemes. The abscissa represents the objective J_1 and the ordinate the objective J_2, both described in Eq. (5.1).

Only samples that ensure at least equivalent or better tracking performance $^{\Pi_\text{T}}/_{\Pi_\text{T-ref}} \leq 1$ are shown. The improvement potential in Π_H is substantially larger than for Π_T, which could otherwise lead to cases where a strongly improved hybrid maneuver performance appears to compensate for degraded tracking performance. Such behavior is undesirable, since both maneuvers are required to improve simultaneously. A comprehensive analysis of all samples, including configurations that yield degraded performance, is therefore carried out in the low-level evaluation.

For both metrics J_1 and J_2, the goal is to obtain values that improve both objectives, i.e., $\Delta\Pi < 1$ and $\Delta q_{95} < 1$. These desirable configurations lie in the bottom-left region of the corresponding graph.

Step 1: Method evaluation Comparing the multiple-shooting method ($N_l = 1$) with the collocation method ($N_l = 2$) identifies computational advantages for the shooting

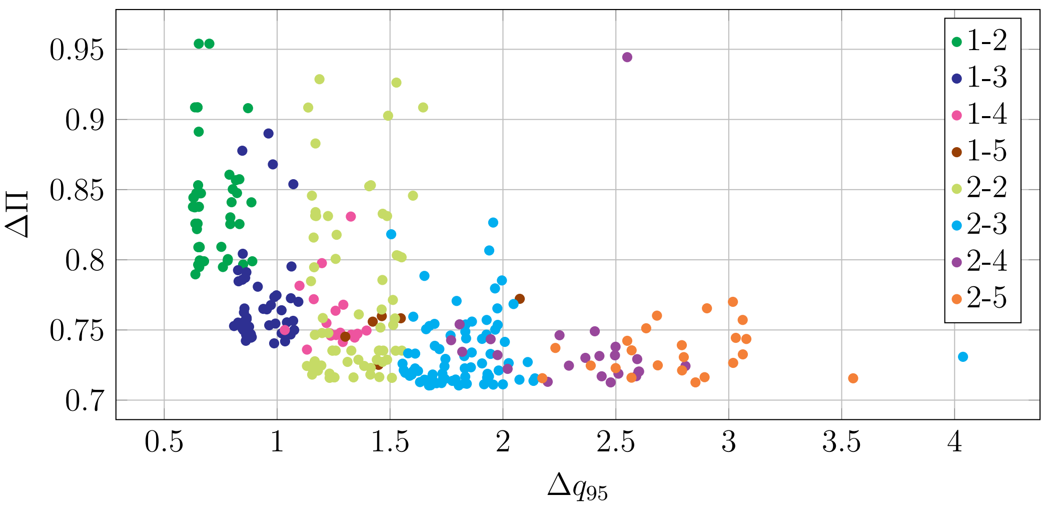


Figure 5.2.: Relative improvement of closed-loop cost against relative change of the 95 % quantile in computation time. Only the samples with improved closed-loop cost $\Delta\Pi < 1$ are shown.

method. Collocation methods require at least about 25 to 30 % more computation time. This result is already expected due to the analysis in the previous chapter. Considering the computation time for the MS approach as a worst-case computation time (explained in Section 5.3.1) underline the shooting methods further to be the most performant option.

On the other hand, the collocation methods have about 5 % enhanced closed-loop costs compared to shooting. Depending on the specific design objectives, either shooting or collocation might be a suitable choice. However, for the requirement in OOS, the significantly improved computation time by $\geq 30\,\%$ outweigh the worse closed-loop cost of $\approx 5\,\%$.

Remark 5.3.1

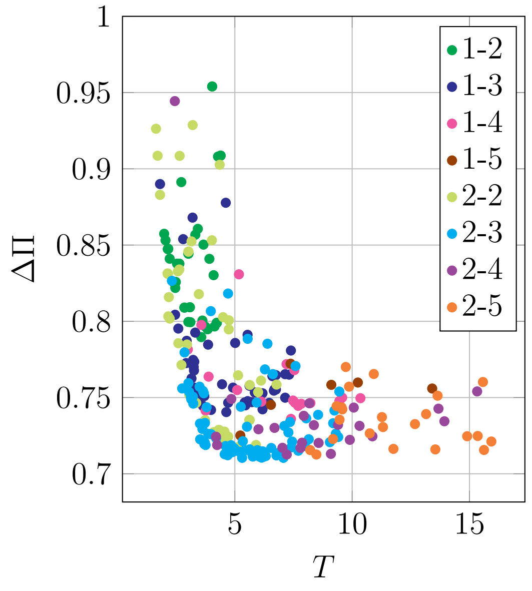
Multiple-Shooting (1-•) should be chosen as a method for the OOS application.

Step 2: Number of intervals evaluation In the previous step the method of choice is evaluated to be MS. Further, evaluating the Pareto objective in Figure 5.2 reveals insight in the optimal choice of intervals. Considering the bottom left of the graph, two methods stand out clearly: (1-2) and (1-3). While the first has the lowest computation time due to the fewest optimization variables included, the second emerges by improved closed-loop performance. Depending on the user-specific priorities, one or the other might be the method of choice. Consequently, the MS method defined on two intervals is further analyzed in Section 5.3.3 and the method defined on three intervals is discussed in detail in Section 5.3.4.

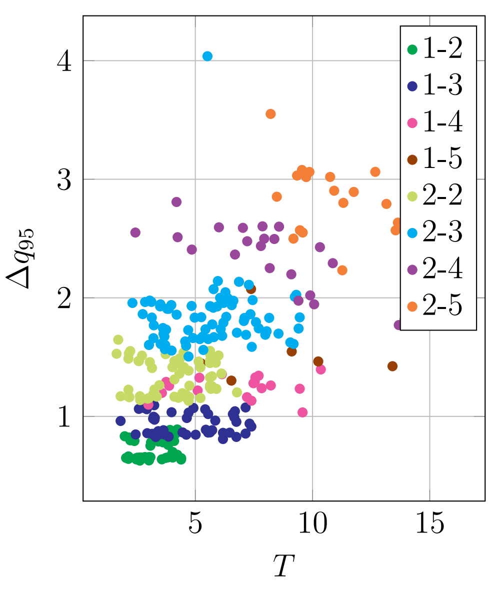
Remark 5.3.2

For MS, the methods (1-2) and (1-3) achieve the best trade-off between Π and q_{95}.

Step 3: High-level Prediction horizon evaluation An overview about the behavior of the methods for different prediction horizon is given in Figure 5.3. Only the samples that achieved for the tracking and the hybrid maneuver an enhanced closed-loop cost are revealed. Plotting the closed-loop cost over the prediction horizon in Figure 5.3a gives



(a) Closed-loop performance



(b) Computation time

Figure 5.3.: Closed-loop cost and computation time over the prediction horizon.

global information about the overall improvement potential of the controller, independent of specific configurations. While for increasing horizons up to $T < 5\,\mathrm{s}$ the closed-loop cost is continuously minimized, no significant enhancement is observed for prediction horizons longer $T \geq 5\,\mathrm{s}$. Methods with more intervals and consequently better accuracy achieve slightly improved closed-loop performance.

Figure 5.3b shows computation time with increasing prediction horizon. Methods with increasing number of intervals N_{int} lead to a larger NLP that consumes more computation time to solve. As further pointed out in the low-level analysis, the number of intervals mainly influences the computation since it defines the number of included optimization variables.

Remark 5.3.3

A prediction horizon $T \geq 5\,\mathrm{s}$ does not produce significantly enhanced closed-loop performance.

5.3.3. Low-level analysis: Method (1-2)

In this section the previously identified solution candidate with $N_l = 1$ and $N_{\mathrm{int}} = 2$ is evaluated.

Step 4: Low-level prediction horizon evaluation Step 4 analyzes the prediction horizon for the specific configuration (1-2). The plot is shown in Figure 5.4, this time including samples with worse closed-loop cost. One can see an optimum for the prediction horizon between $T \in [3, 4.5]\,\mathrm{s}$. All samples with longer prediction horizon worsen the closed-loop performance. This indicates, that for a parameterization with a low number of intervals an upper bound for the prediction horizon exist. Intuitively, the accuracy with only two nodes is not sufficient for prediction horizons longer than 4.5 s.

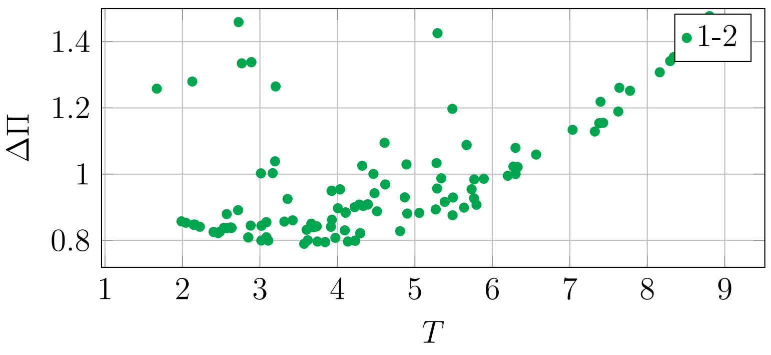


Figure 5.4.: Closed-loop cost over prediction horizon for (1-2).

Remark 5.3.4

A prediction horizon of $T \in [3, 4.5]\,\mathrm{s}$ is recommended for (1-2).

Step 5: Interval spacing evaluation for CL performance Figure 5.5 demonstrates the correlation between $\Delta \boldsymbol{t}_{\mathrm{adp}}$ and $\Delta\Pi$. For achieving optimal closed-loop performance, the first interval spacing should be $\Delta t_{\mathrm{adp},1} \in [1.3, 1.7]\,\mathrm{s}$ while the second interval length should be in the domain $\Delta t_{\mathrm{adp},2} \in [2.4, 2.8]\,\mathrm{s}$. The graph indicates a local minimum in this region.

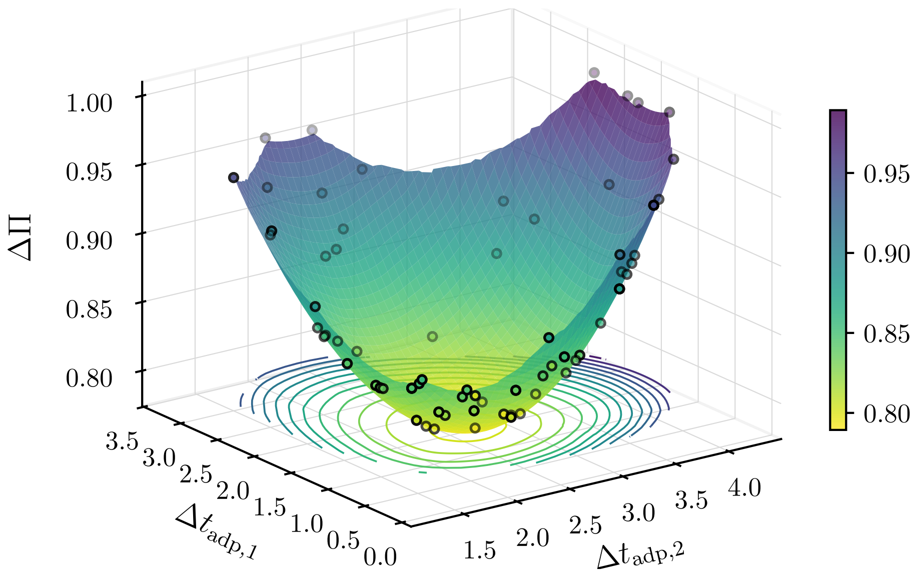


Figure 5.5.: Closed-loop cost of the MS method (1-2) over interval spacing.

Step 6: Interval spacing evaluation for computation time Evaluations on computation time with different interval spacings on two intervals could not identify specific patterns. The computation time remains on a constant level independent of $\boldsymbol{t}_{\text{adp}}$.

Remark 5.3.5

The best performing optimal control method for OOS-MPC in terms of computation time is given as:

$$N_l = 1, \quad N_{\text{int}} = 2, \quad \Delta \boldsymbol{t}_{\text{adp}} = \begin{pmatrix} \Delta t_{\text{adp},1} \in [1.3, 1.7]\,\text{s} \\ \Delta t_{\text{adp},2} \in [2.4, 2.8]\,\text{s} \end{pmatrix}, \quad T = \sum_i \Delta t_{\text{adp},i} \approx 3 - 4.5\,\text{s}.$$

5.3.4. Low-level analysis: Method (1-3)

The second solution candidate is an MS variant defined on three intervals.

Step 4: Low-level prediction horizon evaluation A similar plot for prediction horizon as for the previous solution candidate is given in Figure 5.6. In contrast to the method including only two intervals, the optimum for the prediction horizon is extended to the domain $T \in [4, 7]$s. Horizons outside this region result in a decreasing closed-loop performance for similar reasons as explained previously.

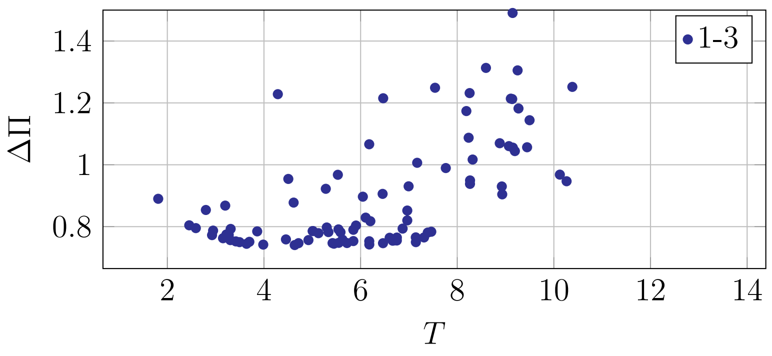
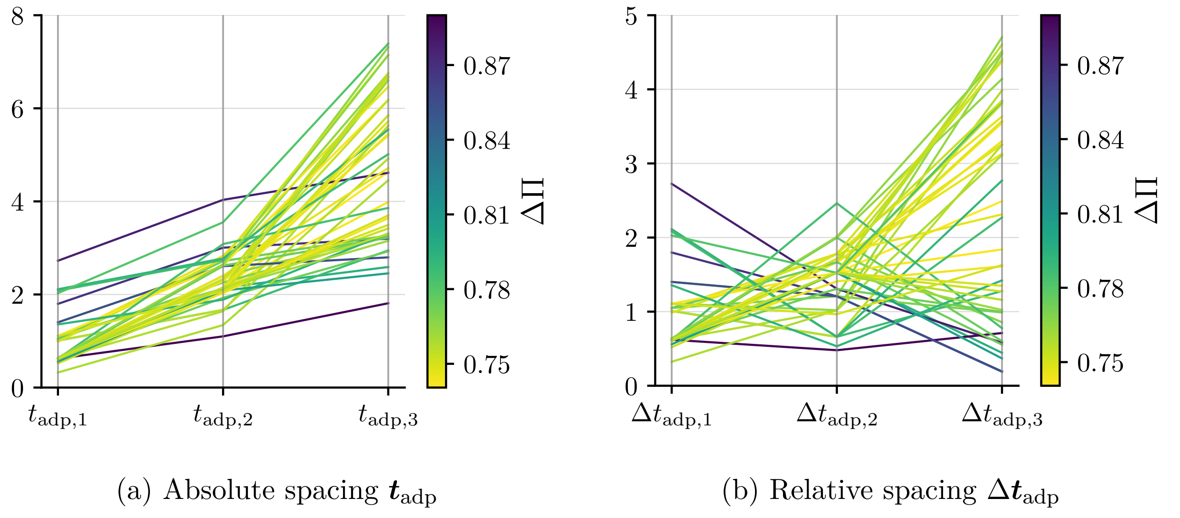


Figure 5.6.: Closed-loop cost over prediction horizon for (1-3).

Remark 5.3.6

For the method (1-3), the prediction horizon should be in the domain $T \in [4, 7]$s.

Step 5: Interval spacing evaluation for CL performance For more than two intervals, the spacing of interval borders cannot be represented in a 3D plot. Instead, a parallel plot is chosen in Figure 5.7. It shows in the left plot the absolute values of interval spacing (good for evaluation of T) and on the right the relative spacing (good for recognizing specific patterns). The colors represent a different closed-loop cost.



(a) Absolute spacing $\boldsymbol{t}_{\text{adp}}$ (b) Relative spacing $\boldsymbol{\Delta t}_{\text{adp}}$

Figure 5.7.: Closed-loop cost of the MS method $N_l = 1$ with three intervals $N_{\text{int}} = 3$ over sampled interval spacing.

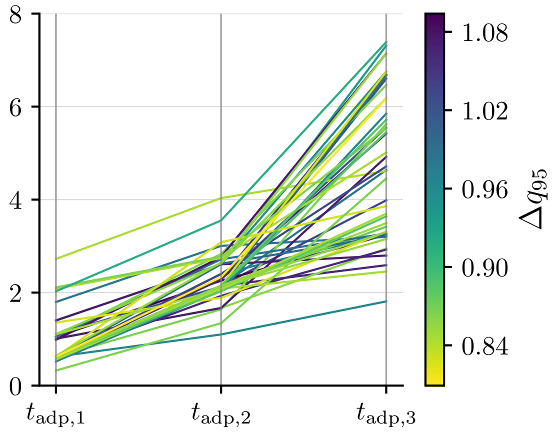
In the left plot 5.7a the observation from Remark 5.3.6 is confirmed. Another trend that can be observed is that all samples with a closed-loop cost $\Delta\Pi < 0.78$ have a first interval length below $t_{\mathrm{adp},1} \leq 1.5\,\mathrm{s}$. Specifically, good results could be achieved with $t_{\mathrm{adp},1} \approx 0.6\,\mathrm{s}$.

Another trend for the spacing of the second interval is identified. It should be in the domain $\Delta t_{\mathrm{adp},2} \in [1, 1.8]\,\mathrm{s}$, as shown in Figure 5.7b. No general trend for the last interval is observed. This leads to the conclusion, that the first two intervals have to align with a specific pattern while the last interval can be chosen to align with the prediction horizon in Remark 5.3.6.

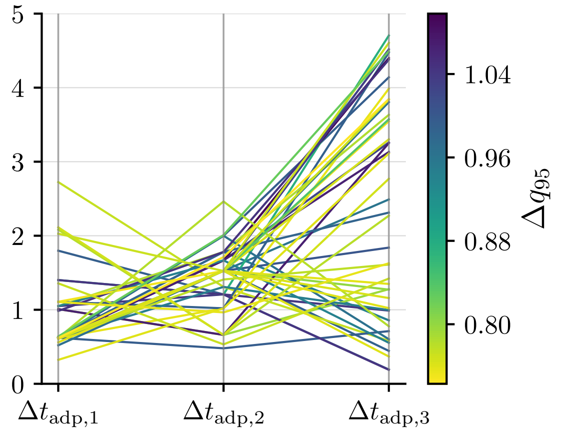
Remark 5.3.7

The first interval length should be below $t_{\mathrm{adp},1} \leq 1.5\,\mathrm{s}$. The second interval should have a relative spacing of $\Delta t_{\mathrm{adp},2} \in [1, 1.8]\,\mathrm{s}$. The third interval can be chosen to achieve an overall prediction horizon between 4 to 7 s.

Step 6: Interval spacing evaluation for computation time In the previous evaluation steps a specific method is chosen, the number of intervals is analyzed, the prediction horizon parameterized, and constraints for spacing of intervals are identified. A remaining question is, if the spacing of the interval impacts the computation time. Figure 5.8 shows the computation time over different absolute and relative interval spacing.



(a) Absolute spacing $\boldsymbol{t}_{\mathrm{adp}}$



(b) Relative spacing $\boldsymbol{\Delta t}_{\mathrm{adp}}$

Figure 5.8.: Computation of the MS method (1-3) over sampled interval spacing.

No general trend can be observed for the combined maneuver including tracking and hybrid scenarios. Specifically, Figure 5.8b reveals no systematic spacing of the interval nodes. When only optimizing for a single objective, such as tracking or a pure escape maneuver, a clear trend might be identified. Due to the mixed approach of finding a controller strategy for both scenarios such a general trend is not found.

Remark 5.3.8

The method with the best trade-off between computation time and closed-loop cost for OOS-MPC is given as:

$$N_l = 1, \quad N_{\text{int}} = 3, \quad \Delta\boldsymbol{t}_{\text{adp}} = \begin{pmatrix} \Delta t_{\text{adp},1} \in [0.1, 1.5] \\ \Delta t_{\text{adp},2} \in [1, 1.8] \\ \Delta t_{\text{adp},3} \in [0.1, 5] \end{pmatrix}, \qquad T = \sum_i \Delta t_{\text{adp},i} \approx 4 - 7\,\text{s}.$$

5.4. Validation

The proposed configuration in Remark 5.3.8 is validated on another scenario. To account for the effect discussed in Section 5.3.1, a MS method omitting extra optimization variables is implemented. The evaluated step-size and interval configuration previously evaluated is hard-coded. Computation analysis on this MS implementation revealed quasi-identical computation time for the methods (2-1) and (3-1). Hence, only the better performing method (3-1) is further validated.

For the validation, a method derived out of Remark 5.3.8 with the parameters

$$N_l = 1, \quad N_{\text{int}} = 3, \quad \boldsymbol{t}_{\text{adp}} = (0.6\,\text{s} \quad 2.1\,\text{s} \quad 4.6\,\text{s})^{\text{T}}$$

is used. In similar notation as for the LGR-LOC*, the improved MS version is annotated with a star (MS*) to distinguish from the baseline.

An overview of the performance increase is indicated in Table 5.3. It reveals a reduction in computation time of ≈ 45 % while achieving enhanced tracking cost of 8 % and of 43 % for the hybrid scenario.

Table 5.3.: Performance characteristics for the improved method MS*.

	Tracking		Hybrid	
	q_{95}	Π	q_{95}	Π
Baseline MS	1.91 s	223.07	2.48 s	117643.69
MS* by BO	1.06 s	205.124	1.30 s	65998.48
	-44.5 %	**-8.05 %**	**-47.6 %**	**-43.9 %**

The validation of the hybrid scenario is performed on an adapted variant to ensure no overfitting of the parameters. More details can be found in following sections.

5.4.1. Validation on a tracking scenario

The tracking case is validated on a similar setting as explained in Section 4.2. The closed-loop plots are shown in Figure 5.11 and 5.12. The improved version MS* shows overall smoother trajectories, specifically for the linear velocities and the corresponding forces. The position tracking error is oscillatory and continuously below 1 cm while the baseline version indicates controller difficulties if the y-position changes nonlinear, i.e. around 50 s. Overall, the behavior is comparable and only minor differences occur.

The computation time is plotted in Figure 5.9. The maximum solver iterations is set to 15 for both approaches. The computation time of MS* is throughout the simulation

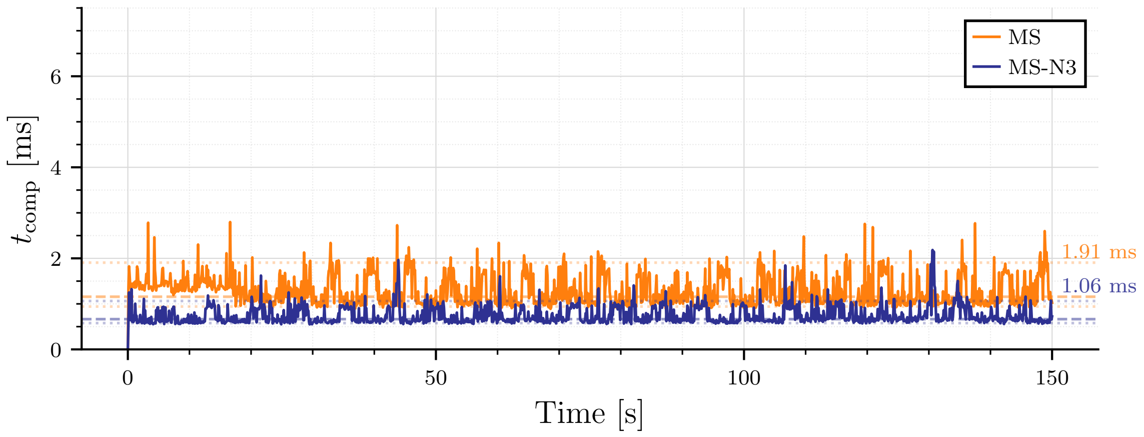


Figure 5.9.: Computation time of MS baseline and MS* on a tracking maneuver. The dotted lines represent the 5 % and 95 % quantile, the dashed lines represent the median.

below the baseline edition. Main reason for the improvement is the reduced number of optimization variables included in the NLP.

5.4.2. Validation on a hybrid scenario

The setup for the hybrid maneuver is slightly adapted. The initial position of the servicer is assumed to have an offset in all three axes before the simulation begins. This represents a more complex scenario and is used to rule out overfitting by the BO routine.

The closed-loop plots are shown in Figure 5.13 and Figure 5.14. The overall closed-loop improvement is over $\geq 40\,\%$. The main reason for the enhanced behavior is the increased prediction horizon. It enables higher linear and angular velocities while still reaching the terminal region. This leads to faster convergence to the corresponding reference pose. During the orientation synchronization with the target at the beginning of the maneuver, the MS* aligns with the target after around 15 s while the baseline version requires approximately 30 s.

The computation time for the hybrid scenario is shown in 5.10. Similarly as for the tracking, the solving time in each iteration is ≈ 45 % below the baseline.

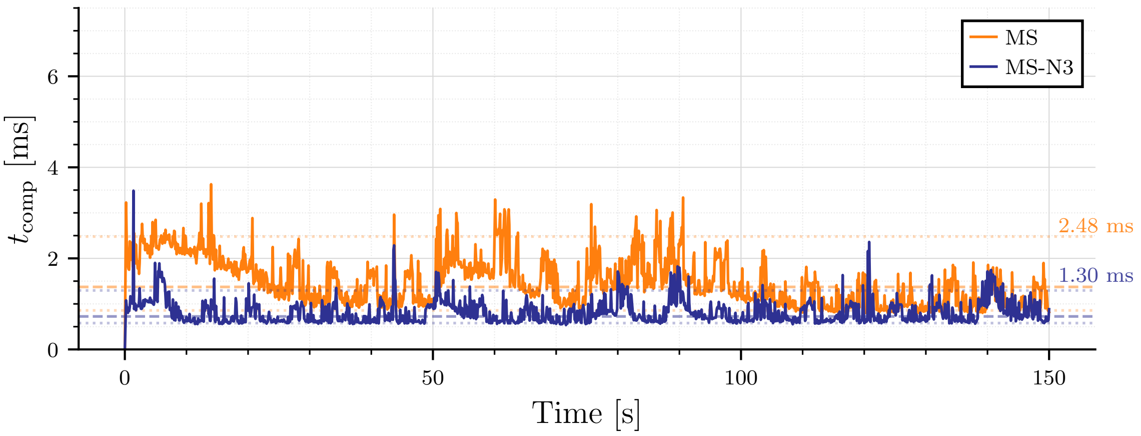


Figure 5.10.: Computation time of MS baseline and MS* on a hybrid maneuver. The dotted lines represent the 5 % and 95 % quantile, the dashed lines represent the median.

5.5. Summary

A BO hyperparameter framework is created to identify the best optimal control method for MPC under a given operating condition. The theoretical basis is the embedding of explicit methods, such as shooting and implicit methods, such as pseudospectral methods in one optimal control formulation. This enabled a global optimization algorithm to find the best method itself, its local scheme and the ideal configuration of the intervals.

The workflow is successfully applied to the OOS framework. The optimal BO-MPC parameterization reduces computation time by approximately 45 %, while also improving closed-loop cost by about 8 % and 44 % in the tracking and hybrid scenarios, respectively.

The framework also exhibits strong potential for broader applicability across diverse control-oriented domains. By further modularising its components, it can enable a more autonomous and scalable workflow for developing and tuning MPC schemes.

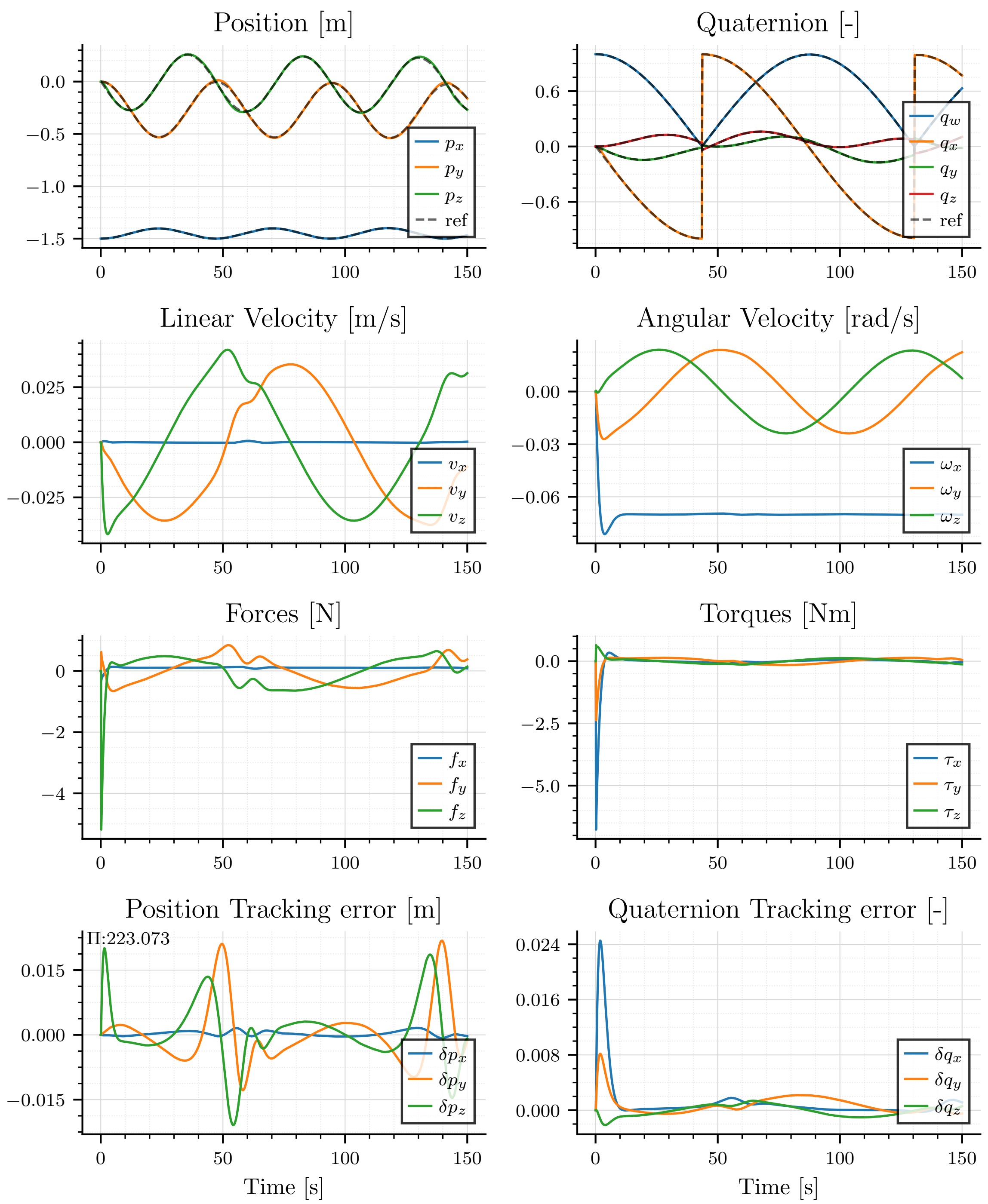


Figure 5.11.: MS baseline in tracking scenario.

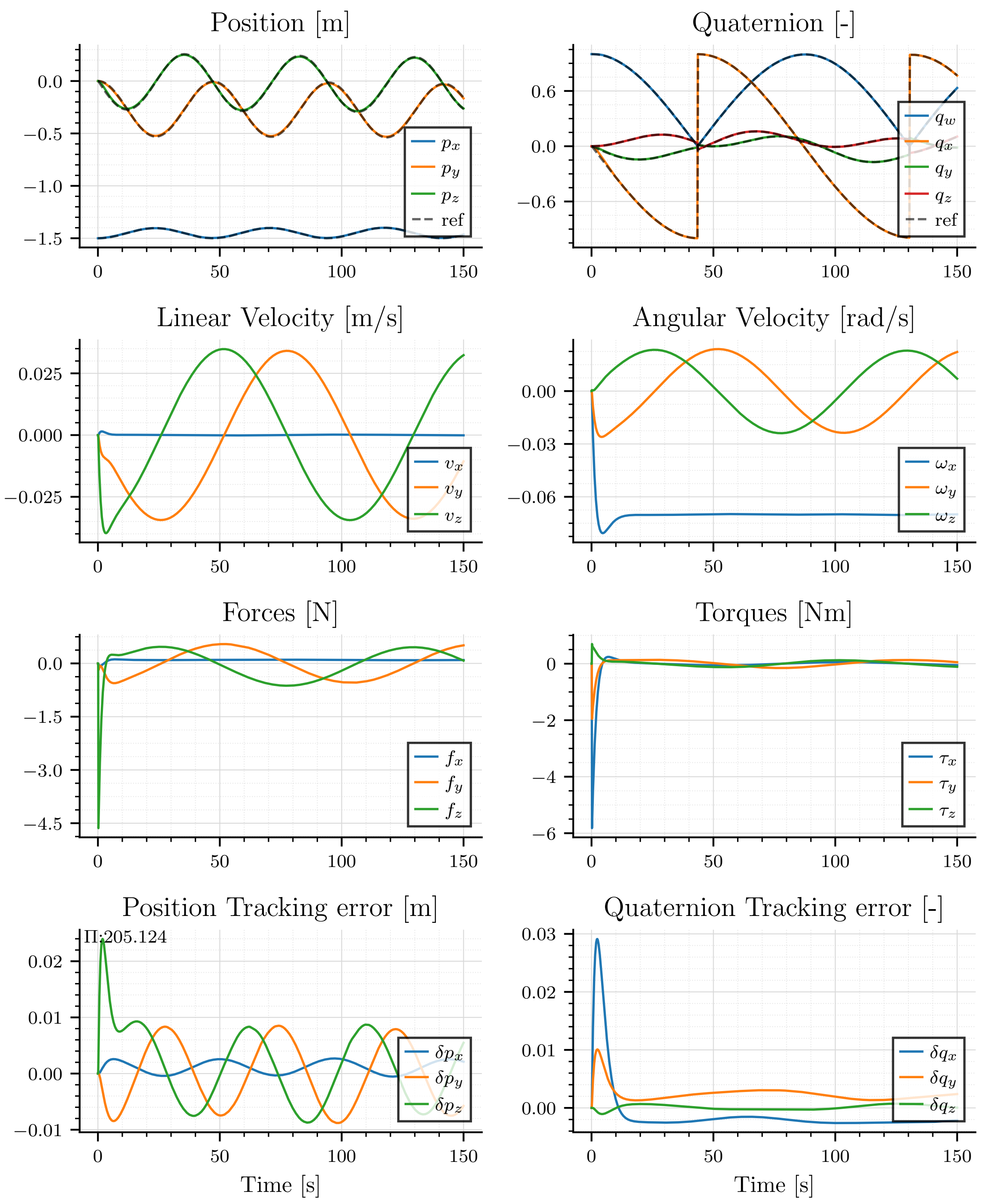


Figure 5.12.: MS* with $N = 3$ in tracking scenario.

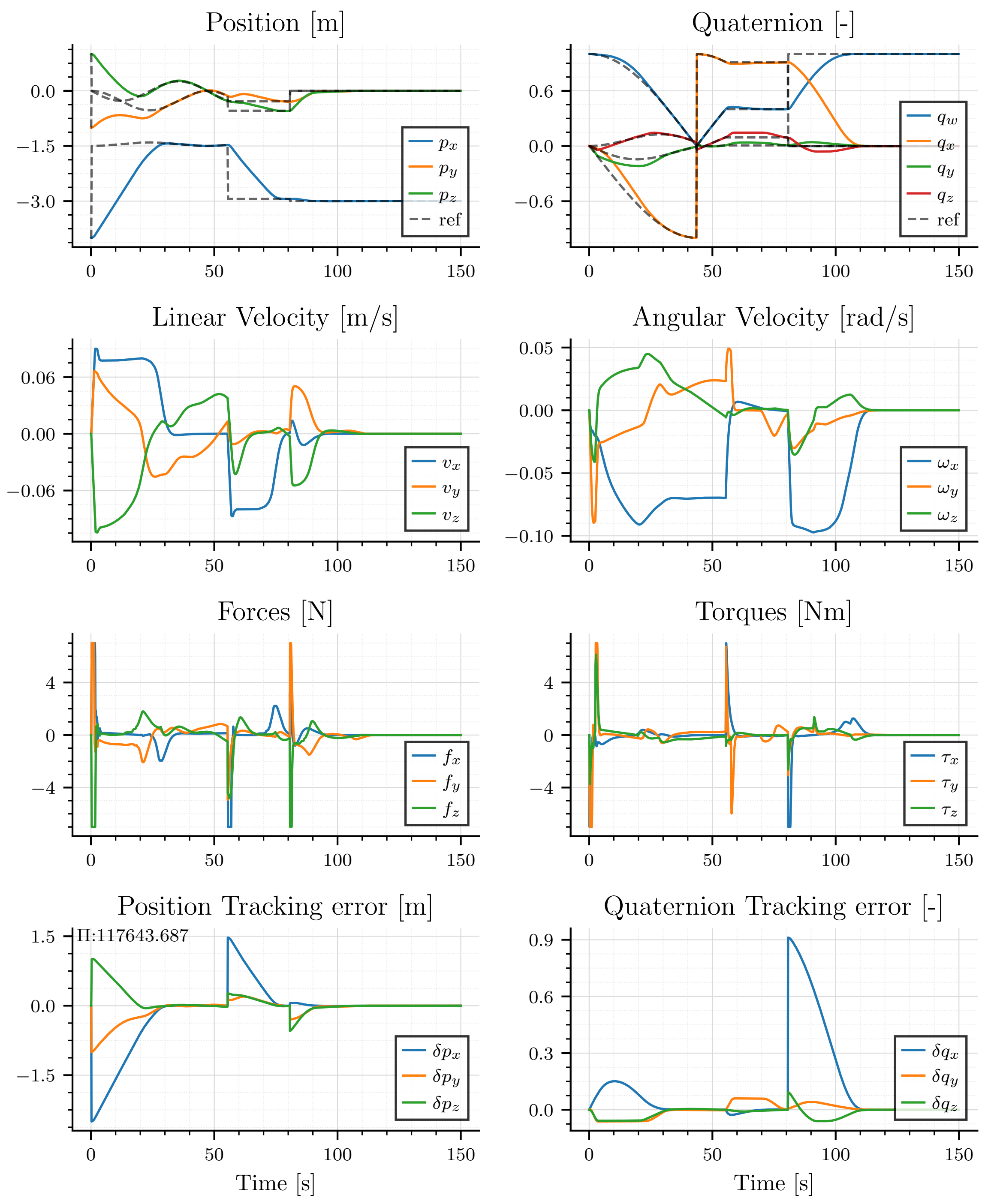


Figure 5.13.: MS baseline in hybrid scenario.

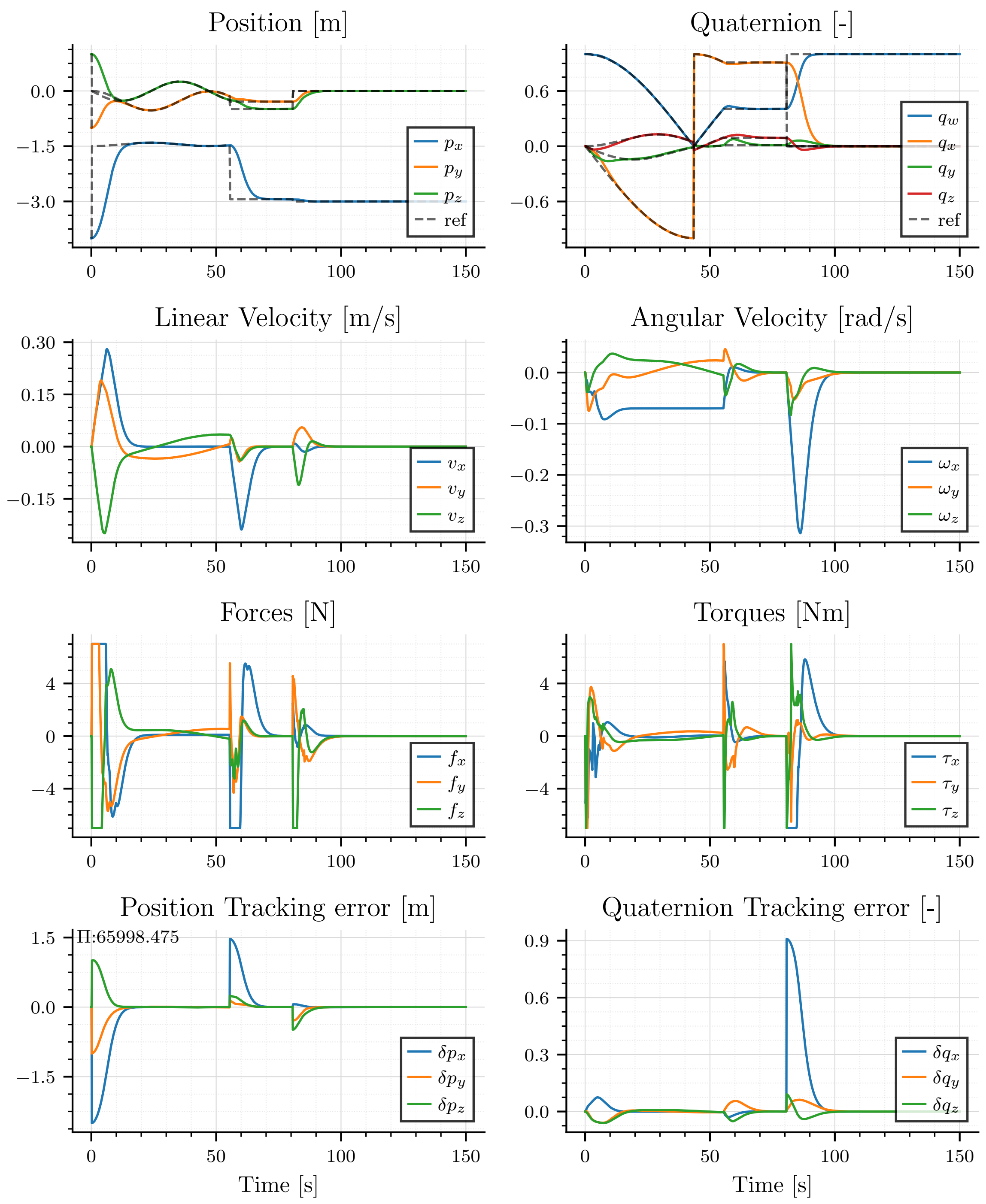


Figure 5.14.: MS* with $N = 3$ in hybrid scenario.

6 Conclusion and Outlook

This thesis provides comparative insights about the question on how to choose the most performant optimal control method for MPC. Specific solver-method combinations and characteristics of the methods were derived in a thorough survey on a benchmark example. The analysis outlined two promising solution candidates that are further implemented in an OOS simulation. Lastly, a universal framework using BO is established that gives a systematic workflow on how to find the optimal method and its configuration for specific design criteria.

The survey of optimal control methods demonstrated the capabilities of each method in combination with different solvers. Apart from explicit schemes such as Multiple-Shooting, mostly implicit collocation method are addressed. Classical equidistant collocation strategies such as Trapezoidal or Hermite-Simpson collocation are implemented aside from non-uniform higher-order local and global pseudospectral schemes. All approaches are evaluated on the nonlinear solvers FATROP [55], IPOPT [57] and SQP [4]. The influence of model dimensionality and couplings is further discussed.

The best trade-off between computational efficiency and closed-loop performance is observed for Multiple-Shooting combined with FATROP. Its sparsity exploitation natively compatible with MS, outperforms the implicit collocation methods that cannot take advantage of this due to their implicit construction. To compensate for this, a local pseudospectral method LGR-LOC* is made compliant by introducing additional optimization variables that preserve the block sparsity of the constraint Jacobian.

Overall, the results of LGR-LOC* on the benchmark example indicate comparable performance as for MS approach. In a consequent step, the above two were analyzed in OOS simulations for a tracking and an escape maneuver. For identical prediction horizons the two methods demonstrated quasi-identical tracking performance. For a hybrid scenario including an escape maneuver the collocation achieved improved closed-loop performance around 8 %. This comes at the cost of increased computation time of 20 % compared to the shooting approach.

Three parameters configure the LGR-LOC* method: the number of collocation nodes per interval, the number of intervals and the spacing of the intervals itself. This description incorporates shooting and collocation approaches in one formulation. Depending on the number of collocation nodes chosen, the LGR-LOC* is either explicit or implicit. This capability enables global optimization to find the most performant optimal control method without implementing each formulation separately.

A BO hyperparameter framework is developed that seeks the optimal choice of the above three mentioned parameters. It evaluates the performance along a multi-objective cost consisting out of the computation time and the closed-loop behavior. For all configurations the sparsity pattern by FATROP is preserved. As a final output, one retrieves the most performant parameterization for the optimal control method in terms of computation time and closed-loop behavior.

This framework is applied in OOS to evaluate the most performant MPC configuration in a tracking and an escape maneuver. A multiple shooting method with three intervals and customized interval spacing is evaluated as the most suitable choice. The combination of using a low number of discretization nodes improves the computation time by 45 % while simultaneously, non-uniform spacing of the interval borders enhances the closed-loop performance for a hybrid scenario about 44 %. The above workflow can be applied to various kinds of MPC-applications with different performance metrics.

Outlook The general characteristics of shooting and collocation methods in the MPC context are compared. Various methods and solver combinations outlined the most-suitable options for real-time implementations. A further valuable contribution to the survey would be the analysis of a more performant SQP solver, such as Acados [56].

The hyperparameter optimization strategy provides a universal framework that optimizes for the most suitable optimal control method without the need to implement each individually. This fact facilitates the design process of MPC significantly. Tailored to the specific MPC formulation (system dynamics, cost function, constraints), either shooting or collocation approaches might be the method of choice. Specifically for stiff ODE the proposed Radau collocation scheme provides a stable alternative to shooting [47, 24]. The used scheme includes the left endpoint of the interval (Radau IA). Only because of this property the derivation of an explicit multiple-shooting was achieved. However, Radau methods that include the right endpoint have even better properties with stiff ODE [24]. Thus, extending the scheme to those methods might be a valuable alternative when dealing with stiff systems. On the negative side, the explicit shooting formulation turns into an implicit backward Euler scheme. This vanishes the key feature of the proposed workflow to integrate explicit and implicit formulations in one formulation.

An additional degree of freedom can be introduced in the BO framework when assuming a non-constant number of collocation nodes in each interval. This adapts the integration accuracy individually in each interval.

Another contribution to a more thorough workflow for finding the most performant MPC is to incorporate the weighting matrices $\mathbf{Q}, \mathbf{R}$ of the cost functional within the BO. This might further improve the closed-loop performance, specifically in combination with choosing the best optimal control method for a specific cost functional.

More specifically for the use-case of MPC in OOS, the results of the hyperparameter optimization merge the fields of trajectory optimization and MPC. The global analysis of the prediction horizon in Figure 5.3 exhibits, that a longer prediction into the future not necessarily improves the control performance further. Instead, the closed-loop performance stagnates when increasing the horizon. A detailed comparison for MPC incorporating trajectory planning and control as a unified module versus a sequential approach using MPC as a reference controller that receives a reference trajectory as an input would further reveal insightful details.

Appendices

A

Appendix: Pseudospectral methods for MPC in On-Orbit Servicing

A.1. Tracking maneuver supplement figures

The closed-loop plot in the tracking case for the MS method can be found in Figure A.1. It shows closely identical behavior as for the LGR-LOC* method shown in Figure 4.3.

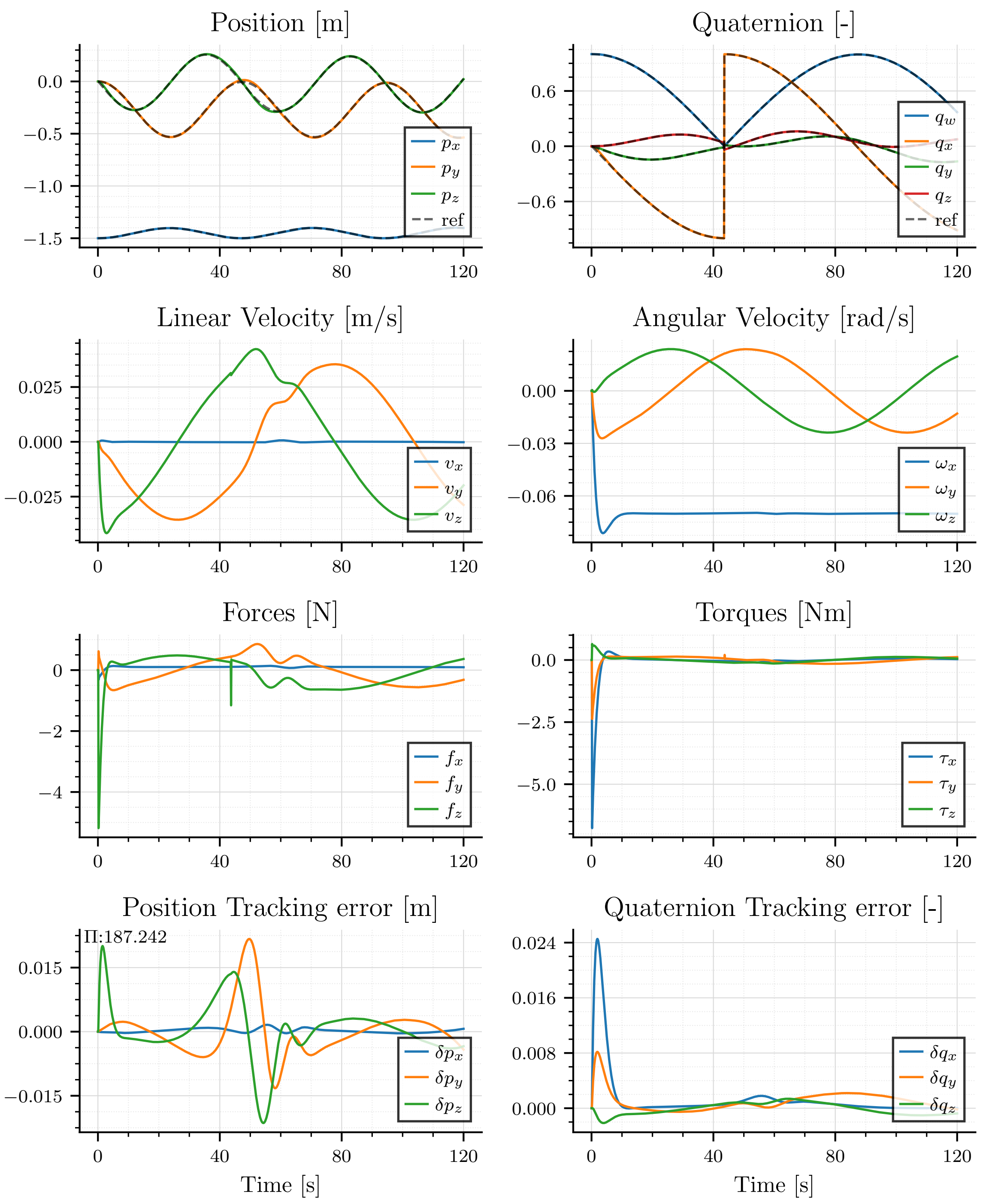


Figure A.1.: Closed-loop plot for the MS method for a tracking maneuver.

B

Appendix: Finding the optimal optimal control method

B.1. Derivation of shooting and collocation methods from LGR-LOC*

This chapter derives the formulation for shooting and collocation methods of the LGR-LOC* method. The node at the current time step is denoted as x_1, the next interval border is denoted as x_2 and all collocation nodes in between the borders are denoted as x_{1i}. The step size h represents the interval length of the corresponding interval $h = t_{\text{adp},2} - t_{\text{adp},1}$. The system dynamics f evaluated at a given node x_i are abbreviated by $f(x_i, u_i) = f_i$ or $f(x_{ij}, u_i) = f_{ij}$.

B.1.1. Derivation of Multiple Shooting: $N_l = 1$

The differentiation matrix for $N_l = 1$, that only includes one node $\tau = -1$ leads to the following constraints (according to Eq. (3.16)):

$$\frac{2}{h}\begin{pmatrix}-\frac{1}{2} & \frac{1}{2}\end{pmatrix}\begin{pmatrix}x_1\\x_2\end{pmatrix} = \begin{pmatrix}f_1\end{pmatrix} \tag{B.1}$$

This is the trivial case that does not require solving a nonlinear system of equations, as i.e. in Section B.1.2. Instead, an explicit formulation for the state of the next interval is retrieved:

$$x_2 - x_1 = hf_1 \qquad \text{(Euler-Integration)} \tag{B.2}$$

The relation in (B.2) is equivalent to the explicit formulation of Euler-Integration method derived in Eq. (2.7). This method is embedded in the Multiple-Shooting (MS) method.

So speaking, the LGR-LOC* method with $N_l = 1$ is mathematically identical to MS when using identical intervals.

This unique property can only be ensured when using the Radau method IA. Since Radau methods do only contain one endpoint of the interval one differs among Radau IA and Radau IIA methods. The first one does not contain the right endpoint, while the second one does not contain the left endpoint [24]. If one were deciding on using Radau IIA methods, the above equivalence of forward Euler-Integration rule would change to Euler-backward resulting in an implicit formulation. This would therefore differ from classical Multiple shooting approaches in an explicit scheme.

B.1.2. Derivation of an adapted Trapezoidal Rule: $N_l = 2$

The differentiation matrix for $N_l = 2$ for any interval is given by evaluating Eq. (2.46) at the Legendre-Gauss-Radau nodes $\tau = [-1, 0.33]$. This leads to the following constraints that ensure the collocation, according to Eq. (3.16):

$$\frac{2}{h}\begin{pmatrix} -\frac{5}{4} & \frac{5}{4} & -1 \\ -\frac{1}{4} & -\frac{3}{4} & 1 \end{pmatrix}\begin{pmatrix} x_1 \\ x_{11} \\ x_2 \end{pmatrix} = \begin{pmatrix} f_1 \\ f_{11} \end{pmatrix} \tag{B.3}$$

The nonlinear system of equations equals to:

$$\frac{2}{h}\left(-\frac{5}{4}x_1 + \frac{5}{4}x_{11} - x_2\right) = f_1 \tag{B.4}$$

$$\frac{2}{h}\left(-\frac{1}{4}x_1 - \frac{3}{4}x_{11} + x_2\right) = f_{11} \tag{B.5}$$

From (B.4) x_{11} can be solved for:

$$x_{11} = -\frac{2h}{9}f_1 + \frac{4}{9}x_2 + \frac{5}{9}x_1$$

Inserting the above expression in (B.5) results in:

$$x_2 - x_1 = \frac{h}{4}(3f_{11} + f_1) \qquad \text{(Radau IA method, see [24])} \tag{B.6}$$

This is the integration rule when using Gauss Radau Quadrature with two nodes [24]. A similarity of Eq. (B.6) with the trapezoidal rule in Eq. (2.26) is evident. Their main difference in construction is the choice of collocation nodes. For the classical trapezoidal rule, equidistant nodes are assumed while for the Radau IA method the roots of an orthogonal Polynomial are used. While the trapezoidal rule has the order $m = 2$ (compare Table 2.1), the Radau IA method is of the order $m = 2N_l - 1 = 3$ and is consequently the more accurate solution.

B.1.3. Derivation of an adapted Hermite-Simpson Rule: $N_l = 3$

In similar fashion as in the previous Section B.1.2, an expression incorporating three collocation nodes $N_l = 3$ can be derived. A detailed derivation is skipped here, but it follows the approach for the adapted trapezoidal rule. The final expression results in

$$x_2 - x_1 = \frac{h}{36}\left(4f_1 + (16+\sqrt{6})f_{11} + (16-\sqrt{6})f_{12}\right) \tag{B.7}$$

that is an adapted version of the Hermite-Simpson rule in (2.36). The derived formula in (B.7) is of the order $m = 2N_l - 1 = 5$ which is one order higher than the Hermite-Simpson rule (see Table 2.1) [24]. This specific subversion of the LGR-LOC* can therefore be seen as a specific type of Hermite-Simpson collocation.

List of Figures

List of Tables

Bibliography

[1] Takuya Akiba et al. “Optuna: A Next-generation Hyperparameter Optimization Framework”. 2019. DOI: 10.48550/arXiv.1907.10902. arXiv: 1907.10902 [cs].

[2] Mohammad Alizadeh and Zheng H. Zhu. “A Comprehensive Survey of Space Robotic Manipulators for On-Orbit Servicing”. In: *Frontiers in Robotics and AI* 11 (2024), p. 1470950. DOI: 10.3389/frobt.2024.1470950.

[3] Jean Pierre Allamaa, Panagiotis Patrinos, Herman Van Der Auweraer, and Tong Duy Son. “Safety Envelope for Orthogonal Collocation Methods in Embedded Optimal Control”. In: *2023 European Control Conference (ECC)*. 2023 European Control Conference (ECC). Bucharest, Romania: IEEE, 2023, pp. 1–7. DOI: 10.23919/ECC57647.2023.10178116.

[4] Joel A. E. Andersson et al. “CasADi: A Software Framework for Nonlinear Optimization and Optimal Control”. In: *Mathematical Programming Computation* 11.1 (2019), pp. 1–36. DOI: 10.1007/s12532-018-0139-4.

[5] Saeid Bayat and James T. Allison. “LGR-MPC: A User-Friendly Software Based on Legendre-Gauss-Radau Pseudo Spectral Method for Solving Model Predictive Control Problems”. 2023. DOI: 10.48550/arXiv.2310.15960. arXiv: 2310.15960 [eess].

[6] James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. “Algorithms for Hyper-Parameter Optimization”. In: *Proceedings of the 25th International Conference on Neural Information Processing Systems*. NIPS’11. Granada, Spain: Curran Associates Inc., 2011, pp. 2546–2554.

[7] John T. Betts. “Survey of Numerical Methods for Trajectory Optimization”. In: *Journal of Guidance, Control, and Dynamics* 21.2 (1998), pp. 193–207. DOI: 10.2514/2.4231.

[8] John T. Betts. *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming*. 2nd ed. Society for Industrial and Applied Mathematics, 2010. DOI: 10.1137/1.9780898718577.

[9] 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), pp. 1603–1608. DOI: 10.1016/S1474-6670(17)61205-9.

[10] Arthur E Bryson. *Applied Optimal Control*. Vol. 2. New York, NY: Taylor & Francis, 1975.

[11] H Chen and Frank Allgower. “A Quasi-Infinite Horizon Nonlinear Model Predictive Control Scheme with Guaranteed Stability”. In: *Automatica* 34.10 (1998), pp. 1205–1217. DOI: `10.1016/S0005-1098(98)00073-9`.

[12] Christopher L. Darby, William W. Hager, and Anil V. Rao. “An Hp-adaptive Pseudospectral Method for Solving Optimal Control Problems”. In: *Optimal Control Applications and Methods* 32.4 (2011), pp. 476–502. DOI: `10.1002/oca.957`.

[13] 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*. Ed. by Moritz Diehl and Katja Mombaur. Vol. 340. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 65–93.

[14] Evgeniya Drozdova, Siegbert Hopfgarten, Evgeny Lazutkin, and Pu Li. “Autonomous Driving of a Mobile Robot Using a Combined Multiple-Shooting and Collocation Method”. In: *IFAC-PapersOnLine* 49.15 (2016), pp. 193–198. DOI: `10.1016/j.ifacol.2016.07.731`.

[15] G. Elnagar, M.A. Kazemi, and M. Razzaghi. “The Pseudospectral Legendre Method for Discretizing Optimal Control Problems”. In: *IEEE Transactions on Automatic Control* 40.10 (1995), pp. 1793–1796. DOI: `10.1109/9.467672`.

[16] Fariba Fahroo and I. Michael Ross. “Advances in Pseudospectral Methods for Optimal Control”. In: *AIAA Guidance, Navigation and Control Conference and Exhibit*. AIAA Guidance, Navigation and Control Conference and Exhibit. Honolulu, Hawaii: American Institute of Aeronautics and Astronautics, 2008. DOI: `10.2514/6.2008-7309`.

[17] Feng Gao et al. “Accurate Pseudospectral Optimization of Nonlinear Model Predictive Control for High-Performance Motion Planning”. In: *IEEE Transactions on Intelligent Vehicles* 8.2 (2023), pp. 1034–1045. DOI: `10.1109/TIV.2022.3153633`.

[18] Javier García-Heras, Manuel Soler, and Francisco J. Sáez. “Collocation Methods to Minimum-Fuel Trajectory Problems with Required Time of Arrival in ATM”. In: *Journal of Aerospace Information Systems* 13.7 (2016), pp. 243–265. DOI: `10.2514/1.I010401`.

[19] Divya Garg et al. “An Overview of Three Pseudospectral Methods for the Numerical Solution of Optimal Control Problems”. In: *Advances in the Astronautical Sciences* 135 (2009).

[20] Divya Garg et al. “Direct Trajectory Optimization and Costate Estimation of Finite-Horizon and Infinite-Horizon Optimal Control Problems Using a Radau Pseudospectral Method”. In: *Computational Optimization and Applications* 49.2 (2011), pp. 335–358. DOI: `10.1007/s10589-009-9291-0`.

[21] German Aerospace Center (DLR). *On-Orbit Servicing Simulator for Orbital Robotics (OOS-SIM)*. 2024. URL: `https://event.dlr.de/en/ila2024/oos-sim/` (visited on 07/10/2025).

[22] Herbert Goldstein, JL Safko, and C Poole. *Classical Mechanics*. 3rd. Pearson Education, Inc, 2002.

[23] Lars Grüne and Jürgen Pannek. *Nonlinear Model Predictive Control*. Communications and Control Engineering. Cham: Springer International Publishing, 2017. DOI: 10.1007/978-3-319-46024-6.

[24] Ernst Hairer and Gerhard Wanner. *Solving Ordinary Differential Equations II: Stiff and Differential-algebraic Problems*. 2nd ed. Vol. 14. Springer Series in Computational Mathematics. Springer Berlin Heidelberg, 1996. DOI: 10.1007/978-3-642-05221-7.

[25] Geoffrey Huntington. "Advancement and Analysis of a Gauss Pseudospectral Transcription for Optimal Control". PhD thesis. Massachusetts: Massachusetts Institute of Technology, 2008.

[26] Geoffrey Huntington, David Benson, and Anil Rao. "A Comparison of Accuracy and Computational Efficiency of Three Pseudospectral Methods". In: *AIAA Guidance, Navigation and Control Conference and Exhibit*. AIAA Guidance, Navigation and Control Conference and Exhibit. Hilton Head, South Carolina: American Institute of Aeronautics and Astronautics, 2007. DOI: 10.2514/6.2007-6405.

[27] Geoffrey T. Huntington and Anil V. Rao. "Comparison of Global and Local Collocation Methods for Optimal Control". In: *Journal of Guidance, Control, and Dynamics* 31.2 (2008), pp. 432–436. DOI: 10.2514/1.30915.

[28] Matthew Kelly. "An Introduction to Trajectory Optimization: How to Do Your Own Direct Collocation". In: *SIAM Review* 59.4 (2017), pp. 849–904. DOI: 10.1137/16M1062569.

[29] Donald J. Kessler and Burton G. Cour-Palais. "Collision Frequency of Artificial Satellites: The Creation of a Debris Belt". In: *Journal of Geophysical Research: Space Physics* 83.A6 (1978), pp. 2637–2646. DOI: 10.1029/JA083iA06p02637.

[30] Donald E Kirk. *Optimal Control Theory: An Introduction*. Carmel, California: Courier Corporation, 2004.

[31] Johannes Köhler, Matthias A. Müller, and Frank Allgöwer. "A Nonlinear Model Predictive Control Framework Using Reference Generic Terminal Ingredients – Extended Version". In: *IEEE Transactions on Automatic Control* 65.8 (8 2020), pp. 3576–3583. DOI: 10.1109/TAC.2019.2949350. arXiv: 1909.12765 [eess].

[32] Peter Kötting. "Model Predictive Control Based Coordinated Control for Free-Flying Space Manipulator Systems". In: *IFAC-PapersOnLine* 58.18 (18 2024), pp. 133–138. DOI: 10.1016/j.ifacol.2024.09.021.

[33] Roberto Lampariello. "Orbital Robotics". In: *Robotics Goes MOOC: Impact*. Ed. by Bruno Siciliano. Cham: Springer Nature Switzerland, 2025, pp. 227–323. DOI: 10.1007/978-3-319-77267-7.

[34] P. J. Larcombe. “On the Control of a Two-Dimensional Multi-Link Inverted Pendulum: The Form of the Dynamic Equations from Choice of Co-Ordinate System”. In: *International Journal of Systems Science* 23.12 (1992), pp. 2265–2289. DOI: `10.1080/00207729208949454`.

[35] K. Lee, W. H. Moase, and C. Manzie. “Mesh Adaptation in Direct Collocated Nonlinear Model Predictive Control”. In: *International Journal of Robust and Nonlinear Control* 28.15 (2018), pp. 4624–4634. DOI: `10.1002/rnc.4235`.

[36] Fengjin Liu, William W. Hager, and Anil V. Rao. “Adaptive Mesh Refinement Method for Optimal Control Using Nonsmoothness Detection and Mesh Size Reduction”. In: *Journal of the Franklin Institute* 352.10 (2015), pp. 4081–4106. DOI: `10.1016/j.jfranklin.2015.05.028`.

[37] Yingjie Liu, Dawei Cui, and Wen Peng. “Minimum-Time Lane Changing Problem of Vehicle Handling Inverse Dynamics Based on Adaptive Mesh Refinement and Collocation Optimization Method”. In: *Journal of Vibroengineering* 25.6 (2023), pp. 1198–1216. DOI: `10.21595/jve.2023.23085`.

[38] Boyu Ma, Zainan Jiang, Yang Liu, and Zongwu Xie. “Advances in Space Robots for On-Orbit Servicing: A Comprehensive Review”. In: *Advanced Intelligent Systems* 5.8 (2023), p. 2200397. DOI: `10.1002/aisy.202200397`.

[39] Amrith Mariappan and John L. Crassidis. “Kessler’s Syndrome: A Challenge to Humanity”. In: *Frontiers in Space Technologies* 4 (2023), p. 1309940. DOI: `10.3389/frspt.2023.1309940`.

[40] F. Landis Markley and John L. Crassidis. *Fundamentals of Spacecraft Attitude Determination and Control*. New York, NY: Springer New York, 2014. DOI: `10.1007/978-1-4939-0802-8`.

[41] Inc. Mathworks. *Matlab Version: 24.2.0.2740171 (R2024b) Update 1*. Natick, Massachusetts, United States: The MathWorks Inc., 2024.

[42] MOSEK ApS. *MOSEK Optimization Suite Version 10.0*. manual. Copenhagen, Denmark: MOSEK ApS, 2023.

[43] Claus-Dieter Munz and Thomas Westermann. *Numerische Behandlung gewöhnlicher und partieller Differenzialgleichungen: Ein anwendungsorientiertes Lehrbuch für Ingenieure*. Berlin, Heidelberg: Springer, 2019. DOI: `10.1007/978-3-662-55886-7`.

[44] Yoshihiko Ozaki, Yuki Tanigaki, Shuhei Watanabe, and Masaki Onishi. “Multiobjective Tree-Structured Parzen Estimator for Computationally Expensive Optimization Problems”. In: *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*. Gecco ’20. Cancún, Mexico: Association for Computing Machinery, 2020, pp. 533–541. DOI: `10.1145/3377930.3389817`.

[45] Michael A. Patterson, William W. Hager, and Anil V. Rao. “A *Ph* Mesh Refinement Method for Optimal Control”. In: *Optimal Control Applications and Methods* 36.4 (2014), pp. 398–421. DOI: `10.1002/oca.2114`.

[46] Anil Rao. “A Survey of Numerical Methods for Optimal Control”. In: *Advances in the Astronautical Sciences* 135 (2010).

[47] James Blake Rawlings, David Q. Mayne, and Moritz Diehl. *Model Predictive Control: Theory, Computation, and Design*. 2nd edition. Madison, Wisconsin: Nob Hill Publishing, 2017. 623 pp.

[48] Thomas Richter, Henry Von Wahl, and Thomas Wick. *Einführung in die Numerische Mathematik: Begriffe, Konzepte und zahlreiche Anwendungsbeispiele*. Berlin, Heidelberg: Springer, 2024. DOI: `10.1007/978-3-662-69582-1`.

[49] Guido Sanchez et al. “MPC for Nonlinear Systems: A Comparative Review of Discretization Methods”. In: *2017 XVII Workshop on Information Processing and Control (RPIC)*. 2017 XVII Workshop on Information Processing and Control (RPIC). Mar del Plata: IEEE, 2017, pp. 1–6. DOI: `10.23919/rpic.2017.8214333`.

[50] Jie Shen, Tao Tang, and Li-Lian Wang. *Spectral Methods: Algorithms, Analysis and Applications*. Vol. 41. Springer Series in Computational Mathematics. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011. DOI: `10.1007/978-3-540-71041-7`.

[51] Bruno Siciliano, Lorenzo Sciavicco, Luigi Villani, and Giuseppe Oriolo. *Robotics*. Advanced Textbooks in Control and Signal Processing. London: Springer, 2009. DOI: `10.1007/978-1-84628-642-1`.

[52] B. Stellato et al. “OSQP: An Operator Splitting Solver for Quadratic Programs”. In: *Mathematical Programming Computation* 12.4 (2020), pp. 637–672. DOI: `10.1007/s12532-020-00179-2`.

[53] Qichao Sun, Juanjuan Xu, and Huanshui Zhang. “Guidance for Hypersonic Reentry Using Nonlinear Model Predictive Control and Radau Pseudospectral Method”. In: *Optimal Control Applications and Methods* 46.4 (2025), pp. 1417–1428. DOI: `10.1002/oca.3267`.

[54] Lloyd N. Trefethen. *Spectral Methods in MATLAB*. Software, Environments, Tools 10. Philadelphia, Pa: Society for Industrial and Applied Mathematics, 2000. 165 pp. DOI: `10.1137/1.9780898719598`.

[55] Lander Vanroye, Ajay Sathya, Joris De Schutter, and Wilm Decré. “FATROP : A Fast Constrained Optimal Control Problem Solver for Robot Trajectory Optimization and Control”. 2023. DOI: `10.48550/arXiv.2303.16746`. arXiv: `2303.16746 [math]`.

[56] Robin Verschueren et al. “Acados: A Modular Open-Source Framework for Fast Embedded Optimal Control”. 2020. DOI: `10.48550/arXiv.1910.13753`. arXiv: `1910.13753 [math]`.

[57] Andreas Wächter and Lorenz T. Biegler. “On the Implementation of an Interior-Point Filter Line-Search Algorithm for Large-Scale Nonlinear Programming”. In: *Mathematical Programming* 106.1 (2006), pp. 25–57. DOI: `10.1007/s10107-004-0559-y`.

[58] Qian Wang, Shunli Li, Yanquan Zhang, and Min Cheng. “Spacecraft Close Proximity to Noncooperative Target Based on Pseudospectral Convex Method”. In: *Journal of Aerospace Engineering* 37.4 (2024), p. 04024034. DOI: `10.1061/JAEEEZ.ASENG-5099`.

[59] Isaac E. Weintraub, Richard G. Cobb, William Baker, and Meir Pachter. “Direct Methods Comparison for the Active Target Defense Scenario”. In: *AIAA Scitech 2020 Forum*. AIAA Scitech 2020 Forum. Orlando, FL: American Institute of Aeronautics and Astronautics, 2020. DOI: `10.2514/6.2020-0612`.