TUM School of Computation, Information and Technology
Technische Universität München

Correct-by-Construction Controller Synthesis for Safety-Critical Systems

Victor Gaßmann

Vollständiger Abdruck der von der TUM School of Computation, Information and Technology der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften (Dr.-Ing.)

genehmigten Dissertation.

Vorsitz:

Prof. Dr. Debarghya Ghoshdastidar

Prüfende der Dissertation:

1. Prof. Dr.-Ing. Matthias Althoff
2. Prof. Dr. Majid Zamani,
   University of Colorado Boulder, USA

Die Dissertation wurde am 01.02.2024 bei der Technischen Universität München eingereicht und durch die TUM School of Computation, Information and Technology am 08.04.2024 angenommen.

Foreword

This thesis summarizes my research over the past years at the Technical University of Munich. First, I would like to thank my supervisor Prof. Matthias Althoff who gave me the chance to pursue this PhD. His academic guidance and invaluable feedback has profoundly impacted the quality of my work, for which I extend my sincere appreciation and gratitude.

I also thank all of my colleagues for the great moments we had together during my time at the chair. The atmosphere was always welcoming and all the fun activities, afternoons with cake, and stimulating conversations were a nice change of pace, especially when things did not go according to plan. I would especially like to thank Lukas Schäfer and Mark Wetzlinger, who always had an open ear and provided me with many invaluable comments and suggestions; I always very much enjoyed our fruitful and honest conversations. I would also like to thank Dr. Niklas Kochdumper and Adrian Kulmburg for their helpful suggestions and valuable insight.

Last, but certainly not least, I am especially grateful to my family and friends. Thank you to my parents Theresia and Volker who always encouraged and supported me. And thank you to my brother David and his wife Lena who were always very inclusive, and showed sympathy and understanding when I had hard times.

Abstract

Due to recent advances in the field of formal verification, the safety of complex autonomous systems, such as a spaceship attempting to dock or a robot collaborating with humans, may be formally verified. The verification of such reach-avoid problems, where a set of initial states is to be steered to a given target under input and state constraints, is often non-constructive since it only provides a yes-or-no answer. As a result, first synthesizing the controller and then adapting it to guarantee safety turns out to be difficult. In this thesis, we thus combine numerical optimization and set-based reachability analysis to synthesize controllers that ensure formal guarantees by construction.

The efficient optimization of these controllers requires differentiability of the considered synthesis problems. Hence, we first introduce an abstraction of these synthesis problems, for which differentiability can be shown and which then implies differentiability of all considered synthesis problems.

Since an exact formulation of these problems is often hard to solve, we need to derive tractable approximations thereof. As we compute with sets, this often requires the use of different set representations, as they are closed under different set operations. To that end, we present the efficient conversions between different set representations. Additionally, we describe set containment checks from the literature, which are required to check input and constraint satisfaction for sets of states.

To synthesize piecewise constant controllers, we start by extending existing approaches from previous work. To improve controller performance, for instance, we extend the class of control laws that can be synthesized. Further, we propose a novel synthesis algorithm based on trust regions, which iteratively synthesizes a piecewise constant controller under input and state constraints. In contrast to previous work, this synthesis algorithm does not require additional user input to achieve non-conservative results for state-constrained systems. All presented algorithms have polynomial complexity in the state dimension and we demonstrate their advantages compared to other state-of-the-art approaches using numerical experiments.

To incorporate continuous state feedback, we present a novel algorithm for the combined synthesis of a piecewise constant feedforward controller with continuous state feedback. In contrast to other state-of-the-art approaches, we do not synthesize the feedforward controller and the state feedback separately but combine the synthesis into a single optimization problem and thus avoid the introduction of additional user-provided algorithm parameters. The complexity of the proposed algorithm is polynomial in the state dimension. As numerical experiments demonstrate, it outperforms existing synthesis approaches while requiring reduced user input compared to the current state of the art.

Zusammenfassung

Die Sicherheit komplexer autonomer Systeme – wie beispielsweise das Andocken von Raumschiffen oder die Kollaboration zwischen Mensch und Roboter – kann dank jüngster Fortschritte im Bereich der formalen Verifikation sichergestellt werden. Der Sicherheitsbeweis solcher "reach-avoid"-Probleme, bei welchen die initiale Zustandsmenge eines Systems zu einem gegebenen Zielzustand unter Einhaltung gegebener Spezifikationen gesteuert werden soll, liefert dabei häufig lediglich eine nicht-konstruktive Ja-Nein-Aussage. Somit ist die Trennung von Design und anschließender Verifikation des geregelten Systems oft schwierig. In dieser Arbeit kombinieren wir daher numerische Optimierung mit mengenbasierter Erreichbarkeitsanalyse, wodurch die so berechneten Regler die gewünschten formalen Garantien "by design" erfüllen.

Eine effiziente Optimierung dieser Regler setzt allerdings die Differenzierbarkeit des zugrundeliegenden Synthese-Problems voraus. Daher führen wir zu Beginn ein abstrahiertes Synthese-Problem ein, welches alle relevanten Optimierungsprobleme generalisiert und zeigen die Differenzierbarkeit dieser Abstrahierung, was damit direkt die Differenzierbarkeit aller abstrahierten Synthese-Probleme impliziert.

Da eine exakte Formulierung dieser Probleme oft schwierig zu lösen ist, müssen effizientere Approximationen konstruiert werden. Da wir in dieser Arbeit mit Mengen rechnen, ist dazu oft die Zuhilfenahme verschiedener Mengenrepräsentationen nötig, da unterschiedliche Repräsentationen unter verschiedenen Mengenoperationen geschlossen sind. Daher zeigen wir die effiziente Konvertierung verschiedener Mengenrepräsentationen. Zur Überprüfung von Zustands- und Eingangsbeschränkungen für Mengen beschreiben wir außerdem Methoden zur Überprüfung der Teilmengen-Eigenschaft zweier Mengen aus der Literatur.

Zur Synthese von stückweise konstanten Regelgesetzen beginnen wir zunächst mit der Erweiterung existierender Ansätze zur Regler-Synthese aus vorangegangenen Arbeiten. Um die Leistung der berechneten Regler zu verbessern, erweitern wir beispielsweise die Klasse an berechenbaren Regelgesetzen. Wir präsentieren zudem einen neuartigen Synthese-Algorithmus, welcher – basierend auf sogenannten "Trust Regions" – iterativ ein stückweise konstantes Regelgesetz berechnet, welches sowohl Zustands- als auch Eingangsspezifikationen berücksichtigt. Dabei benötigt der präsentierte Algorithmus – im Gegensatz zu vorhandenen Ansätzen – neben den gegebenen Spezifikationen keine zusätzlichen Angaben des Nutzers, um performante Regelgesetze zu berechnen. Die beschriebenen Methoden haben hierbei alle eine polynomielle Laufzeit im Bezug auf die Zustandsdimension. Wir demonstrieren die Vorteile der entwickelten Algorithmen im Vergleich zu existierenden Methoden mit Hilfe numerischer Experimente.

Zur Kombination des stückweise konstanten Regelgesetzes mit einem kontinuierlichem Feedback-Regler beschreiben wir dann einen Synthese-Algorithmus für das kombinierte

Regelgesetz. Im Gegensatz zu bekannten Ansätzen synthetisieren wir dabei den stückweise konstanten und den kontinuierlichen Feedback-Regler in einem Optimierungsproblem, wodurch wir zusätzliche Parameter vermeiden, die vom Nutzer spezifiziert werden müssten. Der beschriebene Algorithmus hat polynomielle Laufzeit im Bezug auf die Zustandsdimension, benötigt im Vergleich zu existierenden Ansätzen reduzierten Nutzereingriff, und zeigt in numerischen Experimenten eine verbesserte Regelperformance.

Contents

List of Figures

List of Tables

Acronyms

ACADO	automatic control and dynamic optimization[1] [55]
AI	artifical intelligence
AROC	automated reachset optimal control[2] [63]
CBF	control barrier function
CLF	control Lyapunov function
CORA	continuous reachability analyzer[3] [2]
GSC	generator-space control
HJB	Hamilton-Jacobi-Bellman
iPGSC	iterative polynomial generator-space control
IPOPT	interior point optimizer[4] [111]
iPROC	iterative polynomial reachset optimal control
KKT	Karush-Kuhn-Tucker
LCQ	Linearity constraint qualification
LIDAR	light detection and ranging
LMI	linear matrix inequality
LP	linear program
LQR	linear-quadratic regulator
LTI	linear time-invariant
MATLAB	matrix Laboratory
MFCQ	Mangasarian-Fromovitz constraint qualification
MILP	mixed-integer linear program
MIQP	mixed-integer quadratic program
MPC	model predictive control
MVEE	minimum-volume enclosing ellipsoid
MVIE	maximum-volume inscribed ellipsoid

[1] `https://acado.github.io/`
[2] `https://aroc.in.tum.de/`
[3] `https://cora.in.tum.de/`
[4] `https://github.com/coin-or/Ipopt`

ODE	ordinary differential equation
PCA	principle component analysis
PGSC	polynomial generator-space control
ROC	reachset optimal control
SC	Slater's condition
SDP	semi-definite program

Notations

In this thesis, we denote scalars and vectors with lowercase letters, matrices with uppercase letters, and sets with uppercase calligraphic letters.

Definitions

Number Sets

The set of real numbers, non-negative real numbers, and positive real numbers is denoted by $\mathbb{R}$, $\mathbb{R}_{\geq 0}$, and $\mathbb{R}_+$. Further, the set of complex numbers is denoted by $\mathbb{C}$. Additionally, $\mathbb{N}$ and $\mathbb{N}_+$ denote the sets of natural and positive natural numbers. Lastly, the sets of symmetric, symmetric semi-positive definite, and symmetric positive definite matrices are given by $\mathbb{S}$, $\mathbb{S}_+$, and $\mathbb{S}_{++}$.

Vectors, Matrices, and Sets

The element of a matrix $M \in \mathbb{R}^{n \times m}$ in the i-th row and j-th column for $1 \leq i \leq n$ and $1 \leq j \leq m$ is denoted by M_{ij}; further, we denote with $M_{(i,:)}$ and $M_{(:,j)}$ the i-th row and j-th column of M, and define the shorthand $M^T_{(i,:)} = \left(M_{(i,:)}\right)^T$ and $M^T_{(:,j)} = \left(M_{(:,j)}\right)^T$. For a vector $v \in \mathbb{R}^n$, we similarly denote with $v_i \in \mathbb{R}$ the i-th component of $v \in \mathbb{R}^n$ for $1 \leq i \leq n$. For $a \in \mathbb{N}$ and $b \in \mathbb{N}$ with $a \leq b$, the vector of all integers between a and b (both included) is denoted by $a : b = \begin{bmatrix} a, & a+1, & \ldots, & b \end{bmatrix}$; for a matrix $M \in \mathbb{R}^{n \times m}$, we extend this notation and denote with $M_{(:,a:)} = M_{(:,a:m)}$ the matrix consisting of the a-th to the last column with $a \leq m$. Similarly, we denote with $\mathcal{M}_{(a:b)}$ the projection of the set $\mathcal{M} \subseteq \mathbb{R}^n$ onto its a-th to b-th dimension, where $a \leq b \leq n$. For two matrices of equal size, inequality relations between them are to be interpreted element-wise. Moreover, 1_n and $I_n = \begin{bmatrix} e^{(1)}_{(n)}, & \ldots, & e^{(n)}_{(n)} \end{bmatrix}$ denote the n-dimensional all-ones vector and the n-dimensional identity matrix, where $e^{(i)}_{(n)} \in \mathbb{R}^n$ denotes the i-th unit vector of dimension n for $1 \leq i \leq n$.

Mathematical Operators

For a given vector $v \in \mathbb{R}^n$, $\operatorname{diag}(v) \in \mathbb{R}^{n \times n}$ denotes the diagonal matrix with entries of v on its diagonal. Further, $\operatorname{mod}(v, k) = \begin{bmatrix} \operatorname{mod}(v_1, k), & \operatorname{mod}(v_2, k), & \ldots, & \operatorname{mod}(v_n, k) \end{bmatrix}$ denotes the modulo operation on all components of $v \in \mathbb{R}^n$ where $k \in \mathbb{R}$. The trace of a square matrix $M \in \mathbb{R}^{n \times n}$ is defined as $\operatorname{trace}(M) = \sum_{i=1}^n M_{ii}$. For two sets

$\mathcal{A} \in \mathbb{N}^n$ and $\mathcal{B} \in \mathbb{N}^n$, their set difference is given by $\mathcal{A} \setminus \mathcal{B} = \{a \in \mathcal{A} \mid a \notin \mathcal{B}\}$. Moreover, $|\mathcal{S}| \in \mathbb{N}$ denotes the cardinality of a set $\mathcal{S} \in \mathbb{N}^n$. Further, the empty set is denoted by $\emptyset$. The transpose of a real matrix $R \in \mathbb{R}^{m \times n}$ and the conjugate transpose of a complex matrix $C \in \mathbb{C}^{m \times n}$ are denoted by R^T and C^H, respectively. The minimum and maximum entry of a given matrix $A \in \mathbb{R}^{n \times m}$ are denoted by $\min(A)$ and $\max(A)$, respectively; for $B \in \mathbb{R}^{n \times m}$, we further denote with $\min(A, B) \in \mathbb{R}^{n \times m}$ ("max" analogous) the matrix that results from taking the minimum over each matrix element separately. For a finite number of vectors $v^{(i)} \in \mathbb{R}^n$ with $1 \leq i \leq m$, we denote by $\min_{1 \leq i \leq m} v^{(i)} = \left[\min_{1 \leq i \leq m} v_1^{(i)}, \quad \ldots, \quad \min_{1 \leq i \leq m} v_n^{(i)}\right]^T$ the finite minimization over each vector component, where the result is again collected in an n-dimensional vector (analogous for max). Further, $A \boxtimes B \in \mathbb{R}^{mp \times nq}$ denotes the Kronecker product of matrices $A \in \mathbb{R}^{m \times n}$ and $B \in \mathbb{R}^{p \times q}$, and $C \boxplus D = C \boxtimes I_m + I_n \boxtimes D$ denotes the Kronecker sum for $C \in \mathbb{R}^{n \times n}$ and $D \in \mathbb{R}^{m \times m}$. For a set of m points $\left\{v^{(1)}, v^{(2)}, \ldots, v^{(m)}\right\}$ with $v^{(i)} \in \mathbb{R}^n$ and $1 \leq i \leq m$, we denote its convex hull, i.e., the smallest convex set containing all $v^{(i)}$, with $\operatorname{convh}\left(\left\{v^{(1)}, v^{(2)}, \ldots, v^{(m)}\right\}\right) \subset \mathbb{R}^n$. For a complex matrix $C \in \mathbb{C}^{n \times m}$, $\operatorname{real}(C) \in \mathbb{R}^{n \times m}$ and $\operatorname{imag}(C)$ denote the real and imaginary part of C, respectively. For a matrix $M \in \mathbb{S}_+^{n \times n}$, we alternatively write $M \succeq 0$.

Functions

For a function $f : \mathbb{R} \mapsto \mathbb{R}^n$ and time $t \in \mathbb{R}_{\geq 0}$, its time derivative is denoted by $\dot{f}(t) = \left[\frac{df_1(t)}{dt}, \quad \ldots, \quad \frac{df_n(t)}{dt}\right]^T$. For once and twice continuously differentiable functions $f : \mathbb{R}^n \mapsto \mathbb{R}^m$ and $g : \mathbb{R}^n \mapsto \mathbb{R}$ with vector $x \in \mathbb{R}^n$, the Jacobian matrix $J_f(x) \in \mathbb{R}^{m \times n}$ of g is defined by $[J_f(x)]_{ij} = \frac{\partial f_i(x)}{\partial x_j}$ for $1 \leq i \leq m$ and $1 \leq j \leq n$, and the Hessian matrix $H_g(x) \in \mathbb{R}^{n \times n}$ of g is defined by $[H_g(x)]_{ij} = \frac{\partial^2 g_i(x)}{\partial x_i \partial x_j}$ for $1 \leq i \leq n$ and $1 \leq j \leq n$.

List of Symbols

Scalars

Symbol	Dimension	Description
t	$\in \mathbb{R}_{\geq 0}$	current point in time
t_f	$\in \mathbb{R}_+$	time horizon length
m_s	$\in \mathbb{N}_+$	number of discrete-time feedback steps
m_c	$\in \mathbb{N}_+$	number of piecewise constant control inputs per feedback step
t_s	$\in \mathbb{R}$	time between discrete-time feedback
t_c	$\in \mathbb{R}_+$	duration of piecewise constant control inputs per feedback step
a	$\in \mathbb{N}_+$	number of monomials for the controller template
κ	$\in \mathbb{N}_+$	controller order

Symbol	Dimension	Description
π	$\in \mathbb{N}_+$	abstraction order for reachable set computations
o	$\in \mathbb{N}_+$	reduction order for zonotopes and polynomial zonotopes
γ	$\in (0, 1]$	trust-region radius for piecewise constant controller parameters P
η	$\in (0, 1]$	trust-region radius for feedback controller parameters Q and R
m_d	$\in \mathbb{N}_+$	number of steps for the time discretization of time-varying matrices
n_x	$\in \mathbb{N}_+$	state dimension
n_u	$\in \mathbb{N}_+$	input dimension
n_w	$\in \mathbb{N}_+$	disturbance dimension
$o_{\mathcal{X}_\mathrm{f}}$	$\in \mathbb{N}_+$	number of halfspaces making up the final state constraints
$o_\mathcal{X}$	$\in \mathbb{N}_+$	number of halfspaces making up the state constraints
$o_\mathcal{U}$	$\in \mathbb{N}_+$	number of halfspaces making up the input constraints
h	$\in \mathbb{N}_+$	extended optimization horizon
m_r	$\in \mathbb{N}_+$	number of steps for reachability analysis
μ	$\in \mathbb{R}_+$	optimality tolerance
σ	$\in \mathbb{R}_+$	constraint penality multiplier
ζ	$\in \mathbb{R}_+$	input penalty factor
l_max	$\in \mathbb{N}_+$	maximum number of iterations for a given algorithm
n_z	$\in \mathbb{N}_+$	number of controller parameters for the combined controller
ω	$\in \mathbb{R}_+$	exponent for the computational complexity of matrix multiplication

Vectors

Symbol	Dimension	Description
$x(t)$	$\in \mathbb{R}^{n_x}$	state vector
$u(t)$	$\in \mathbb{R}^{n_u}$	controllable input vector
$w(t)$	$\in \mathbb{R}^{n_w}$	uncontrollable input vector (disturbance)
x_f	$\in \mathbb{R}^{n_x}$	target state
x_ref	$\in \mathbb{R}^{n_x}$	reference state trajectory
u_ref	$\in \mathbb{R}^{n_u}$	reference input trajectory
ϑ	$\in \mathbb{R}^h_{\geq 0}$	weight vector for the objective function using the extended horizon h

Matrices

Symbol	Dimension	Description
P	$\in [-1, 1]^{m_\mathrm{c} m_\mathrm{s} n_u \times a}$	controller parameter matrix for all $m_\mathrm{c} m_\mathrm{s}$ steps
Q_ref	$\in \mathbb{S}_+^{n_x \times n_x}$	state weighting matrix for reference trajectory

Symbol	Dimension	Description
R_{ref}	$\in \mathbb{S}_{+}^{n_u \times n_u}$	input weighting matrix for reference trajectory
K	$\in \mathbb{R}^{n_u \times n_x}$	state feedback matrix
Q	$\in \mathbb{S}_{++}^{n_x \times n_x}$	state weighting matrix for LQR control
R	$\in \mathbb{S}_{++}^{n_u \times n_u}$	input weighting matrix for LQR control

Sets

Symbol	Dimension	Description
$\mathcal{R}^{(e)}$	$\subset \mathbb{R}^{n_x}$	exact reachable set
$\mathcal{R}$	$\subset \mathbb{R}^{n_x}$	outer approximation of the reachable set
$\tilde{\mathcal{R}}$	$\subset \mathbb{R}^{n_x}$	approximation (without formal guarantees) of the reachable set
$\mathcal{X}^{(0)}$	$\subset \mathbb{R}^{n_x}$	set of initial states
$\bar{\mathcal{X}}^{(0)}$	$\subset \mathbb{R}^{n_x}$	parallelotope outer approximation of $\mathcal{X}^{(0)}$
$\mathcal{U}$	$\subset \mathbb{R}^{n_u}$	input constraint set
$\mathcal{W}$	$\subset \mathbb{R}^{n_w}$	disturbance set
$\mathcal{X}$	$\subseteq \mathbb{R}^{n_x}$	state constraint set
$\mathcal{X}_\mathrm{f}$	$\subseteq \mathbb{R}^{n_x}$	final state constraint set
$\tilde{\mathcal{X}}$	$\subseteq \mathbb{R}^{n_x}$	adapted state constraint set
$\tilde{\mathcal{U}}$	$\subset \mathbb{R}^{n_u}$	adapted input constraint set with $\tilde{\mathcal{U}} \subseteq \mathcal{U}$
$\tilde{\mathcal{X}}_\mathrm{f}$	$\subseteq \mathbb{R}^{n_x}$	adapted final state constraint set
$\mathcal{M}^{(Q)}$	$\subset \mathbb{S}_{++}^{n_x \times n_x}$	bounded set of state weighting matrices for LQR control
$\mathcal{M}^{(R)}$	$\subset \mathbb{S}_{++}^{n_u \times n_u}$	bounded set of input weighting matrices for LQR control

Computing Platform and Implementation

All numerical experiments and computations are carried out in the matrix Laboratory (MATLAB) 2022b on an Intel(R) Core(TM) i7-9700K with 16 GB of RAM. Further, we implemented all presented set operations and conversions in the continuous reachability analyzer[1] [2] (CORA) toolbox, and all proposed synthesis algorithms in the automated reachset optimal control[2] [63] (AROC) toolbox, which are both publicly available.

[1] `https://cora.in.tum.de/`
[2] `https://aroc.in.tum.de/`

1 Introduction

1.1 Motivation

In 1965, Gordon Moore, co-founder of one of today's biggest semiconductor chip manufacturers in the world, posited that the number of components on a chip of fixed size would double every year. His revised version, which stated that this number would double every two years and which has held true every since, became known as Moore's law and is de facto synonymous with the exponential rise of computational power we have seen over the last decades. Incidentally, the birth of Moore's law in 1965 coincided with the publication of the first machine learning paper. While back then, machine learning was restricted to very basic applications, today's availability of computational resources predicted by Moore has truly enabled the application of artifical intelligence (AI) to nearly every aspect of our lives, with applications ranging from smart kitchen appliances and vacuuming robots to highly sophisticated chat bots.

Among many, autonomous driving is one prominent area of research that benefited from the increasing capabilities of AI. However, while smart vacuuming robots may have some form of human interaction, guaranteeing the safety of their machine-learning-guided journey through one's home is not safety critical. In contrast, AI systems which enable fully autonomous driving are highly safety critical. However, the high complexity of such AI systems, often using millions or even billions of neurons, makes it hard or even impossible to verify them. As a result, these autonomous vehicles are required to validate their safety to within some statistical margin by extensive testing. For self-driving cars, this involves driving thousands or even millions of kilometers in varying environments. Furthermore, changes to any safety-critical system in the car can require additional testing, or even demand a complete restart of the testing campaign.

Instead of only validating a given system through extensive testing, one may formally verify it to obtain guarantees about its behavior. With the increase of digital computers in all aspects of our society, cyber-physical systems, which combine both physical and software components, are becoming increasingly popular to model these systems. Reachability analysis is one popular formal verification tool to verify cyber-physical systems: The reachable set contains all possible future states of a system that can be reached, starting from a given set of initial states under any possible disturbance realization. A state hereby collects the necessary information of a system to describe it fully. For autonomous vehicles, this means that its subsystems can be formally verified by checking intersections of unsafe states with the reachable set; changes to a subsystem then only require the re-verification of that subsystem but leaves the formal guarantees of all other subsystems untouched. Additionally, reachability analysis can account for uncertainty in

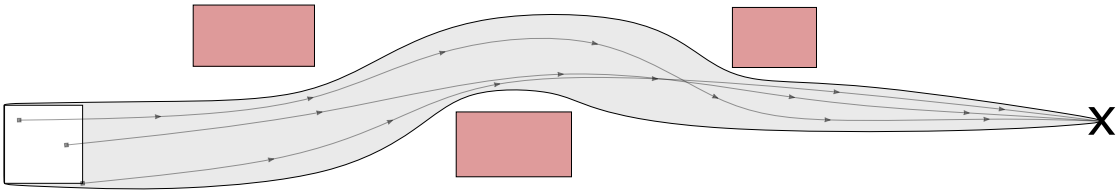


Figure 1.1: Concept of a reach-avoid problem: From a given initial set of states (white), the system is steered to the target state ("x") within a given time horizon while avoiding all obstacles (red). The grey area hereby is the reachable set, where the grey lines visualize possible trajectories for a specific initial state (grey square).

the system, such as measurement errors of radar or light detection and ranging (LIDAR) sensors.

However, reachable sets of more complex systems cannot be computed exactly. Therefore, an outer approximation of the reachable set, which is an approximation containing the exact reachable set, is computed. That said, autonomous systems, such as autonomous cars, are often first designed, i.e., a controller that executes a certain task is synthesized, and only then verified. As a result, tuning the controller manually in order to verify safety may be impractical for highly complex systems. Therefore, instead of verifying existing controllers, we focus in this thesis on combining numerical optimization and reachability analysis to obtain correct-by-construction controllers that already provide formal guarantees about the behavior of the closed-loop system.

A wide range of these tasks can be classified as reach-avoid problems, which are visualized in Fig. 1.1: For a given system with a set of initial states, we try to find a controller that, within a given time horizon, avoids all obstacles and steers the system state close to a given target state while minimizing the required control effort. In previous work, the combination of reachability analysis and optimization theory for the synthesis proved to be a powerful combination as it allows to optimize over a set of initial states instead of a single initial state. That said, the computation of reachable sets is computationally expensive when executed repeatedly. Thus, in this thesis, we propose novel synthesis approaches that avoid the computation of reachable sets as much as possible while still using their formal verification properties.

1.2 State of the Art

The task of synthesizing a controller for a given system has already been extensively explored in literature. We subsequently summarize the most important work.

1.2.1 Optimal Control

When only a single initial state needs to be controlled, many approaches in the field of optimal control have been proposed [92]. For a given system of possibly nonlinear ordinary differential equations (ODEs), optimal control approaches try to find a function for the input vector subject to these ODEs that minimizes a functional while respecting

constraints on the input and state of the system. When there are no constraints on a linear time-invariant (LTI) system and one optimizes over a quadratic cost function in both state and input, Kalman derived in his seminal paper [56] the optimal control law – now known as linear-quadratic regulator (LQR) control – using Pointryagin's maximum principle, which provides necessary (but not sufficient) conditions for an optimal control law. For general nonlinear optimal control problems, however, it becomes necessary to use numerical methods [105]. We differentiate between two approaches: Starting from the first-order necessary conditions of the original optimal control problem given by the Hamiltonian boundary-value problem, indirect methods numerically solve this boundary-value problem for given boundary values and thus reduce the original problem to the solution of a system of nonlinear equations [92]. Due to the recent progress in optimization theory, however, indirect methods [105, 96] have become less popular. In contrast, direct methods parameterize the input and state functions using a finite number of parameters and then solve the resulting finite-dimensional optimization problem. Here, methods include direct multiple shooting [13], which subdivides the time horizon of the optimal control problem into multiple steps and subsequently parameterizes the control function, and has, e.g., been applied to optimal robot control [26]. Further, direct collocation methods [104] parameterize both the input and state vector and have, e.g., been used for the estimation of muscle forces during motion [43]. There exist many software packages, such as the automatic control and dynamic optimization[1] [55] (ACADO) toolbox and CasADi [10], that implement these optimal control methods while providing user-friendly interfaces in multiple programming languages.

As an extension to Pointryagin's maximum principle, the Hamilton-Jacobi-Bellman (HJB) equations provide necessary and sufficient conditions for an optimal control law over the entire state space instead of a single trajectory [57]. Further, they allow the direct inclusion of state and input constraints into the problem. However, while there have been attempts to reduce the curse of dimensionality when trying to solve the HJB equations, solving them requires the solution of partial differential equations and thus still has worst-case exponential complexity in the number of state variables [85].

1.2.2 Model Predictive Control

When using a direct method for the solution of an optimal control problem for a given initial state, choosing the input to be piecewise constant as a parameterization means that the algorithm returns a sequence of piecewise constant control inputs that minimizes the given objective function for the given initial state. In model predictive control (MPC), this procedure is applied iteratively: For a given initial state, the sequence of control inputs is computed and the first input is applied to the system. After one step, the system state is again measured and the input sequence is recomputed. Since this procedure easily allows for the incorporation of state and input constraints into the optimal control problem, MPC has been used extensively in industry for the past decades [82, 93]. When the given system has inherent uncertainty, tube-based MPC guarantees

[1] `https://acado.github.io/`

that the system stays within a tube around a reference trajectory [80, 72, 79, 91, 78]. However, since MPC repeatedly solves an optimal control problem for a given state, its application requires optimization online. To alleviate some of that computation cost online, explicit MPC approaches parameterize the optimal control problem in the initial state and solve the resulting parametric optimization problem offline [1, 30], albeit at the cost of high computation times when trying to solve high-dimensional, parametric, non-convex optimization problems. With major advances in optimization theory, more recent work thus focuses again on solving the optimal control problem online, which allows the online controller synthesis for LTI systems in real-time [44, 119, 102].

1.2.3 Abstraction-Based Synthesis

Similarly to the HJB equations, controller synthesis using model abstraction – and particularly symbolic model abstraction – allows for the synthesis of controllers for sets of initial states instead of a single initial state. In model abstraction, the continuous or hybrid system is abstracted to a finite-state system which has a finite number of discrete states [107, 88, 89, 94, 58, 39]. For such automata, standard algorithms for finite systems can be employed to synthesize a controller for the abstracted system; the resulting controller then needs to be suitably refined such that we obtain a controller for the original system [95]. When abstracting the state space with a uniform grid, the number of abstracted finite states grows exponentially in the number of state variables. Additionally, early work made assumptions about the system class [58, 46, 12] or the stability of the system [42, 90]; recent work [118, 95] has managed to avoid such assumptions. Recently, attempts have been made to also circumvent the curse of dimensionality due to state space discretization; however, the proposed techniques either use mixed-integer linear program (MILP) with exponential worst-case complexity in the number of integer variables [116] or focus on linear systems [87].

1.2.4 Reachability-Based Synthesis

Reachability analysis has recently attracted increased attention in the formal verification community [9, 4, 23, 5, 21, 6]. Previous work has already combined reachability analysis with optimization to solve reach-avoid problems: The authors in [100] synthesize a continuous state feedback controller by formulating a non-convex optimization problem, where in each optimization iteration an outer approximation to the reachable set is computed, which in turn is used to compute the current value of the objective function and the constraints. In [99], the authors consider the controller synthesis for nonlinear system dynamics by first computing a piecewise constant control sequence for each vertex of the initial set. These sequences are then used in the final controller to form a piecewise constant input sequence for any initial state. However, the number of vertices of the initial set grows exponentially with the system dimension. Thus, the authors in [99] instead parameterize the controller using zonotopes and compute a parameterized reachable set based on the linearized dynamics to obtain a synthesis problem which can be formulated as a linear program (LP). This parameterization grows linearly in the

state dimension and thus can efficiently synthesize piecewise constant controllers for disturbed nonlinear systems under input and state constraints. While the parameterization of the control input by a zonotope means that input constraints can be easily verified, the parameterized reachable set is only an approximation and thus the inclusion of state constraints requires user input to achieve non-conservative results. Furthermore, while this approach can be applied iteratively to realize discrete-time state feedback, it fails to efficiently reject larger disturbances. Therefore, the authors combine the feedback synthesis from [100] and the feedforward synthesis from [99] for the synthesis of a combined piecewise constant feedforward controller with continuous state feedback [101], albeit at the cost of again computing an outer approximation to the reachable set of the current controller in each optimization iteration.

1.3 Outline

We propose several new algorithms for formally verified controller synthesis of constrained, disturbed nonlinear systems. Since we make extensive use of set-based computations, we further introduce required conversions between set representations in order to pose the synthesis problems efficiently.

Chapter 2: Background We introduce all necessary background in Chap. 2. Specifically, we start by introducing mathematical standard operations in Sec. 2.1 and the theory of differentials in Sec. 2.2. We then describe the basics of optimization theory in Sec. 2.3 and continue with a review of the well-known linear-quadratic regulator (LQR) control scheme in Sec. 2.4. Since we compute with sets, we describe in Sec. 2.5 and Sec. 2.6 all necessary set operations and set representations. In Sec. 2.7, we introduce the concept of reachable sets and give a short overview over the relevant algorithms for this thesis. In Sec. 2.8, we then briefly discuss how the representation complexity of zonotopes and polynomial zonotopes can be controlled, which is, e.g., necessary to keep reachability computations efficient.

Chapter 3: Abstracted Synthesis Problem In Chap. 3, we discuss the abstracted synthesis problem, i.e., an optimization problem which generalizes all major optimization problems of subsequent chapters and which we use to derive important properties. First, we introduce reach-avoid problems and define the general synthesis problem in Sec. 3.1. Because many optimization solvers find possible extrema by solving for the first-order critical points that are characterized by the Karush-Kuhn-Tucker (KKT) conditions, we then define the abstracted synthesis problem in Sec. 3.2 and derive a smooth optimization problem, for which we prove that it shares its global minimum with the original optimization problem. Lastly, we discuss the effect of different norms in optimization in Sec. 3.3.

Chapter 4: Set Conversions and Set Containment In order to efficiently pose and solve optimization problems of subsequent chapters, we further introduce existing and

novel conversions between different set representations in Chap. 4. First, we introduce all necessary set conversions in Sec. 4.1. In order to check, e.g., the constraint satisfaction of input or state constraints, we then shortly describe existing approaches from the literature to check set containment in Sec. 4.2.

Chapter 5: Piecewise Constant Controller Synthesis We describe existing and novel algorithms for the synthesis of piecewise constant controllers, i.e., controllers which are constant in time for a given time interval, in Chap. 5. After the introduction of the problem statement in Sec. 5.1, we first review the generator-space control (GSC) approach from [99] in Sec. 5.2 and then propose the polynomial generator-space control (PGSC) approach as a novel extension of GSC in Sec. 5.3. Because both GSC and PGSC require additional user inputs when state contraints are to be enforced, we introduce iterative polynomial generator-space control (iPGSC), a novel iterative synthesis approach using trust regions, in Sec. 5.4. We demonstrate the applicability of our algorithms using numerical examples.

Chapter 6: Piecewise Constant Controller Synthesis with Continuous State Feedback Since piecewise constant controllers cannot quickly counteract disturbances due to their inherent discrete-time feedback, we discuss the synthesis of piecewise constant feedforward controllers with continuous state feedback in Chap. 6. We start by introducing the problem statement in Sec. 6.1 and then describe the reachset optimal control (ROC) approach from [101] in Sec. 6.2, where feedforward synthesis using GSC is combined with the synthesis of a continuous linear feedback term. Because ROC executes the feedforward and feedback synthesis sequentially, i.e., the feedforward controller is synthesized first and then kept fixed during the feedback controller optimization, we introduce iterative polynomial reachset optimal control (iPROC) in Sec. 6.3, which – for the first time – optimizes over the feedforward and feedback controller simultaneously reduces the required user input compared to ROC. The applicability of our algorithms is demonstrated with numerical benchmarks.

2 Background

In this chapter, we introduce all concepts that are relevant for this thesis. We start by introducing necessary definitions and well-known theorems in Sec. 2.1 and then briefly describe differentials in Sec. 2.2. In Sec. 2.3, we then give an overview of optimization theory and introduce the well-known LQR controller in Sec. 2.4. In Sec. 2.5, we define all necessary set operations and introduce all required set representations in Sec. 2.6. We continue with a short review of the relevant reachability algorithms from the literature in Sec. 2.7 and conclude with the concept of order reduction in Sec. 2.8, which is required for the efficient application of reachability analysis.

2.1 Standard Operations

All controllers in this thesis are synthesized using different numerical optimization techniques. Since numerical solvers will generally not reach an optimum exactly but only approximately, we measure numerical tolerances as defined next.

Definition 2.1 (Numerical Tolerance)**.** Given two matrices $A \in \mathbb{R}^{n \times m}$ and $B \in \mathbb{R}^{n \times m}$, we define the tolerance between A and B as the minimum between the absolute tolerance and the relative tolerance, i.e.

$$\operatorname{tol}(A, B) = \max_{\substack{1 \leq i \leq n \\ 1 \leq j \leq m}} \min \left(|A_{ij} - B_{ij}|, \lim_{\epsilon \to 0^+} \frac{|A_{ij} - B_{ij}|}{\min(|A_{ij}|, |B_{ij}|) + \epsilon} \right),$$

where "$\lim_{\epsilon \to 0^+}$" denotes the limit from $\epsilon > 0$ to 0. ■

For later convenience, we subsequently introduce the vectorization and convex combination operation.

Definition 2.2 (Vectorization)**.** The column-wise vectorization of a given matrix $X \in \mathbb{R}^{n \times m}$ into a vector $x \in \mathbb{R}^{nm}$ is given by

$$x = X_{(:)} = \begin{bmatrix} X_{(:,1)} \\ X_{(:,2)} \\ \vdots \\ X_{(:,m)} \end{bmatrix}.$$

■

For the vectorization of matrix products, it further holds that [74, Chap. 2, Sec. 4]

$$[ABC]_{(:)} = \left(C^T \boxtimes A\right) B_{(:)} \tag{2.1a}$$
$$= \left(C^T B^T \boxtimes I_m\right) A_{(:)} \tag{2.1b}$$
$$= \left(I_q \boxtimes AB\right) C_{(:)}, \tag{2.1c}$$
$$A_{(:)}^T D_{(:)} = \operatorname{trace}\left(A^T D\right), \tag{2.1d}$$

for $A \in \mathbb{R}^{m\times n}$, $B \in \mathbb{R}^{n\times o}$, $C \in \mathbb{R}^{o\times q}$, and $D \in \mathbb{R}^{n\times m}$.

Definition 2.3 (Convex Combination)**.** The convex combination of two sets $A \subseteq \mathbb{R}^n$ and $\mathcal{B} \subseteq \mathbb{R}^n$ is defined by

$$\operatorname{convc}(\mathcal{A}, \mathcal{B}) = \{la + (1-l)\, b \mid a \in \mathcal{A}, b \in \mathcal{B}, l \in [0,1]\}.$$

■

Next, we quickly review well-known theorems that will be required later.

Theorem 2.1 (Singular Value Decomposition [54, Th. 2.6.3])**.** *The singular value decomposition of a matrix $A \in \mathbb{R}^{n\times m}$ is*

$$A = U\Sigma V^H,$$

where $U \in \mathbb{C}^{n\times n}$ and $V \in \mathbb{C}^{m\times m}$ are unitary matrices, and $\Sigma \in \mathbb{R}^{n\times m}$ is a rectangular matrix of zeros except for $\Sigma_{(i,i)} = \sigma_1$, $1 \leq i \leq \min(n,m)$, with $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_{\min(n,m)} \geq 0$ being the singular values of A.

We sometimes use the shorthand $\sigma(A)$ to refer to the vector of singular values of A.

More widely known is the eigen decomposition, which we introduce next.

Theorem 2.2 (Eigen decomposition [54, Th. 1.3.7, adapted])**.** *Let $A \in \mathbb{R}^{n\times n}$ be a diagonalizable matrix, i.e., there exists an invertible matrix $P \in \mathbb{R}^{n\times n}$ and a diagonal matrix $D \in \mathbb{R}^{n\times n}$ such that $P^{-1}AP = D$. Then we can decompose A as*

$$A = Q\Lambda Q^{-1},$$

where $\Lambda = \operatorname{diag}\left(\begin{bmatrix}\lambda_1, & \lambda_2, & \ldots, & \lambda_n\end{bmatrix}\right) \in \mathbb{R}^{n\times n}$, and $Q \in \mathbb{C}^{n\times n}$ is an invertible matrix. For $1 \leq i \leq n$, $Q_{(:,i)} \in \mathbb{R}^n$ and $\lambda_i \in \mathbb{R}$ denote the eigenvector and corresponding eigenvalue of A, where $AQ_{(:,i)} = \lambda_i Q_{(:,i)}$ holds.

We denote with $\lambda(A)$ the vector of eigenvalues. Next, we introduce the square root of a positive semi-definite matrix.

Theorem 2.3 (Matrix Square Root [54, Th. 7.2.6, adapted])**.** *A square root of a positive semi-definite matrix $Q \in \mathbb{S}_+^{n\times n}$ is*

$$\sqrt{Q} = Q^{\frac{1}{2}} = U \operatorname{diag}\left(\sqrt{\sigma(Q)}\right),$$

such that $Q = \sqrt{Q}\sqrt{Q}$, where $Q = U \operatorname{diag}(\sigma(Q))\, U^T$ with singular values $\sigma(Q) \in \mathbb{R}_{\geq 0}^n$.

2.2 Differentials

Whenever matrices are involved, taking their derivative with respect to a vector, i.e., the derivative of each matrix element, is often cumbersome. Differentials offer a convenient formulation for taking derivatives that also works well for matrices. Hence, we introduce differentials and relevant properties thereof in this section – which is inspired by [74, Chap. 18] – starting with a simple example.

Example 2.1. Let $f(x) = \begin{bmatrix} x_1^2, & x_2^2 \end{bmatrix}^T$ with $x \in \mathbb{R}^2$. Its Jacobian matrix is given by

$$J_f(x) = \begin{bmatrix} 2x_1, & 0 \\ 0, & 2x_2 \end{bmatrix},$$

with $J_f(x) \in \mathbb{R}^{2\times 2}$ since each component of f is differentiated with respect to each component of x, i.e., we have $2 \cdot 2 = 4$ derivatives. In contrast, the differential of f is

$$\mathrm{d}f = \mathrm{d}\begin{bmatrix} x_1^2 \\ x_2^2 \end{bmatrix} = \begin{bmatrix} \begin{bmatrix} 2x_1, & 0 \\ 0, & 2x_2 \end{bmatrix} \mathrm{d}x \end{bmatrix}.$$

Immediately, it is obvious that $\mathrm{d}f \in \mathbb{R}^2$ and thus the differential has the same dimensionality as f, whereas $\frac{df(x)}{dx} \in \mathbb{R}^{2\times 2}$. Further, one can directly identify the Jacobian matrix from the above expression. ■

This identification of the Jacobian matrix from the differential is possible in general.

Theorem 2.4 (First Identification Theorem [74, Th. 5.6, adapted])**.** *Let $x \in \mathbb{R}^n$ and $f : \mathbb{R}^n \mapsto \mathbb{R}^m$. Then*

$$\mathrm{d}f = A(x)\,\mathrm{d}x \iff J_f(x) = A(x).$$

In a nutshell, the differential $\mathrm{d}f$ does not represent the derivative directly but rather is a linear approximation around $f(x)$ with an infinitesimally small variation around $f(x)$, i.e., $\mathrm{d}f = A(x)\,\mathrm{d}x$, where $\mathrm{d}x$ is that very small deviation. The fact that the dimension of differentials stays constant makes them particularly useful when handling matrix derivatives and simplifies the following rules, which will be helpful in subsequent chapters.

Proposition 2.1 (Collection of Differential Rules [74, Chap. 18])**.** *Let $A \in \mathbb{R}^{n\times m}$ be a constant matrix, $F \in \mathbb{R}^{n\times m}$, $G \in \mathbb{R}^{n\times m}$, $H \in \mathbb{R}^{m\times p}$, $X \in \mathbb{R}^{n\times n}$ some non-constant functions, and $c \in \mathbb{R}$ some scalar. We have*

$$\begin{aligned}
\mathrm{d}A &= 0, \\
\mathrm{d}(cF) &= c\,\mathrm{d}F, \\
\mathrm{d}\left(F^T\right) &= (\mathrm{d}F)^T, \\
\mathrm{d}(F+G) &= \mathrm{d}F + \mathrm{d}G, \\
\mathrm{d}(FH) &= (\mathrm{d}F)\,H + F\,\mathrm{d}H, \\
\mathrm{d}X^{-1} &= -X^{-1}(\mathrm{d}X)\,X^{-1} \textit{ (for nonsingular } X\textit{).}
\end{aligned}$$

For a scalar field $f : \mathbb{R}^n \mapsto \mathbb{R}$, the Hessian matrix is well-defined and can also be computed using differentials as the following example demonstrates.

Example 2.2. Let $f(x) = x_1^2 x_2$. We have

$$\begin{aligned}
\mathrm{d}^2 f &= \mathrm{d}^2 \left(x_1^2 x_2 \right) \\
&= \mathrm{d} \left(\begin{bmatrix} 2x_1 x_2, & x_1^2 \end{bmatrix} \mathrm{d}x \right) \\
&= \mathrm{d} \begin{bmatrix} 2x_1 x_2, & x_1^2 \end{bmatrix} \mathrm{d}x + \begin{bmatrix} 2x_1 x_2, & x_1^2 \end{bmatrix} \mathrm{d}^2 x \\
&= \begin{bmatrix} \mathrm{d}\left(2x_1 x_2\right), & \mathrm{d}\left(x_1^2\right) \end{bmatrix} \mathrm{d}x \\
&= \left[\mathrm{d}x^T \begin{bmatrix} 2x_2 \\ 2x_1 \end{bmatrix}, \quad dx^T \begin{bmatrix} 2x_1 \\ 0 \end{bmatrix} \right] \mathrm{d}x \\
&= \mathrm{d}x^T \begin{bmatrix} 2x_2 \\ 2x_1 \end{bmatrix} \mathrm{d}x_1 + \mathrm{d}x^T \begin{bmatrix} 2x_1 \\ 0 \end{bmatrix} \mathrm{d}x_2 \\
&= \mathrm{d}x^T \left(\begin{bmatrix} 2x_2 \\ 2x_1 \end{bmatrix} \begin{bmatrix} 1, & 0 \end{bmatrix} + \begin{bmatrix} 2x_1 \\ 0 \end{bmatrix} \begin{bmatrix} 0, & 1 \end{bmatrix} \right) \mathrm{d}x \\
&= \mathrm{d}x^T \begin{bmatrix} 2x_2, & 2x_1 \\ 2x_1, & 0 \end{bmatrix} \mathrm{d}x.
\end{aligned}$$

Computing the Hessian matrix of $f(x)$, we see that $H_f(x) = \begin{bmatrix} 2x_2, & 2x_1 \\ 2x_1, & 0 \end{bmatrix}$. ■

Ex. 2.2 can be generalized as follows:

Theorem 2.5 (Second Identification Theorem [74, Th. 6.6, adapted])**.** *Let $f : \mathbb{R}^n \mapsto \mathbb{R}$ with $x \in \mathbb{R}^n$. We have*

$$\mathrm{d}^2 f = (\mathrm{d}x)^T B(x) \, \mathrm{d}x \iff H_f(x) = \frac{B(x) + B(x)^T}{2}.$$

To compute with differentials, we can interpret them as tensors. Let $dA_{ij} = \frac{dA_{ij}(x)}{dx} \in \mathbb{R}^{1 \times n}$ for $1 \leq i \leq m$ and $1 \leq j \leq o$, where $A(x) \in \mathbb{R}^{m \times o}$ denotes a matrix function in $x \in \mathbb{R}^n$ with differentiable matrix elements $A_{ij}(x)$. Then

$$[\mathrm{d}A]_{ij} = dA_{ij} \, \mathrm{d}x,$$

and we can alternatively collect the Jacobian matrix of each matrix element with respect to x in the first two dimensions of a 4D tensor, where the last two dimensions reflect the dimensions of A. Thus, the differential $\mathrm{d}A$ can be represented by the tensor $dA \in \mathbb{R}^{1 \times n \times m \times o}$. Similarly, let $d^2 A_{ij} = \frac{d^2 A_{ij}(x)}{dx^2} \in \mathbb{R}^{n \times n}$. Then

$$\mathrm{d}^2 A_{ij} = \mathrm{d}x^T dA_{ij} \, \mathrm{d}x,$$

and thus $\mathrm{d}^2 A$ can – analogously to the first differential – be interpreted as the 4D tensor $d^2 A \in \mathbb{R}^{n \times n \times m \times o}$.

It remains to define arithmetic operations on these differential tensors. To that end, let $B(x) \in \mathbb{R}^{m\times o}$ be a twice differentiable matrix function. We have

$$\begin{aligned}\mathrm{d}\left[A+B\right]_{ij} &= \left[dA+dB\right]_{ij}\mathrm{d}x,\\ \mathrm{d}^2\left[A+B\right]_{ij} &= \mathrm{d}x^T\left[d^2A+d^2B\right]_{ij}\mathrm{d}x,\end{aligned}$$

and thus the tensor representation of the sum of two differentials is equivalent to the sum of the two tensor representations. Further, let $C(x) \in \mathbb{R}^{o\times l}$ and $D \in \mathbb{R}^{q\times m}$. Since

$$\begin{aligned}\left[D\,\mathrm{d}A\right]_{ij} &= \sum_{k=1}^{m} D_{ik}\,\mathrm{d}A_{kj}\\ &= \left(\sum_{k=1}^{m} D_{ik}dA_{kj}\right)\mathrm{d}x,\ 1\le i\le q,\ 1\le j\le o,\\ \left[\mathrm{d}AC\right]_{ij} &= \sum_{k=1}^{o} \mathrm{d}A_{ik}C_{kj}\\ &= \left(\sum_{k=1}^{m} dA_{ik}C_{kj}\right)\mathrm{d}x,\ 1\le i\le m,\ 1\le j\le l,\\ \left[\mathrm{d}A\,\mathrm{d}C\right]_{ij} &= \sum_{k=1}^{o} \mathrm{d}A_{ik}\,\mathrm{d}C_{kj}\\ &= \sum_{k=1}^{o} dA_{ik}\,\mathrm{d}x\,dC_{kj}\,\mathrm{d}x\\ &= \mathrm{d}x^T\left(\sum_{k=1}^{o} dA_{ik}^T dC_{kj}\right)\mathrm{d}x,\ 1\le i\le o,\ 1\le j\le l,\end{aligned}$$

we define the following arithmetic operations for these tensors:

Definition 2.4 (Arithmetic Operations for Differential Tensors)**.** Let $A(x) \in \mathbb{R}^{m\times o}$, $B(x) \in \mathbb{R}^{m\times o}$, $C(x) \in \mathbb{R}^{o\times l}$, $D(x) \in \mathbb{R}^{q\times m}$, and $x \in \mathbb{R}^n$ with the corresponding differentials represented as the tensors $dA \in \mathbb{R}^{1\times n\times m\times o}$, $dB \in \mathbb{R}^{1\times n\times m\times o}$, $dC \in \mathbb{R}^{1\times n\times o\times l}$, and $dD \in \mathbb{R}^{1\times n\times q\times m}$ (dependency on x omitted for readability). We define

$$\left[dA+dB\right]_{ij} = dA_{ij}+dB_{ij}, \quad 1\le i\le m,\ 1\le j\le o, \tag{2.2}$$

$$\left[DdA\right]_{ij} = \sum_{k=1}^{m} D_{ik}dA_{kj}, \quad 1\le i\le q,\ 1\le j\le o, \tag{2.3}$$

$$\left[dAC\right]_{ij} = \sum_{k=1}^{m} dA_{ik}C_{kj}, \quad 1\le i\le m,\ 1\le j\le l, \tag{2.4}$$

$$\left[dAdC\right]_{ij} = \sum_{k=1}^{o} dA_{ik}^T dC_{kj}, \quad 1\le i\le m,\ 1\le j\le l. \tag{2.5}$$

■

Thus, any differential expressions with sums, products or transpose operations can be directly computed by simply replacing the differentials with their differential tensor counterparts and executing the corresponding operations as defined in Def. 2.4.

2.3 Optimization Theory

We extensively use optimization in this thesis to find optimal controllers. Thus, we briefly introduce necessary concepts from optimization theory. We only deal with differentiable optimization problems and therefore assume that all involved functions are sufficiently smooth, which generally means that they are at least twice continuously differentiable. We start with the definition of a smooth optimization problem.

Definition 2.5. Let $f : \mathbb{R}^n \mapsto \mathbb{R}$, $g : \mathbb{R}^n \mapsto \mathbb{R}^m$, and $h : \mathbb{R}^n \mapsto \mathbb{R}^o$ be sufficiently smooth functions. A smooth optimization problem can be written as

$$p^* = \arg\min_x f(x), \tag{2.6a}$$

$$\text{s.t. } g(x) \leq 0, \tag{2.6b}$$

$$h(x) = 0, \tag{2.6c}$$

where $x \in \mathbb{R}^n$ denotes the multivariate optimization variable, $f(x)$ is the objective function, $g(x)$ are the inequality constraints, $h(x)$ denote the equality constraints, and p^* is the optimal objective value at the optimizer x^*. Further, we call the set

$$\mathcal{S} = \{x \in \mathbb{R}^n \mid g(x) \leq 0,\ h(x) = 0\},$$

the feasible set of (2.6). ■

In this thesis, we only consider optimization problems for which an optimum exists, i.e., $p^* > -\infty$. When it is not clear that the optimal objective value is finite, limit expressions ("inf" and "sup") need to be used instead of " min" and "max" operations. Further, we only consider minimization problems in this section, as any maximization problem can be transformed into a minimization problem via

$$\max_{x \in \mathcal{S}} f(x) = -\min_{x \in \mathcal{S}} -f(x).$$

To solve such smooth optimization problems, practically all known efficient algorithms use the Jacobian matrix and sometimes Hessian matrix information of the objective function and the constraints, respectively. Methods that solely rely on first-order derivatives are called first-order methods while second-order methods additionally use the Hessian matrices of the objective function and constraints.

The remainder of this section is structured as follows: In Sec. 2.3.1, we introduce the Karush-Kuhn-Tucker (KKT) optimal conditions for the original optimization problem in (2.6): Any optimizer of (2.6) necessarily fulfills these optimality conditions, assuming that any applicable constraint qualification – discussed next in Sec. 2.3.2 – holds. We then describe convex optimization as an important subclass in Sec. 2.3.3, followed by a discussion of non-convex optimization problems in Sec. 2.3.4. In Sec. 2.3.5, we then

introduce the dual problem, whose maximum always bounds the minimum of the original problem from below, and which is always a convex optimization problem. We conclude the section with a short review of popular reformulation techniques for optimization problems in Sec. 2.3.6.

2.3.1 Karush-Kuhn-Tucker Conditions

While (2.6) can be intuitively interpreted, it remains unclear how it can be solved in general. Instead of trying to solve (2.6) directly, one may solve for the optimizer using the first-order KKT optimality conditions, which we introduce in this section.

For the introduction of these optimality conditions, we first require the Lagrangian function of (2.6).

Definition 2.6 (Lagrangian Function [14, Chap. 5, adapted])**.** Let f, g, and h be defined as in Def. 2.5. The Lagrangian of (2.6) is

$$\phi\left(x, \mu, \lambda\right) = f\left(x\right) + \mu^T g\left(x\right) + \lambda^T h\left(x\right), \tag{2.7}$$

where $\lambda \in \mathbb{R}^m$ and $\mu \in \mathbb{R}^o$ are the Lagrange multipliers for the inequality and equality constraints, respectively. ■

Given that (2.6) is regular at a critical point x^*, i.e., it satisfies an appropriate constraint qualification (see Sec. 2.3.2), one can use the Lagrangian function to derive the first-order necessary KKT conditions for optimality of x^*, which are given by [14, Sec. 5.5]

$$\nabla_x f\left(x^*\right) + \sum_{i=1}^{m} \mu_i^* \nabla_x g_i\left(x^*\right) + \sum_{j=1}^{o} \lambda_j^* \nabla_x h_j\left(x^*\right) = 0 \text{ (stationarity)}, \tag{2.8a}$$

$$\mu^{*T} g\left(x^*\right) = 0 \text{ (complementary slackness)}, \tag{2.8b}$$

$$\mu^* \geq 0 \text{ (dual feasibilty)}, \tag{2.8c}$$

$$g\left(x^*\right) \leq 0 \text{ (primal feasibilty)}, \tag{2.8d}$$

$$h\left(x^*\right) = 0 \text{ (primal feasibilty (cont.))}, \tag{2.8e}$$

where $\mu^* \in \mathbb{R}^m_{\geq 0}$ and $\lambda^* \in \mathbb{R}^o$ are the optimal Lagrange multipliers for (2.6b) and (2.6c), respectively. Intuitively, complementary slackness means that whenever the i-th constraint is inactive, i.e., $g_i\left(x^*\right) < 0$, $\mu_i = 0$ is enforced by (2.8b) so that $g_i\left(x^*\right)$ is treated as if it was not included in the optimization problem.

2.3.2 Constraint Qualifications

As described in Sec. 2.3.1, the KKT conditions are only necessary optimality conditions when regularity conditions, also called constraint qualifications, hold. In this section, we introduce an important constraint qualification for convex optimization problems and then introduce a constraint qualification which is also applicable to non-convex optimization problems that is used in a later section.

2.3.2.1 Slater's Condition

Arguably the most well-known constraint qualification is Slater's condition (SC).

Proposition 2.2 (Slater's Condition [14, Sec. 5.2.3, adapted])**.** *Consider a convex minimization problem as defined in Def. 2.9. If there exists an interior point $x \in \mathbb{R}^n$, i.e., $h(x) = 0$ and $g(x) < 0$, then Slater's condition (SC) holds.*

This constraint qualification is widely used, e.g., to prove strong duality for instances of semi-definite programs (SDPs) (strong duality does not generally hold for SDPs even though they are convex) since finding an interior point within the feasible set is often easier than checking other constraint qualifications.

2.3.2.2 Mangasarian-Fromovitz Constraint Qualification

In this section, we provide the Mangasarian-Fromovitz constraint qualification (MFCQ) which will prove useful in a later section (see Sec. 3.2.2).

Proposition 2.3 (Mangasarian-Fromovitz Constraint Qualification [76, Sec. 3, adapted])**.** *Let a smooth optimization problem be given as in* (2.6) *with critical point $x^* \in \mathbb{R}^n$, inequality constraints $g(x^*) \leq 0$, and equality constraints $h(x^*) = 0$, where $g : \mathbb{R}^n \mapsto \mathbb{R}^m$ and $h : \mathbb{R}^n \mapsto \mathbb{R}^o$. Further, let $\mathcal{A} = \{a \in \{1, ..., m\} \mid g_a(x^*) = 0\}$ denote the set of active inequality constraints. If the gradients $\nabla_x h_j(x^*)$ for $1 \leq j \leq o$ of the equality constraints are linearly independent and there exists a vector $d \in \mathbb{R}^n$ such that*

$$\nabla_x g_a(x^*)^T d < 0, a \in \mathcal{A},$$
$$\nabla_x h_j(x^*)^T d = 0, 1 \leq j \leq o,$$

then x^* satisfies the Mangasarian-Fromovitz constraint qualification (MFCQ).

2.3.3 Convex Optimization

Being essential in a variety of everyday applications, convex optimization is widely adopted since, once a problem has been formulated as a convex optimization problem, one can generally (with a few edge cases as an exception) efficiently solve the problem [14, Sec. 1.3.2]. In more technical terms, convex optimization is such an important field in optimization since all stationary points of the Lagrangian are global optima under a suitable constraint qualification (see Sec. 2.3.2). Furthermore, there exist many mature numerical solvers for convex optimization problems, such as MOSEK[1], Gurobi[2], or SDPT3[3], to only name a few. For the definition of a convex optimization problem, we first introduce convex sets and convex functions. For a thorough discussion on convex optimization, we refer the interested reader to [14].

[1] `https://www.mosek.com/`
[2] `https://www.gurobi.com/`
[3] `https://blog.nus.edu.sg/mattohkc/softwares/sdpt3/`

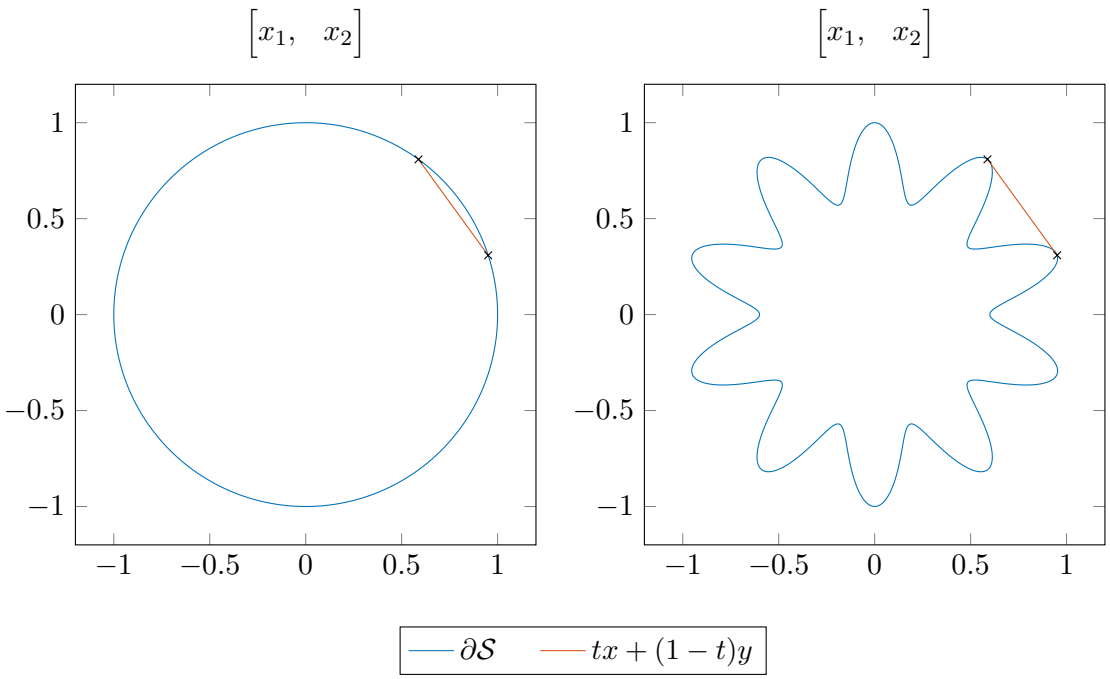


Figure 2.1: Example of a convex set given by $\left\{ r \begin{bmatrix} \cos\phi \\ \sin\phi \end{bmatrix} \;\middle|\; 0 \leq r \leq 1,\ 0 \leq \phi \leq 2\pi \right\}$ and non-convex set $\left\{ r\,(0.8 + 0.2\cos(10\phi)) \begin{bmatrix} \cos\phi \\ \sin\phi \end{bmatrix} \;\middle|\; 0 \leq r \leq 1,\ 0 \leq \phi \leq 2\pi \right\}$. Additionally, we show the line connecting $x = \begin{bmatrix} \cos\left(\frac{\pi}{10}\right) \\ \sin\left(\frac{\pi}{10}\right) \end{bmatrix}$ and $y = \begin{bmatrix} \cos\left(3\frac{\pi}{10}\right) \\ \sin\left(3\frac{\pi}{10}\right) \end{bmatrix}$.

Definition 2.7 (Convex Set [14, Sec. 2.1.4])**.** A set $\mathcal{S} \subseteq \mathbb{R}^n$ is said to be convex if and only if

$$\forall x, y \in \mathcal{S}\, \forall t \in [0,1] : tx + (1-t)y \in \mathcal{S}. \tag{2.9}$$

■

Fig. 2.1 shows an example for a convex and a non-convex set, respectively, where the second set is not convex as it violates (2.9).

Definition 2.8 (Convex Function [14, Sec. 3.1.1])**.** A function $g : \mathcal{G} \subseteq \mathbb{R}^n \to \mathbb{R}^m$ is convex if and only if

$$\forall x, y \in \mathcal{G}\, \forall t \in [0,1] : g\left(tx + (1-t)y\right) \leq tg\left(x\right) + (1-t)g\left(y\right), \tag{2.10}$$

where $\mathcal{G}$ is a convex set as introduced in Def. 2.7. ■

A function g is concave if $-g$ is convex. Fig. 2.2 shows an example for one convex and non-convex function. Clearly, the second function is not convex as (2.10) is violated for $x^{(1)} = 0.1$ and $x^{(2)} = 0.9$. Geometrically speaking, the second function is not convex since not all points on the line connecting $x^{(1)}$ and $x^{(2)}$ are above their respective function values (see (2.10)). We are now ready to define a convex optimization problem.

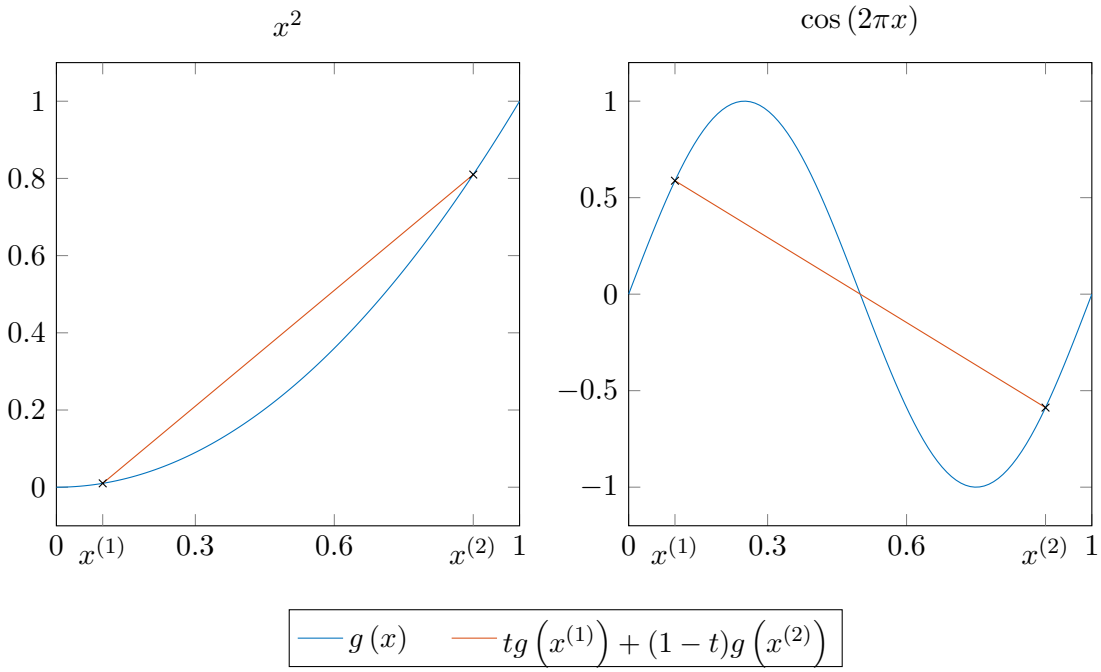


Figure 2.2: Example of a convex function ($g(x) = x^2$) and non-convex function ($g(x) = \cos(2\pi x)$) evaluated over $x \in [0, 1]$. Additionally we show the line connecting $\left(x^{(1)}, g\left(x^{(1)}\right)\right)$ to $\left(x^{(2)}, g\left(x^{(2)}\right)\right)$.

Definition 2.9 (Convex Minimization Problem)**.** We say that (2.6) is a convex minimization problem if and only if f and g are convex and h is affine. ■

Put differently, both the objective function f and the feasible set $\mathcal{S}$ have to be convex. Next, we briefly introduce important subclasses of convex optimization problems which are relevant for this thesis. For a more detailed introduction, see, e.g., [14].

2.3.3.1 Linear Program

The objective and constraint functions in linear programs (LPs) are linear in the optimization variable, i.e.

$$\min_x c^T x, \tag{2.11a}$$
$$\text{s.t. } Cx \leq d, \tag{2.11b}$$
$$Ax = b, \tag{2.11c}$$

with optimization variable $x \in \mathbb{R}^n$, where $c \in \mathbb{R}^n$, $C \in \mathbb{R}^{m \times n}$, $d \in \mathbb{R}^m$, $A \in \mathbb{R}^{q \times n}$, and $b \in \mathbb{R}^q$. Linear programs can be solved reliably and efficiently, even for a large number of optimization variables and constraints [14, Sec. 1.2.2], [22]. Since we assume that all optimization problems have a finite minimum, strong duality (see Prop. 2.4) always holds [25, Sec. 7.4].

2.3.3.2 Semi-Definite Program

With SDPs, one can formulate optimization problems of the form

$$\min_{x, X^{(i)}} \quad c^T x + \sum_{i=1}^{m} \text{trace}\left(C^{(i)}, X^{(i)}\right), \tag{2.12a}$$
$$\text{s.t.} \quad \underline{b}_j \leq {l^{(j)}}^T x + \sum_{i=1}^{N} \text{trace}\left(A^{(ij)}, X^{(i)}\right) \leq \bar{b}_j,\ 1 \leq j \leq q, \tag{2.12b}$$
$$\underline{d} \leq x \leq \bar{d}, \tag{2.12c}$$
$$x \in \mathcal{K}, \tag{2.12d}$$
$$X^{(i)} \succeq 0,\ i \in \{1, ..., N\}, \tag{2.12e}$$

where $x \in \mathbb{R}^n$ contains n scalar optimization variables, and $X^{(i)} \in \mathbb{S}_+^{n \times n}$ are m positive semi-definite optimization matrices. Further, we have $c \in \mathbb{R}^n$, $C^{(i)} \in \mathbb{S}^{n \times n}$, $\underline{b} \in \mathbb{R}^q$, $\bar{b} \in \mathbb{R}^q$, $l^{(j)} \in \mathbb{R}^n$, $A^{(ij)} \in \mathbb{S}^{n \times n}$, $\underline{d} \in \mathbb{R}^n$, $\bar{d} \in \mathbb{R}^n$, and $\mathcal{K} \subseteq \mathbb{R}^n$ with the Cartesian product of convex cones[4] (e.g., quadratic cone, rotated quadratic cone etc), where $\text{trace}\left(C^{(i)}, X^{(i)}\right) = \sum_{k,l} C_{kl}^{(i)} X_{kl}^{(i)}$ denotes the Frobenius inner product ($\square_{kl}$ denotes the element in the k-th row and l-th column).

In contrast to LPs, strong duality for SDPs additionally requires Slater's condition (see Prop. 2.2) to hold [51, Sec. 2.2]. Under that assumption, however, SDPs can be efficiently solved [110].

[4] for details, see, e.g.: `https://docs.mosek.com/MOSEKModelingCookbook-a4paper.pdf`

2.3.4 Non-Convex Optimization

While convex optimization offers great advantages, optimization problems in practice often cannot be formulated as a convex optimization problem. Thus, we introduce non-convex optimization problems in this section.

Definition 2.10 (Non-Convex Optimization Problem)**.** Let the optimization problem as defined in Def. 2.5 be given. We say that (2.6) is a non-convex optimization problem if it is not a convex optimization problem according to Def. 2.9. ■

In general, non-convex optimization problems cannot be solved efficiently to global optimality in general. As it turns out, even checking whether a feasible solution is a local minimum is in the NP-complete complexity class [84]. That said, when second-order methods are used, the number of function evaluations of the objective function and the constraint function required to converge to an ϵ-critical point of the optimization problem is polynomially bounded in the inverse of the accuracy parameter $\epsilon > 0$ [18]. A critical point hereby is a feasible point of (2.6) that is either a (global or local) minimum or a saddle point of (2.6).

2.3.5 Lagrangian Duality

When the optimization problem at hand cannot be reformulated as a convex optimization problem, finding a solution is substantially more difficult. However, instead of trying to find the optimum p^*, one may be satisfied with a lower bound on the original optimization problem in (2.6). In this section, we thus introduce the Lagrangian dual problem which is always a convex optimization problem and provides a lower bound on the optimum p^*.

Definition 2.11 (Lagrangian Dual Problem [14, Chap. 5, adapted])**.** Let f, g, and h be defined as in Def. 2.5, and let ϕ be defined as in (2.7). Further, let $\tilde{\mathcal{S}}$ collect any additional constraints on x that we do not wish to dualize. The dual problem of (2.6) is

$$d^* = \sup_{\lambda \geq 0, \mu} \inf_{x \in \tilde{\mathcal{S}}} \phi(x, \mu, \lambda), \tag{2.13}$$

where $\lambda \in \mathbb{R}^m_{\geq 0}$ and $\mu \in \mathbb{R}^o$ are the Lagrange multipliers for the inequality and equality constraints, respectively. ■

We collect relevant properties of the primal-dual relationship of (2.6) and (2.13) in the following proposition.

Proposition 2.4 (Primal-Dual Relationship [14, Chap. 5, adapted])**.** *Let the primal and dual problem be given as in* (2.6) *and* (2.13)*. The following properties are true:*

- *The dual problem is always a convex optimization problem, even if the primal problem is non-convex.*
- *The dual optimal value bounds the primal optimal value from below, i.e.,* $d^* \leq p^*$ *(weak duality).*

- *If the primal problem is convex and constraint qualifications (see Sec. 2.3.2) hold, then $d^* = p^*$, i.e., there is zero duality gap between the primal and the dual problem (strong duality).*

2.3.6 Modeling

When formulating optimization problems, it often happens that this formulation cannot be directly fed to numerical optimization solvers. Examples include the objective function or constraints containing absolute value expressions or maximizations over multiple functions; then the optimization problem is no longer differentiable. Subsequently, we derive equivalent but smooth reformulations of such optimization problems.

2.3.6.1 Absolute Values

Let

$$\min_x \left\{ f(x) + \sum_{i=1}^{q} |r_i(x)| \right\}, \tag{2.14a}$$

$$\text{s.t. } g(x) + \sum_{j=1}^{p} |c_j(x)| \leq 0, \tag{2.14b}$$

be given, where $x \in \mathbb{R}^n$, $f : \mathbb{R}^n \mapsto \mathbb{R}$, $r : \mathbb{R}^n \mapsto \mathbb{R}^q$, $g : \mathbb{R}^n \mapsto \mathbb{R}$, $c : \mathbb{R}^n \mapsto \mathbb{R}^p$, and we assume that all functions are sufficiently smooth. We only consider a single inequality constraint here as the following derivation can be easily extended to multiple inequality constraints. The non-differentiability of (2.14) due to the absolute values can be avoided by transforming (2.14) into the equivalent but differentiable optimization problem

$$\min_{x,s,v} \left\{ f(x) + \sum_{i=1}^{q} s_i \right\}, \tag{2.15a}$$

$$\text{s.t. } g(x) + \sum_{j=1}^{p} v_j \leq 0, \tag{2.15b}$$

$$\begin{bmatrix} +r(x) \\ -r(x) \end{bmatrix} \leq \begin{bmatrix} s \\ s \end{bmatrix}, \tag{2.15c}$$

$$\begin{bmatrix} +c(x) \\ -c(x) \end{bmatrix} \leq \begin{bmatrix} v \\ v \end{bmatrix}, \tag{2.15d}$$

with $s \in \mathbb{R}^q$ and $v \in \mathbb{R}^p$. This transformation can be justified as follows:

Let $\hat{x}$ denote the minimizer of (2.14) and let (x^*, s^*, v^*) denote the minimizer of (2.15). Further, we define

$$\mathcal{S} = \left\{ x \in \mathbb{R}^n \;\middle|\; g(x) + 1_p^T |c(x)| \leq 0 \right\},$$
$$\tilde{\mathcal{S}} = \left\{ x \in \mathbb{R}^n \;\middle|\; \exists v \geq |c(x)| : g(x) + 1_p^T v \leq 0 \right\},$$

where $\tilde{\mathcal{S}} = \mathcal{S}$ follows since choosing $v = |c(x)|$ is always possible. We have

$$\begin{aligned} f(\hat{x}) + 1_q^T |r(\hat{x})| &= \min_{x \in \mathcal{S}} f(x) + 1_q^T |r(x)| = \min_{x \in \tilde{\mathcal{S}}} f(x) + 1_q^T |r(x)| \\ &\leq \min_{\substack{x \in \tilde{\mathcal{S}} \\ |r(x)| \leq s}} f(x) + 1_q^T s = f(x^*) + 1_q^T s^*. \end{aligned} \tag{2.16}$$

On the other hand, $(\hat{x}, |r(\hat{x})|, |c(\hat{x})|)$ is a feasible point of (2.15), and by definition

$$f(x^*) + 1_q^T s^* \leq f(\hat{x}) + 1_q^T |r(\hat{x})|. \tag{2.17}$$

Combining (2.16) and (2.17) yields the equivalence of (2.14) and (2.15).

2.3.6.2 Finite Maximization

Let

$$\min_x f(x) + \max_{1 \leq i \leq q} r_i(x), \tag{2.18a}$$

$$\text{s.t. } g(x) + \max_{1 \leq j \leq p} c_j(x) \leq 0, \tag{2.18b}$$

with $x \in \mathbb{R}^n$, $f : \mathbb{R}^n \mapsto \mathbb{R}$, $r : \mathbb{R}^n \mapsto \mathbb{R}^q$, $g : \mathbb{R}^n \mapsto \mathbb{R}$, $c : \mathbb{R}^n \mapsto \mathbb{R}^p$, and where all functions are assumed to be sufficiently smooth. We limit ourselves to one inequality constraint, as an extension to multiple inequality constraints is straightforward. Further, we define $\hat{x}$ as a minimizer of (2.14) and (x^*, s^*, v^*) as a minimizer of (2.15). To avoid the maximizations in (2.18), we reformulate the problem as

$$\min_{x,s,v} f(x) + s, \tag{2.19a}$$

$$\text{s.t. } g(x) + v \leq 0, \tag{2.19b}$$

$$r(x) \leq s 1_q, \tag{2.19c}$$

$$c(x) \leq v 1_p, \tag{2.19d}$$

with $s \in \mathbb{R}$ and $v \in \mathbb{R}$. To show equivalence of (2.18) and (2.19), we first define

$$\mathcal{S} = \left\{ x \in \mathbb{R}^n \;\middle|\; g(x) + \max_{1 \leq j \leq p} c_j(x) \leq 0 \right\},$$

$$\tilde{\mathcal{S}} = \left\{ x \in \mathbb{R}^n \;\middle|\; \exists v \geq \max_{1 \leq j \leq p} c_j(x) : g(x) + v \leq 0 \right\},$$

from $\tilde{\mathcal{S}} = \mathcal{S}$ follows since choosing $v = \max_{1 \leq j \leq p} c_j(x)$ is always possible. Further, we have

$$\begin{aligned} f(\hat{x}) + \max_{1 \leq i \leq q} r_i(\hat{x}) &= \min_{x \in \mathcal{S}} f(x) + \max_{1 \leq i \leq q} r_i(x) = \min_{x \in \tilde{\mathcal{S}}} f(x) + \max_{1 \leq i \leq q} r_i(x) \\ &\leq \min_{\substack{x \in \tilde{\mathcal{S}}, s \\ \max_{1 \leq i \leq q} r_i(x) \leq s}} f(x) + s = f(x^*) + s^*. \end{aligned} \tag{2.20}$$

On the other hand, $(\hat{x}, \max_{1\leq i\leq q} r_i(\hat{x}), \max_{1\leq j\leq p} c_i(\hat{x}))$ is a feasible point of (2.19), and thus by definition

$$f(x^*) + \max_{1\leq i\leq q} r_i(x^*) \leq f(\hat{x}) + \max_{1\leq i\leq q} r_i(\hat{x}), \tag{2.21}$$

which yields the equivalence of (2.18) and (2.19) by combining (2.20) and (2.21).

2.4 Linear-Quadratic Regulator

In this section, we introduce LQR control [56], which will be used in later sections of this thesis. Let

$$\dot{x}(t) = Ax(t) + Bu(t),$$

be a given LTI system (A, B), where $x(t) \in \mathbb{R}^{n_x}$ denotes the state at time $t \in \mathbb{R}_{\geq 0}$, $u(t) \in \mathbb{R}^{n_u}$ denotes the controllable input, $A \in \mathbb{R}^{n_x \times n_x}$ is the system matrix, and $B \in \mathbb{R}^{n_x \times n_u}$ denotes the input matrix. For the introduction of LQR control, we first require the concept of controllability.

Corollary 2.1 (Controllability [56, adapted])**.** *An LTI system (A, B) as defined above, for which*

$$\operatorname{rank}\left(\begin{bmatrix} B, & AB, & A^2B, & \ldots, & A^{n_x-1}B \end{bmatrix}\right) = n_x,$$

holds, is controllable.

Put differently, an LTI system is controllable if and only if there exists an input trajectory such that any finite state $x(t)$ can be reached from any initial state $x(0)$ in finite time. Thus, assume that the LTI system (A, B) is controllable as per Corr. 2.1. The unique, optimal controller which minimizes

$$\min_{u(t)} \int_0^\infty \left(x(t)^T Q x(t) + u(t)^T R u(t)\right) \mathrm{d}t, \tag{2.22}$$

with $Q \in \mathbb{S}_{++}^{n_x \times n_x}$ and $R \in \mathbb{S}_{++}^{n_u \times n_u}$ (weaker conditions for Q and R are possible) is

$$u(t) = Kx(t), \tag{2.23}$$

where

$$K = -R^{-1}B^T X, \tag{2.24}$$

and $X \in \mathbb{S}_{++}^{n_x \times n_x}$ is the unique, positive-definite solution to the algebraic Riccati equation

$$A^T X + XA - XBR^{-1}B^T X + Q = 0. \tag{2.25}$$

For later convenience, we further state that solving the algebraic Riccati equation has the same computational complexity as computing the eigendecomposition of an n_x-by-n_x dimensional matrix [69]; thus, computing the gain matrix K for a given system has the same computational complexity since it otherwise only involves matrix multiplications.

2.5 Set Operations

In this thesis, we synthesize controllers for sets of initial states instead of a single initial state. As a result, we need to perform arithmetic operations on sets, which we define next.

Definition 2.12 (Minkowski Sum)**.** The Minkowski sum of two sets $\mathcal{A} \subseteq \mathbb{R}^n$ and $\mathcal{B} \subseteq \mathbb{R}^n$ is defined by

$$\mathcal{A} \oplus \mathcal{B} = \{a + b \mid a \in \mathcal{A},\ b \in \mathcal{B}\}.$$

■

Definition 2.13 (Linear Map)**.** The linear map of a set $\mathcal{S} \subseteq \mathbb{R}^n$ with matrix $M \in \mathbb{R}^{m \times n}$ is given by

$$M\mathcal{S} = \{Ms \mid s \in \mathcal{S}\}.$$

■

Definition 2.14 (Quadratic Map [5])**.** The quadratic map of a set $\mathcal{S} \subseteq \mathbb{R}^n$ with a set $\mathcal{Q} = \left\{Q^{(1)}, \ldots, Q^{(m)}\right\}$ of matrices $Q^{(i)} \in \mathbb{R}^{n \times n}$ for $1 \leq i \leq m$ is defined as

$$\mathrm{sq}(\mathcal{S}, \mathcal{Q}) = \left\{x \in \mathbb{R}^m \;\middle|\; x_i = s^T Q^{(i)} s,\ 1 \leq i \leq m,\ s \in \mathcal{S}\right\}.$$

■

In addition to operations that again produce a set, we further define the following operations on sets which return a scalar value.

Definition 2.15 (Support Function [41, Def. 1, adapted])**.** The support function of a convex set $\mathcal{S} \subseteq \mathbb{R}^n$ in direction $l \in \mathbb{R}^n$ is defined by

$$\rho_{\mathcal{S}}(l) = \sup_{s \in \mathcal{S}} l^T s.$$

■

For a set $\mathcal{S}(z)$ parameterized in some vector $z \in \mathbb{R}^n$, we denote its support function with $\rho_{\mathcal{S}}(l, z)$. For later convenience, we further define the short-hand

$$\rho_{\mathcal{S}}(L) = \left[\rho_{\mathcal{S}}\left(L^T_{(1,:)}\right),\ \ \rho_{\mathcal{S}}\left(L^T_{(2,:)}\right),\ \ \ldots,\ \ \rho_{\mathcal{S}}\left(L^T_{(m,:)}\right)\right]^T,$$

where $L \in \mathbb{R}^{m \times n}$.

Lastly, we later also require a measure of similarity for two sets, for which the Hausdorff distance is defined next.

Definition 2.16 (Hausdorff Distance [113, Def. 2])**.** The Hausdorff distance of two non-empty sets $\mathcal{X} \subseteq \mathbb{R}^n$ and $\mathcal{Y} \subseteq \mathbb{R}^n$ is defined by

$$d_H(\mathcal{X}, \mathcal{Y}) = \max\left(\sup_{x \in \mathcal{X}} \inf_{y \in \mathcal{Y}} \|x - y\|_2, \sup_{y \in \mathcal{Y}} \inf_{x \in \mathcal{X}} \|x - y\|_2\right).$$

■

Intuitively, we can say that the smaller the Hausdorff distance between two sets, the more similar (e.g. visually) these two sets are.

Table 2.1: Closedness of intervals ($\mathcal{I}$), zonotopes ($\mathcal{Z}$), polynomial zonotopes ($\mathcal{PZ}$), H-polytopes ($\mathcal{H}$), and ellipsoids ($\mathcal{E}$) under common operations [6, Tab. 1]. We differentiate between: Result can be efficiently and exactly computed (✓); result cannot be exactly represented (✗).

	$\mathcal{I}$	$\mathcal{Z}$	$\mathcal{PZ}$	$\mathcal{H}$	$\mathcal{E}$
Linear Map	✗	✓	✓	✗	✓
Minkowski Sum	✓	✓	✓	✗	✗
Quadratic Map	✗	✗	✓	✗	✗

2.6 Set Representations

In order to compute with sets, we need to represent them appropriately. Ideally, we find a set representation that is closed under all relevant operations: A set representation is said to be closed under an operation if the result of applying said operation can again be represented in the same set representation. That said, while being closed under relevant operations is important, we also need to be able to perform these set operations efficiently. As a result, there exist many different set representations in the literature which are closed under different set operations and have varying computational effort (see Tab. 2.1). Choosing the right set representation thus heavily depends on the algorithm we wish to implement. In this section, we therefore introduce all set representations which are relevant in this thesis.

Intuitively, the subsequently introduced set representations differ in how they parameterize an element of a set. The set itself is then generated by including all states that can be parameterized in that particular way.

Definition 2.17 (Set Generation [35, Def. 3])**.** We define

$$\{s(M)\}_M = \left\{ s(M) \;\middle|\; M \in [-1,1]^{m\times n} \right\} = \mathcal{S},$$

where $s : [-1,1]^{m\times n} \to \mathbb{R}^n$ is the generating function of $\mathcal{S} \subseteq \mathbb{R}^n$ with dependent factors $M \in [-1,1]^{m\times n}$. We say that $\mathcal{S}$ is generated by $s(M)$ over M. ■

Thus, the generating function allows us to express any element of a set in a unified way, i.e., by only varying M within the unit hypercube.

2.6.1 Intervals

As the most fundamental of set representations, intervals are widely used due to their compact representation size.

Definition 2.18 (Interval [4, Def. 2.4, adapted])**.** The interval with lower bound $\underline{x} \in \mathbb{R}^n$ and upper bound $\bar{x} \in \mathbb{R}^n$ is defined as

$$\mathcal{I} = [\underline{x}, \bar{x}] = \left\{ \frac{1}{2}(\bar{x} + \underline{x}) + \frac{1}{2}(\bar{x} - \underline{x})\,\beta \right\}_\beta = \{x \in \mathbb{R}^n \mid \underline{x} \leq x \leq \bar{x}\},$$

where inequalities are to be interpreted element-wise and $\underline{x} \leq \bar{x}$. ■

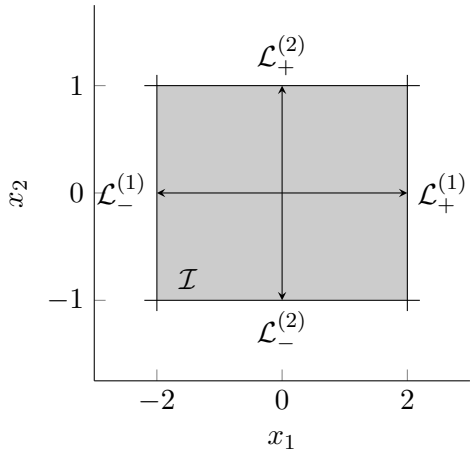


Figure 2.3: Visualization of the interval $\mathcal{I} = \left[\left[-2, \ -1\right]^T, \left[2, \ 1\right]^T\right]$, where $\mathcal{L}_\pm^{(i)} = \left\{x \in \mathbb{R}^2 \;\middle|\; \pm {e_{(2)}^{(i)}}^T x = \rho_\mathcal{I}\left(\pm e_{(2)}^{(i)}\right)\right\}$ for $1 \leq i \leq 2$ defines the hyperplanes which bound this interval.

Fig. 2.3 visualizes a simple interval $\mathcal{I} = \left[\begin{bmatrix}-2\\-1\end{bmatrix}, \begin{bmatrix}2\\1\end{bmatrix}\right]$.

Minkowski sum [4, Sec. 2.6, adapted]: Given two intervals $\mathcal{I} = [\underline{x}, \bar{x}] \subseteq \mathbb{R}^n$ and $\mathcal{L} = \left[\underline{l}, \bar{l}\right] \subseteq \mathbb{R}^n$, their Minkowski sum is

$$\mathcal{I} \oplus \mathcal{L} = \left[\underline{x} + \underline{l}, \bar{x} + \bar{l}\right].$$

Linear map: Since intervals are axis-aligned boxes and because linear transformations generally include rotations, intervals are not closed under linear maps. That said, the tight outer approximation of a linear map $M \in \mathbb{R}^{m \times n}$ of an interval $\mathcal{I} \subseteq \mathbb{R}^n$ is given by

$$M\mathcal{I} \subseteq \left\{x \in \mathbb{R}^m \;\middle|\; \begin{bmatrix}I_m\\-I_m\end{bmatrix} x \leq \rho_{M\mathcal{I}}\left(\begin{bmatrix}I_m\\-I_m\end{bmatrix}\right)\right\} = \left[-\rho_{M\mathcal{I}}\left(-I_m\right), \rho_{M\mathcal{I}}\left(I_m\right)\right],$$

since $\forall x \in M\mathcal{I} : {\pm e_m^{(i)}}^T x \leq \rho_{M\mathcal{I}}\left(\pm e_m^{(i)}\right)$ for $1 \leq i \leq m$ holds by definition (see Def. 2.15). Here, $\rho_{M\mathcal{I}}(\cdot)$ can be computed by first representing $M\mathcal{I}$ as a zonotope and then computing the resulting support function (see Sec. 2.6.5).

Quadratic map: Naturally, intervals cannot be closed under quadratic maps as they are convex sets. That said, one may simply write the interval as a zonotope and apply the outer approximation of the quadratic map for zonotopes from [7, Lem. 1].

Support function: Since each dimension of an interval $\mathcal{I} \subseteq \mathbb{R}^n$ is independent, it follows from Def. 2.15 that its support function in direction $l \in \mathbb{R}^n$ is given by

$$\rho_{\mathcal{I}}(l) = \max_{\underline{x} \leq x \leq \bar{x}} \sum_{k=1}^{n} l_k x_k = \sum_{k=1}^{n} \max_{\underline{x}_k \leq x_k \leq \bar{x}_k} l_k x_k = \sum_{k=1}^{n} \begin{cases} l_k \bar{x}_k, & \text{sign}(x_k) \geq 0, \\ l_k \underline{x}_k, & \text{sign}(x_k) < 0 \end{cases}.$$

2.6.2 Arbitrary Convex Set

While support functions as defined in Def. 2.15 are often used as set operations, they can also be used to represent arbitrary convex sets:

$$\mathcal{S} = \left\{ x \in \mathbb{R}^n \;\middle|\; \forall l \in \mathbb{R}^n : l^T x \leq \rho_{\mathcal{S}}(l) \right\}.$$

Fig. 2.3 visualizes the construction of a 2D interval using support functions.

Minkowski sum [41, Prop. 2, adapted]: The Minkowski sum between two convex sets $\mathcal{A} \in \mathbb{R}^n$ and $\mathcal{B} \in \mathbb{R}^n$ is given by

$$\mathcal{A} \oplus \mathcal{B} = \left\{ x \in \mathbb{R}^n \;\middle|\; \forall l \in \mathbb{R}^n : l^T x \leq \rho_{\mathcal{A}}(l) + \rho_{\mathcal{B}}(l) \right\}, \tag{2.26}$$

i.e., $\rho_{\mathcal{A} \oplus \mathcal{B}}(l) = \rho_{\mathcal{A}}(l) + \rho_{\mathcal{B}}(l)$ for $l \in \mathbb{R}^n$.

Linear map [41, Prop. 2, adapted]: The linear map $M \in \mathbb{R}^{m \times n}$ of a convex set $\mathcal{A} \subseteq \mathbb{R}^n$ represented using support functions is given by

$$M\mathcal{A} = \left\{ x \in \mathbb{R}^m \;\middle|\; \forall l \in \mathbb{R}^m : l^T x \leq \rho_{\mathcal{A}}\left(M^T l\right) \right\}, \tag{2.27}$$

i.e., $\rho_{M\mathcal{A}}(l) = \rho_{\mathcal{A}}\left(M^T l\right)$ for $l \in \mathbb{R}^m$.

2.6.3 Ellipsoids

Next, we define ellipsoids which – while being similarly compact in representation size as intervals – further allow for arbitrary rotations.

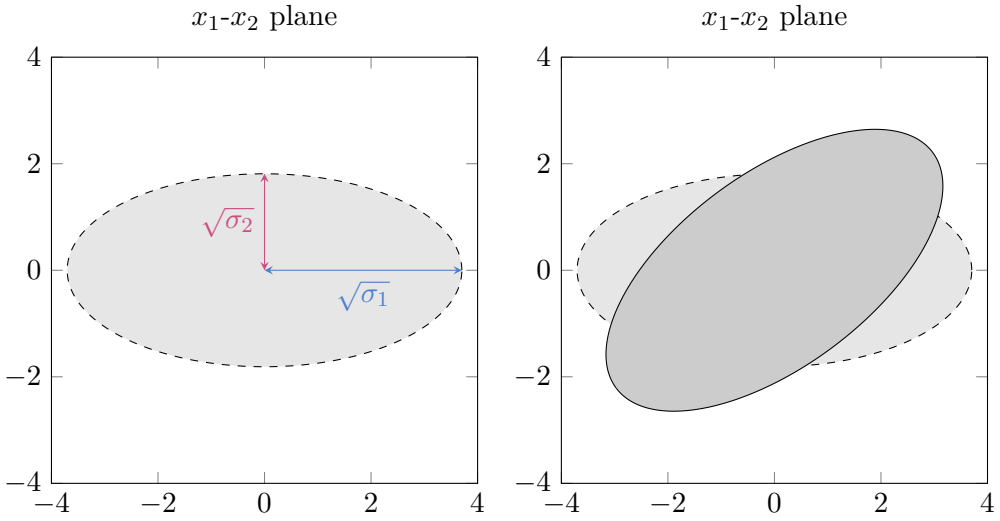
Definition 2.19 (Non-Degenerate Ellipsoid [67, Def. A.1])**.** The non-degenerate ellipsoid $\langle c, Q \rangle_E$ with center $c \in \mathbb{R}^n$ and shape matrix $Q \in \mathbb{S}_{++}^{n \times n}$ is given by

$$\langle c, Q \rangle_E = \left\{ x \in \mathbb{R}^n \;\middle|\; (x - c)^T Q^{-1} (x - c) \leq 1 \right\}.$$

■

The construction of an ellipsoid in demonstrated subsequently.

Example 2.3. Let $\mathcal{E} = \langle 0, Q \rangle_E = \left\langle 0, \begin{bmatrix} 10, & 5 \\ 5, & 7 \end{bmatrix} \right\rangle_E$. Since $Q \succ 0$, one can construct the ellipsoid easily using the singular value decomposition, which is visualized in Fig. 2.4: With the singular value decomposition given by $Q = U^T \operatorname{diag}(\sigma) U$, we first construct



(a) Construction of minor and major axes using singular values of shape matrix

(b) Rotated final ellipsoid

Figure 2.4: Construction of the ellipsoid from Ex. 2.3.

an axis-aligned ellipsoid $\langle 0, \operatorname{diag}(\sigma)\rangle_E$ which extends $\sqrt{\sigma}_1$ in the first and $\sqrt{\sigma}_2$ in the second unit direction. The original ellipsoid is then constructed by a simple rotation since $U\langle 0, \operatorname{diag}(\sigma)\rangle_E = \left\langle 0, U \operatorname{diag}(\sigma) U^T \right\rangle_E = \mathcal{E}$ (see below for details on the linear map of an ellipsoid). ■

Minkowski sum: Ellipsoids are not closed under the Minkowski sum. However, for m ellipsoids $\left\langle q^{(i)}, Q^{(i)} \right\rangle_E \subset \mathbb{R}^n$ with $1 \leq i \leq m$, an efficient outer approximation $\left\langle -\hat{W}^{-1}\hat{b}, \hat{W}^{-1} \right\rangle_E$ can be found as the solution of the SDP [15, Sec. 3.7.4] with

$$
\begin{aligned}
\hat{W}, \hat{b} = \arg\min_{W,b,\tau} \; & \log\det W^{-1}, \\
\text{s.t.} \; & \begin{bmatrix} {E^{(0)}}^T W E^{(0)}, & {E^{(0)}}^T b, & 0 \\ \left({E^{(0)}}^T b\right)^T, & -1, & b^T \\ 0, & b, & -W \end{bmatrix} - \sum_{i=1}^{m} \tau_i \begin{bmatrix} \tilde{A}^{(i)}, & \tilde{b}^{(i)}, & 0 \\ {\tilde{b}^{(i)}}^T, & c^{(i)}, & 0 \\ 0, & 0, & 0 \end{bmatrix} \preceq 0, \\
& \tau_i \geq 0, \; \forall i \in \{1, ..., m\}, \\
& W \succ 0,
\end{aligned}
$$

where

$$
\begin{aligned}
E^{(i)} &= \left[0_{n\times((i-1)n)}, \quad I_n, \quad 0_{n\times((m-i)n)}\right], \\
E^{(0)} &= \sum_{i=1}^{m} E^{(i)}, \\
\tilde{A}^{(i)} &= {E^{(i)}}^T A^{(i)} E^{(i)}, \\
\tilde{b}^{(i)} &= {E^{(i)}}^T b^{(i)}, \\
A^{(i)} &= {Q^{(i)}}^{-1}, \\
b^{(i)} &= -{Q^{(i)}}^{-1} q^{(i)}, \\
c^{(i)} &= {q^{(i)}}^T {Q^{(i)}}^{-1} q^{(i)}.
\end{aligned}
$$

Here, $0_{q\times p}$ denotes the q-by-p-dimensional zero matrix. For alternative approaches to the computation of the Minkowski sum, we refer the reader to [47].

Linear map [66, Sec. II]: For $A \in \mathbb{R}^{m\times n}$, the linear map of an ellipsoid $\langle c, Q\rangle_E \subset \mathbb{R}^n$ is

$$A \langle c, Q\rangle_E = \left\langle Ac, AQA^T \right\rangle_E. \tag{2.28}$$

Quadratic map: Since ellipsoids are convex sets, they cannot be closed under the quadratic map.

Support function [66, Sec. II]: The support function of an ellipsoid $\mathcal{E} = \langle c, Q\rangle_E \subset \mathbb{R}^n$ for a direction $l \in \mathbb{R}^n$ is

$$\rho_{\mathcal{E}}(l) = l^T c + \sqrt{l^T Q l}. \tag{2.29}$$

Using the support function of an ellipsoid allows for a definition which includes degenerate ellipsoids.

Definition 2.20 (Ellipsoid [67, Def. A.3])**.** The ellipsoid $\mathcal{E} = \langle c, Q\rangle_E$ with center $c \in \mathbb{R}^n$ and shape matrix $Q \in \mathbb{S}_+^{n\times n}$ is given by

$$\mathcal{E} = \left\{ x \in \mathbb{R}^n \;\middle|\; \forall l \in \mathbb{R}^n : l^T x \le \rho_{\mathcal{E}}(l) \right\}.$$

■

2.6.4 H-Polytopes

Typically, one distinguishes between V-polytopes, which are represented using their vertices, and H-polytopes, which enclose a given polytope with a finite number of halfspaces (see Fig. 2.5). In this thesis, we only require convex H-polytopes as defined next.

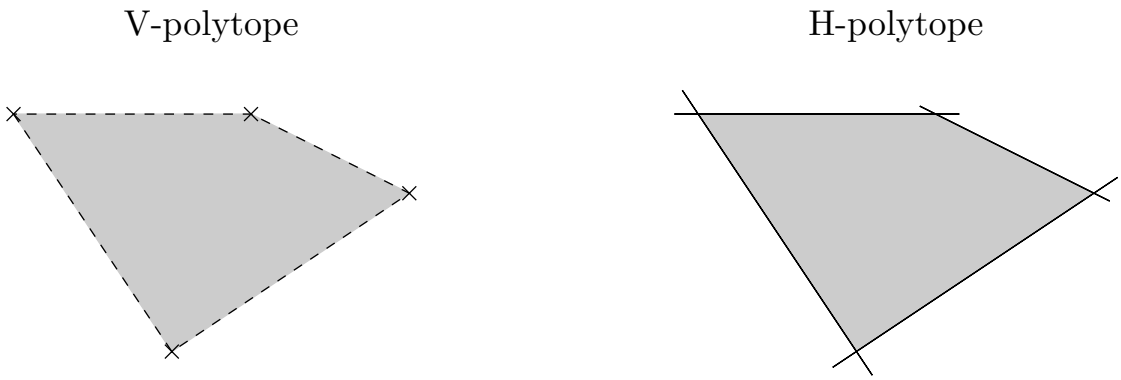


Figure 2.5: Visualization of a two-dimensional polytope, defined by its vertices (left) and its half-spaces (right).

Definition 2.21 (H-Polytope [4, Def. 2.21, adapted])**.** A polytope $\mathcal{H} = \langle A, b \rangle_H \subseteq \mathbb{R}^n$ with its halfspaces contained in $A \in \mathbb{R}^{m \times n}$ and its offsets given in $b \in \mathbb{R}^m$ is defined as

$$\mathcal{H} = \{x \in \mathbb{R}^n \mid Ax \leq b\}.$$

■

Other representations for polytopes exist, such as the Z-representation which uses polynomial zonotopes to represent polytopes [60].

Minkowski sum: H-polytopes are closed under the Minkowski sum since V-polytopes are closed under linear maps and both representations are equivalent [120, Th. 1.1]. However, the Minkowski sum between two H-polytopes is NP-hard since the conversion to an V-polytope requires vertex enumeration [109, Sec. 3].

That said, an efficiently computable outer approximation of the Minkowski sum $\mathcal{A} \oplus \mathcal{B}$ for two H-polytopes $\mathcal{A} \subseteq \mathbb{R}^n$ and $\mathcal{B} \subseteq \mathbb{R}^n$ is given by

$$\mathcal{A} \oplus \mathcal{B} \subseteq \langle L, \rho_{\mathcal{A} \oplus \mathcal{B}}(L) \rangle_H = \langle L, \rho_{\mathcal{A}}(L) + \rho_{\mathcal{A}}(L) \rangle_H,$$

where $L \in \mathbb{R}^{m \times n}$: With a given choice of directions in L, $\rho_{\mathcal{A} \oplus \mathcal{B}}(L) = \rho_{\mathcal{A}}(L) + \rho_{\mathcal{A}}(L)$ follows from (2.26) and thus the result follows since $\forall c \in \mathcal{A} \oplus \mathcal{B} : L_{(i,:)} c \leq \rho_{\mathcal{A} \oplus \mathcal{B}}\left(L_{(i,:)}^T\right)$ holds due to Def. 2.15 for $1 \leq i \leq m$ in. Thus, the outer approximation of $\mathcal{A} \oplus \mathcal{B}$ is also tight in directions $L_{(i,:)}^T$.

Linear map: H-polytopes are closed under linear maps since V-polytopes are closed under linear maps and both representations are equivalent [120, Th. 1.1]. However, a linear transformation generally rotates and projects the polytope, and computing the projection of an H-polytope is NP-hard [108].

Similarly to its Minkowski sum, however, an efficiently computable outer approximation of the linear map $M\mathcal{A}$ of an H-polytope $\mathcal{A} \subseteq \mathbb{R}^n$ with matrix $M \in \mathbb{R}^{q \times n}$, which is

tight in m directions collected in $L \in \mathbb{R}^{m \times q}$, is given by

$$M\mathcal{A} \subseteq \langle L, \rho_{M\mathcal{A}}(L) \rangle_H = \langle L, \rho_{\mathcal{A}}(LM) \rangle_H .$$

This follows, analogously to the Minkowski sum above, from Def. 2.15 and (2.27).

Quadratic Map: Since H-polytopes are convex sets, they cannot be closed under the quadratic map.

Support function: The support function of an H-polytope $\mathcal{H} = \langle A, b \rangle_H$ follows directly from Def. 2.15 as

$$\rho_{\mathcal{H}}(l) = \max_{Ax \leq b} l^T x.$$

2.6.5 Zonotopes

Even though intervals and ellipsoids can be compactly represented, they lack representational power. Zonotopes are a popular choice for set-based computations, as they are closed under linear maps and the Minkowski sum.

Definition 2.22 (Zonotope [40, Def. 1, adapted])**.** A zonotope $\mathcal{Z} = \langle c, G \rangle_Z$ with center $c \in \mathbb{R}^n$ and generator matrix $G \in \mathbb{R}^{n \times m}$ consisting of generators $G_{(:,i)}$ for $1 \leq i \leq m$ is given by

$$\mathcal{Z} = \langle c, G \rangle_Z = \{ c + G\nu \}_\nu ,$$

where $c + G\nu$ is its generating function with dependent factors $\nu \in [-1, 1]^m$. ■

We call an n-dimensional zonotope with n generators a parallelotope. We illustrate the construction of a zonotope in the following example.

Example 2.4. Let $\mathcal{Z} = \left\langle 0, \begin{bmatrix} 1, & 3, & 1 \\ 2, & 0, & 1 \end{bmatrix} \right\rangle_Z$ be given. Fig. 2.6 shows the construction of the zonotope for all three generators: Since

$$\mathcal{Z} = \left\langle 0, \begin{bmatrix} 1 \\ 2 \end{bmatrix} \right\rangle_Z \oplus \left\langle 0, \begin{bmatrix} 3 \\ 0 \end{bmatrix} \right\rangle_Z \oplus \left\langle 0, \begin{bmatrix} 1 \\ 1 \end{bmatrix} \right\rangle_Z ,$$

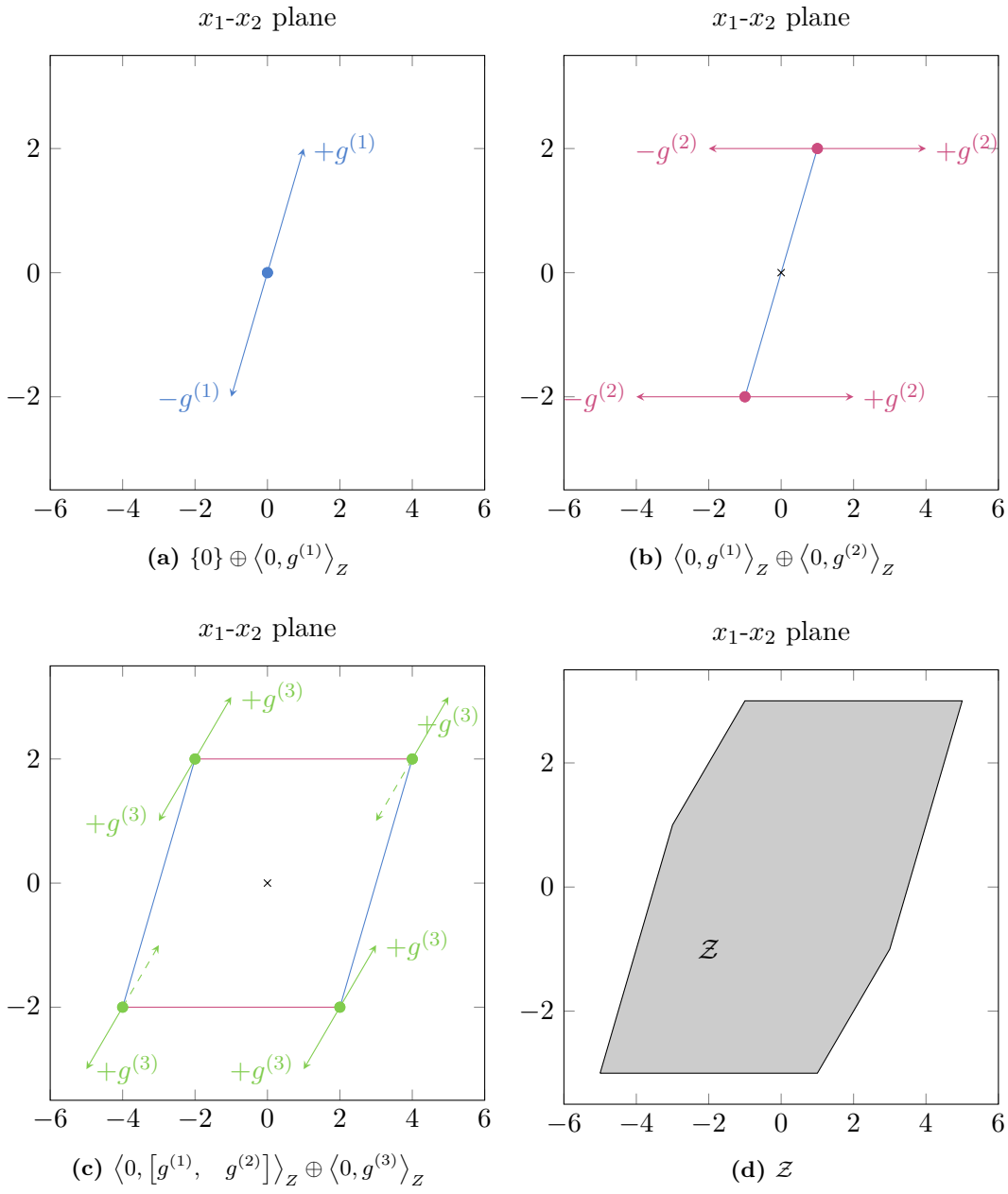
the zonotope can be constructed by performing sequential Minkowski additions of each generator. ■

Minkowski sum [40, Sec. 2]: Given two zonotopes $\langle c, G \rangle_Z \subset \mathbb{R}^n$ and $\langle d, M \rangle_Z \subset \mathbb{R}^n$, their Minkowski sum is

$$\langle c, G \rangle_Z \oplus \langle d, M \rangle_Z = \left\langle c + d, \begin{bmatrix} G, & M \end{bmatrix} \right\rangle_Z .$$

Linear map [40, Sec. 2]: The linear map $A \in \mathbb{R}^{m \times n}$ of a zonotope $\langle c, G \rangle_Z \subset \mathbb{R}^n$ is

$$A \langle c, G \rangle_Z = \langle Ac, AG \rangle_Z .$$



(a) $\{0\} \oplus \langle 0, g^{(1)} \rangle_Z$ **(b)** $\langle 0, g^{(1)} \rangle_Z \oplus \langle 0, g^{(2)} \rangle_Z$

(c) $\langle 0, [g^{(1)}, \quad g^{(2)}] \rangle_Z \oplus \langle 0, g^{(3)} \rangle_Z$ **(d)** $\mathcal{Z}$

Figure 2.6: Construction of the zonotope $\mathcal{Z} = [g^{(1)}, \quad g^{(2)}, \quad g^{(3)}]$ from Ex. 2.4.

Quadratic map: Since sets described with support functions are convex by definition, they cannot be closed under quadratic maps. That said, one can apply the efficient outer approximation from [7, Lem. 1].

Support function [45, Prop 2.2, adapted]: The support function of a zonotope $\mathcal{Z} = \langle c, G \rangle_Z \subset \mathbb{R}^n$ in direction $l \in \mathbb{R}^n$ is given by

$$\rho_{\mathcal{Z}}(l) = c^T l + \left\| G^T l \right\|_1 . \tag{2.30}$$

2.6.6 Polynomial Zonotopes

As an extension to zonotopes, polynomial zonotopes were first introduced in [5] to represent non-convex reachable sets (see Sec. 2.7.3).

Definition 2.23 (Polynomial Zonotope [61, Def. 1, adapted])**.** Let $c \in \mathbb{R}^n$ be the starting point, $G \in \mathbb{R}^{n \times m}$ the dependent generator matrix with generators $G_{(:,i)} \in \mathbb{R}^n$ for $1 \leq i \leq m$, $G_{\text{indep}} \in \mathbb{R}^{n \times o}$ the independent generator matrix, and $E \in \mathbb{N}^{d \times m}$ the exponent matrix. We define the generating function of a polynomial zonotope as

$$g(\nu, \xi) = c + \sum_{i=1}^{m} G_{(:,i)} \nu^{E_{(:,i)}} + G_{\text{indep}} \xi,$$

with dependent factors $\nu \in [-1, 1]^d$ and independent factors $\xi \in [-1, 1]^o$. The polynomial zonotope is then constructed as $\mathcal{PZ} = \{g(\nu, \xi)\}_{\nu,\xi}$ with the shorthand $\mathcal{PZ} = \langle c, G, G_{\text{indep}}, E \rangle_{PZ}$. ■

The following example demonstrates the construction of a simple, non-convex polynomial zonotope.

Example 2.5. Let

$$\mathcal{PZ} = \left\langle 0, \begin{bmatrix} 1, & 2, & 1 \\ 2, & 1, & 0 \end{bmatrix}, 0, \begin{bmatrix} 1, & 0, & 2 \\ 0, & 1, & 0 \end{bmatrix} \right\rangle_{PZ} = \left\{ \begin{bmatrix} 1 \\ 2 \end{bmatrix} \beta_1 + \begin{bmatrix} 2 \\ 1 \end{bmatrix} \beta_2 + \begin{bmatrix} 1 \\ 0 \end{bmatrix} \beta_1^2 \right\}_{\beta},$$

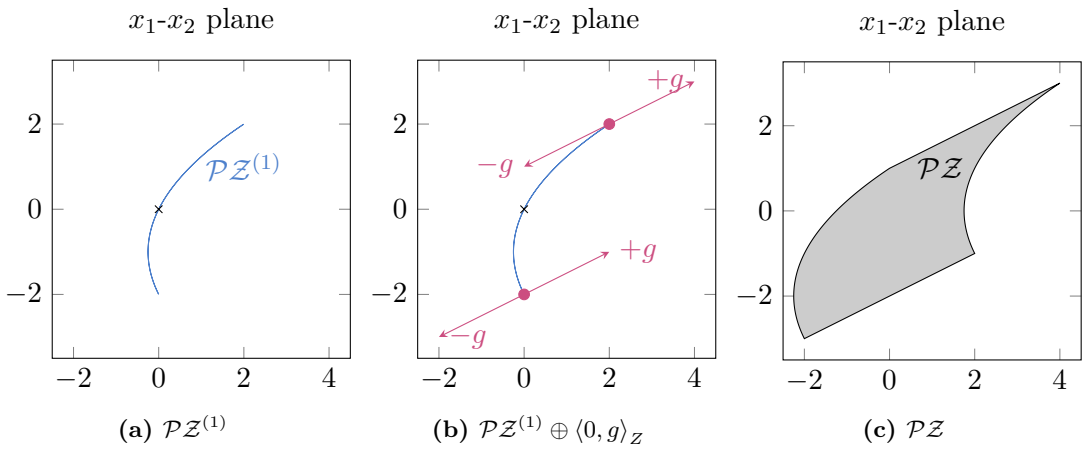
where $\beta \in [-1, 1]^2$. Since

$$\mathcal{PZ} = \mathcal{PZ}^{(1)} \oplus \langle 0, g \rangle_Z = \left\{ \begin{bmatrix} 1 \\ 2 \end{bmatrix} \beta_1 + \begin{bmatrix} 1 \\ 0 \end{bmatrix} \beta_1^2 \right\}_{\beta_1} \oplus \left\langle 0, \begin{bmatrix} 2 \\ 1 \end{bmatrix} \right\rangle_Z ,$$

the construction of $\mathcal{PZ}$ is similar to that of a zonotope, with the difference being that the first generator is not linear. Fig. 2.7 shows its construction.

■

We introduce the following relevant operations on polynomial zonotopes (see [61] for details).



(a) $\mathcal{PZ}^{(1)}$ **(b)** $\mathcal{PZ}^{(1)} \oplus \langle 0, g \rangle_Z$ **(c)** $\mathcal{PZ}$

Figure 2.7: Construction of the non-convex polynomial zonotope $\mathcal{PZ} = \mathcal{PZ}^{(1)} \oplus \langle 0, g \rangle_Z$ from Ex. 2.5.

Minkowski sum [61, Prop. 9, adapted]: Given two polynomial zonotopes $\mathcal{A} = \langle c, G, G_{\text{indep}}, E \rangle_{PZ} \subset \mathbb{R}^n$ and $\mathcal{B} = \langle d, M, M_{\text{indep}}, O \rangle_{PZ} \subset \mathbb{R}^n$, their Minkowski sum is given by

$$\mathcal{A} \oplus \mathcal{B} = \left\langle c + d, \begin{bmatrix} G & , M \end{bmatrix}, \begin{bmatrix} G_{\text{indep}} & , M_{\text{indep}} \end{bmatrix}, \text{blkdiag}\left(E, O\right) \right\rangle_{PZ}.$$

Exact sum: Given $\mathcal{A} = \{a\left(\xi, \alpha\right)\}_{\xi,\alpha} \subset \mathbb{R}^n$ and $\mathcal{B} = \{b\left(\xi, \beta\right)\}_{\xi,\beta} \subset \mathbb{R}^n$ as polynomial zonotopes, the exact sum of $\mathcal{A}$ and $\mathcal{B}$ is

$$\mathcal{A} \oplus_e \mathcal{B} = \{a\left(\xi, \alpha\right) + b\left(\xi, \beta\right)\}_{\xi,\alpha,\beta}.$$

Note that $\mathcal{A} \oplus_e \mathcal{B} \subseteq \mathcal{A} \oplus \mathcal{B}$ since $\{a\left(\xi, \alpha\right) + b\left(\xi, \beta\right)\}_{\xi,\alpha,\beta} \subseteq \{a\left(\xi, \alpha\right)\}_{\xi,\alpha} \oplus \{b\left(\xi, \beta\right)\}_{\xi,\beta}$ where $\mathcal{A} \oplus_e \mathcal{B} = \mathcal{A} \oplus \mathcal{B}$ if and only if the generating functions a and b do not share any dependent factors. We refer the reader to [61, Prop. 10] for details.

Linear map [61, Prop. 8, adapted]: The linear map $A \in \mathbb{R}^{m \times n}$ of an n-dimensional polynomial zonotope $\langle c, G, G_{\text{rest}}, E \rangle_{PZ}$ is

$$A \left\langle c, G, G_{\text{rest}}, E \right\rangle_{PZ} = \left\langle Ac, AG, AG_{\text{rest}}, E \right\rangle_{PZ}.$$

Quadratic map: The quadratic map of a polynomial zonotope $\mathcal{A} = \{a\left(\alpha\right)\}_{\alpha} \subset \mathbb{R}^n$, following the definition in Def. 2.14, is given by

$$\text{sq}\left(\mathcal{A}, \mathcal{Q}\right) = \left\{ \begin{bmatrix} a\left(\alpha\right)^T Q^{(1)} a\left(\alpha\right) \\ \vdots \\ a\left(\alpha\right)^T Q^{(m)} a\left(\alpha\right) \end{bmatrix} \right\}_{\alpha},$$

where $\mathcal{Q}$ collects m matrices $Q^{(i)} \in \mathbb{R}^n$ for $1 \leq i \leq m$. For an efficient implementation of this quadratic map, we refer the reader to [61, Prop. 12, Prop. 13].

Support function: Since a polynomial zonotope in general represents a non-convex set, its support function cannot be evaluated exactly as the evaluation of the support function essentially requires range bounding of a polynomial, which is NP-hard in general [33]. We refer the reader to [61, Prop. 7] for an efficient outer approximation.

That said, using an efficient outer approximation of a polynomial zonotope to a zonotope – introduced in Sec. 4.1.4 – one can use the support function value of a zonotope (see Sec. 2.6.5) as an upper bound.

Subset property: Let $\mathcal{X}^{(0)} = \left\{x^{(0)}(\beta)\right\}_\beta \subset \mathbb{R}^n$ be a polynomial zonotope and let $\mathcal{R} = \{r(\beta)\}_\beta$ be the resulting polynomial zonotope of an algorithm (A) that takes $\mathcal{X}^{(0)}$ as an input and returns $\mathcal{R}$ by applying a finite number of elementary set operations, such as Minkowski sums or linear maps. The subset property of polynomial zonotopes as described in [62] allows us the extract the solution of (A) for a subset of $\mathcal{X}^{(0)}$ without executing (A) again, as the following example demonstrates.

Example 2.6 (Subset property of polynomial zontopes)**.** Let $\mathcal{X}^{(0)} = \left\{x^{(0)}(\beta)\right\}_\beta = \{\beta\}_\beta$ for $\beta \in [-1,1]^2$ and assume that the algorithm (A) computes

$$\mathcal{R} = \{r(\beta)\}_\beta = \left\{\begin{bmatrix}1\\2\end{bmatrix}\right\} \oplus \operatorname{sq}\left(\mathcal{X}^{(0)}, \mathcal{Q}\right) = \left\{\begin{bmatrix}1\\2\end{bmatrix}\right\} \oplus \left\{\begin{bmatrix}2\beta_1\beta_2\\ \beta_2^2\end{bmatrix}\right\}_\beta = \left\{\begin{bmatrix}1+2\beta_1\beta_2\\ 2+\beta_2^2\end{bmatrix}\right\}_\beta,$$

for

$$\mathcal{Q} = \left\{\begin{bmatrix}0, & 1\\ 1, & 0\end{bmatrix}, \begin{bmatrix}0, & 0\\ 0, & 1\end{bmatrix}\right\}.$$

Now, we want to compute the output $\breve{\mathcal{R}}$ of algorithm (A) for a subset

$$\breve{\mathcal{X}}^{(0)} = \left\{x^{(0)}(b(\beta))\right\}_\beta = \left\{\begin{bmatrix}b_1(\beta)\\ b_2(\beta)\end{bmatrix}\right\}_\beta = \left\{\frac{1}{2}\beta\right\}_\beta,$$

where $b(\beta) = \frac{1}{2}\beta$. Since we have an analytic expression for the generating function $r(\beta)$ of $\mathcal{R}$, this output is simply given by substituting β with $b(\beta)$, i.e., we have

$$\breve{\mathcal{R}} = \{r(b(\beta))\}_\beta = \left\{\begin{bmatrix}1+\frac{1}{2}\beta_1\beta_2\\ 2+\frac{1}{4}\beta_2^2\end{bmatrix}\right\}_\beta.$$

Fig. 2.8 visualizes $\mathcal{X}^{(0)}$ and its output $\mathcal{R}$ as well as the subset $\breve{\mathcal{X}}^{(0)} \subseteq \mathcal{X}^{(0)}$ and the corresponding output $\breve{\mathcal{R}}$.

■

In this example, $\mathcal{R}$ can be computed exactly since it only involves a single quadratic map. However, an algorithm often contains many more elementary operations, which causes the representation of the resulting polynomial zonotope to grow. To reduce this complexity, reduction techniques (see Sec. 2.8) are necessary.

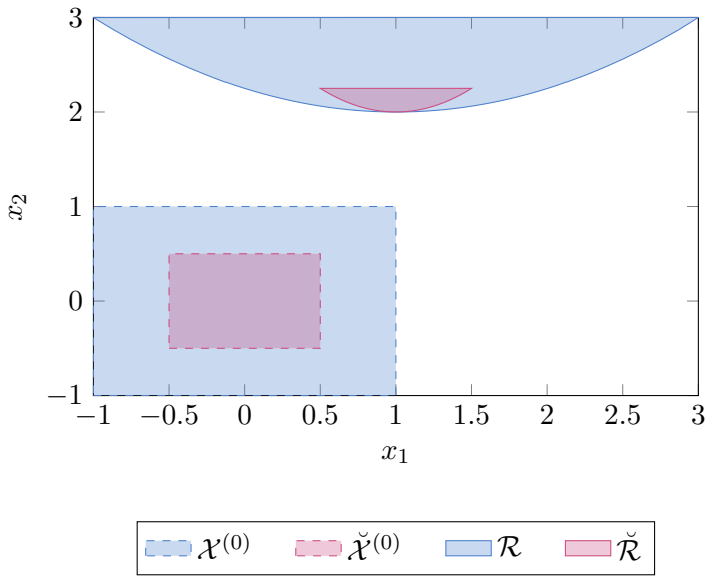


Figure 2.8: Illustration of the subset property of polynomial zonotopes using a 2D example.

2.7 Reachability Analysis

In order to synthesize a controller for a given task, the underlying model needs to be mathematically described so that predictions about its future evolution can be made, based on which a controller can be synthesized. Since we compute with sets of initial states instead of a single initial state, this chapter introduces reachability analysis: For a given set of initial states and a set of inputs, reachability analysis computes a set of future states that is guaranteed to contain all possible future states of a given system. Here, we only provide a short introduction to the most relevant aspects of reachability analysis; we refer the reader to [4, 59] for more details.

Definition 2.24 (System)**.** An n_x-dimensional system with state $x(t) \in \mathbb{R}^{n_x}$, controllable input $u(x(t), t) \in \mathbb{R}^{n_u}$, and uncontrollable disturbance $w(t) \in \mathbb{R}^{n_w}$ is governed by the system of ordinary differential equations (ODEs)

$$\dot{x}(t) = f(x(t), u(x(t), t), w(t)), \tag{2.31}$$

with $t \in \mathbb{R}_{\geq 0}$ being the time and where $\xi(t, x(0), u(\cdot, \cdot), w(\cdot))$ denotes the solution of (2.31) at time t starting at the initial state $x(0)$. We assume that f is sufficiently smooth. ■

Without loss of generality, we assume that any input u dependent on the state has already been substituted into f and define

$$\dot{x} = f_{\text{cl}}(x, w), \tag{2.32}$$

which absorbs the fixed controller. With slight abuse of notation, the solution $x(t)$ of the closed-loop dynamics in (2.32) is then denoted by $\xi(t, x(0), w(\cdot))$. The reachable set is then defined as follows:

Definition 2.25 (Reachable Set)**.** The reachable set at time $t \in \mathbb{R}_{\geq 0}$ for an initial set $\mathcal{X}^{(0)} \subset \mathbb{R}^{n_x}$ and a set of possible disturbances $\mathcal{W} \subset \mathbb{R}^{n_w}$ is defined by

$$\mathcal{R}^{(e)}(t) = \left\{ x(t) \in \mathbb{R}^{n_x} \;\middle|\; \forall \tau \in [0,t] : \dot{x}(\tau) = f_{\text{cl}}(x(\tau), w(\tau)),\ x(0) \in \mathcal{X}^{(0)}, w(\tau) \in \mathcal{W} \right\}.$$

■

Fig. 1.1 visualizes the main idea of reachability analysis: For a given initial state and disturbance $w(t)$, the reachable states are given by the trajectory of the system for these values. The reachable set is then the union of all trajectories starting in $\mathcal{X}^{(0)}$ with any realization of the disturbance $w(t)$. Since in general, computing the exact reachable set $\mathcal{R}^{(e)}$ is not tractable [68], we use an outer approximation $\mathcal{R}$ of the reachable set for formal verification, or an approximation $\tilde{\mathcal{R}}$ (without formal guarantees) for optimization.

We now briefly introduce the reachability algorithms relevant for this thesis from [9, 5, 4]. All algorithms presented subsequently are available in the continuous reachability analyzer[5] [2] (CORA), a MATLAB toolbox for reachability analysis and set computations. We compute the reachable sets for $t \in [0, t_{\text{f}}]$, where $t_{\text{f}} \in \mathbb{R}_+$ is the final time and $r = \frac{t_{\text{f}}}{m_r}$ is the duration of one reachability step with $m_r \in \mathbb{N}_+$ being the number of reachability steps.

2.7.1 Linear Time-Invariant Systems

In this section, we introduce reachability analysis for an LTI system (see [4, Sec. 3.2] for details)

$$\dot{x}(t) = Ax(t) + w(t). \tag{2.33}$$

The well-known solution of (2.33) given $x(0) \in \mathcal{X}^{(0)}$ is

$$x(t) = e^{At}x(0) + \int_0^t e^{A(t-\tau)}w(\tau)\,\mathrm{d}\tau. \tag{2.34}$$

Interpreting (2.34) in a set-based manner yields the reachable set

$$\mathcal{R}^{(e)}(t) = \mathcal{H}^{(e)}(t) \oplus \mathcal{P}^{(e)}(t), \tag{2.35}$$

where

$$\mathcal{H}^{(e)}(t) = \left\{ e^{At}x(0) \;\middle|\; x(0) \in \mathcal{X}^{(0)} \right\}, \tag{2.36}$$

$$\mathcal{P}^{(e)}(t) = \left\{ \int_0^t e^{A(t-\tau)}w(\tau)\,\mathrm{d}\tau \;\middle|\; \forall \tau \in [0,t] : w(\tau) \in \mathcal{W} \right\}. \tag{2.37}$$

Subsequently, we first focus an a sub interval $t \in [0, r] \subset [0, t_{\text{f}}]$. The extension to $t \in [0, t_{\text{f}}]$ is then made at the end of this section.

[5] `https://cora.in.tum.de/`

Time-point solution: While the homogeneous solution directly follows as $\mathcal{H}^{(e)}(r) = \mathcal{H}(r) = e^{Ar}\mathcal{X}^{(0)}$ from (2.36), only an outer approximation $\mathcal{P}(r) \supseteq \mathcal{P}^{(e)}(r)$ of the exact particular solution $\mathcal{P}^{(e)}(r)$ in (2.37) can be computed using [4, Th 3.1], i.e.

$$\mathcal{P}(r) = \bigoplus_{k=0}^{\mu} \left(\frac{A^k r^{k+1}}{(k+1)!} \mathcal{W} \right) \oplus \mathcal{I}_{\text{rem}}(r)\, r\mathcal{W}, \tag{2.38}$$

with

$$\mathcal{I}_{\text{rem}}(r) = [-1, 1]^{n_x \times n_x} \frac{(\|A\|_\infty r)^{\mu+1}}{(\mu+1)!} \frac{1}{1-\epsilon},\ \epsilon = \frac{\|A\|_\infty r}{\mu+2},$$

and where it is required that $\epsilon < 1$. As a result, μ and the number of time steps m_r are typically chosen large enough to ensure $\epsilon < 1$. Plugging $\mathcal{H}(r)$ and $\mathcal{P}(r)$ from (2.38) into (2.35) yields an outer approximation of the time-point reachable set at $t = r$ by

$$\mathcal{R}(r) = \mathcal{H}(r) \oplus \mathcal{P}(r) \supseteq \mathcal{R}^{(e)}(r).$$

Time-interval solution: In [4, Sec. 3.2.2], it is proven that $\mathcal{P}([0, r]) = \mathcal{P}(r)$ if $0 \in \mathcal{W}$. Hence, we introduce $\tilde{w} \in \mathcal{W}$ so that $0 \in \tilde{\mathcal{W}} = \{-\tilde{w}\} \oplus \mathcal{W}$ and define $\tilde{\mathcal{P}}(r) = \tilde{\mathcal{P}}([0, r])$ analogously to (2.38) using $\tilde{\mathcal{W}}$. Thus, it only remains to find an outer approximation of the time-interval homogeneous solution $\mathcal{H}^{(e)}([0, r])$ and the time-interval solution $\mathcal{P}_{\tilde{w}}^{(e)}([0, r])$ due to the constant disturbance $\tilde{w}$. If we approximate the homogeneous solution $x^{(h)}(t)$ and the constant disturbance solution $x_{\tilde{w}}(t) \in \mathcal{P}_{\tilde{w}}^{(e)}(t)$ for $t \in [0, r]$ as

$$x^{(h)}(t) \approx x(0) + \frac{t}{r}\left(e^{Ar}x(0) - x(0)\right) = \left(1 - \frac{t}{r}\right) x(0) + \frac{t}{r} e^{Ar} x(0), \tag{2.39}$$

$$x_{\tilde{w}}(t) \approx \frac{t}{r} \int_0^r e^{A\tau}\tilde{w}\, \mathrm{d}\tau = \left(1 - \frac{t}{r}\right) 0 + \frac{t}{r} x_{\tilde{w}}(r), \tag{2.40}$$

summation of (2.37) and (2.39) and its set-based evaluation for $t \in [0, r]$ yields (see Def. 2.3)

$$\mathcal{H}^{(e)}([0, r]) \oplus \mathcal{P}_{\tilde{w}}^{(e)}([0, r]) \approx \text{convc}\left(\mathcal{X}^{(0)}, e^{Ar}\mathcal{X}^{(0)} \oplus \mathcal{P}_{\tilde{w}}^{(e)}(r)\right).$$

Since (2.39) and (2.40) only approximate the time interval solutions, we additionally need to compute interval matrices $\mathcal{F}$ and $\tilde{\mathcal{F}}$ (see [4, Sec. 3.2] for details) to account for the curvature of solutions, so that an outer approximation is given by

$$\mathcal{H}^{(e)}([0, r]) \oplus \mathcal{P}_{\tilde{w}}^{(e)}([0, r]) \subseteq \text{convc}\left(\mathcal{X}^{(0)}, e^{Ar}\mathcal{X}^{(0)} \oplus \mathcal{P}_{\tilde{w}}(r)\right) \oplus \mathcal{F}\mathcal{X}^{(0)} \oplus \tilde{\mathcal{F}}\tilde{w},$$

where $\mathcal{P}_{\tilde{w}}(r) \supseteq \mathcal{P}_{\tilde{w}}^{(e)}(r) = \int_0^r e^{A\tau}\tilde{w}\, \mathrm{d}\tau$ is computed using, e.g., (2.38) with $\mathcal{W} = \{\tilde{w}\}$, since $\mathcal{P}_{\tilde{w}}^{(e)}(r)$ generally cannot be computed exactly. Thus, the time-interval solution is given by

$$\mathcal{R}([0, r]) = \text{convc}\left(\mathcal{X}^{(0)}, e^{Ar}\mathcal{X}^{(0)} \oplus \mathcal{P}_{\tilde{w}}(r)\right) \oplus \mathcal{F}\mathcal{X}^{(0)} \oplus \tilde{\mathcal{F}}\tilde{w} \oplus \tilde{\mathcal{P}}(r) \supseteq \mathcal{R}^{(e)}([0, r]).$$

Fig. 2.9 visualizes the computation of the reachable set, where we assume that $0 \in \mathcal{W}$, i.e., $\tilde{w} = 0$, for illustrative purposes:

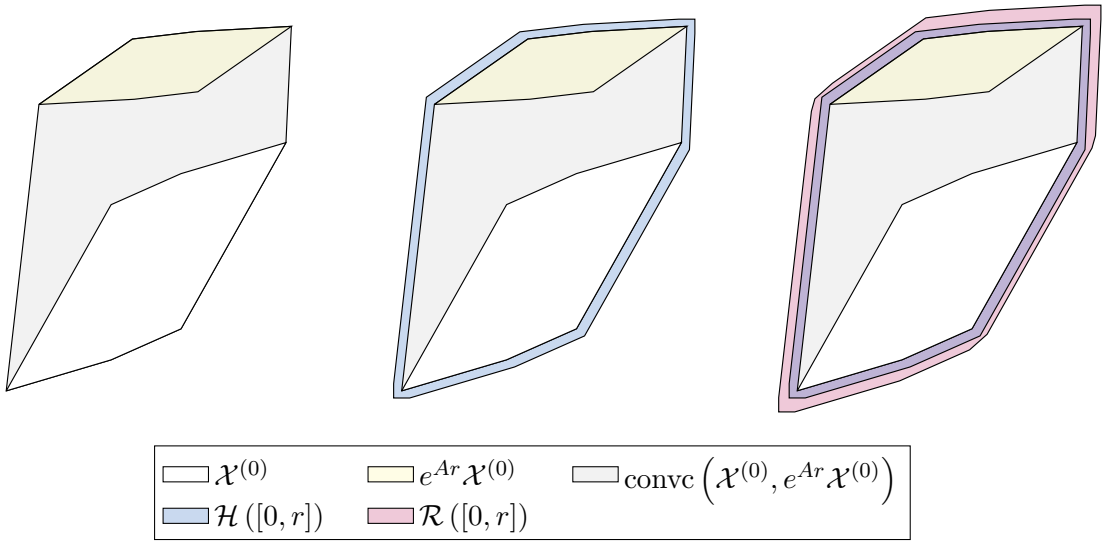


Figure 2.9: One step of reachability analysis for linear systems.

(i) First, we approximate the set of homogeneous solutions $\mathcal{H}\left([0,r]\right)$ for $t \in [0,r]$ as convc $\left(\mathcal{X}^{(0)}, e^{Ar}\mathcal{X}^{(0)}\right)$.

(ii) To arrive at an outer approximation of the homogeneous solution for $t \in [0,r]$, we additionally need to account for the curvature of trajectories, which yields $\mathcal{H}\left([0,r]\right) = \text{convc}\left(\mathcal{X}^{(0)}, e^{Ar}\mathcal{X}^{(0)}\right) \oplus \mathcal{F}\mathcal{X}^{(0)}$.

(iii) As a last step, we add the effect of the disturbance, i.e., the particular time-interval solution, which finally yields the time-interval reachable set $\mathcal{R}\left([0,r]\right) = \text{convc}\left(\mathcal{X}^{(0)}, e^{Ar}\mathcal{X}^{(0)}\right) \oplus \mathcal{F}\mathcal{X}^{(0)} \oplus \mathcal{P}\left(r\right)$.

Following [4, Sec. 3.2.2], one can then iteratively compute the outer approximations $\mathcal{R}\left([k,k+1]\,r\right)$ of the subsequent time-interval reachable sets for $0 \leq k \leq m_r - 1$, and we obtain $\mathcal{R}\left([0,t_\text{f}]\right) = \bigcup_{k=0}^{m_r-1} \mathcal{R}\left([k,k+1]\,r\right)$. If we use zonotopes for these iterative computations, increasing the number of reachability steps m_r increases the number of required Minkowski additions, so that the number of generators grows. To keep computations efficient, this requires reduction techniques (see Sec. 2.8).

Linear systems have gained much attention recently. Most notably, significant improvements to set algorithm parameters, such as the number of reachability steps, automatically [112]. Furthermore, polynomial-time algorithms in the state-dimension for fully automated verification of LTI systems have been proposed in [114] and can even be extended to include safe and unsafe set specifications [115].

2.7.2 Nonlinear Systems using Conservative Linearization

Next, we look at a reachability algorithm for nonlinear systems from [9], i.e., we again have $\dot{x} = f_{\text{cl}}(x, w) = f_{\text{cl}}(z)$ with $z = \begin{bmatrix} x^T, & w^T \end{bmatrix}^T$. While the reachability computations are similar to Sec. 2.7.1, we linearize the nonlinear system in each reachability step for $t \in [k, k+1]\, r$ with $0 \leq k \leq m_r - 1$, which yields

$$\dot{x}(t) = c(z^*) + A(z^*)\, x(t) + B(z^*)\, w(t) + l(k), \tag{2.41}$$

with

$$\begin{aligned} A(z^*) &= \left.\frac{\partial f_{\text{cl}}(z)}{\partial x}\right|_{z=z^*}, \\ B(z^*) &= \left.\frac{\partial f_{\text{cl}}(z)}{\partial w}\right|_{z=z^*}, \\ c(z^*) &= f_{\text{cl}}(z^*) - A(z^*)\, x^* - B(z^*)\, w^*, \end{aligned}$$

where $l(k) \in \mathcal{L}(k)$ with $\mathcal{L}(k)$ being the Lagrange remainder to account for linearization errors, $z^* = \begin{bmatrix} x^{*T}, & w^{*T} \end{bmatrix}^T$ is the linearization point, and

$$\begin{aligned} \mathcal{L}(k) = \Bigg\{ l \in \mathbb{R}^{n_x} \Bigg| l_i = \frac{1}{2}(z - z^*)^T \frac{\partial^2 f_{\text{cl}i}(\chi(z, z^*, \alpha))}{\partial z^2}(z - z^*), \\ z \in \mathcal{R}([k, k+1]\, r) \times \mathcal{W}, \alpha \in [0, 1] \Bigg\}, \end{aligned} \tag{2.42}$$

with $\chi(z, z^*, \alpha) = z^* + \alpha(z - z^*)$ and $\mathcal{R}([k, k+1]\, r)$ denoting the outer approximation of the reachable set for $t \in [k, k+1]\, r$ that we wish to compute. The Lagrange remainder can be evaluated using interval arithmetic [9, Sec. V]. Further, since the Lagrange remainder is constant in (2.41), the superposition principle can be applied to compute the reachable set $\mathcal{R}_{\text{lin}}([k, k+1]\, r)$ of the corresponding LTI system for $l(k) = 0$ according to Sec. 2.7.1, and the reachable set $\mathcal{R}_{\text{err}}([k, k+1]\, r)$ due to the linearization error separately. However, the Lagrange remainder set computation in (2.42) requires the reachable set $\mathcal{R}([k, k+1]\, r)$, whose computation in turn requires $\mathcal{L}(k)$.

To break this circular dependency, we assume an initial candidate $\bar{\mathcal{L}}(k)$ (e.g. $\bar{\mathcal{L}}(k) = \{0\}$) for the Lagrange remainder and use the superposition principle to arrive at a reachable set candidate

$$\bar{\mathcal{R}}([k, k+1]\, r) = \mathcal{R}_{\text{lin}}([k, k+1]\, r) \oplus \bar{\mathcal{R}}_{\text{err}}([k, k+1]\, r),$$

where $\bar{\mathcal{R}}_{\text{err}}([k, k+1]\, r)$ is the reachable set due to the linearization errors captured in $\bar{\mathcal{L}}(k)$ (can be computed using (2.38)). Then, we evaluate (2.42) with $\bar{\mathcal{R}}([k, k+1]\, r)$ to obtain $\mathcal{L}(k)$ and check if $\mathcal{L}(k) \subseteq \bar{\mathcal{L}}(k)$. If this check fails, we enlarge $\bar{\mathcal{L}}(k)$ and try again; otherwise, $\mathcal{R}([k, k+1]\, r) = \bar{\mathcal{R}}([k, k+1]\, r)$ is an outer approximation of the reachable set.

2.7.3 Nonlinear Systems using Conservative Polynomialization

In general, reachable sets of nonlinear systems may be non-convex. Therefore, we provide a brief overview on abstracting a nonlinear system $\dot{x} = f(x, w)$ for reachability analysis using conservative polynomialization and polynomial zonotopes as proposed in [5]. To improve readability, we focus on the first reachability step, i.e., we assume $t \in [0, r]$ without loss of generality; the remaining steps for $t \in [r, t_\mathrm{f}]$ follow analogously.

We start by computing a Taylor expansion of $f_\mathrm{cl}(z)$ of order $\pi \in \mathbb{N}_+$ around the linearization point $z^* = \begin{bmatrix} x^{*T}, & w^{*T} \end{bmatrix}^T$ with $x^* \in \mathbb{R}^{n_x}$ and $w^* \in \mathbb{R}^{n_w}$ being linearization points, which yields

$$\dot{x} = A(z^*)\, x + v(z(t), z^*, w(t)) + l, \tag{2.43}$$

for $l \in \mathcal{L}$, and where the Jacobian of f is given by $A(z^*)$, the expression $v(z(t), z^*, w(t))$ contains all remaining terms of the Taylor expansion up to (and including) order π, and $\mathcal{L}$ is the Lagrange remainder evaluated over the rest term of the Taylor expansion (as in (2.42)). Using, e.g., a quadratic abstraction, the advantage of using polynomial zonotopes becomes clear since – in contrast to zonotopes – they are closed under quadratic maps (see Tab. 2.1). With $w^\Delta(t) = w(t) - \tilde{w}$ for $\tilde{w} \in \mathcal{W}$ chosen such that $0 \in \mathcal{W}^\Delta = \{-\tilde{w}\} \oplus \mathcal{W}$ and $z^\Delta(t) = z(t) - z(0)$, we split

$$v(z(t), z^*, w(t)) = v^{(0)}(z(0), z^*, \tilde{w}) + v^\Delta\left(z^\Delta(t), z(0), w^\Delta(t)\right),$$

into $v^{(0)}(z(0), z^*, \tilde{w})$ only dependent on known values $z(0)$, z^*, $\tilde{w}$ at $t = 0$, and $v^\Delta\left(z^\Delta(t), z(0), w^\Delta(t)\right)$ for $t \in [0, r]$, which depends on $z(t)$ and thus the reachable set $\mathcal{R}([0, r])$ which we aim to compute. Therefore, (2.43) can be rewritten as

$$\dot{x}(t) = A(z^*)\, x(t) + v^{(0)}(z(0), z^*, \tilde{w}) + \tilde{l}, \tag{2.44}$$

for $\tilde{l} \in \tilde{\mathcal{L}}$, where

$$\begin{aligned}
\tilde{\mathcal{L}} &= \mathcal{V}^\Delta([0, r]) \oplus \mathcal{L}, \\
\mathcal{V}^\Delta([0, r]) &= \Big\{ v^\Delta\left(z^\Delta, z(0), w^\Delta\right) \;\Big|\; z^\Delta \in \mathcal{R}^\Delta([0, r]) \times \mathcal{W}^\Delta,\ z(0) \in \mathcal{X}^{(0)} \times \{0\}, \\
&\qquad w^\Delta \in \mathcal{W}^\Delta \Big\}, \\
\mathcal{R}^\Delta([0, r]) &= \Big\{ \xi(t, x(0), w(\cdot)) - x(0) \;\Big|\; t \in [0, r], x(0) \in \mathcal{X}^{(0)}, w(t) \in \mathcal{W} \Big\},
\end{aligned}$$

where $\xi(t, x(0), w(\cdot))$ again denotes the state solution. Since (2.44) can be interpreted as an LTI system with uncertainty $\tilde{l} \in \tilde{\mathcal{L}}$ and additional constant $v^{(0)}(z(0), z^*, \tilde{w})$, its reachable set can be computed using the methods from Sec. 2.7.1 . In contrast to (2.41), however, (2.44) captures some nonlinearities in $v^{(0)}(z(0), z^*, \tilde{w})$.

Remark: By choosing the reachability step r smaller, the set $\tilde{\mathcal{L}}$ shrinks and thus the accuracy of the approximation is increased [5], albeit at the cost of additional computational effort.

2.8 Reduction Techniques

When calculating reachable sets (see Sec. 2.7), reduction techniques are essential to keep computations efficient. In this thesis, we mainly use zonotopes and polynomial zonotopes for set-based computations. In this section, we thus introduce the concept of reduction using a simple zonotope example. We refer the reader to [64], [59, Sec. 2.6] for details on zonotope order reduction techniques and to [61, Sec. 2.E], [59, Sec. 3.1.7] for polynomial zonotope order reduction techniques.

For a given set $\mathcal{S} \subset \mathbb{R}^n$ (zonotope or polynomial zonotope) with m generators, we denote by $\mathcal{S} \downarrow_o \supseteq \mathcal{S}$ the reduced set with a smaller number of generators $\tilde{m} \leq m$, where $o = \frac{\tilde{m}}{n}$ denotes the generator order of the new set. We demonstrate the concept of order reduction in the following example:

Example 2.7 (Zonotope Order Reduction)**.** Let

$$\mathcal{Z} = \left\langle \begin{bmatrix} -1 \\ 1 \end{bmatrix} \begin{bmatrix} -6, & -2, & 2, & 0, & 3, & -1 \\ 3, & 4, & 0, & -2, & 0, & -2 \end{bmatrix} \right\rangle_Z,$$

and we want to compute $\mathcal{Z} \downarrow_2$, i.e., the reduced zonotope should have $2 \cdot 2 = 4$ generators. Here, we choose the order reduction method from [40]: For a given zonotope $\mathcal{Z} = \langle c, G \rangle_Z$ of order $o = \frac{m}{n} \in \mathbb{N}$ with $c \in \mathbb{R}^n$ and $G \in \mathbb{R}^{n \times m}$, we sort all generators $G_{(:,i)}$ for $1 \leq i \leq m$ according to the metric $l(g) = \|g\|_1 - \|g\|_\infty$. To compute $\mathcal{Z} \downarrow_q$ with $q \leq o$, one then chooses the $n(q-1)$ smallest generators according to the metric l, and adds the resulting interval outer approximation back to the remaining generators.

In this example, we select the 4 smallest generators of $\mathcal{Z}$, which correspond to the last 4 generators, i.e., $\begin{bmatrix} 2, & 0, & 3, & -1 \\ 0, & -2, & 0, & -2 \end{bmatrix}$. The resulting box enclosure is then given by

$$\left\langle \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \operatorname{diag}\left(\left| \begin{bmatrix} 2 \\ 0 \end{bmatrix} \right| + \left| \begin{bmatrix} 0 \\ -2 \end{bmatrix} \right| + \left| \begin{bmatrix} 3 \\ 0 \end{bmatrix} \right| \left| \begin{bmatrix} -1 \\ -2 \end{bmatrix} \right| \right) \right\rangle_Z = \left\langle \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 6, & 0 \\ 0, & 4 \end{bmatrix} \right\rangle_Z.$$

As a result, we have

$$\mathcal{Z} \downarrow_2 = \left\langle \begin{bmatrix} -1 \\ 1 \end{bmatrix}, \begin{bmatrix} -6, & -2, & 6, & 0 \\ 3, & 4, & 0, & 4 \end{bmatrix} \right\rangle_Z.$$

Fig. 2.10 shows both the original and the reduced zonotope.

■

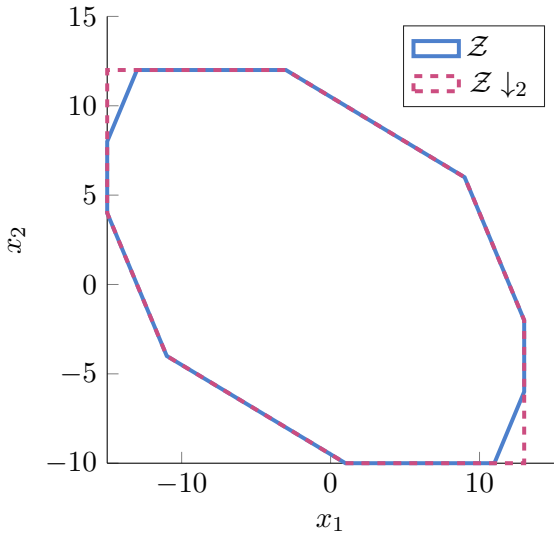


Figure 2.10: Simple reduction approach for a 2D zonotope.

3 The Abstracted Synthesis Problem

In this thesis, we synthesize controllers that steer a system from any initial state, chosen from a bounded set, close to a given target state while formally guaranteeing that no constraints are violated. We solve these so-called reach-avoid problems by combining optimization theory with reachability analysis. Here, optimization theory is generally used to formulate an optimization problem in order to synthesize the controller, and reachability analysis is used for formal verification. While subsequent chapters introduce various algorithms to synthesize a controller for these reach-avoid problems, they all share a common structure; this allows the introduction of the abstracted synthesis problem, which captures this common structure. Since practically all available efficient optimization solvers require differentiability, we derive the smooth but equivalent formulation of the abstracted problem. Furthermore, we show that the first-order necessary conditions for this smooth problem are always necessary.

This chapter is structured as follows: First, we describe the general synthesis problem we solve in this thesis in Sec. 3.1. In Sec. 3.2, we then discuss the abstracted problem, from which we derive an equivalent but smooth formulation and show that the first-order optimality conditions of this smooth problem are always necessary. Lastly, we briefly motivate the use of the 1-norm when synthesizing controllers using reachable sets in Sec. 3.3.

3.1 Problem Statement

All synthesis problems in this work are so-called reach-avoid problems.

Definition 3.1 (Reach-Avoid Problem)**.** Let the system $\dot{x}(t) = f(x(t), u(x(t), t), w(t))$ be given as in Def. 2.24, and let $x(0) \in \mathcal{X}^{(0)}$, $u(x(t), t) \in \mathcal{U}$, $x(t) \in \mathcal{X}$, $x(t_\mathrm{f}) \in \mathcal{X}_\mathrm{f}$, and $w(t) \in \mathcal{W}$, where $\mathcal{X}^{(0)} \subset \mathbb{R}^{n_x}$ is the set of bounded initial states, $\mathcal{U} \subset \mathbb{R}^{n_u}$ is the bounded set of admissible inputs, $\mathcal{X} \subseteq \mathbb{R}^{n_x}$ denotes the set of admissible states, $\mathcal{X}_\mathrm{f} \subseteq \mathbb{R}^{n_x}$ are the admissible states at final time $t_\mathrm{f} \in \mathbb{R}_+$, and $\mathcal{W} \subset \mathbb{R}^{n_w}$ is the set of bounded, uncontrollable disturbances. We define the problem of steering the system f from a given set of initial states $\mathcal{X}^{(0)}$ to a target state $x_\mathrm{f} \in \mathcal{X}_\mathrm{f}$ within time t_f – while respecting the input and state constraints – as a reach-avoid problem. ■

Fig. 1.1 visualizes this concept. In this thesis, we solve a specific instance of such a reach-avoid problem, given by

$$\hat{u}_{\text{ctrl}}\left(x\left(t\right),t\right) = \arg\min_{u_{\text{ctrl}}(x(t),t)} \max_{x(t)} \left\{ \left\| x\left(t_{\text{f}}\right) - x_{\text{f}} \right\|_1 + \zeta \int_0^{t_{\text{f}}} \left\| u_{\text{ctrl}}\left(x\left(\tau\right),\tau\right) \right\|_1 \mathrm{d}\tau \right\}, \tag{3.1a}$$

$$\text{s.t. } x\left(t\right) \in \mathcal{R}\left(t\right), \tag{3.1b}$$

$$\forall t \in [0, t_{\text{f}}] : u_{\text{ctrl}}\left(x\left(t\right),t\right) \in \mathcal{U}, \tag{3.1c}$$

$$\mathcal{R}\left([0,t_{\text{f}}]\right) \subseteq \mathcal{X}, \tag{3.1d}$$

$$\mathcal{R}\left(t_{\text{f}}\right) \subseteq \mathcal{X}_{\text{f}}, \tag{3.1e}$$

where $\mathcal{R}\left(t\right)$ denotes the (not necessarily exact) reachable set of states $x\left(t\right)$ at time t for the controller term $u_{\text{ctrl}}\left(x\left(t\right),t\right)$ under disturbances $w\left(t\right) \in \mathcal{W}$, and $\zeta \in \mathbb{R}_+$ is a weighting factor to penalize large inputs. No assumptions about the statistical nature of $\mathcal{W}$ are made, only that $\mathcal{W} = \langle c_{\mathcal{W}}, G_{\mathcal{W}} \rangle_Z$ is given as a zonotope. That said, we only consider zero-centered disturbance zonotopes $\mathcal{W} = \langle 0, G_{\mathcal{W}} \rangle_Z$ since any non-zero center $c_{\mathcal{W}}$ can be absorbed by the dynamics f. We focus on state, final state, and input constraints over the time interval $t \in [0, t_{\text{f}}]$; while an extension to consider different constraints over a finite number of time points or time intervals is conceptually straightforward, we omit further constraints for notational simplicity. Naturally, we not only want to satisfy constraints, but also steer the final state $x\left(t_{\text{f}}\right)$ as close as possible to a target state $x_{\text{f}} \in \mathbb{R}^{n_x}$ while ideally minimizing the expended control effort. This is reflected in (3.1a) by finding the controller that minimizes the maximum deviation of $x\left(t_{\text{f}}\right)$ from x_{f} while penalizing large inputs. For computational reasons, we choose the 1-norm (see Sec. 3.3).

Remark: The solution to (3.1) may be used for the computation of motion primitives to plan formally verified trajectories using a maneuver automaton [52]. As an illustrative example, imagine a take-over maneuver of an autonomous vehicle (see Fig. 3.1): To perform the take-over (see Fig. 3.1a), a suitable controller has to be found. Instead of trying find a suitable controller for an uncertain initial state (Fig. 3.1b), the trajectory is split in time into left and right turn primitives for $t \in \left[0, t^{(1)}\right]$ and $t \in \left[t^{(1)}, t_{\text{f}}\right]$ (see Fig. 3.1c); here, $x_{\text{f}}^{(1)}$ at $t = t^{(1)}$ is used as an intermediate target state. When only considering static constraints, a controller for a given motion primitive and its corresponding reachable set $\mathcal{R}\left(\cdot\right)$ can be computed offline for various different initial sets and target states. These motion primitives are then collected into a maneuver automaton which links appropriate motion primitives so that they can be concatenated online to form a wide range of trajectories while respecting dynamic constraints (see [98, Sec. 1.2] for a detailed example on a maneuver automaton). Linking motion primitives requires – among other factors – that the final reachable set is contained in the initial set of the next motion primitive (see Fig. 3.1c). While this thesis does not consider maneuver automata for trajectory planning, it motivates both the fact that we minimize

the worst-case deviation from the target state in (3.1a) and allow final state constraints in the form of $\mathcal{X}_\mathrm{f}$.

3.2 Abstraction

We now introduce the abstraction of the synthesis problem from (3.1). To enable efficient numerical optimization, we then derive its smooth reformulation in Sec. 3.2.1 and show in Sec. 3.2.2 that the KKT conditions are always necessary. Since these properties hold for the abstracted synthesis problem, they follow for all subsequent synthesis problems as they can be generalized to this abstraction.

Thus, we define the abstracted synthesis problem as

$$\hat{z}, \hat{s} = \arg\min_{z,s} J(z,s), \tag{3.2a}$$

$$\text{s.t. } g(z) \leq s, \tag{3.2b}$$

$$\underline{z} \leq z \leq \bar{z}, \tag{3.2c}$$

$$s \geq 0, \tag{3.2d}$$

with

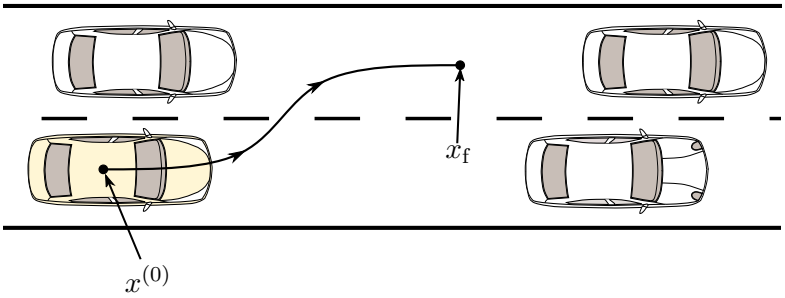
$$J(z,s) = l^T |h(z)| + 1_{n_y}^T \max\left(\underline{R}\rho(z), \bar{R}\rho(z)\right) + \sigma 1_{n_s}^T s, \tag{3.3}$$

$$g_m(z) = \max_{1\leq k\leq n_k} \left\{\max\left(F_{(m)}^{(k)}\left(Ah(z) + M|h(z)|\right)\right) + \max\left(G_{(m)}^{(k)}\rho(z)\right)\right\} - b_m, \tag{3.4}$$

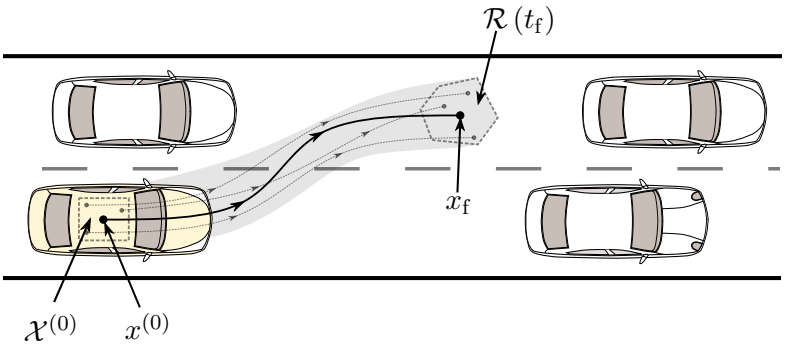
for $1 \leq m \leq n_s$, where $A \in \mathbb{R}^{n_r \times n_w}$, $M \in \mathbb{R}_{\geq 0}^{n_r \times n_w}$, $\underline{R} \in \mathbb{R}^{n_y \times g}$, $\bar{R} \in \mathbb{R}^{n_y \times g}$, $F_{(m)}^{(k)} \in \mathbb{R}^{q \times n_r}$, and $G_{(m)}^{(k)} \in \mathbb{R}^{q \times g}$ are matrices, $l \in \mathbb{R}_{\geq 0}^{n_w}$ is a vector, and $\sigma \in \mathbb{R}_+$ is a given scalar; further, $h(z) \in \mathbb{R}^{n_w}$ and $\rho(z) \in \mathbb{R}^g$ are sufficiently smooth functions, $z \in \mathbb{R}^{n_z}$ and $\hat{z} \in \mathbb{R}^{n_z}$ as well as $s \in \mathbb{R}^{n_s}$ and $\hat{s} \in \mathbb{R}^{n_s}$ are the optimization variables and their respective optimizers with $\underline{z} \in \mathbb{R}^{n_z}$ and $\bar{z} \in \mathbb{R}^{n_z}$ for $\underline{z} < \bar{z}$ denoting lower and upper bounds on z, and $\sigma \in \mathbb{R}_+$. If $\underline{z}_i = \bar{z}_i$ for $1 \leq i \leq n_z$, z_i can simply be removed from the optimization problem since then $\hat{z}_i = \underline{z}_i = \bar{z}_i$.

We briefly motivate the structure of (3.2) here; the intuition behind the exact choice for (3.2) will become clear in subsequent chapters (see, e.g., Sec. 6.3.5): Instead of maximizing over the unknown controller function directly as in (3.1), we fix the structure of the controller by parameterizing it in the controller parameters z. We use $h(z)$ to, e.g., collect generators and centers of reachable sets which are required for the optimization. Any additional differentiable functions not originating from reachable sets are collected in $\rho(z)$.

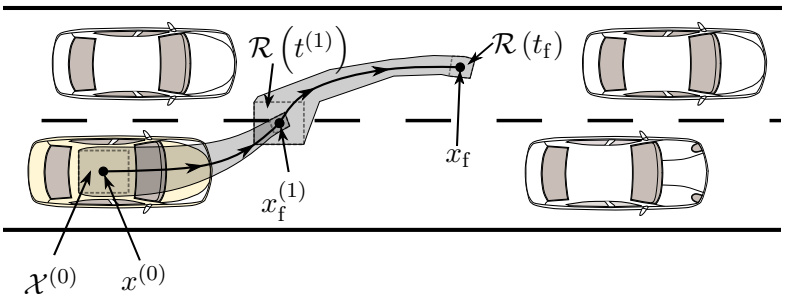
The abstracted objective function in (3.3) is obtained by outer approximating the maximization over the state in (3.1a). The constraints (3.1c) to (3.1e) of the original synthesis problem are collected in (3.2b). The maximizations in (3.2b) may be necessary if, e.g., generators from reachable sets for different time intervals are to be considered in a single constraint.



(a) Desired trajectory for single initial state



(b) Desired reachable set for a set of initial states



(c) Concatenation of appropriate motion primitives to solve take-over maneuver

Figure 3.1: Illustration of the concatenation of motion primitives online (adapted from [77]).

Since the original synthesis problem contained constraints on both input and states, the optimization problem in (3.1) and therefore also its abstraction in (3.2) may not have a feasible solution. Thus, we also use the slack variable $s \geq 0$ in (3.2d) to relax the constraints in (3.2b) so that we always find a feasible solution; by choosing σ large enough, any constraint violation is therefore penalized in (3.3) to prioritize no constraint violation over the minimization of the objective function for $s = 0$. Lastly, synthesis problems of subsequent chapters restrict the set of controller parameters z to a bounded set, which is reflected in (3.2c).

3.2.1 Smooth Reformulation

Many efficient numerical solvers require the optimization problem at hand to be sufficiently smooth, i.e., the objective function as well as all constraints need to be sufficiently often differentiable. Using insights from Sec. 2.3.6.1 and Sec. 2.3.6.2, the abstracted synthesis problem in (3.2) can be replaced by the smooth optimization problem

$$\min_{z,s,w,y,V,T,p} J_{\text{smooth}}(z,s,w,y), \tag{3.5a}$$

$$\text{s.t.} \quad \begin{bmatrix} +h(z) \\ -h(z) \end{bmatrix} \leq \begin{bmatrix} w \\ w \end{bmatrix}, \tag{3.5b}$$

$$\begin{bmatrix} \underline{R}\rho(z) \\ \bar{R}\rho(z) \end{bmatrix} \leq \begin{bmatrix} y \\ y \end{bmatrix}, \tag{3.5c}$$

$$F_{(m)}^{(k)}(Ah(z) + Mw) \leq 1_q V_{km}, \tag{3.5d}$$

$$G_{(m)}^{(k)}\rho(z) \leq 1_q T_{km}, \tag{3.5e}$$

$$V_{(:,m)} + T_{(:,m)} \leq 1_{n_k} p_m, \tag{3.5f}$$

$$p - b \leq s, \tag{3.5g}$$

$$\underline{z} \leq z \leq \bar{z}, \tag{3.5h}$$

$$s \geq 0, \tag{3.5i}$$

for $1 \leq m \leq n_s$ and $1 \leq k \leq n_k$ with

$$J_{\text{smooth}}(z,s,w,y) = l^T w + 1_{n_y}^T y + \sigma 1_{n_s}^T s, \tag{3.6}$$

where $w \in \mathbb{R}^{n_w}$, $y \in \mathbb{R}^{n_y}$, $V \in \mathbb{R}^{n_k \times n_s}$, $T \in \mathbb{R}^{n_k \times n_s}$ and $p \in \mathbb{R}^{n_s}$ are auxiliary variables to resolve absolute values and maximization expressions. The equivalence of (3.2) and (3.5) is stated in the following proposition.

Proposition 3.1 (Equivalence of the Smooth Reformulation)**.** *The abstracted synthesis problem in* (3.2) *and its reformulation in* (3.5) *attain the same global minimum. Additionally, the objective function* (3.6) *at any critical point* $\left(\hat{z}, \hat{s}, \hat{w}, \hat{y}, \hat{V}, \hat{T}, \hat{p}\right)$ *of the reformulated problem can written as*

$$\tilde{J}(\hat{z}) = J_{\text{smooth}}(\hat{z}, \hat{s}, \hat{w}, \hat{y}) = J(\hat{z}, \max(0, g(\hat{z}))).$$

Proof. See App. A.1.1. □

3.2.2 Regularity of Smooth Reformulation

Because the abstracted synthesis problem can be rewritten as a smooth optimization problem, efficient gradient-based solvers can be used for its solution. That said, many solvers implicitly solve optimization problems by finding an approximate solution to the corresponding first-order Karush-Kuhn-Tucker (KKT) conditions. However, the KKT conditions are only necessary conditions if the optimization problem at hand is regular at a critical point (see Sec. 2.3.1). As it turns out, this holds for the smooth reformulation.

Proposition 3.2 (Regularity of Abstracted Synthesis Problem)**.** *The KKT conditions are necessary for the smooth optimization problem in* (3.5) *at any critical point.*

Proof. See App. A.1.2. □

3.3 Different Norms for Optimization

As subsequent chapters will show, we frequently make use of norm functions inside the objective function of a controller synthesis problem. Thus, briefly motivate the choice of the 1-norm for optimization over reachable sets in this thesis.

In the following, we will make use of the well-known norm inequalities

$$\|x\|_\infty \leq \|x\|_2 \leq \|x\|_1 \leq \sqrt{n}\,\|x\|_2 \leq n\,\|x\|_\infty\,, \tag{3.7}$$

for $x \in \mathbb{R}^n$. As is clear from (3.7), the 2-norm is a better approximation of the infinity norm than the 1-norm, especially if $\exists i \in \{1,...,n\}\, \forall j \in \{1,...,n\} \setminus \{i\} : |x_i| \gg |x_j|$, since then $\|x\|_2 \approx |x_i| = \|x\|_\infty$. However, both the Euclidean and the infinity norm have a downside when optimizing sizes of sets, as the following example demonstrates.

Example 3.1. Let the zonotope $\mathcal{R}(p) = \left\langle 0, \begin{bmatrix} 0.001p + 1000, & -p+1 \end{bmatrix} \right\rangle_Z$ for $p \in [-1,1]$ be given. We are interested in finding the optimal $\hat{p}$ which minimizes the size of $\mathcal{R}(p)$. Since $\mathcal{R}(p)$ is zero-centered, one way to minimize the size of $\mathcal{R}(p)$ is to minimize the norm of its generators by solving

$$\hat{p}* = \arg\min_{p\in[-1,1]} \left\| \begin{bmatrix} 0.001p + 1000 \\ -p+1 \end{bmatrix} \right\|,$$

i.e., we shrink the size of $\mathcal{R}(p)$ by minimizing the magnitude of each generator. Using different norms, we have

$$\left\| \begin{bmatrix} 0.001p + 1000 \\ -p+1 \end{bmatrix} \right\|_\infty = \max\left(|0.001p + 1000|\,, |-p+1|\right) = |0.001p + 1000|\,, \tag{3.8}$$

$$\left\| \begin{bmatrix} 0.001p + 1000 \\ -p+1 \end{bmatrix} \right\|_2 = \sqrt{(0.001p + 1000)^2 + (-p+1)^2} \approx |0.001p + 1000|\,, \tag{3.9}$$

$$\left\| \begin{bmatrix} 0.001p + 1000 \\ -p+1 \end{bmatrix} \right\|_1 = |0.001p + 1000| + |-p+1|\,. \tag{3.10}$$

Since the first generator is much bigger than the second generator, a subsequent minimization over p will effectively keep the objective value for (3.8) and (3.9) constant since $\min_{p\in[-1,1]} |0.001p + 1000| = 999.999 \approx 1000$. As a result, $\hat{p} = -1$, which does not really affect the size of the set in the first dimension, and even increases the size in the second dimension.

The $\|\cdot\|_1$ norm in (3.10), however, values reduction in all dimensions equally, and thus reduces the size in the second dimension significantly ($\hat{p} = 1$) while only marginally increasing the set size in the first dimension. ■

Since we frequently optimize over (reachable) sets in this thesis, we choose 1-norms for their optimization as motivated by Ex. 3.1.

3.4 Summary

In this chapter, we introduced the abstracted synthesis problem as a generalization of many synthesis problems of subsequent chapters. By deriving an equivalent but smooth formulation of this abstracted problem and proving that its Karush-Kuhn-Tucker (KKT) first-order optimality conditions are always necessary, we showed that any problem that can be written in the form of the abstracted problem can be solved by off-the-shelf optimization solvers.

We reviewed reach-avoid problems as the category of synthesis problems we solve in this thesis. In reach-avoid problems, we try to steer a given system for a set of initial states to a target state within a specified time horizon while ensuring both input and state constraints. This is reflected in the general problem formulation, where – for the controller to be synthesized – we penalize the distance of the resulting center of the reachable set to the target and furthermore minimize the size of the final reachable set while also minimizing the accumulated control effort and satisfying input and state constraints.

Since the abstracted problem is not continuously differentiable, we described a smooth formulation to allow for efficient optimization. Because many optimization solvers solve the KKT optimality conditions instead of the optimization problem, we further showed that these optimality conditions are necessary for any critical point of the smooth abstracted problem. As optimization problems for the controller synthesis in subsequent chapters can be written in the form of the abstracted problem, this ensures that these problems can be solved by a variety of off-the-shelf optimization solvers.

All synthesis problems described in this thesis use 1-norms to bound the size of the respective (reachable) sets. As a result, we motivated the choice of the 1-norm by comparing the impact of different norms on the optimization of reachable set sizes.

4 Set Conversions and Set Containment

Depending on the application, the choice of set representation can make a significant difference as different set representations are closed under different set operations (see Tab. 2.1) or have varying computational complexities. Thus, set conversions are essential when converting the original intractable synthesis problem into a tractable optimization problem: Examples include the conversion of a non-convex reachable set given by a polynomial zonotope to a simple zonotope to efficiently bound the size of the reachable set, or the conversion from a zonotope to an ellipsoid to avoid an increasing number of generators when many Minkowski sums are required (see Sec. 6.3). Similarly, efficiently checking the containment of these sets within each other is essential, e.g., when using a given reachable set to check state constraints.

Thus, we first introduce all necessary set conversion methods in Sec. 4.1. We then conclude the chapter with a short discussion of relevant set containment methods in Sec. 4.2.

4.1 Set Conversions

Since not all sets are closed under all operations, conversions between set representations can become necessary. In Sec. 4.1.1 and Sec. 4.1.2, we first introduce inner and outer approximations of zonotopes by ellipsoids and vice versa, where we refer the interested reader to [34] for their implementation in CORA. While we do not need all zonotope-ellipsoid set conversions introduced here in subsequent chapters, we include them for the sake of completeness. Then, we describe an outer approximation of an H-polytope by a parallelotope in Sec. 4.1.3, and introduce the outer approximation of a polynomial zonotope by a zonotope from [61] in Sec. 4.1.4.

To keep the presentation simple, we assume for the remainder of this section that all sets are non-degenerate, i.e., they have non-zero volumes and refer the reader to [34, Sec. 4.1.3] for a possible approach to handle non-degenerate sets.

4.1.1 Zonotope To Ellipsoid

In this section, we describe approximations of a zonotope by an ellipsoid from our work in [36]: In Sec. 4.1.1.1, we derive an outer approximation of a given zonotope by an ellipsoid, and then discuss an inner approximation of a zonotope by an ellipsoid in Sec. 4.1.1.2. First, we compute a template ellipsoid to describe its shape and then scale this template appropriately using the maximum and minimum zonotope norms, which are defined next.

Definition 4.1 (Squared Maximum Zonotope Norm)**.** The squared maximum norm from any point on the boundary of a zonotope $\langle c, G\rangle_Z$ to its center c is defined by

$$\lceil \langle c, G\rangle_Z \rceil^2 = \max_{\|\beta\|_\infty \leq 1} \beta^T G^T G \beta. \tag{4.1}$$

■

Definition 4.2 (Squared Minimum Zonotope Norm)**.** The squared minimum norm from any point on the boundary of a zonotope $\langle c, G\rangle_Z$ to its center c is defined by the radius of the largest hyper-sphere $\langle c, \lfloor \langle c, G\rangle_Z \rfloor^2 I_n\rangle_E$ still contained in $\langle c, G\rangle_Z$, where

$$\lfloor \langle c, G\rangle_Z \rfloor^2 = \max_{r \geq 0} r, \text{ s.t. } \rho_{\langle 0, rI_n\rangle_E}(A) \leq b, \tag{4.2}$$

and where $\langle A, b\rangle_H$ with $A \in \mathbb{R}^{o\times n}$ and $b \in \mathbb{R}^o$ is the half-space representation of $\langle 0, G\rangle_Z$. ■

When the halfspace representation for the zonotope is given, the optimization problem in Def. 4.2 can be avoided.

Proposition 4.1 (Squared Minimum Zonotope Norm)**.** *Let the zonotope $\mathcal{Z} \subset \mathbb{R}^n$ and its half-space representation $\langle A, b\rangle_H$ with $A \in \mathbb{R}^{o\times n}$ and $b \in \mathbb{R}^o$ be given, where we assume without loss of generality that $\left\|A_{(i,:)}\right\|_2 > 0$ for all $1 \leq i \leq o$. Then*

$$\lfloor \mathcal{Z} \rfloor^2 = \min_{1\leq i\leq o} \frac{b_i^2}{\left\|A_{(i,:)}^T\right\|_2^2}.$$

Proof. Since $\rho_{\langle 0, rI_n\rangle_E}\left(A_{(i,:)}^T\right) = \sqrt{A_{(i,:)} r I_n A_{(i,:)}^T} = \sqrt{r}\left\|A_{(i,:)}\right\|_2$ for $1 \leq i \leq o$, it follows from (4.2) that $\sqrt{r}\left\|A_{(i,:)}\right\|_2 \leq b_i \iff r \leq \frac{b_i^2}{\|A_{(i,:)}\|_2^2}$, which concludes the proof. □

4.1.1.1 Outer Approximation

If we want to find an ellipsoid $\langle e_{\min}, E_{\min}\rangle_E$ enclosing a zonotope $\langle c, G\rangle_Z \subset \mathbb{R}^n$ with minimum volume (also called the Löwner-John ellipsoid), it is clear that $e_{\min} = c$ must hold since both zonotopes and ellipsoids are centrally symmetric. The optimal shape matrix $E_{\min}$ is then found by solving the SDP [14, Sec. 8.4.1]

$$E_{\min} = \arg\min_{E \succ 0} -\log\det E, \tag{4.3a}$$

$$\text{s.t. } \left(v^{(i)} - c\right)^T E^{-1}\left(v^{(i)} - c\right) \leq 1,\ 1 \leq i \leq m, \tag{4.3b}$$

with $v^{(i)} \in \mathbb{R}^n$ for $1 \leq i \leq m$ being the vertices of $\langle c, G\rangle_Z$. While even more scalable approaches exist to compute this minimum-volume enclosing ellipsoid (MVEE) [106], they still require us to have the vertex representation of $\langle c, G\rangle_Z$ available, which is intractable to compute for higher dimensions [32, Theorem 2.5]. Therefore, we subsequently derive a sub-optimal solution that does not require the vertex representation of a given zonotope.

To that end, we first describe an initial guess for the relative proportions of the sub-optimal shape matrix $\bar{E} \in \mathbb{S}_{++}^{n\times n}$, and then use the idea of the maximum zonotope norm (see Def. 4.1) to find a scaling factor $\hat{r} \in \mathbb{R}_+$ to ensure $\langle c, G\rangle_Z \subseteq \left\langle c, \hat{r}\bar{E}\right\rangle_E$.

Shape matrix: In [19], a variant of Goffin's algorithm for zonotopes is used to compute an approximation to the MVEE. While their resulting approximation is somewhat conservative, they propose the following initial guess [19, Sec. 4.1]: The zonotope $\langle 0, G\rangle_Z$ with $G \in \mathbb{R}^{n\times m}$ can be interpreted as the projection of the m-dimensional hypercube under G, where the smallest hyper-ball enclosing this hypercube is given by $\langle 0, mI_m\rangle_E$. Applying the linear map G to both hypercube and hyper-ball yields the original zonotope and the enclosing ellipsoid $G\langle 0, mI_n\rangle_E = \left\langle 0, mGG^T\right\rangle_E$, where the shape matrix we are looking for is then given by $\bar{E} = GG^T$.

This choice of $\bar{E}$ can also be motivated by showing that the covariance matrix over all vertex candidates of $\langle 0, G\rangle_Z$ is identical to $\bar{E}$: Let $B \in \{-1, 1\}^{m\times 2^{m-1}}$, where $B_{i1} = 1$ and we alternate the sign of the i-th row $B_{(i,:)}$ every 2^{i-1} entries for $1 \leq i \leq m$. For example, this results in

$$B = \begin{bmatrix} 1, & -1, & 1, & -1 \\ 1, & 1, & -1, & -1 \\ 1, & 1, & 1, & 1 \end{bmatrix},$$

for $m = 3$. As a result, $L = G\begin{bmatrix} B, & -B \end{bmatrix} \in \mathbb{R}^{n\times 2^m}$ contains all 2^m possible vertex candidates of $\langle 0, G\rangle_Z$. Since the column-wise mean $\sum_{k=1}^{2^m} L_{(:,k)} = 0_n$ by construction, we can use principle component analysis (PCA) to construct the covariance matrix of all 2^m samples collected in L, which is given by[1]

$$\begin{aligned} Q_L &= \frac{1}{2^m} LL^T \\ &= \frac{2}{2^m} GBB^T G^T \\ &= GG^T, \end{aligned}$$

since

$$B_{(i,:)} \left[B^T\right]_{(:,j)} = \begin{cases} 2^{m-1}, & i = j, \\ 0, & \text{otherwise} \end{cases}, \tag{4.4}$$

for $1 \leq i \leq m$ and $1 \leq j \leq m$ due to the construction of B. The ellipsoid $\langle 0, Q_L\rangle_E = \left\langle 0, GG^T\right\rangle_E$ induced by the covariance matrix $Q_L \in \mathbb{S}_+^{n\times n}$ then describes the principal components of all samples in L. Fig. 4.1 shows the distribution of these 2^m samples for a random two-dimensional zonotope $\langle c, G\rangle_Z$ with ten generators, where the covariance matrix is scaled so that $\langle c, G\rangle_Z$ is tightly enclosed (see Th. 4.1).

Radius: With $\bar{E}$ given, it only remains to find an appropriate scaling factor $\hat{r} > 0$ to ensure that $\langle c, G\rangle_Z \subseteq \left\langle c, \hat{r}\bar{E}\right\rangle_E$. As it turns out, the maximum zonotope norm will be useful in finding such a scaling factor. The maximum squared zonotope norm from Def. 4.1 can then be rewritten as a mixed-integer quadratic program (MIQP).

[1] for details, see Sec. IV.C in `https://arxiv.org/abs/1404.1100`

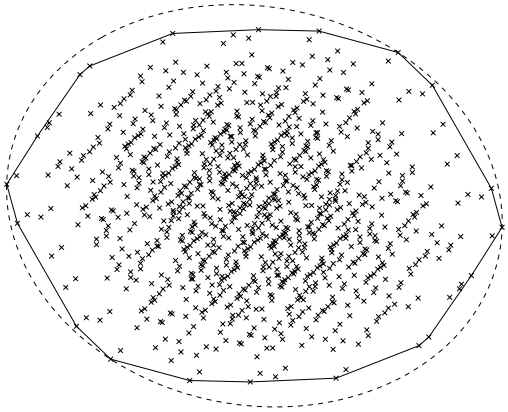


Figure 4.1: All candidate vertex points (black x) of a random two-dimensional zonotope with ten generators (solid black) and the corresponding covariance matrix (dashed black), scaled using Th. 4.1.

Proposition 4.2. *Given the zonotope* $\langle \cdot, G \rangle_Z$ *with generator matrix* $G \in \mathbb{R}^{n \times m}$*, we have*

$$\lceil \langle \cdot, G \rangle_Z \rceil^2 = - \min_{\beta \in \{-1,1\}^m} \beta^T M \beta + \lambda_{\max} m, \tag{4.5}$$

where $M = -\left(G^T G - \lambda_{\max} I\right) \succeq 0$ *and* $\lambda_{\max} = \max\left(\lambda\left(G^T G\right)\right)$.

Proof. From $\beta^T \beta = m$, it follows that $\beta^T G^T G \beta = \beta^T \left(G^T G - \lambda_{\max} I\right) \beta + \lambda_{\max} m$. Because $G^T G \succeq 0$, using $M \succeq 0$ together with Def. 4.1 provides the desired result. □

While (4.5) in Prop. 4.2 is a MIQP and thus has a worst-case complexity which is polynomial in the number of generators [86], it is often quickly solvable for smaller values of m: Finding $\lceil \langle c, G \rangle_Z \rceil^2$ to scale the covariance matrix in Fig. 4.1 took around 0.05 s. That said, computing (4.5) exactly quickly becomes intractable for larger m. Interestingly, [31] showed that solving (4.5) was polynomial in the number of generators of $G \in \mathbb{R}^{n \times m}$ if $\operatorname{rank} G = d$ was fixed. They argue that (4.5) can be solved by enumerating all extreme points of $\langle c, G \rangle_Z$, and that the number of these extreme points m_v can be bounded from above by $m_v \leq \sum_{k=0}^{d-1} 2\binom{m-1}{k}$ [31, Th. 3.1]. By using the known upper bound for the binomial coefficient $\binom{m-1}{k} \leq \left(\frac{e(m-1)}{k}\right)^k$ [24, Eq. (C.3)], it follows for $m \geq 1$ that

$$m_v \leq \sum_{k=0}^{d-1} 2 \binom{m-1}{k} \leq \sum_{k=0}^{d-1} 2 \left(\frac{e\,(m-1)}{k}\right)^k,$$

and therefore m_v is of complexity $O\left(m^{d-1}\right)$ in m if d is fixed. However, generally $d = n$, and therefore this approach still scales exponentially with the number of dimensions.

Hence, we now propose an upper bound on the maximum zonotope norm, which can similarly be found in [75]. Since the proof was omitted in [75], we provide it here for completeness.

Lemma 4.1 (Bound on Maximum Zonotope Norm [75, Eq. (3)], adapted)**.** *Let the zonotope* $\langle \cdot, G \rangle_Z$ *with* $G \in \mathbb{R}^{n \times m}$ *and* $\operatorname{rank}(G) = n$ *be given. Then the SDP*

$$\hat{r} = \min_{\lambda \geq 0} 1_m^T \lambda, \tag{4.6a}$$

$$s.t.\ \operatorname{diag}(\lambda) - G^T G \succeq 0, \tag{4.6b}$$

with $\lambda \in \mathbb{R}^m$ *provides an upper bound* $\hat{r} \geq \lceil \langle \cdot, G \rangle_Z \rceil^2$.

Proof. The exact squared maximum zonotope norm from Def. 4.1 can be equivalently computed by solving

$$\max_{\beta_i^2 \leq 1,\ 1 \leq i \leq m} \beta^T G^T G \beta,$$

since $G^T G \succeq 0$ and thus the maximum is attained for $\beta_i^2 = 1$. Its Lagrangian dual function is given by

$$g(\lambda) = \inf_{\beta \in \mathbb{R}^m} \left(\beta^T G^T G \beta + \sum_{i=1}^{m} \lambda_i \left(\beta_i^2 - 1 \right) \right) = \inf_{\beta \in \mathbb{R}^m} \beta^T \left(\operatorname{diag}(\lambda) - G^T G \right) \beta - 1_m^T \lambda,$$

where $\lambda \in \mathbb{R}^m$; the subsequent maximization $\sup_{\lambda \geq 0} g(\lambda)$ to form the Lagrangian dual problem then results in (4.6) since

$$g(\lambda) = \begin{cases} -1_m^T \lambda, & \operatorname{diag}(\lambda) - G^T G \succeq 0, \\ -\infty, & \text{otherwise} \end{cases},$$

which concludes the proof. □

Naturally, there generally is a duality gap between (4.1) and (4.6) since the former is not a convex optimization problem. Rewriting (4.1) with $B = \beta \beta^T$, we obtain

$$\max_B \operatorname{trace}\left(G^T G B \right), \tag{4.7a}$$

$$\text{s.t. } B \succeq 0, \tag{4.7b}$$

$$\operatorname{trace}\left(\operatorname{diag}\left(e_(m)^{(i)} \right) B \right) = 1,\ 1 \leq i \leq m, \tag{4.7c}$$

$$\operatorname{rank}(B) = 1. \tag{4.7d}$$

If we drop the rank constraint (4.7d) and formulate its Lagrangian dual problem (see Def. 2.11), we arrive at (4.6). Since there is a strictly feasible B (e.g., $B = I_m$), strong duality holds according to SC (see Prop. 2.2). As a result, the gap between (4.1) and (4.6) can be attributed to the "loss" of (4.7d) in (4.6). A result for tightening this duality gap is provided in [75, 48]. We are now ready to state the main result of this section.

Theorem 4.1 (Enclosing Ellipsoid)**.** *An n-dimensional enclosing ellipsoid of the non-degenerate zonotope* $\mathcal{Z} = \langle c, G \rangle_Z$ *is given by*

$$\hat{\mathcal{E}}(\mathcal{Z}) = \begin{cases} \left\langle c, \hat{r} \bar{E} \right\rangle_E, & m > n, \\ \left\langle c, n \bar{E} \right\rangle_E, & m = n \end{cases},$$

where $\hat{r}$ is an upper bound on the maximum squared zonotope norm $\lceil \bar{E}^{-\frac{1}{2}} \langle c, G \rangle_Z \rceil^2$ using, e.g., Lem. 4.1, and $\bar{E} = GG^T$.

Proof. Let $m > n$. Define $T = \bar{E}^{-\frac{1}{2}}$ and compute $T \left\langle c, \bar{E} \right\rangle_E = \langle Tc, I_n \rangle_E$ and $T \langle c, G \rangle_Z$, where T exists since $\mathcal{Z}$ is non-degenerate. Let $\hat{r} \geq \lceil T \langle c, G \rangle_Z \rceil^2$ denote an upper bound on the squared maximum zonotope norm of $T \langle c, G \rangle_Z$. Then $\max_{\tilde{x} \in T\langle c,G \rangle_Z} \|\tilde{x} - Tc\|_2^2 \leq \hat{r}$ by definition, $T \langle c, G \rangle_Z \subseteq \langle Tc, \hat{r} I_n \rangle_E$, and applying T^{-1} to both $T \langle c, G \rangle_Z$ and $\langle Tc, \hat{r} I_n \rangle_E$ yields the desired result.

Now let $m = n$. The MVEE enclosing an n-dimensional hypercube is the hypersphere with radius $\sqrt{n}$. Applying G, the minimum-volume enclosing ellipsoid of $\langle c, G \rangle_Z$ is $\left\langle c, nGG^T \right\rangle_E$. □

4.1.1.2 Inner Approximation

Finding the maximum-volume inscribed ellipsoid (MVIE) $\langle e_{\max}, E_{\max} \rangle_E$ for a zonotope $\langle c, G \rangle_Z \subset \mathbb{R}$ with center $c \in \mathbb{R}^n$ and generator matrix $G \in \mathbb{R}^{n \times m}$, where $\langle A, b \rangle_H$ for $A \in \mathbb{R}^{q \times n}$ and $b \in \mathbb{R}^q$ denotes its half-space representation, can be formulated as the SDP [14, Sec. 8.4.2]

$$\hat{B} = \arg\min_{B \succ 0} \; -\log\det B, \tag{4.8a}$$

$$\text{s.t.} \;\; \left\| B A_{(i,:)}^T \right\|_2 + A_{(i,:)} c \leq b_i, \;\; 1 \leq i \leq q, \tag{4.8b}$$

where $E_{\max} = \hat{B}\hat{B}$ and again $e_{\max} = c$ due to the central symmetry of ellipsoids and zonotopes. That said, the half-space representation of a zonotope is generally not available and existing algorithms produce a number of half-spaces exponential in the number of dimensions for a given zonotope [3, Th. 2].

In this section, we thus compute an approximation to $\langle e_{\max}, E_{\max} \rangle_E$, analogously to Sec. 4.1.1.1, but now use a lower bound on the minimum zonotope norm from Def. 4.2 to scale the shape matrix appropriately since its exact value requires the half-space representation of the zonotope (see Prop. 4.1). We again use $\bar{E} = GG^T$, and propose the following lemma to find a lower bound on the minimum zonotope norm for scaling.

Lemma 4.2 (Lower Bound on Mininum Zonotope Norm)**.** *Given is a zonotope $\mathcal{Z} = \langle c, G \rangle_Z \subset \mathbb{R}^n$ with center $c \in \mathbb{R}^n$ and generator matrix $G \in \mathbb{R}^{n \times m}$. Using the test from [19, Sec. 4.4], we can find a lower bound $\check{r} \in \mathbb{R}_+$ on $\lfloor \mathcal{Z} \rfloor^2$ as*

$$\check{r} = \frac{\hat{\nu}^2}{n} \leq \lfloor \mathcal{Z} \rfloor^2, \tag{4.9}$$

where $\hat{\nu}$ is optimizer of the LP

$$\hat{\nu} = \arg\max_{\nu, B} \; \nu, \tag{4.10a}$$

$$s.t. \;\; \nu I_n = GB, \tag{4.10b}$$

$$B \in [-1, 1]^{m \times n}. \tag{4.10c}$$

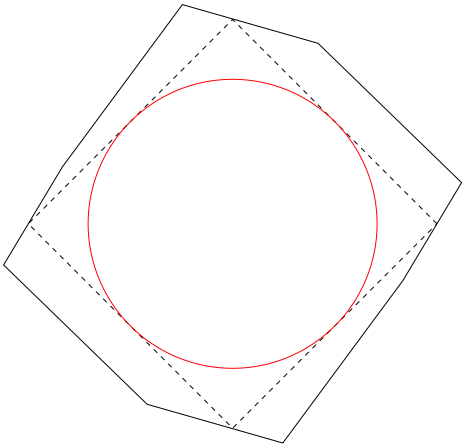


Figure 4.2: Lower bound computation of the minimum zonotope norm as described in Lem. 4.2: For a given n-dimensional zonotope (black), we compute its boundary points along all $2n$ unit directions, which define a convex polytope (dashed black). Using the MVEE of this polytope, we apply the Löwner-John property of symmetric sets (see (4.11)) to obtain an ellipsoidal inner approximation (red).

Proof. By solving (4.10) for ν and due to the symmetry of $\mathcal{Z}$, we have

$$\mathcal{V} = \operatorname{convh}\left(\pm\hat{\nu} e_{(n)}^{(1)}, \ldots, \pm\hat{\nu} e_{(n)}^{(n)}\right) \subseteq \mathcal{Z}.$$

To inscribe a hyper-ball into $\mathcal{V}$ and thus find a lower bound on the squared minimum zonotope norm by its radius, we can use the Löwner-John ellipsoidal approximation for symmetric sets [14, Sec. 8.4.1]

$$\frac{1}{\sqrt{n}} \left\langle 0, \hat{\nu}^2 I_n \right\rangle_E \subseteq \mathcal{V} \subseteq \left\langle 0, \hat{\nu}^2 I_n \right\rangle_E, \tag{4.11}$$

since $\mathcal{V} = \{-v \mid v \in \mathcal{V}\}$ is symmetric by construction, where $\langle 0, \hat{\nu}^2 I_n \rangle_E$ is the MVEE (or Löwner-John ellipsoid) of $\mathcal{V}$. Applying the definition of the linear map for ellipsoids (see Sec. 2.6.3) to (4.11), it follows that $\left\langle 0, \frac{\hat{\nu}^2}{n} I_n \right\rangle_E \subseteq \mathcal{V} \subseteq \mathcal{Z}$, which concludes the proof. □

The application of Lem. 4.2 to a 2D zonotope is visualized in Fig. 4.2. Since Lem. 4.2 essentially only requires the solution of an LP, for which algorithms with complexity polynomial in the number of optimization variables exist [22], a lower bound on the minimum zonotope norm can be efficiently computed. That said, since we cannot efficiently test whether a given hyper-ball is contained in a zonotope, we need to resort to checking whether its enclosing hyper-cube is contained, which naturally adds conservatism. We can now inscribe an ellipsoid into a zonotope.

Theorem 4.2 (Inscribed Ellipsoid)**.** *An n-dimensional ellipsoid inscribed into the non-degenerate zonotope $\mathcal{Z} = \langle c, G \rangle_Z$ with center $c \in \mathbb{R}^n$ and generator matrix $G \in \mathbb{R}^{n \times m}$ is given by*

$$\check{\mathcal{E}}(\mathcal{Z}) = \begin{cases} \left\langle c, \check{r}\bar{E} \right\rangle_E, & m > n, \\ \left\langle c, \bar{E} \right\rangle_E, & m = n \end{cases},$$

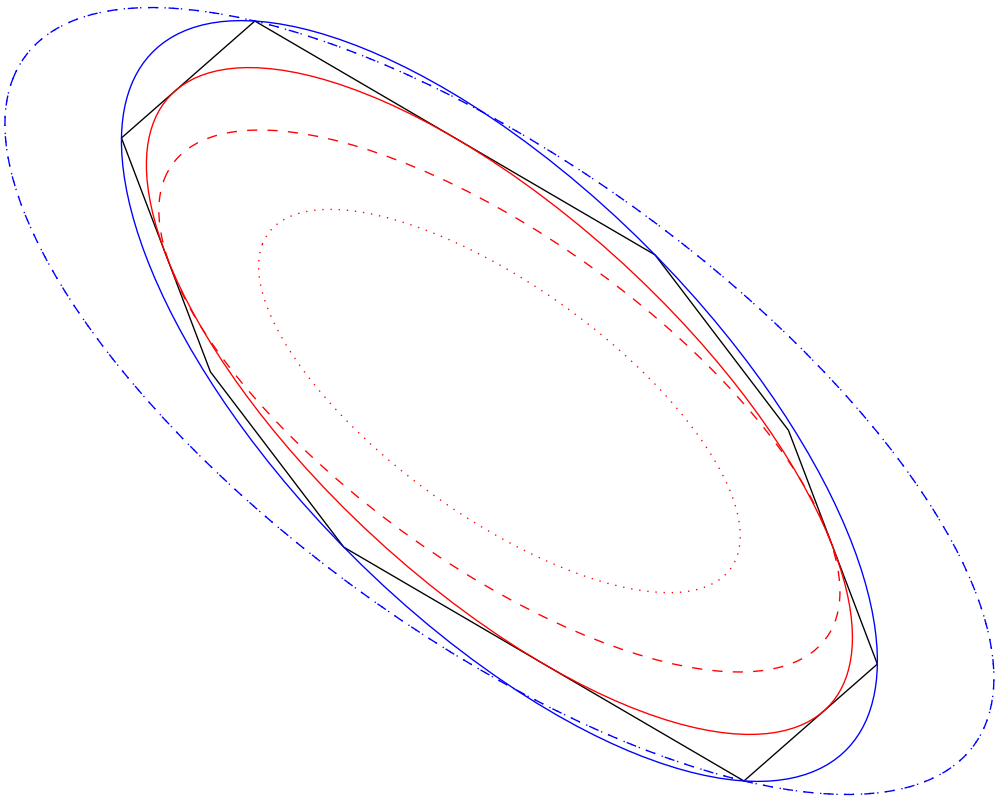


Figure 4.3: Different enclosing and inscribed ellipsoids for a given two-dimensional zonotope. Shown are the MVEE and MVIE (solid blue and red), the approximated enclosing and inscribed ellipsoid with exact zonotope norm (dashed blue and red), and the approximated enclosing and inscribed ellipsoids with bounded zonotope norm (dotted blue and red). For this example, the enclosing ellipsoid with exact and bounded zonotope norm coincide.

where $\check{r} \leq \lfloor \bar{E}^{-\frac{1}{2}} \langle c, G \rangle_Z \rfloor^2$ *can, e.g., be computed using Lem. 4.2, and* $\bar{E} = GG^T$.

Proof. Let $T = \bar{E}^{-\frac{1}{2}}$ and further let $\check{r} \leq \lfloor T \langle c, G \rangle_Z \rfloor^2$ be a lower bound on the respective squared minimum zonotope norm. The hyper-sphere $\langle Tc, \check{r} I_n \rangle_E$ is still contained in $T \langle c, G \rangle_Z$ by definition of the minimum zonotope norm and thus applying the inverse transform T^{-1} yields the desired result.

Now let $m = n$. The MVIE inscribed into an n-dimensional hypercube is the hyper-sphere with radius 1. Applying G, the minimum-volume enclosing ellipsoid of $\langle c, G \rangle_Z$ is $\left\langle c, \bar{E} \right\rangle_E$. □

Fig. 4.3 visualizes different enclosing as well as inscribed ellipsoids for a given two-dimensional zonotope.

4.1.2 Ellipsoid to Zonotope

In this section, we now describe approximations of a given ellipsoid by a zonotope for a given number of generators from our work in [36]: In Sec. 4.1.2.1, we derive an outer approximation of an ellipsoid with a zonotope, and then describe an inner approximation of an ellipsoid with a zonotope in Sec. 4.1.2.2. Similarly to Sec. 4.1.1, we compute these approximations by finding a template zonotope approximating the general shape of the ellipsoid, and then using the maximum and minimum zonotope norms (Defs. 4.1 and 4.2) to scale them appropriately.

To that end, we first describe how a unit hyper-ball can be approximated with a zonotope and then apply this approximation to derive inner and outer approximations for general ellipsoids. Thus, we momentarily assume that the given ellipsoid is a hyper-ball, as we can always apply the appropriate linear maps to achieve that. Intuitively, a zonotope with $m \in \mathbb{N}_+$ generators, sampled equally distant from the surface of a hyper-sphere, might approximate a hyper-sphere increasingly closely (in the Hausdorff distance). Since computing equally distant points on a hyper-sphere is impossible for arbitrary m [117], we first approximate this distribution by sampling m uniformly distributed points on the surface of a hyper-ball [83]: Let the random variable Y be normally distributed, and denote with $y^{(i)} \in \mathbb{R}^n$ for $1 \leq i \leq m$ any m independent realizations of Y. Defining $x^{(i)} = \frac{y^{(i)}}{\left\| y^{(i)} \right\|_2}$, the vectors $x^{(i)}$ will be uniformly distributed on the surface of the unit hyper-ball for large enough m. This approach does not necessarily result in a uniform distribution for small m in practice. In [71], it is claimed that a polynomial-time deterministic sampling approach is found that solves this distribution problem approximately. Fig. 4.4 visualizes the two variants of approximating a hyper-ball with a zonotope. However, its derivation seems to have a flaw[2]; thus, we use the probabilistic sampling method to obtain an approximately uniform distribution on the surface of the hyper-sphere, since it was proven that a zonotope constructed from these sampled surface points approximates a hyper-sphere increasingly closer for an increasing number of samples [73, 3., Theorem].

4.1.2.1 Outer Approximation

With the shape of the ellipsoid given by samples of the hyper-sphere as described previously, an outer approximation of an ellipsoid by a zonotope is described next.

[2] I thank Adrian Kulmberg for bringing the following issue to my attention: From [71, Eq. (3.5.8)] and [71, Eq. (3.5.11)], it follows that $\rho_H\left(\mathcal{N}_0\left(\frac{1}{2}\right)\right) = 1 + \frac{1}{2^{\frac{1}{2}}-1}$, which is undefined and thus causes a problem in [71, Lem. 3.5.23], which in turn causes a problem in [71, Th. 5.4.1]. Thus, this invalidates [36, Lem. 4] from our original paper since we used [71, Th. 5.4.1]. However, this issue has no consequence on the remainder of our paper since we can simply use the probabilistic sampling method to obtain an approximation of the hyper-sphere by a zonotope as described in [36, Sec. V].

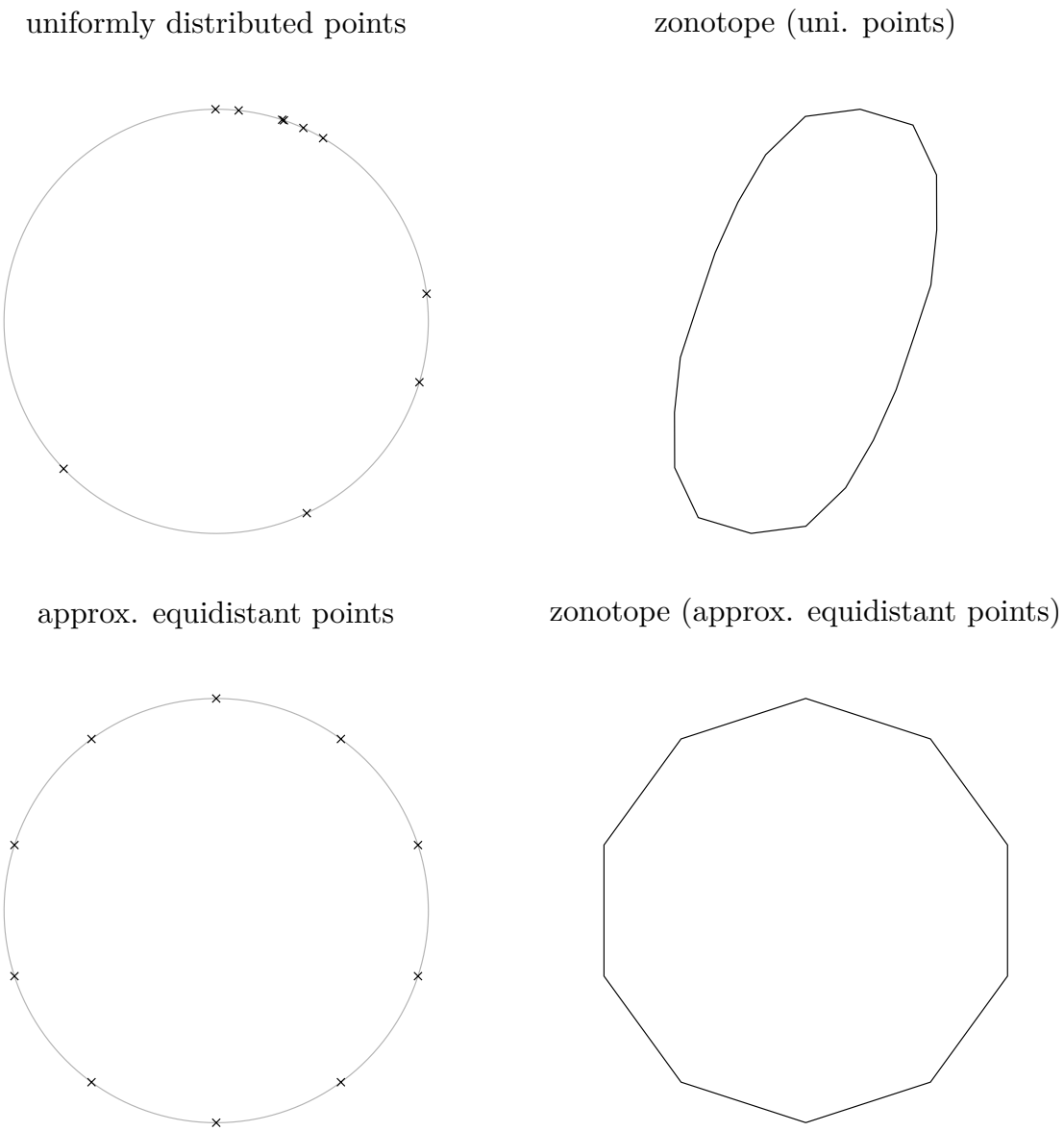


Figure 4.4: Uniform sampling of sphere directions and approximately equi-distant sampling of sphere directions and their resulting zonotopes.

Theorem 4.3 (Enclosing Zonotope)**.** *An n-dimensional zonotope with $m \geq n$ generators enclosing the ellipsoid $\langle c, Q \rangle_E$ is given by*

$$\hat{\mathcal{Z}}\left(\langle c, Q \rangle_E\right)_m = \begin{cases} \left\langle c, \frac{1}{\sqrt{\check{r}}} \sqrt{Q} S \right\rangle_Z, & m > n, \\ \langle c, \sqrt{Q} \rangle_Z, & m = n \end{cases},$$

where $S \in \mathbb{R}^{n \times m}$ collects the points $S_{(:,i)}$ on the surface of the hyper-ball for $1 \leq i \leq m$ and $\check{r}$ is a lower bound of $\lfloor \langle \cdot, S \rangle_Z \rfloor^2$ using Lem. 4.2.

Proof. Let $m > n$. With $T = Q^{-\frac{1}{2}}$, we have $T \langle c, Q \rangle_E = \langle Tc, I_n \rangle_E$. Further, we know that $\langle Tc, I_n \rangle_E \subseteq \left\langle Tc, \frac{1}{\sqrt{\check{r}}} S \right\rangle_Z$ due to the definition of $\check{r}$. Applying the inverse transform T^{-1} then gives the desired result.

Now let $m = n$. A zonotope enclosing a hyper-ball $\langle 0, I_n \rangle_E$ is given by $\langle 0, I_n \rangle_Z$, and therefore $\hat{\mathcal{Z}}\left(\langle c, Q \rangle_E\right)_n = \sqrt{Q} \langle 0, I_n \rangle_Z \oplus \{c\}$ since $\langle c, Q \rangle_E = \sqrt{Q} \langle 0, I_n \rangle_E \oplus \{c\}$. □

4.1.2.2 Inner Approximation

The inner approximation similarly follows by using the maximum zonotope norm.

Theorem 4.4 (Inscribed Zonotope)**.** *An n-dimensional zonotope with $m \geq n$ generators inscribed into the ellipsoid $\langle c, Q \rangle_E$ is given by*

$$\check{\mathcal{Z}}\left(\langle c, Q \rangle_E\right)_m = \begin{cases} \left\langle c, \frac{1}{\sqrt{\hat{r}}} \sqrt{Q} S \right\rangle_Z, & m > n, \\ \left\langle c, \frac{1}{\sqrt{n}} \sqrt{Q} \right\rangle_Z, & m = n \end{cases},$$

where $S \in \mathbb{R}^{n \times m}$ collects the points $S_{(:,i)}$ on the surface of the hyper-ball for $1 \leq i \leq m$ and $\hat{r}$ is an upper bound on $\lceil \langle \cdot, S \rangle_Z \rceil^2$ using Lem. 4.1.

Proof. Let $m > n$. Setting $T = Q^{-\frac{1}{2}}$, we have $T \langle c, Q \rangle_E = \langle Tc, I_n \rangle_E$. Further, we know that $\left\langle Tc, \frac{1}{\sqrt{\hat{r}}} S \right\rangle_Z \subseteq \langle Tc, I_n \rangle_E$ due to the definition of $\hat{r}$. Applying the inverse transform T^{-1} then gives the desired result.

Now let $m = n$. A zonotope inscribed into a hyper-sphere $\langle 0, I_n \rangle_E$ is given by $\left\langle 0, \frac{1}{\sqrt{n}} I_n \right\rangle_Z$, and therefore $\check{\mathcal{Z}}\left(\langle c, Q \rangle_E\right)_n = \sqrt{Q} \left\langle 0, \frac{1}{\sqrt{n}} I_n \right\rangle_Z \oplus \{c\}$ since it holds that $\langle c, Q \rangle_E = \sqrt{Q} \langle 0, I_n \rangle_E \oplus \{c\}$. □

Fig. 4.5 visualizes enclosing and inscribed zonotopes for a given two-dimensional ellipsoid.

For computation times and tightness results on the zonotope-to-ellipsoid and ellipsoid-to-zonotope conversions from Sec. 4.1.1 and Sec. 4.1.2, we refer the reader to [36, Sec. VI] as exact numbers are not relevant for the content of this thesis.

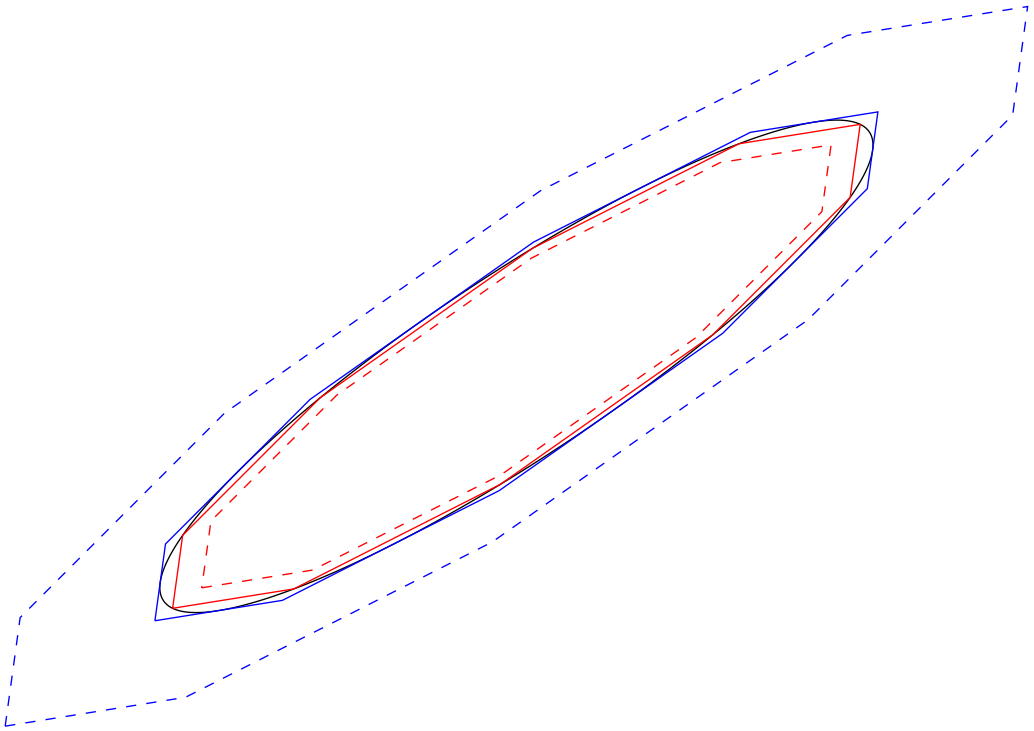


Figure 4.5: Different enclosing and inscribed zonotopes for a given two-dimensional ellipsoid. Shown are the approximated enclosing (see Th. 4.3) and inscribed (see Th. 4.4) zonotopes with exact zonotope norm (solid blue and red), and the approximated enclosing and inscribed zonotopes with bounded zonotope norm (dashed blue and red). All zonotopes have $m = 10$ generators.

4.1.3 H-Polytope to Parallelotope

In this thesis, we specify constraints for the reach-avoid problem using H-polytopes. An outer approximation of a bounded H-polytope with a parallelotope, as introduced in our work in [35], is described next.

Proposition 4.3 (H-Polytope to Parallelotope)**.** *Let $\mathcal{M} = \langle A, b \rangle_H$ with $A \in \mathbb{R}^{o \times n}$ and $b \in \mathbb{R}^o$ be given as a bounded polytope. Then a parallelotope outer approximation of $\mathcal{M}$ is given by $\mathcal{Z} = Q^{\frac{1}{2}} \langle \tilde{c}, \operatorname{diag} \tilde{r} \rangle_Z \oplus \{c\}$, where*

$$\begin{aligned} \tilde{c} &= \frac{1}{2} \left(\rho_{\tilde{\mathcal{M}}}(I) + \rho_{\tilde{\mathcal{M}}}(-I) \right), \\ \tilde{r} &= \frac{1}{2} \left(\rho_{\tilde{\mathcal{M}}}(I) - \rho_{\tilde{\mathcal{M}}}(-I) \right), \\ \tilde{M} &= \left\langle A Q^{\frac{1}{2}}, b - Ac \right\rangle_H, \end{aligned}$$

and $\langle c, Q \rangle_E$ is the MVIE of $\mathcal{M}$ [14, Sec. 8.4.2].

Proof. We shift $\mathcal{M}$ by c so that $0 \in \mathcal{M} \oplus \{-c\}$, and use the shape matrix Q to transform $\mathcal{M}$ into roughly a hypercube. Then $Q^{-\frac{1}{2}} \left(\langle c, Q \rangle_E \oplus \{-c\} \right) = \langle 0, I_n \rangle_E$ and it holds that $\tilde{\mathcal{M}} = Q^{-\frac{1}{2}} \left(\mathcal{M} \oplus \{-c\} \right) = \left\langle A Q^{\frac{1}{2}}, b - Ac \right\rangle_H$. A tight parallelotope enclosing $\tilde{\mathcal{M}}$ is given by $\tilde{\mathcal{Z}} = \langle \tilde{c}, \operatorname{diag} \tilde{r} \rangle_Z$ with $\tilde{c}$ and $\tilde{r}$ given as above. Applying the inverse transform and shift yields $\mathcal{Z} = Q^{\frac{1}{2}} \tilde{\mathcal{Z}} \oplus \{c\}$ which concludes the proof. □

Since we use the MVIE as a shape template for the enclosing parallelotope, the approximation quality of Prop. 4.3 depends on a shape similarity of the inscribed and the enclosed ellipsoid, which is not necessarily given. Depending on the required approximation quality, Prop. 4.3 may thus not be approxiate; since Prop. 4.3 is used in this thesis for normalization purposes only, this is of no concern here.

4.1.4 Polynomial Zonotope to Zonotope

Reachable sets of nonlinear systems are often computed using polynomial zonotopes to represent non-convex sets (see Sec. 2.7.3). However, for operations such as the support function, only outer approximations can be computed, which often requires the solution of a (possibly non-convex) optimization problem. Alternatively, one can enclose a polynomial zonotope with a zonotope and then use its support function.

Proposition 4.4 (Polynomial Zonotope to Zonotope [61, Prop. 5, adapted])**.** *A zonotope outer approximation of the polynomial zonotope $\mathcal{PZ} = \langle c, G, G_{\text{rest}}, E \rangle_{PZ}$ with starting point $c \in \mathbb{R}^n$, generator matrix $G \in \mathbb{R}^{n \times m}$, rest matrix $G_{\text{rest}} \in \mathbb{R}^{n \times r}$, and exponent matrix $E \in \mathbb{N}^{q \times m}$ is given by*

$$\hat{\mathcal{Z}}(\mathcal{PZ}) = \left\langle c + \frac{1}{2} \sum_{k \in \mathcal{I}_{\text{even}}} G_{(:,k)}, \left[G_{(:,\mathcal{I}_{\text{odd}})}, \quad \tfrac{1}{2} G_{(:,\mathcal{I}_{\text{even}})} \right], G_{\text{rest}}, I_m \right\rangle_{PZ},$$

where

$$\mathcal{I}_{\text{even}} = \left\{ i \in \{1, ..., m\} \;\middle|\; \text{mod}\left(E_{(:,i)}, 2\right) = 0 \right\},$$
$$\mathcal{I}_{\text{odd}} = \{1, ..., m\} \setminus \mathcal{I}_{\text{even}},$$

and $G_{(:,\mathcal{I}_{\text{odd}})} \in \mathbb{R}^{n \times |\mathcal{I}_{\text{odd}}|}$ *and* $G_{(:,\mathcal{I}_{\text{even}})} \in \mathbb{R}^{n \times |\mathcal{I}_{\text{even}}|}$ *collect all corresponding odd and even generators.*

By inspection, Prop. 4.4 is of at most polynomial complexity in the state dimension (see also [61, Prop. 5]). For a demonstration of the tightness of the resulting zonotope, see, e.g., [61, Fig. 3].

4.2 Set Containment

In this thesis, we want to solve controller synthesis problems under input and state constraints. As a result, efficient methods to check the containment of a set within, e.g., a given constraint set, are necessary.

Thus, we first introduce a containment check from previous work for two zonotpes in Sec. 4.2.2. Since constraints in this thesis will be given as H-polytopes, we then describe the containment check of a zonotope in an H-polytope and briefly motivate the superiority of this check over the zonotope-zonotope-containment for our purposes.

4.2.1 Zonotope in Zonotope

Since zonotopes are a popular choice for reachable sets in general, it is only natural to first try zonotopes as constraint sets.

Proposition 4.5 (Zonotope in Zonotope [97, Cor. 4, adapted])**.** *Given are two zonotopes* $\mathcal{X} = \langle \bar{x}, X \rangle_Z$ *with* $\bar{x} \in \mathbb{R}^n$*,* $X \in \mathbb{R}^{n \times m}$*, and* $\mathcal{Y} = \langle \bar{y}, Y \rangle_Z$ *with* $\bar{y} \in \mathbb{R}^n$*,* $Y \in \mathbb{R}^{n \times q}$*. Then* $\mathcal{X} \subseteq \mathbb{Y}$ *if there exists* $\Gamma \in \mathbb{R}^{q \times m}$ *and* $\beta \in \mathbb{R}^q$ *such that*

$$X = Y\Gamma,$$
$$\bar{y} - \bar{x} = Y\beta,$$
$$\left\| \left[\Gamma^T_{(:)}, \quad \beta^T \right] \right\|_\infty \leq 1.$$

Thus, Prop. 4.5 gives numerical conditions which can be easily checked using any linear programming solver. Note, however, that Prop. 4.5 is not sufficient and necessary, i.e., failure to find Γ and β that fulfill Prop. 4.5 does not necessarily mean that $\mathcal{X}$ is not contained in $\mathcal{Y}$.

4.2.2 Zonotope in H-Polytope

As mentioned in Sec. 4.2.1, the zonotope-zontope-containment check is only sufficient. When using such a containment check for controller synthesis to check constraints, this

might result in an overly conservative controller. Since we later assume that all constraint sets are represented as H-polytopes anyway, we next present an efficient way to check the containment of a zonotope within an H-polytope which is both necessary and sufficient.

Proposition 4.6 (Zonotope in Polytope [98, Lem. 1, adapted])**.** *A zonotope* $\mathcal{Z} = \langle c, G \rangle_Z$ *with* $c \in \mathbb{R}^n$ *and* $G \in \mathbb{R}^{n \times p}$ *is contained within a given H-polytope* $\mathcal{H} = \langle A, b \rangle_H$ *with* $A \in \mathbb{R}^{m \times n}$ *and* $b \in \mathbb{R}^m$ *if and only if*

$$\rho_{\mathcal{Z}}(A) \leq b.$$

Proof. We provide an alternative to the proof in [98] here.

We have

$$\begin{aligned} & \mathcal{Z} \subseteq \mathcal{H} \\ \iff & \forall l \in \mathbb{R}^n : \rho_{\mathcal{Z}}(l) \leq \rho_{\mathcal{H}}(l) \\ \iff & \rho_{\mathcal{Z}}(A) \leq \rho_{\mathcal{H}}(A), \end{aligned}$$

where the last equivalence follows from the fact that all halfspaces defining $\mathcal{H}$ are contained in A, which concludes the proof since $\rho_{\mathcal{H}}(A) = b$ (see Def. 2.15). □

4.3 Summary

In this chapter, we both reviewed and introduced novel set conversion methods in Sec. 4.1 which are necessary for later chapters. Additionally, we also described a zonotope-in-zonotope and zonotope-in-polytope containment check from the literature in Sec. 4.2 and motivated why the zonotope-in-polytope check is superior for our purposes.

In Sec. 4.1, we described – for the first time – ellipsoid-zonotope and zonotope-ellipsoid inner and outer approximations from our work in [36]: By combining upper and lower bounds on the maximum and minimum zonotope norms with appropriate initial shapes for both ellipsoids and zonotopes, we were able to find efficient inner and outer approximations for given zonotopes and ellipsoids. Furthermore, we described a novel method for efficiently outer approximating an H-polytope by a parallelotope and review the existing zonotope outer approximation of a given polynomial zonotope from [61].

In order to check constraints for the controller synthesis in subsequent chapters, we then introduced two existing set containment checks in Sec. 4.2: First, we reviewed the zonotope-zonotope containment checks from [97]. Because the zonotope-in-zonotope check is only sufficient, we avoid possible conservatism by specifying all constraints as H-polytopes in this thesis and thus introduced a zonotope-in-H-polytope check from [101] which is both necessary and sufficient.

5 Piecewise Constant Controller Synthesis

In order to arrive at an efficient method to synthesize controllers for nonlinear systems, it is often necessary to restrict the form of the controller one wishes to compute. In this chapter, formally correct controller synthesis using a combination of reachability analysis and optimization is discussed, where the controllers to be synthesized are piecewise constant in time.

In Sec. 5.1, we first introduce the general problem we aim to solve, and introduce an approach to the solution of this problem using a linearization of the system dynamics from previous work [99] in Sec. 5.2. In Sec. 5.3, we then extend this approach to polynomial controllers with higher-order reachability abstractions. Finally, we propose a novel iterative solution approach in Sec. 5.4 which can handle larger input sets efficiently and does not require manual tuning of the state constraint sets.

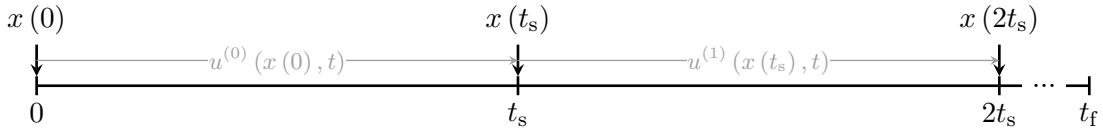
5.1 Problem Statement

Generally, the synthesis problem in (3.1) is not efficiently solvable for an arbitrary controller $u_{\text{ctrl}}(x(t), t)$. Therefore, one has to specify a parameterization for the controller to be synthesized. To that end, we start by subdividing the time horizon $[0, t_{\text{f}}]$ into $m_{\text{s}} \in \mathbb{N}_+$ intervals of length $t_{\text{s}} = \frac{t_{\text{f}}}{m_{\text{s}}}$ (see Fig. 5.1). The duration t_{s} denotes the time between state feedback, i.e., $u_{\text{ctrl}}(x(t), t) = u^{(i)}(x(it_{\text{s}}), t)$ for $t \in [i, i+1]\, t_{\text{s}}$ and $0 \leq i \leq m_{\text{s}} - 1$. We further split the time interval $[i, i+1]\, t_{\text{s}}$ using $t_{\text{c}} = \frac{t_{\text{s}}}{m_{\text{c}}}$, where t_{c} is the duration between piecewise constant controllers, i.e., $u_{\text{ctrl}}(x(t), t) = u_{\text{pc}}^{(i,j)}(x(it_{\text{s}}))$ for $t \in it_{\text{s}} + [j, j+1]\, t_{\text{c}}$ and $0 \leq j \leq m_{\text{c}} - 1$.

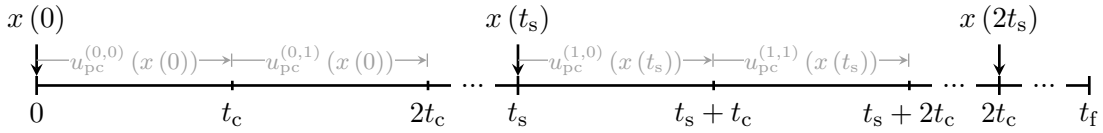
To enable optimization over these controllers $u_{\text{pc}}^{(i,j)}(x(it_{\text{s}}))$, we parameterize them in the controller parameters $P^{(i,j)} \in \mathbb{R}^{n_u \times a}$ and write $u\left(x(it_{\text{s}}), P^{(i,j)}\right)$, where $a \in \mathbb{N}_+$ is the number of controller parameters per input dimension (will be discussed in detail in subsequent sections). Let

$$
\begin{aligned}
P^{(i)} &= \left[P^{(i,0)^T}, \quad P^{(i,1)^T}, \quad \ldots, \quad P^{(i,m_{\text{c}}-1)^T}\right]^T, \\
P &= \left[P^{(0)^T}, \quad P^{(1)^T}, \quad \ldots, \quad P^{(m_{\text{s}}-1)^T}\right]^T.
\end{aligned}
$$

and denote with $\mathcal{R}(t, P)$ the closed-loop reachable set at time t using the controller $u\left(x(it_{\text{s}}), P^{(i,j)}\right)$ for $t \in it_{\text{s}} + [j, j+1]\, t_{\text{c}}$ with $0 \leq i \leq m_{\text{s}} - 1$ and $0 \leq i \leq m_{\text{c}} - 1$, which is now dependent on P. Lastly, we assume that the bounded input constraint set $\mathcal{U} = \langle C_{\mathcal{U}}, d_{\mathcal{U}} \rangle_H$ with $C_{\mathcal{U}} \in \mathbb{R}^{o_{\mathcal{U}} \times n_u}$ and $d_{\mathcal{U}} \in \mathbb{R}^{o_{\mathcal{U}}}$, state constraint set $\mathcal{X} = \langle C_{\mathcal{X}}, d_{\mathcal{X}} \rangle_H$



(a) Visualization of the duration $t_\mathrm{s} = \frac{t_\mathrm{f}}{m_\mathrm{s}}$ between state measurements.



(b) Visualization of the duration $t_\mathrm{c} = \frac{t_\mathrm{s}}{m_\mathrm{c}}$ between piecewise constant controller switching.

Figure 5.1: Visualization of the time discretization for discrete-time feedback and piecewise constant control inputs for each sampling interval.

with $C_\mathcal{X} \in \mathbb{R}^{o_\mathcal{X} \times n_x}$ and $d_\mathcal{X} \in \mathbb{R}^{o_\mathcal{X}}$, and final state constraint set $\mathcal{X}_\mathrm{f} = \langle C_{\mathcal{X}_\mathrm{f}}, d_{\mathcal{X}_\mathrm{f}} \rangle_H$ with $C_{\mathcal{X}_\mathrm{f}} \in \mathbb{R}^{o_{\mathcal{X}_\mathrm{f}} \times n_x}$ and $d_{\mathcal{X}_\mathrm{f}} \in \mathbb{R}^{o_{\mathcal{X}_\mathrm{f}}}$ are given as H-polytopes. We can then rewrite (3.1) as

$$\hat{P} = \arg\min_{P} \max_{x(\cdot,P) \in \mathcal{R}(\cdot,P)} \left\{ \left\| x\left(t_\mathrm{f}, P\right) - x_\mathrm{f} \right\|_1 + \zeta \sum_{i=0}^{m_\mathrm{s}-1} \sum_{j=0}^{m_\mathrm{c}-1} \left\| u\left(x\left(i t_\mathrm{s}, P\right), P^{(i,j)} \right) \right\|_1 \right\}, \tag{5.1a}$$

$$\text{s.t. } \forall i \in \{0, ..., m_\mathrm{s} - 1\} \, \forall j \in \{0, ..., m_\mathrm{c} - 1\} : u\left(x\left(i t_\mathrm{s}, P\right), P^{(i,j)} \right) \subseteq \mathcal{U}, \tag{5.1b}$$

$$\mathcal{R}\left([0, t_\mathrm{f}], P\right) \subseteq \mathcal{X}, \tag{5.1c}$$

$$\mathcal{R}\left(t_\mathrm{f}, P\right) \subseteq \mathcal{X}_\mathrm{f}. \tag{5.1d}$$

We remark that the inclusion of state constraints means that a feasible solution might not exist.

5.2 Generator-Space Control

In this section, we introduce the generator-space control (GSC) approach from [98, 99] to synthesize $m_\mathrm{s} m_\mathrm{c}$ piecewise constant controllers while ensuring state and input constraints. Fig. 5.2 visualizes the concept of GSC, while Alg. 1 describes the general procedure which is briefly summarized subsequently:

(i) First, a reference trajectory – steering the system from the center of the initial set close to the target state x_f while respecting constraints – is computed, which yields the intermediate target states $x_\mathrm{f}^{(1:m_\mathrm{s})}$ (l. 2).

(ii) We then iteratively synthesize m_c controllers for all m_s steps. We use a moving horizon $h \in \mathbb{N}_+$ with $1 \leq h \leq m_\mathrm{s} - i$ starting from $t = i t_\mathrm{s}$: This allows us to account for later changes in the nonlinear dynamics f which might not be visible for $t \in [0, t_\mathrm{s}]$. The algorithm than iterates as follows:

  (a) Using the outer approximation of the closed-loop reachable set which is available from the last iteration, we compute the parameterized reachable set, which is an approximation to the closed-loop reachable set for $t \in i t_\mathrm{s} + [0, h t_\mathrm{s}]$, parameterized in the controller parameters we wish to determine (l. 5–7).

  (b) This parameterized reachable set, along with the intermediate target states collected in $x_\mathrm{f}^{(i+(1:h))}$, is then used to compute the controller parameters subject to input and state constraints by solving a linear program (LP) (l. 8).

  (c) While we compute the parameters for all controllers until $t = (i + h)\, t_\mathrm{s}$, we only use the first m_c piecewise controllers and compute the closed-loop reachable set using the optimized controllers until $t = (i + 1)\, t_\mathrm{s}$ (l. 9).

As explained in Sec. 5.2.4 in detail, the efficient inclusion of state constraints in GSC requires the user to provide an adapted state constraint set $\tilde{\mathcal{X}} \subseteq \mathbb{R}^{n_x}$ and $\tilde{\mathcal{X}}_\mathrm{f} \subseteq \mathbb{R}^{n_x}$.

Algorithm 1 Generator-space control

1: **function** GENERATORSPACECONTROL($\mathcal{X}^{(0)}, \mathcal{U}, \mathcal{X}, \tilde{\mathcal{X}}, \mathcal{X}_\mathrm{f}, \tilde{\mathcal{X}}_\mathrm{f}, \mathcal{W}, m_\mathrm{s}, m_\mathrm{c}, x_\mathrm{f}, h, t_\mathrm{f}$)
2: $\quad [x_\mathrm{f}, u_\mathrm{ref}] =$ REFERENCETRAJECTORY($\mathcal{X}^{(0)}, \mathcal{U}, \mathcal{X}, \mathcal{X}_\mathrm{f}, m_\mathrm{s}, m_\mathrm{c}, x_\mathrm{f}, t_\mathrm{f}$) ▷ Sec. 5.2.1
3: $\quad \bar{\mathcal{R}} = \mathcal{X}^{(0)}$
4: $\quad$ **for** $i = 0; i < m_\mathrm{s}; i{+}{+}$ **do**
5: $\quad\quad \bar{\mathcal{R}} = \bar{\mathcal{R}} \downarrow_o$ ▷ $o \in \mathbb{N}_+$ (reduction order)
6: $\quad\quad h = \min(h, m_\mathrm{s} - i)$
7: $\quad\quad \tilde{\mathcal{R}}^{(i)} =$ PARAMREACH($\bar{\mathcal{R}}, h$) ▷ Sec. 5.2.3
8: $\quad\quad \hat{P}^{(i)} =$ COMPUTECTRL($\tilde{\mathcal{R}}^{(i)}, \mathcal{U}, \tilde{\mathcal{X}}, \tilde{\mathcal{X}}_\mathrm{f}, x_\mathrm{f}^{(i+(1:h))}, u_\mathrm{ref}^{(i m_\mathrm{c} + (0:h m_\mathrm{c} - 1))}$) ▷ Sec. 5.2.4
9: $\quad\quad \mathcal{R}([i, i+1]\, t_\mathrm{s}) =$ REACH($\bar{\mathcal{R}}, \hat{P}^{(i)}, t_\mathrm{s}$) ▷ Sec. 5.2.5
10: $\quad\quad \bar{\mathcal{R}} = \mathcal{R}((i+1) t_\mathrm{s})$
11: $\quad$ **end for**
12: $\quad$ **return** $\hat{P}$, $\mathcal{R}([0, t_\mathrm{f}])$
13: **end function**

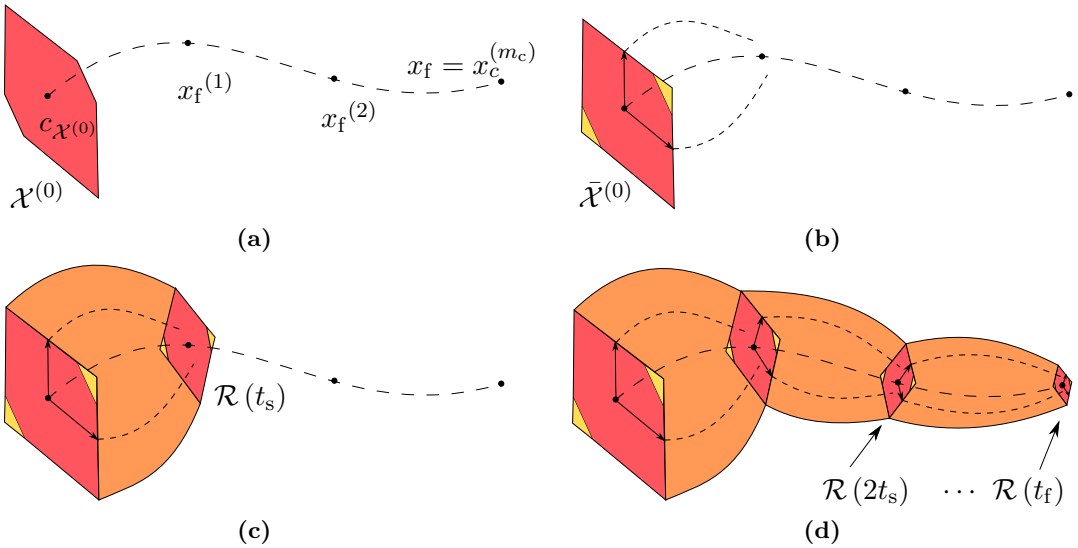


Figure 5.2: Overview of the GSC approach [101, adapted]: First, a reference trajectory is computed (Fig. 5.2a), which is then used to subdivide the overall synthesis problem into m_s parts and define the intermediate target states and their corresponding control inputs. Using this first intermediate target state $x_\mathrm{f}^{(1)}$, the synthesis problem (5.1) is solved for $x_\mathrm{f}^{(1)}$ (Fig. 5.2b). The closed-loop reachable set $\mathcal{R}(t_\mathrm{s})$ for the newly computed controller is then computed (Fig. 5.2c). Applying the same solution process iteratively – where $\mathcal{R}(t_\mathrm{s})$ is considered the new initial set – we can compute all closed-loop reachable sets, i.e., we obtain $\mathcal{R}([0, t_\mathrm{f}])$.

We start by introducing the reference trajectory computation in Sec. 5.2.1 and describe the proposed controller template in Sec. 5.2.2. For this template, we then compute its parameterized reachable set in Sec. 5.2.3 and introduce the synthesis of the controller in Sec. 5.2.4, where its closed-loop reachable set computation is then discussed in Sec. 5.2.5. Finally, we briefly discuss both the offline and online complexity of the GSC algorithm in Sec. 5.2.6 and shortly discuss its properties in Sec. 5.2.7.

Remark: We will not evaluate the performance of the GSC approach in this section but defer its numerical experiments to the next section, where it will be compared to a novel extension of GSC. We refer the reader to [99, Sec. 4] for a numerical comparison of GSC against other synthesis algorithms.

5.2.1 Reference Trajectory

In this section, we want to compute $m_\mathrm{s} m_\mathrm{c}$ piecewise constant control inputs such that the center $c_{\mathcal{X}^{(0)}}$ of the initial set $\mathcal{X}^{(0)} = \langle c_{\mathcal{X}^{(0)}}, G_{\mathcal{X}^{(0)}} \rangle_Z$ is steered closed to the target state x_f in time t_f. These $m_\mathrm{s} m_\mathrm{c}$ constant control inputs can be computed by solving

$$u_\mathrm{ref}^{(0:m_\mathrm{s}m_\mathrm{c}-1)} = \arg \min_{u^{(0:m_\mathrm{s}m_\mathrm{c}-1)}} \left(\left(x\left(t_\mathrm{f}\right) - x_\mathrm{f}\right)^T Q_\mathrm{ref} \left(x\left(t_\mathrm{f}\right) - x_\mathrm{f}\right) + \sum_{l=0}^{m_\mathrm{s}m_\mathrm{c}-1} {u^{(l)}}^T R_\mathrm{ref} u^{(l)} \right), \tag{5.2a}$$

$$\text{s.t. } \forall t \in [k, k+1]\, t_\mathrm{c} : \dot{x}\left(t\right) = f\left(x\left(t\right), u^{(k)}, 0\right) \tag{5.2b}$$

$$x\left(0\right) = c_{\mathcal{X}^{(0)}}, \tag{5.2c}$$

$$\forall t \in [0, t_\mathrm{f}] : x\left(t\right) \in \mathcal{X}, \tag{5.2d}$$

$$u^{(k)} \in \mathcal{U}, \tag{5.2e}$$

$$0 \leq k \leq m_\mathrm{s} m_\mathrm{c} - 1, \tag{5.2f}$$

where $Q_\mathrm{ref} \in \mathbb{S}_+^{n_x \times n_x}$ and $R_\mathrm{ref} \in \mathbb{S}_+^{n_u \times n_u}$ are user-specified weighting matrices. There exists software, such as the automatic control and dynamic optimization[1] [55] (ACADO) toolbox, which can efficiently solve (5.2). The corresponding state solutions are then given by $x_\mathrm{ref}^{(k+1)} = \xi\left(\left(k+1\right) t_\mathrm{c}, c_{\mathcal{X}^{(0)}}, u_\mathrm{ref}^{(0:k)}, 0\right)$ for $0 \leq k \leq m_\mathrm{s} m_\mathrm{c} - 1$, where the intermediate target states are given by $x_\mathrm{f}^{(1+i)} = \xi\left(\left(1+i\right) t_\mathrm{s}, c_{\mathcal{X}^{(0)}}, u_\mathrm{ref}^{(0:(1+i)m_\mathrm{c}-1)}, 0\right)$ for $0 \leq i \leq m_\mathrm{s} - 1$.

5.2.2 Controller Template

Before we can compute a parameterization of the reachable set in the controller parameters, we first need to define the parameterization of the controller. Since we assume the same parameterization for all $m_\mathrm{s} m_\mathrm{c}$ piecewise constant controllers, we restrict ourselves to the first piecewise constant controller for $t \in [0, t_\mathrm{c}]$, i.e., we find the template

[1] `https://acado.github.io/`

for $u\left(x\left(0\right), P^{(0,0)}\right)$. With slight abuse of notation, we assume for this section that $P = P^{(0,0)}$ for easier readability.

In Sec. 5.2.1, we already computed (sub-)optimal control inputs for $\mathcal{X}^{(0)} = \{c_{\mathcal{X}^{(0)}}\}$, i.e., when the initial set contains only its center. That said, the initial set $\mathcal{X}^{(0)}$ generally contains infinitely many initial states, and thus computing a control input trajectory $u_{\text{ref}}^{(0:m_\text{c}m_\text{s}-1)}$ for every state in $\mathcal{X}^{(0)}$ using Sec. 5.2.1 is not feasible. However, using a single input trajectory for every $x\left(0\right) \in \mathcal{X}^{(0)}$ is clearly not ideal, as the optimal input trajectory to reach x_f generally depends on the chosen initial state.

For the efficient computation of a controller for every initial state, the authors in [99, Sec. 3.2] introduce the controller parameterization

$$\bar{u}\left(\beta, P\right) = P_{(:,1)} + P_{(:,2:)}\beta = P\begin{bmatrix}1\\ \beta\end{bmatrix}, \tag{5.3}$$

where $P \in \mathbb{R}^{n_u \times (1+l)}$ and $\beta \in [-1,1]^l$ are the dependent factors of the generating function $x^{(0)}\left(\beta\right) = c_{\mathcal{X}^{(0)}} + G_{\mathcal{X}^{(0)}}\beta$ of the initial set $\mathcal{X}^{(0)} = \left\{x^{(0)}\left(\beta\right)\right\}_\beta$: By making (5.3) dependent on the initial state $x^{(0)}\left(\beta\right)$ through β, we obtain different control inputs for different initial states. We remark here that the optimal control law is assumed to be linear in β due to (5.3); this assumption will be removed in Sec. 5.3. The choice to parameterized the controller template in β instead of the initial state $x\left(0\right)$ directly has numerical advantages since $\beta \in [-1,1]^l$. Additionally, we parameterize the controller in P, which allows us to later synthesize an optimal controller by finding the optimal value for P.

For the online application of the controller, however, only the initial state $x\left(0\right)$ is given and thus we require the input parameterized in $x\left(0\right)$ and not β. If $\mathcal{X}^{(0)}$ is given as a non-degenerate parallelotope, the controller template parameterized in $x\left(0\right)$ then follows by substitution of $\beta = G_{\mathcal{X}^{(0)}}^{-1}\left(x\left(0\right) - c_{\mathcal{X}^{(0)}}\right)$ into (5.3), yielding (compare to [99, Eq. (15)])

$$u\left(x\left(0\right), P\right) = \bar{u}\left(G_{\mathcal{X}^{(0)}}^{-1}\left(x\left(0\right) - c_{\mathcal{X}^{(0)}}\right), P\right). \tag{5.4}$$

If $\mathcal{X}^{(0)}$ is a general zonotope, one can also replace $\mathcal{X}^{(0)}$ with its parallelotope outer approximation $\bar{\mathcal{X}}^{(0)} = \mathcal{X}^{(0)}\downarrow_1$ so that the controller template can still be directly parameterized in $\bar{x}\left(0\right) \in \bar{\mathcal{X}}^{(0)}$ (β then describes the dependent factor of the parallelotope outer approximation); however, this comes at the cost of possibly losing some input capacity: We need to ensure that $\forall x\left(0\right) \in \mathcal{X}^{(0)} : u\left(x\left(0\right), P\right) \subseteq \mathcal{U}$. If we now parameterize the template in $\bar{x}\left(0\right)$, this is still enforced, even though we introduce artificial initial states, i.e., there exist $\bar{x}\left(0\right) \notin \mathcal{X}^{(0)}$. As an alternative to this parallelotope enclosure, the factors β for a corresponding $x\left(0\right)$ can also be efficiently computed online by solving $x\left(0\right) = c_{\mathcal{X}^{(0)}} + G_{\mathcal{X}^{(0)}}\beta$ with $\left\|\beta\right\|_\infty \leq 1$ using linear programming, which does not result in a potential loss of input capacity.

Reference Input for the Center State: In Sec. 5.2.1, we compute a reference input trajectory for the center $c_{\mathcal{X}^{(0)}}$ of the initial set. If we wish to use this reference input

directly in the controller template, we have $u_{\text{ref}}^{(0)} = P \begin{bmatrix} 1 \\ 0 \end{bmatrix} \implies P_{(:,1)} = u_{\text{ref}}^{(0)}$ for $\beta = 0$ since $c_{\mathcal{X}^{(0)}} = c_{\mathcal{X}^{(0)}} + G_{\mathcal{X}^{(0)}} 0$. Thus, the controller template simply returns the input trajectory for the initial state and thus steers the reachable set along the reference trajectory.

5.2.3 Parameterized Reachable Set

The solution of (5.1) requires (an approximation of) the closed-loop reachable using the controller template from (5.3), parameterized in the controller parameters. Since the m_{s} optimization problems can be solved sequentially (see Alg. 1), we assume without loss of generality that we start from the initial set $\mathcal{X}^{(0)}$, generated by $x^{(0)}(\beta) = c_{\mathcal{X}^{(0)}} + G_{\mathcal{X}^{(0)}} \beta$ for center $c_{\mathcal{X}^{(0)}} \in \mathbb{R}^{n_x}$ and generator matrix $G_{\mathcal{X}^{(0)}} \in \mathbb{R}^{n_x \times l}$ with $\beta \in [-1, 1]^l$. As discussed in [98], we do not only need this parameterized set for $t \in [0, t_{\text{s}}]$, but rather $t \in [0, ht_{\text{s}}]$ since we optimize over an extended optimization horizon to account for later changes in the nonlinear dynamics f which might not be visible for $t \in [0, t_{\text{s}}]$. That said, we first find the parameterized reachable set for $t \in [0, t_{\text{s}}]$, and then extend this to the moving horizon of length h.

To that end, we linearize and time-discretize the undisturbed nonlinear dynamics $\dot{x} = f(x, u, 0)$ (see Def. 2.24), yielding (also compare to [99, Sec. 3.1])

$$x^{(j+1)} = c_d^{(j)} + A_d^{(j)} x^{(j)} + B_d^{(j)} u^{(j)}, \tag{5.5}$$

with

$$\begin{aligned}
A_d^{(j)} &= e^{A^{(j)} t_{\text{c}}}, \\
B_d^{(j)} &= \int_0^{t_{\text{c}}} e^{A^{(j)} \tau} \, \mathrm{d}\tau B^{(j)}, \\
c_d^{(j)} &= \int_0^{t_{\text{c}}} e^{A^{(j)} \tau} \, \mathrm{d}\tau c^{(j)}, \\
A^{(j)} &= \left. \frac{\partial f(x, u, 0)}{\partial x} \right|_{\substack{x = \bar{x}^{(j)} \\ u = \bar{u}^{(j)}}}, \\
B^{(j)} &= \left. \frac{\partial f(x, u, 0)}{\partial u} \right|_{\substack{x = \bar{x}^{(j)} \\ u = \bar{u}^{(j)}}}, \\
c^{(j)} &= f\left(\bar{x}^{(j)}, \bar{u}^{(j)}, 0\right) - A^{(j)} \bar{x}^{(j)} - B^{(j)} \bar{u}^{(j)},
\end{aligned}$$

where $x^{(0)} \in \mathcal{X}^{(0)}$, and $\bar{x}^{(j)} = \frac{1}{2}\left(x_{\text{ref}}^{(j)} + x_{\text{ref}}^{(j+1)}\right)$ and $\bar{u}^{(j)} = u_{\text{ref}}^{(j)}$ are the linearization points for $0 \leq j \leq m_{\text{c}} - 1$ (see Sec. 5.2.1). Evaluation of (5.5) using the corresponding generating functions yields the recursive relation

$$\tilde{r}\left((j+1) t_{\text{c}}, P, \beta\right) = c_d^{(j)} + A_d^{(j)} \tilde{r}\left(j t_{\text{c}}, P, \beta\right) + B_d^{(j)} \bar{u}\left(\beta, P^{(0,j)}\right), \tag{5.6}$$

where $\tilde{\mathcal{R}}(t,P) = \{\tilde{r}(t,P,\beta)\}_\beta$ and $\tilde{r}(0,P,\beta) = x^{(0)}(\beta) = c_{\mathcal{X}^{(0)}} + G_{\mathcal{X}^{(0)}}\beta$. For later convenience, we define

$$\tilde{\mathcal{R}}(jt_\mathrm{c},P) = \left\langle \tilde{c}(jt_\mathrm{c},P), \tilde{G}(jt_\mathrm{c},P) \right\rangle_Z,$$
$$\left\{ \bar{u}\left(\beta, P^{(i,j)}\right) \right\}_\beta = \left\langle c_u\left(P^{(i,j)}\right), G_u\left(P^{(i,j)}\right) \right\rangle_Z.$$

Extended Horizon: We notice from (5.6) that β alone generates $\tilde{\mathcal{R}}(jt_\mathrm{c},P)$ since both $\mathcal{X}^{(0)}$ and $\left\{ \bar{u}\left(\beta, P^{(0,j)}\right) \right\}_\beta$ are generated by β. As a result, the parameterized reachable set at $t = (k+1)t_\mathrm{s}$ for $0 \leq k \leq h-1$ can be found by simply replacing $\mathcal{X}^{(0)}$ with $\tilde{\mathcal{R}}(kt_\mathrm{s},P)$ and executing the aforementioned steps again.

5.2.4 Controller Computation

With the parameterized reachable set computed in Sec. 5.2.3, we are now ready to approximate (5.1) as proposed in [99]. We first transform the objective function in (5.1) into a tractable approximation and then briefly discuss how the constraints of (5.1) can be adapted such that the resulting optimization problem can be posed as an LP. Due to the iterative nature of the GSC algorithm, we again assume without loss of generality that $t = 0$, i.e., we start at the initial set $\mathcal{X}^{(0)}$.

Objective Function: We bound (5.1a) over the extended optimization horizon from above as (also compare to [99, Eq. (9), (10)], [98, Lem. 2])

$$\max_{x(\cdot,P)\in\tilde{\mathcal{R}}(\cdot,P)} \sum_{k=1}^{h} \vartheta_k \left(\left\| x(kt_\mathrm{s},P) - x_\mathrm{f}^{(k)} \right\|_1 + \zeta \sum_{j=0}^{m_\mathrm{c}-1} \left\| u\left(x\left((k-1)t_\mathrm{s},P\right), P^{(k-1,j)} \right) \right\|_1 \right) \tag{5.7}$$

$$\leq \sum_{k=1}^{h} \vartheta_k \left(\left\| \begin{bmatrix} \tilde{c}(kt_\mathrm{c},P) - x_\mathrm{f}^{(k)} \\ \tilde{G}(kt_\mathrm{c},P)_{(:)} \end{bmatrix} \right\|_1 + \zeta \sum_{j=0}^{m_\mathrm{c}-1} \left\| \begin{bmatrix} c_u\left(P^{(k-1,j)}\right) \\ G_u\left(P^{(k-1,j)}\right)_{(:)} \end{bmatrix} \right\|_1 \right), \tag{5.8}$$

as the cost function, where $\zeta \in \mathbb{R}_+$ penalizes the overall input size, and $\vartheta \in \mathbb{R}^h_{\geq 0}$ weighs the size of the reachable set at different points in time over the extended horizon h. The upper bound from the triangle inequality to arrive at (5.8) corresponds to a maximizing $x(kt_\mathrm{s},P)$ which in general does not exist, as the following example demonstrates.

Example 5.1. Let $h = 1$, $x_\mathrm{f} = 0$, $\vartheta = \vartheta_1 = 1$, $\zeta = 0$, and $\tilde{\mathcal{R}}(t_\mathrm{f},P) = \{\tilde{r}(t_\mathrm{f},\beta)\}_\beta$ with $\tilde{r}(t_\mathrm{f},\beta) = \begin{bmatrix} 2 \\ -2 \end{bmatrix}\beta_1 + \begin{bmatrix} 1 \\ 1 \end{bmatrix}\beta_2$. Then (5.7) can be equivalently written as

$$\max_{\|\beta\|_\infty \leq 1} \|\tilde{r}(t_\mathrm{f},\beta) - x_\mathrm{f}\|_1 = \max_{\|\beta\|_\infty \leq 1} \left(|2\beta_1 + \beta_2| + |-2\beta_1 + \beta_2| \right)$$
$$= 4.$$

In contrast, (5.8) yields

$$\max_{\|\beta\|_\infty \leq 1} \|\tilde{r}(t_\mathrm{f}, \beta) - x_\mathrm{f}\|_1 \leq \left\| \begin{bmatrix} 2 \\ -2 \end{bmatrix} \right\|_1 + \left\| \begin{bmatrix} 1 \\ 1 \end{bmatrix} \right\|_1 = 6.$$

■

Since the objective function is based on the linearized parameterized reachable set, its approximation quality tends to diminish with an increasing prediction horizon. As a result, one typically chooses $\vartheta_1 \geq \vartheta_2 \geq \cdots \geq \vartheta_h$, and we have a direct approximation of (5.1a) for $\vartheta_1 = \vartheta_2 = \cdots = \vartheta_{h-1} = 0$, $\vartheta_h = 1$, and $h = m_\mathrm{s}$.

Constraints: Using the approximated, parameterized reachable set instead of an outer approximation, the constraints in (5.1b) to (5.1d) for the extended horizon $t \in [0, ht_\mathrm{s}]$ are

$$\forall i \in \{0, ..., h-1\} \, \forall j \in \{0, ..., m_\mathrm{c} - 1\} : \left\{ \bar{u}\left(\beta, P^{(i,j)}\right) \right\}_\beta \subseteq \mathcal{U}, \tag{5.9}$$

$$\tilde{\mathcal{R}}([0, ht_\mathrm{s}], P) \subseteq \mathcal{X}, \tag{5.10}$$

$$\tilde{\mathcal{R}}(ht_\mathrm{s}, P) \subseteq \mathcal{X}_\mathrm{f}. \tag{5.11}$$

Using Prop. 4.6, the input constraint $\{\bar{u}(P, \beta)\}_\beta \subseteq \mathcal{U} = \langle C_\mathcal{U}, d_\mathcal{U} \rangle_H$ can be written as

$$\left\{ \bar{u}\left(\beta, P^{(i,j)}\right) \right\}_\beta \subseteq \mathcal{U} \iff \rho_u\left(C_\mathcal{U}, P^{(i,j)}\right) \leq d_\mathcal{U},$$

which can be encoded as a convex constraint by reformulating the absolute values using auxiliary variables (see Sec. 2.3.6.1) since $\left\{ \bar{u}\left(\beta, P^{(i,j)}\right) \right\}_\beta = \hat{\mathcal{Z}}\left(\left\{ \bar{u}\left(\beta, P^{(i,j)}\right) \right\}_\beta \right)$; here, $\rho_u\left(\cdot, P^{(i,j)}\right)$ denotes the support function of $\left\{ \bar{u}\left(\beta, P^{(i,j)}\right) \right\}_\beta$ and $\hat{\mathcal{Z}}(\cdot)$ denotes the outer approximation of a polynomial zontope by a zonotope as described in Sec. 4.1.4.

To check state constraints, we only have the approximate time-point solutions

$$\tilde{x}(it_\mathrm{s} + jt_\mathrm{c}, P) \in \tilde{\mathcal{R}}(it_\mathrm{s} + jt_\mathrm{c}, P), \; 0 \leq i \leq m_\mathrm{s} - 1, \; 0 \leq j \leq m_\mathrm{c},$$

available. While the set of applied inputs $\{\bar{u}(P, \beta)\}_\beta$ is known exactly and thus the input constraints can be checked using a zonotope outer approximation as described above, the set of states is only approximate. Therefore, we cannot directly enforce the state constraints contained in $\mathcal{X}$: It might happen that the approximation of the parameterized reachable set is much smaller than an outer approximation of the reachable set, in which case the resulting controller will most likely violate the state constraints. Therefore, let $\tilde{\mathcal{X}} = \langle C_{\tilde{\mathcal{X}}}, d_{\tilde{\mathcal{X}}} \rangle_H \subseteq \mathbb{R}^{n_x}$ with $C_{\tilde{\mathcal{X}}} \in \mathbb{R}^{o_\mathcal{X} \times n_x}$ and $d_{\tilde{\mathcal{X}}} \in \mathbb{R}^{o_\mathcal{X}}$ denote the user-provided state constraint set which is adapted to the approximated parameterized reachable set (see Sec. 5.2.7 for details).

We now approximate the interval constraint (5.10) with $m_\mathrm{c}h$ time-point constraints and apply Prop. 4.6, yielding

$$\max_{0\leq k\leq hm_\mathrm{c}} \tilde{\rho}_x\left(C_{\tilde{\mathcal{X}}}, kt_\mathrm{c}, P\right) \leq d_{\tilde{\mathcal{X}}},$$

where $\tilde{\rho}_x\left(\cdot, t, P\right)$ denotes the support function of $\tilde{\mathcal{R}}\left(t, P\right)$. Finally, the final state constraints can be rewritten using again Prop. 4.6 as

$$\tilde{\rho}_x\left(C_{\tilde{\mathcal{X}}_\mathrm{f}}, ht_\mathrm{s}, P\right) \leq d_{\tilde{\mathcal{X}}_\mathrm{f}},$$

where $\tilde{\mathcal{X}}_\mathrm{f} = \left\langle C_{\tilde{\mathcal{X}}_\mathrm{f}}, d_{\tilde{\mathcal{X}}_\mathrm{f}}\right\rangle_H \subseteq \mathbb{R}^{n_x}$ with $C_{\tilde{\mathcal{X}}_\mathrm{f}} \in \mathbb{R}^{o_{\mathcal{X}_\mathrm{f}}\times n_x}$ and $d_{\tilde{\mathcal{X}}_\mathrm{f}} \in \mathbb{R}^{o_{\mathcal{X}_\mathrm{f}}}$ are user-provided, adapted final state constraints.

Optimization Problem: The optimization problem for the extended horizon h is then given by (also compare to [99, Eq. (12), (13)], [98, Th. 3])

$$\hat{P}^{(0:h-1)} = \arg\min_P \sum_{k=1}^{h} \vartheta_k \left(\left\| \begin{bmatrix} \tilde{c}\left(kt_\mathrm{c}, P\right) - x_\mathrm{f}^{(k)} \\ \left[\tilde{G}\left(kt_\mathrm{c}, P\right)\right]_{(:)} \end{bmatrix} \right\|_1 + \zeta \sum_{j=0}^{m_\mathrm{c}-1} \left\| \begin{bmatrix} c_u\left(P^{(k-1,j)}\right) \\ \left[G_u\left(P^{(k-1,j)}\right)\right]_{(:)} \end{bmatrix} \right\|_1 \right), \tag{5.12a}$$

$$\text{s.t.} \quad \max_{\substack{0\leq i\leq h-1 \\ 0\leq j\leq m_\mathrm{c}-1}} \rho_u\left(C_\mathcal{U}, P^{(i,j)}\right) \leq d_\mathcal{U}, \tag{5.12b}$$

$$\max_{0\leq k\leq hm_\mathrm{c}} \tilde{\rho}_x\left(C_{\tilde{\mathcal{X}}}, kt_\mathrm{c}, P\right) \leq d_{\tilde{\mathcal{X}}}, \tag{5.12c}$$

$$\tilde{\rho}_x\left(C_{\tilde{\mathcal{X}}_\mathrm{f}}, t_\mathrm{f}, P\right) \leq d_{\tilde{\mathcal{X}}_\mathrm{f}}. \tag{5.12d}$$

The absolute values and maximizations can be reformulated using the techniques from Sec. 2.3.6 which then yields an LP. While we obtain optimal controller parameters for all h steps, we only apply the first m_c piecewise constant controllers corresponding to $\hat{P}^{(0)}$ until $t = t_\mathrm{s}$ (see Alg. 1).

5.2.5 Closed-Loop Reachable Set

Subsequently, we show how an outer approximation of the closed-loop reachable set can be computed. Without loss of generality, we first focus on the reachable set computation for $t \in [0, t_\mathrm{s}]$. Given an optimal $\hat{P}^{(0)}$ as the solution to (5.12) for $t \in [0, t_\mathrm{s}]$, the controller is (see (5.3)) $\bar{u}\left(\beta, \hat{P}^{(0,j)}\right)$ for $0 \leq j \leq m_\mathrm{c} - 1$. The initial set is given by $\mathcal{X}^{(0)} = \langle c_{\mathcal{X}^{(0)}}, G_{\mathcal{X}^{(0)}}\rangle_Z = \left\{x^{(0)}\left(\beta\right)\right\}_\beta$.

We mainly use two different algorithms to compute reachable sets in this thesis: In Sec. 5.2.5.1, we describe the reachable set computation using the conservative linearization approach from [9] described in Sec. 2.7.2, and in Sec. 5.2.5.2 we introduce the reachable set computation for polynomial zonotopes using the conservative polynomialization approach from [5], which we introduced in Sec. 2.7.3.

5.2.5.1 Conservative Linearization

In Sec. 2.7, we discussed how the reachable set for given closed-loop system dynamics can be computed (also compare to [98, Sec. 3.3.6]). Thus, we introduce the extended dynamics

$$f_{\text{ext}}\left(x_{\text{ext}}, w\right) = \begin{bmatrix} f\left(x, u, w\right) \\ 0 \\ 0 \end{bmatrix},$$

with the extended state $x_{\text{ext}} = \begin{bmatrix} x^T, & u^T, & \beta^T \end{bmatrix}^T$ and the generating function of the extended initial set

$$x_{\text{ext}}^{(0)}\left(\beta\right) = \begin{bmatrix} x^{(0)}\left(\beta\right) \\ \bar{u}\left(\beta, \hat{P}^{(0,0)}\right) \\ \beta \end{bmatrix}.$$

Since the computed controller is dependent on β, we include the input as an additional state variable, which allows us to define $x_{\text{ext}}^{(0)}\left(\beta\right)$ such that we always apply the corresponding pair of initial state and applied input. For $t \in [0, t_{\text{c}}]$ with $t_{\text{c}} = \frac{t_{\text{f}}}{m_{\text{s}} m_{\text{c}}}$, we can then compute the extended reachable set $\mathcal{R}_{\text{ext}}\left(t\right) = \left\{ r_{\text{ext}}\left(t, \tilde{\beta}\right) \right\}_{\tilde{\beta}}$ using the approach in Sec. 2.7.2. At $t = t_{\text{c}}$, we need to halt the reachability computations since the controller and thus the extended dynamics change. Due to reduction operations during reachability analysis and because reachability analysis with zonotopes does not preserve dependencies, the generating function of the extended reachable set is in general now dependent on a new dependent factor $\tilde{\beta}$ instead of β. Hence, in order to apply the correct control input, we need a mapping from the new dependent factors $\tilde{\beta}$ to the desired dependent factors β. Fortunately, this is available in the extended generating function of the reachable set at $t = t_{\text{c}}$, since by definition of the extended state we have

$$\beta\left(\tilde{\beta}\right) = \left[r_{\text{ext}}\left(t_{\text{c}}, \tilde{\beta}\right) \right]_{n_x + n_u + (1:n_x)}. \tag{5.13}$$

Substitution of (5.13) into the controller template then yields $\bar{u}\left(\beta\left(\tilde{\beta}\right), \hat{P}^{(0,1)}\right)$, which is now parameterized in the current dependent factor $\tilde{\beta}$. Thus, we can define the generating function for the new extended initial set as

$$x_{\text{ext}}^{(1)}\left(\tilde{\beta}\right) = \begin{bmatrix} \left[r_{\text{ext}}\left(t_{\text{c}}, \tilde{\beta}\right) \right]_{(1:n_x)} \\ \bar{u}\left(\beta\left(\tilde{\beta}\right), \hat{P}^{(0,1)}\right) \\ \beta\left(\tilde{\beta}\right) \end{bmatrix},$$

from which $\mathcal{R}_{\text{ext}}\left(2t_{\text{c}}\right)$ can be computed. The remaining steps until $t = t_{\text{s}}$ follow analogously.

5.2.5.2 Conservative Polynomialization

Using the approach from Sec. 2.7.3, we can make use of the dependency-preserving operations of polynomial zonotopes (see Sec. 2.6.6). We again define an extended state

$x_{\text{ext}} = \begin{bmatrix} x^T, & u^T \end{bmatrix}$ with extended dynamics

$$f_{\text{ext}}(x_{\text{ext}}, w) = \begin{bmatrix} f(x, u, w) \\ 0 \end{bmatrix},$$

and extended initial generating function

$$x_{\text{ext}}^{(0)}(\beta) = \begin{bmatrix} x^{(0)}(\beta) \\ \bar{u}\left(\beta, \hat{P}^{(0,0)}\right) \end{bmatrix}.$$

In contrast to Sec. 5.2.5.1, the extended state does not require the inclusion of β due to the dependency-retaining properties of polynomial zonotopes as implemented in CORA at the time of writing. That said, it is in principle possible to also implement dependency-retaining zonotopes.

The reachable set $\mathcal{R}_{\text{ext}}(t_{\text{c}}) = \{\tilde{r}_{\text{ext}}(t_{\text{c}}, \beta)\}_{\beta} \oplus \langle c_{\text{err}}, G_{\text{err}} \rangle_Z$ is computed by executing the algorithm described in Sec. 2.7.3 until $t = t_{\text{c}}$ using polynomial zonotopes. Here, $\tilde{r}_{\text{ext}}(t_{\text{c}}, \beta)$ is the generating function of the reachable set with the β dependency preserved, and $\{c_{\text{err}} + G_{\text{err}}\delta\}_{\delta}$ represents abstraction errors (see Sec. 2.7.3) and reduction errors (see Sec. 2.8). The new extended initial set is then given by the generating function

$$x_{\text{ext}}^{(1)}(\beta, \delta) = \begin{bmatrix} [\tilde{r}_{\text{ext}}(t_{\text{c}}, \beta) + c_{\text{err}} + G_{\text{err}}\delta]_{(1:n_x)} \\ \bar{u}\left(\beta, \hat{P}^{(0,1)}\right) \end{bmatrix},$$

from which $\mathcal{R}_{\text{ext}}(t)$ for $t \in [1, 2]\, t_{\text{c}}$ can be computed. The remaining steps follow analogously.

5.2.6 Computational Complexity

We first discuss the offline complexity of the controller synthesis and then briefly describe the complexity of applying the computed controller online. We make the following assumptions.

Assumption 5.1. The number of elementary operations $e \in \mathbb{N}_+$ to evaluate the closed-loop dynamics $[f_{\text{cl}}]_i : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \mapsto \mathbb{R}^{n_x}$ for each dimension $1 \leq i \leq n_x$ as well as any of its derivatives, the number of input constraints $o_{\mathcal{U}} \in \mathbb{N}_+$, state constraints $o_{\mathcal{X}} \in \mathbb{N}_+$, and final state constraints $o_{\mathcal{X}_{\text{f}}} \in \mathbb{N}_+$, the number of generators $l \in \mathbb{N}_+$ for any reachable set, and the number of dependent factors of the initial set all grow with at most $O(n)$, i.e.,

$$\begin{aligned} O(e) &= O(n), \\ O(o_{\mathcal{U}}) = O(o_{\mathcal{X}}) = O(o_{\mathcal{X}_{\text{f}}}) &= O(n), \\ O(l) &= O(n), \end{aligned}$$

where $n = \max(n_x, n_u)$. ■

5.2.6.1 Offline Complexity

We derive an upper bound on the computational complexity of GSC in n. Since m_s is fixed, it suffices to inspect the algorithm for $m_\mathrm{s} = 1$.

Reference Trajectory: Generally, the optimization problem (5.2) to compute the reference trajectory is a smooth non-convex optimization problem, which can be solved (using second-order methods) to a first-order critical point with accuracy ϵ using a number of function evaluations polynomially bounded in $\frac{1}{\epsilon}$ [18], where $\epsilon > 0$ is the selected accuracy. On the other hand, while tools like ACADO do not provide bounds on the complexity to compute the reference trajectory in Sec. 5.2.1, they solve these optimization problems in practice very efficiently. For this complexity analysis, we simply define the computational complexity as $O\left(c_\mathrm{ref}\left(n\right)\right)$, where $c_\mathrm{ref}\left(n\right)$ is some function of n.

Reachable Sets: Since the parameterized reachable set in Sec. 5.2.3 is given as a simple analytic function, its computational complexity is clearly lower than the complexity of computing the closed-loop reachable set, which is at most $O\left(n^5\right)$ [59, Sec. 4.1.4] when using the conservative polynomialization approach (see Sec. 5.2.5.2).

Controller Computation: The complexity of the controller computation is dominated by the solution of the optimization problem – which can be posed as a linear program – whose solution complexity is $O\left(c_\mathrm{LP}\left(n\right)\right) = O\left(n^\omega \log\left(\frac{n}{\delta}\right) \log^k\left(n\right)\right)$, where δ is the relative accuracy and $k \in \mathbb{N}_+$ is independent of n [16].

Overall Complexity: The computational offline complexity of GSC is thus at most

$$O\left(c_\mathrm{GSC}^{(\mathrm{off})}\left(n\right)\right) = O\left(c_\mathrm{ref}\left(n\right) + c_\mathrm{LP}\left(n\right) + n^5\right). \tag{5.14}$$

5.2.6.2 Online Complexity

The output of GSC are the optimal controller parameters $\hat{P}$ and the closed-loop reachable set $\mathcal{R}\left([0, t_\mathrm{f}]\right)$. Due to the sequential nature of GSC and because h, m_s, and m_c are fixed, it again suffices to restrict the analysis to $t \in [0, t_\mathrm{c}]$.

If $\mathcal{X}^{(0)}$ is given as a parallelotope (or enclosed by one), obtaining β for a given $x\left(0\right) \in \mathcal{X}^{(0)}$ amounts to a simple matrix vector multiplication with complexity $O\left(n_x^2\right)$ (see (5.4)) and another matrix vector multiplication with complexity $O\left(n_u n_x\right)$ (see (5.3)), so that the overall online complexity is $O\left(n^2\right)$. If we do not compute a parallelotope outer approximation of $\mathcal{R}\left(it_\mathrm{s}\right)$ and its l generators, obtaining β requires the solution of an LP with complexity $O\left(c_\mathrm{LP}\left(l\right)\right) = O\left(c_\mathrm{LP}\left(n\right)\right)$ since $O\left(l\right) = O\left(n\right)$ by Ass. 5.1. Computing $\bar{u}\left(\beta, \hat{P}^{(i,j)}\right)$ is then one matrix-vector multiplication (see (5.3)) with complexity $O\left(n_u l\right) = O\left(n^2\right)$. Thus, the online application of GSC has complexity

$$O\left(c_\mathrm{GSC}^{(\mathrm{on})}\left(n\right)\right) = \begin{cases} O\left(c_\mathrm{LP}\left(n\right)\right), & \text{no parallelotope outer approximation,} \\ O\left(n^2\right), & \text{otherwise} \end{cases}. \tag{5.15}$$

5.2.7 Discussion

Because the GSC approach uses a linearized, parameterized reachable set approximation and because the controller template is piecewise linear in the state, the controller synthesis only requires the solution of (5.12) which can be reformulated as an LP and therefore can be solved efficiently. However, this simplicity also means that the parameterized rechable set approximation might not be very accurate for nonlinear systems with large initial sets. Furthermore, a linear controller might not be enough to achieve the desired control performance.

Additionally, we so far simply accepted that the adapted state and final state constraints $\tilde{\mathcal{X}}$ and $\tilde{\mathcal{X}}_\mathrm{f}$ are given by the user. However, it is still unclear how these sets can be obtained from the original state and final state constraints $\mathcal{X}$ and $\mathcal{X}_\mathrm{f}$, respectively. Since we have no information about the accuracy of the parameterized reachable set approximation and because its accuracy is also dependent on the initial set and the input set, finding $\tilde{\mathcal{X}}$ and $\tilde{\mathcal{X}}_\mathrm{f}$ requires trial and error by the user for each system and each system setup for which a controller needs to be synthesized. As a result, the GSC approach is not trivial to apply for non-experts when state and final state constraints need to be enforced.

5.3 Polynomial Generator-Space Control

Due to the linear approximation of the parameterized reachable set and the limitation to linear control laws in GSC, it may be impossible to achieve satisfactory controller performance, especially when the system dynamics are very nonlinear. In this section, we introduce the polynomial generator-space control (PGSC) approach from our work in [37] which is an extension of the GSC approach. Since PGSC subsumes GSC, we again solve (5.1) by synthesizing $m_\mathrm{s} m_\mathrm{c}$ piecewise constant controllers (see Fig. 5.1). In contrast to GSC, however, compute the parameterized reachable set using arbitrary system abstractions and allow for an extension to polynomial controller templates.

Alg. 2 shows the general procedure for PGSC. The algorithm iteratively computes the following steps to synthesize $m_\mathrm{s} m_\mathrm{c}$ piecewise constant controllers with respect to input, state, and final state constraints:

(i) Given the center of the current initial set, we compute a reference trajectory from that center to the target state (l. 7).

(ii) Let $i \in \{0, ..., m_\mathrm{s} - 1\}$ denote the current iteration. We now compute an approximation to the parameterized reachable set using higher-order abstractions for $t \in [i, i + h]\, t_\mathrm{s}$ (l. 8).

(iii) Since we compute the parameterized reachable set using higher-order abstractions, the resulting set is generally no longer a zonotope, making the optimization problem in the next step possibly non-convex. Therefore, we compute an initial guess for the controller parameters using, e.g., the GSC approach (l. 9).

(iv) Using the updated reference trajectory, the parameterized reachable set, and the computed initial guess for the controller parameters, we solve the synthesis problem to obtain the optimal controller parameters (l. 10).

(v) Lastly, the outer approximation of the closed-loop reachable set for $t \in [i, (i + 1)]\, t_\mathrm{s}$ is computed for the newly synthesized optimal controller (l. 11).

Subsequently, we present the described steps of the algorithm in detail: In Sec. 5.3.1, we show how a drift of the closed-loop reachable set away from the reference trajectory for $m_\mathrm{s} > 1$ can be avoided and then introduce the proposed polynomial controller template in Sec. 5.3.2. In Sec. 5.3.3, this template is then used to compute an approximation to the closed-loop reachable set, parameterized in the controller parameters, over which we then optimize in Sec. 5.3.4 to find the optimal controller parameters. The closed-loop reachable set for the resulting controller is then computed in Sec. 5.3.5. We derive the computational offline and online complexity of the approach in Sec. 5.3.6 and prove in Sec. 5.3.7 that PGSC generalizes GSC under mild technical assumptions. We conclude the section with with a comparison of the novel PGSC approach to the GSC approach using numerical experiments in Sec. 5.3.8 and a brief discussion the algorithm in Sec. 5.3.9.

Algorithm 2 Polynomial generator-space control

function polyGenSpaceControl($\mathcal{X}^{(0)}, \mathcal{U}, \mathcal{X}, \tilde{\mathcal{X}}, \mathcal{X}_\mathrm{f}, \tilde{\mathcal{X}}_\mathrm{f}, \mathcal{W}, m_\mathrm{s}, m_\mathrm{c}, x_\mathrm{f}, t_\mathrm{f}, h, \kappa$)
  $\bar{\mathcal{R}} = \mathcal{X}^{(0)}$
  for $i = 0; i < m_\mathrm{s}; i++$ **do**
    $h = \min(h, m_\mathrm{s} - i)$
    $\bar{\mathcal{R}} = \bar{\mathcal{R}} \downarrow_o$ ▷ $o \in \mathbb{N}_+$ (reduction order)
    $\bar{\mathcal{X}} = \hat{\mathcal{Z}}\left(\bar{\mathcal{R}}\right) \downarrow_1$
    $[x_\mathrm{f}, u_\mathrm{ref}] =$ referenceTrajectory($\bar{\mathcal{X}}, \mathcal{U}, \mathcal{X}, \mathcal{X}_\mathrm{f}, m_\mathrm{s} - i, m_\mathrm{c}, x_\mathrm{f}, t_\mathrm{f}$) ▷ Sec. 5.3.1
    $\tilde{\mathcal{R}}^{(i)} =$ paramReach($\bar{\mathcal{R}}, h, \kappa$) ▷ Sec. 5.3.3
    $\bar{P}^{(i+(1:h))} =$ initGuess($\bar{\mathcal{X}}, \mathcal{U}, \mathcal{X}, \tilde{\mathcal{X}}, \mathcal{X}_\mathrm{f}, \tilde{\mathcal{X}}_\mathrm{f}, \mathcal{W}, m_\mathrm{s} - i, m_\mathrm{c}, x_\mathrm{f}^{(\cdot)}, h, t_\mathrm{f}$) ▷ e.g. GSC (Sec. 5.2)
    $\hat{P}^{(i)} =$ computeCtrl($\bar{P}^{(i+(1:h))}, \tilde{\mathcal{R}}^{(i)}, \mathcal{U}, \tilde{\mathcal{X}}, \tilde{\mathcal{X}}_\mathrm{f}, x_\mathrm{f}, u_\mathrm{ref}, h, \kappa$) ▷ Sec. 5.3.4
    $\mathcal{R}([i, i+1]\, t_\mathrm{s}) =$ reach($\bar{\mathcal{R}}, \hat{P}^{(i)}$) ▷ Sec. 5.3.5
    $\bar{\mathcal{R}} = \mathcal{R}((i+1)t_\mathrm{s})$
  end for
  return $\hat{P} = \left[\hat{P}^{(0)^T}, \ \ldots, \ \hat{P}^{(m_\mathrm{s}-1)^T}\right], \mathcal{R}([0, t_\mathrm{f}])$
end function

5.3.1 Reference Trajectory

For the general computation procedure of the reference trajectory, we refer to Sec. 5.2.1. In this section, we discuss the necessary adaptations for $\mathcal{X}^{(0)}$ being a polynomial zonotope, and then argue why an iterative update of the reference trajectory can be beneficial.

Starting Point: Let again be $t = 0$ without loss of generality. In GSC, $\mathcal{X}^{(0)}$ is given as a zonotope and thus the reference trajectory starts at the center of the initial set. While it need not be the case for nonlinear systems in general, in practice this often means that the reference trajectory runs centrally through the reachable set and ends centrally in the final, closed-loop reachable set at $t = ht_\mathrm{s}$. As a result, it makes sense to use the reference trajectory at $t = kt_\mathrm{c}$ for $0 \leq k \leq hm_\mathrm{c}$ as linearization points for the computation of the abstraction (also see Sec. 5.3.3), based on which the parameterized reachable set is formed, to reduce abstraction errors.

In PGSC, $\mathcal{X}^{(0)}$ is generally given as a polynomial zonotope. If now at least one exponent vector in the exponent matrix of $\mathcal{X}^{(0)}$ has all-even exponents, its starting point no longer represents the center of the initial set due its even exponent vector, as the following example demonstrates.

Example 5.2 (Center Shift of a Polynomial Zonotope)**.** Given is the polynomial zonotope

$$\mathcal{PZ} = \langle c_{\mathcal{PZ}}, \cdot, \cdot, \cdot \rangle_{PZ} = \left\langle \begin{bmatrix} 1 \\ -1 \end{bmatrix}, \begin{bmatrix} 0, & 2 \\ 2, & 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1, & 2 \\ 1, & 4 \end{bmatrix} \right\rangle_{PZ},$$

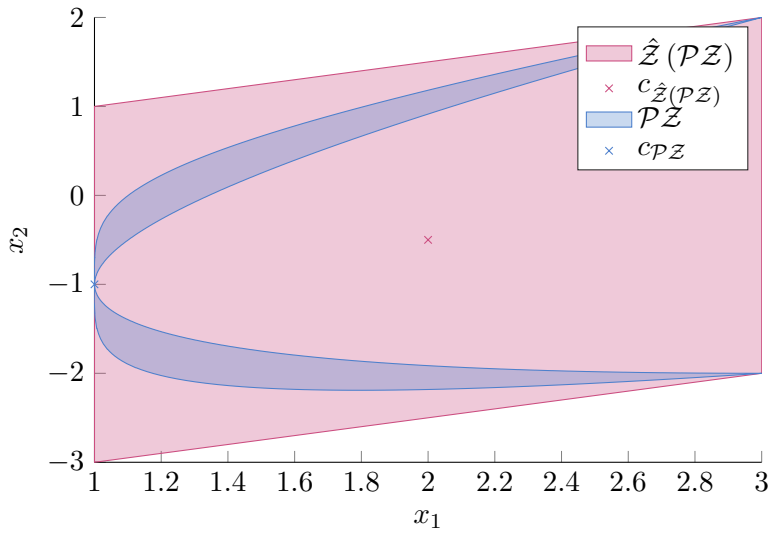


Figure 5.3: Comparison of the starting point of a polynomial zonotope to the center of its corresponding zonotope outer approximation using Prop. 4.4 as an approximation to the geometric center.

with its zonotope outer approximation according to Prop. 4.4 given by

$$\hat{\mathcal{Z}}(\mathcal{PZ}) = \langle c_{\mathcal{Z}}, \cdot \rangle_Z = \left\langle \begin{bmatrix} 2 \\ -\frac{1}{2} \end{bmatrix}, \begin{bmatrix} 0, & 1 \\ 2, & \frac{1}{2} \end{bmatrix} \right\rangle_Z .$$

As is clearly visible in Fig. 5.3, the geometric center is better approximated by the center $c_{\mathcal{Z}}$ of the zonotope outer approximation than by the starting point $c_{\mathcal{PZ}}$ of the polynomial zonotope.

■

Thus, to obtain a reference trajectory that is roughly centered in the convex hull of the reachable set – which is beneficial to reduce the abstraction error during reachability analysis (see Sec. 2.7) – we compute the parallelotope outer approximation

$$\bar{\mathcal{X}}^{(0)} = \langle c_{\bar{\mathcal{X}}^{(0)}}, G_{\bar{\mathcal{X}}^{(0)}} \rangle_Z = \hat{\mathcal{Z}}\left(\mathcal{X}^{(0)}\right) \downarrow_1 ,$$

and use its center $c_{\bar{\mathcal{X}}^{(0)}}$ as an approximation to the geometric center of $\mathcal{X}^{(0)}$ for the computation of the reference trajectory.

Update: Finding a reference trajectory from a given initial state to the target state using optimal control as described in Sec. 5.2.1 works very well in practice: Not only is the solution process efficient, but we also obtain an input trajectory that steers us very close to the target state if such a trajectory exists under input and state constraints. Thus, it often makes sense to use the reference input directly in the controller template as described in Sec. 5.2.2: This not only ensures that the controller steers all initial

states close to the target state, but also reduces the number of optimization variables in the synthesis problem.

To that end, we assume that $t = 0$ and let the reference trajectory from the geometric center $c^{(0)}$ of the initial set to the target state x_f be given by $u_\mathrm{ref}^{(0:m_\mathrm{s}m_\mathrm{c}-1)}$ with intermediate target states $x_\mathrm{f}^{(1+i)}$ for $0 \leq i \leq m_\mathrm{s} - 1$. Now denote with $u\left(c^{(0)}, P^{(0,j)}\right)$ the controller template parameterized in $P^{(0,j)}$ (see Sec. 5.3.2 for details) and let the first m_c controllers be chosen such that $u\left(c^{(0)}, P^{(0,j)}\right) = u_\mathrm{ref}^{(j)}$ for $0 \leq j \leq m_\mathrm{c} - 1$. Thus, the exact state trajectory brings $c^{(0)}$ to the first intermediate target state $x_\mathrm{f}^{(1)}$. However, reachability analysis does not integrate the dynamics exactly, such that the geometric center $c^{(1)}$ of the closed-loop reachable set at $t = t_\mathrm{s}$ is generally not equal to $x_\mathrm{f}^{(1)}$. If we now synthesize the controllers for the next step again such that $u\left(c^{(1)}, P^{(1,j)}\right) = u_\mathrm{ref}^{(m_\mathrm{c}+j)}$, even the exact state trajectory from $c^{(1)}$ under $u_\mathrm{ref}^{(m_\mathrm{c}+(0:m_\mathrm{c}-1))}$ does not reach $x_\mathrm{f}^{(2)}$ exactly since $u_\mathrm{ref}^{(m_\mathrm{c}+(0:m_\mathrm{c}-1))}$ steers $x_\mathrm{f}^{(1)}$ to $x_\mathrm{f}^{(2)}$ and $c^{(1)} \neq x_\mathrm{f}^{(1)}$. To avoid this shift, we update the reference trajectory in each iteration, starting from the geometric center (see Alg. 2, l. 7).

5.3.2 Controller Template

Without loss of generality, we again start at $t = 0$, and define with slight abuse of notation $P = P^{(0,0)}$ to avoid unnecessary indices.

Since $\mathcal{X}^{(0)} = \left\{x^{(0)}(\beta)\right\}_\beta$ is in general represented as a polynomial zonotope, finding β for a given $x(0) \in \mathcal{X}^{(0)}$ for the online application of the controller by solving

$$x(0) = x^{(0)}(\beta), \ \|\beta\|_\infty \leq 1, \tag{5.16}$$

is in general hard since β now is the solution of a polynomial equation under the constraint $\|\beta\|_\infty \leq 1$. Thus, we compute a parallelotope outer approximation $\bar{\mathcal{X}}^{(0)} = \langle c_{\bar{\mathcal{X}}^{(0)}}, G_{\bar{\mathcal{X}}^{(0)}} \rangle_Z = \hat{\mathcal{Z}}\left(\mathcal{X}^{(0)}\right) \downarrow_1 \supseteq \mathcal{X}^{(0)}$ with corresponding dependent factor $\tilde{\beta} \in [-1, 1]^{n_x}$ to obtain the analytic dependence $\tilde{\beta} = G_{\bar{\mathcal{X}}^{(0)}}^{-1}\left(\bar{x}(0) - c_{\bar{\mathcal{X}}^{(0)}}\right)$ for $\bar{x}(0) \in \bar{\mathcal{X}}^{(0)}$. Because $\bar{x}(0) \in \bar{\mathcal{X}}^{(0)} \implies x(0) \in \mathcal{X}^{(0)}$, we use

$$\begin{aligned} & c_{\bar{\mathcal{X}}^{(0)}} + G_{\bar{\mathcal{X}}^{(0)}}\tilde{\beta} = x^{(0)}(\beta) \\ \iff & \tilde{\beta} = G_{\bar{\mathcal{X}}^{(0)}}^{-1}\left(x^{(0)}(\beta) - c_{\bar{\mathcal{X}}^{(0)}}\right), \end{aligned} \tag{5.17}$$

to obtain an analytic dependence between $\tilde{\beta}$ and β. Next, we derive the controller template in $\tilde{\beta}$; by substituting $\tilde{\beta}$ from (5.17) into said template, we also obtain a template in β.

While the linear controller template of the GSC approach in (5.3) is simple, it is numerically ill-conditioned: If, e.g., $\mathcal{U}$ is near-degenerate, different row dimensions of P may have values over multiple orders of magnitude. For example, the thrust force and the ruder angle as inputs to an aircraft model may have wildly different magnitudes. As

a result, a numerically well-conditioned linear controller template of the GSC approach can be written as

$$\tilde{u}_{\text{lin}}\left(\tilde{\beta}, P\right) = c_{\bar{\mathcal{U}}} + G_{\bar{\mathcal{U}}} P \begin{bmatrix} 1 \\ \tilde{\beta} \end{bmatrix}, \tag{5.18}$$

where we used a parallelotope outer approximation $\bar{\mathcal{U}} = \langle c_{\bar{\mathcal{U}}}, G_{\bar{\mathcal{U}}} \rangle_Z \supseteq \mathcal{U}$. Instead of a direct parameterization of the input in P, we use $\bar{\mathcal{U}} = \{\bar{u}(\alpha)\}_\alpha$ with $\bar{u}(\alpha) = c_{\bar{\mathcal{U}}} + G_{\bar{\mathcal{U}}}\alpha$ to parameterize the dependent factor α in P, which achieves a parameterization with evenly scaled P: We have $\mathcal{U} \subseteq \bar{\mathcal{U}} \subseteq \left\{\tilde{u}_{\text{lin}}\left(\tilde{\beta}, P\right)\right\}_{\tilde{\beta},P}$ since $[-1,1]^{n_u} \subseteq \left\{P \begin{bmatrix} 1 \\ \tilde{\beta} \end{bmatrix}\right\}_{\tilde{\beta},P}$. Thus, it suffices to choose $P \in [-1,1]^{n_u \times (1+n_x)}$ to generate all admissible inputs. Note that $\bar{\mathcal{U}}$ is used for scaling purposes only; we still ensure the original input constraints using $\mathcal{U}$. We compute a parallelotope outer approximation of $\mathcal{U}$ since a zonotope outer approximation of $\mathcal{U}$ would increase the dimension of α in $\bar{u}(\alpha)$ beyond n_u. Thus, the row dimension of P would also grow, which in turn increases the number of optimization variables. If $\mathcal{U}$ is also available as a zonotope, $\bar{\mathcal{U}}$ can be obtained using reduction techniques (see Sec. 2.8). Otherwise, $\bar{\mathcal{U}}$ can be obtained using, e.g., Prop. 4.3. A natural extension of (5.18) to realize polynomial controllers then is

$$\tilde{u}\left(\tilde{\beta}, P\right) = c_{\bar{\mathcal{U}}} + G_{\bar{\mathcal{U}}}\left(\sum_{k=1}^{a} P_{(:,k)} \tilde{\beta}^{E_{(:,k)}}\right), \tag{5.19}$$

where $E \in \mathbb{N}^{n_x \times a}$ is a matrix of exponents with the assumption of $E_{(:,1)} = 0$ to model constant offsets, $a \in \mathbb{N}_+$ is the number of monomials in E, and $P \in [-1,1]^{n_u \times a}$. Similarly to the linear template in (5.18), the polynomial template in (5.19) also has uniform scaling in P, i.e., $P \in [-1,1]^{n_u \times a}$ since $[-1,1]^{n_u} \subseteq \left\{\sum_{k=1}^{a} P_{(:,k)} \tilde{\beta}^{E_{(:,k)}}\right\}_{\tilde{\beta},P}$. As a more convenient alternative to defining the exponent matrix directly, we further define the controller order $\kappa \in \mathbb{N}_+$: For a given controller order, we use the exponent matrix $E(\kappa) \in \mathbb{N}^{n_x \times o(\kappa)}$, which collects all $o(\kappa) = \binom{\kappa + n_x - 1}{n_x - 1}$ unique exponent vectors up to the total order κ, i.e., $\forall k \in \{1, ..., o(\kappa)\} : 1_{n_x}^T E(\kappa)_{(:,k)} \leq \kappa$, where $E(\kappa)_{(:,1)} = 0$.

Substitution of (5.17) into (5.19) then yields the controller template in $x(0)$ as

$$u(x(0), P) = \tilde{u}\left(G_{\bar{\mathcal{X}}^{(0)}}^{-1}\left(x(0) - c_{\bar{\mathcal{X}}^{(0)}}\right), P\right), \tag{5.20}$$

and the controller template in β is

$$\bar{u}(\beta, P) = u\left(x^{(0)}(\beta), P\right). \tag{5.21}$$

Reference Input for Center State: If we wish to use the reference input in the controller directly, i.e., set the respective parameters such that $u(c_{\bar{\mathcal{X}}^{(0)}}, P) = u_{\text{ref}}^{(0)}$ (see Sec. 5.3.1 for why $c_{\bar{\mathcal{X}}^{(0)}}$), it follows that $\tilde{u}(0, P) = u_{\text{ref}}^{(0)}$ and thus

$$\begin{aligned} & c_{\bar{\mathcal{U}}} + G_{\bar{\mathcal{U}}} P_{(:,1)} = u_{\text{ref}}^{(0)} \\ \iff & P_{(:,1)} = G_{\bar{\mathcal{U}}}^{-1}\left(u_{\text{ref}}^{(0)} - c_{\bar{\mathcal{U}}}\right), \end{aligned}$$

since $E_{(:,1)} = 0$ by assumption.

Remark: As stated initially, the derivations in this section assumed that we start from the initial set $\mathcal{X}^{(0)}$. If the controller is to be synthesized from $t = it_\mathrm{s}$ for $1 \leq i \leq m_\mathrm{s} - 1$, one can simply replace the initial set with the final, closed-loop reachable set at $t = it_\mathrm{s}$ that is obtained from the closed-loop reachable set computation using the previous controller (see Sec. 5.3.5).

5.3.3 Parameterized Reachable Set

In this section, we use a higher-order polynomial abstraction of our system to compute a parameterized reachable set (see also Sec. 2.7.3). This is in contrast to the GSC approach, where only a linear, time-discretized approximation is used to compute the parameterized reachable set. Similarly to GSC, we compute the parameterized reachable set based on the non-disturbed, nominal system dynamics.

In order to retain the dependency of $\mathcal{X}^{(0)}$ and the controller template on their dependent factor β, we extend the state by the input, which yields the generating function

$$x_\mathrm{ext}^{(0)}(P, \beta) = \begin{bmatrix} x^{(0)}(\beta) \\ \bar{u}\left(\beta, P^{(0,0)}\right) \end{bmatrix}, \tag{5.22}$$

of the extended initial set for $x_\mathrm{ext} = \begin{bmatrix} x^T, & u^T \end{bmatrix}^T$ with the extended flow given by

$$f_\mathrm{ext}(x_\mathrm{ext}) = \begin{bmatrix} f(x, u, 0) \\ 0 \end{bmatrix}.$$

Applying reachability analysis as described in Sec. 2.7.3 while interpreting both β and P in (5.22) as dependent factors for $t \in [0, t_\mathrm{c}]$ yields

$$\mathcal{R}(t, P) = \mathcal{D}(t, P) \oplus \mathcal{Z}_\mathrm{err} = \left\{ \left[r_\mathrm{ext}(t, P, \delta) \right]_{(1:n_x)} \right\}_\delta, \tag{5.23}$$

i.e., we generate $\mathcal{R}(t, P)$ over δ, which collects all dependent and independent factors except for P: While we treat P as a dependent factor during the computation of the parameterized reachable set to retain the dependency of the generating function $r_\mathrm{ext}(t, P, \delta)$ on P, we then generate $r_\mathrm{ext}(t, P, \delta)$ only over δ to obtain $\mathcal{R}(t, P)$ as an expression with P as an argument. We choose the linearization point $\bar{x}_\mathrm{ext}^{(k)} = \begin{bmatrix} \bar{x}^{(k)^T}, & \bar{u}^{(k)^T} \end{bmatrix}^T$ for the computation of $\hat{\mathcal{R}}(t, P)$ with $0 \leq k \leq m_r - 1$, $m_r \in \mathbb{N}_+$ denoting the number of reachability steps in $t \in [0, t_\mathrm{c}]$, and reachability step length $r = \frac{t_\mathrm{c}}{m_r}$, where

$$\bar{x}^{(k)} = \frac{1}{2}\left(\xi\left(kr, c_{\bar{\mathcal{X}}^{(0)}}, u_\mathrm{ref}^{(0)}, 0 \right) + \xi\left((k+1)\, r, c_{\bar{\mathcal{X}}^{(0)}}, u_\mathrm{ref}^{(0)}, 0 \right) \right), \tag{5.24}$$

$$\bar{u}^{(k)} = u_\mathrm{ref}^{(0)}. \tag{5.25}$$

Hence we ensure that the linearization during reachability analysis also occurs along the reference trajectory as in Sec. 5.2.3; specifically, we have $\xi\left(t_\mathrm{c}, c_{\bar{\mathcal{X}}^{(0)}}, u_\mathrm{ref}^{(0)}, 0 \right) = x_\mathrm{ref}^{(1)}$, where $\bar{\mathcal{X}}^{(0)} = \hat{\mathcal{Z}}\left(\mathcal{X}^{(0)} \right) \downarrow_1 = \langle c_{\bar{\mathcal{X}}^{(0)}}, G_{\bar{\mathcal{X}}^{(0)}} \rangle_Z$. The reachable set approximation $\hat{\mathcal{R}}([0, t_\mathrm{c}], P)$ is

then directly available since reachability analysis returns time-point and time-interval solutions.

However, by treating P as a dependent factor in the computation of $r_{\text{ext}}(t, P, \delta)$, we essentially compute the reachable set for all closed-loop controllers parameterized by $P \in [-1,1]^{n_u \times a}$. Since the result of reachability analysis generally worsens for large initial sets (since the abstraction used in reachability analysis is only locally accurate around the linearization points), increasing the size of the bounded input $\mathcal{U}$ causes an increasingly large outer approximation in reachability analysis which is detrimental to the approximation quality of $\hat{\mathcal{R}}(t, P)$, as demonstrated in the following example with $m_{\text{s}} = 1$ for ease of presentation.

Example 5.3. Let us consider a controlled Van-der-Pol oscillator with the dynamics $f(x,u) = \left[x_2, \quad (1-x_1^2)\,x_2 - x_1 + u\right]^T$, $\mathcal{X}^{(0)} = \left\{ \begin{bmatrix} 0.5\beta_1 \\ 0.3\beta_2 \end{bmatrix} \right\}_\beta$, $t_{\text{f}} = 0.5$, $\bar{u}\left(\beta, P^{(0,j)}\right) = G_{\bar{\mathcal{U}}}\left(p_1^{(j)}\beta_1 + p_2^{(j)}\beta_2\right)$ for $P^{(0,j)} = \left[p_1^{(j)}, \quad p_2^{(j)}\right]$ with $0 \leq j \leq m_{\text{c}} - 1 = 4$, $E = I_2$, and the parallelotope outer approximation $\bar{\mathcal{U}} = \langle 0, G_{\bar{\mathcal{U}}} \rangle_Z$.

Let a tight outer approximation $\mathcal{R}^*(t_{\text{f}})$ of the closed-loop reachable set for the controller $\bar{u}^{*(j)}(\beta) = \frac{1}{2}(\beta_1 + \beta_2)$ be given. We now compare this outer approximation to the parameterized reachable sets

$$\mathcal{R}^{(1)}\left(t, \breve{P}\right) = \mathcal{D}^{(1)}\left(t, \breve{P}\right) \oplus \mathcal{Z}_{\text{err}}^{(1)},$$
$$\mathcal{R}^{(2)}\left(t, \hat{P}\right) = \mathcal{D}^{(2)}\left(t, \hat{P}\right) \oplus \mathcal{Z}_{\text{err}}^{(2)},$$

evaluated at two fixed controller parameters $\breve{P} \in [-1,1]^{m_{\text{c}} n_u \times a}$ and $\hat{P} \in [-1,1]^{m_{\text{c}} n_u \times a}$, with increasing input capacities $\bar{\mathcal{U}}^{(1)} = \left\langle 0, \frac{1}{5} \right\rangle_Z$ and $\bar{\mathcal{U}}^{(2)} = \langle 0, 2 \rangle_Z$: The controller templates are hereby given by $\bar{u}^{(1)}\left(\beta, P^{(0,j)}\right) = \frac{1}{5}\left(p_1^{(j)}\beta_1 + p_2^{(j)}\beta_2\right)$ and $\bar{u}^{(2)}\left(\beta, P^{(0,j)}\right) = 2\left(p_1^{(j)}\beta_1 + p_2^{(j)}\beta_2\right)$, from which $\breve{P} = 1_{m_{\text{c}} n_u \times a}$ and $\hat{P} = 0.1 \cdot 1_{m_{\text{c}} n_u \times a}$ to replicate $\bar{u}^{*(j)}(\beta)$ follow. Fig. 5.4 visualizes these sets. Evidently, increasing the input capacity clearly increases the bloating of $\mathcal{R}^{(2)}\left(t, \breve{P}\right)$ compared to $\mathcal{R}^{(1)}\left(t, \hat{P}\right)$. In contrast, $\mathcal{D}^{(1)}\left(t, \breve{P}\right)$ and $\mathcal{D}^{(2)}\left(t, \hat{P}\right)$ are practically identical.

■

Thus, while using $\mathcal{R}(t, P)$ directly as an approximation for the parameterized reachable set can become quite inaccurate for increasing control capacity, the set $\mathcal{D}(t, P)$ remains even for larger input capacities a good approximation as the example demonstrated. Therefore, we propose to use $\tilde{\mathcal{R}}(t, P) = \mathcal{D}(t, P)$ as an approximation to the parameterized reachable set in order to avoid bloating effects for large input sets. The approximated reachable set for $t \in [1,2]\,t_{\text{c}}$ follows by choosing $\tilde{\mathcal{R}}(t_{\text{c}}, P)$ as the initial set for the next step with corresponding input $\left\{ \bar{u}\left(\beta, P^{(0,1)}\right) \right\}_\beta$ and executing the above steps again, where the remaining steps until $t = h t_{\text{s}}$ follow analogously.

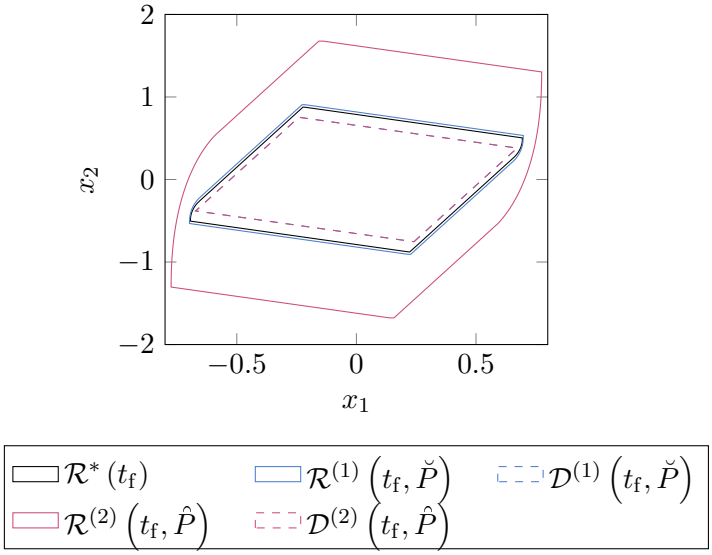


Figure 5.4: Approximation of the tight outer approximation $\mathcal{R}^*(t_\mathrm{f})$ of the reachable set by $\mathcal{R}^{(1)}\left(t, \breve{P}\right) = \mathcal{D}^{(1)}\left(t, \breve{P}\right) \oplus \mathcal{Z}^{(1)}_\mathrm{err}$ for $\breve{P} = 1_{m_c n_u \times a}$ and $\mathcal{R}^{(2)}\left(t, \hat{P}\right) = \mathcal{D}^{(2)}\left(t, \hat{P}\right) \oplus \mathcal{Z}^{(2)}_\mathrm{err}$ for $\hat{P} = 0.1 \cdot 1_{m_c n_u \times a}$. Note that $\mathcal{D}^{(1)}\left(t, \breve{P}\right)$ is not visible as it is visually identical to $\mathcal{D}^{(2)}\left(t, \hat{P}\right)$.

5.3.4 Controller Computation

We are now ready to pose the synthesis problem using the parameterized reachable set computed previously. Similarly to Sec. 5.2, we also consider an extended optimization horizon h. Without loss of generality, we further assume that we start from $t = 0$.

Let the parameterized reachable set $\tilde{\mathcal{R}}(t, P) = \{\tilde{r}(t, P, \beta)\}_\beta$ for $t \in [0, ht_\mathrm{s}]$ from Sec. 5.3.3 be given. Since $\tilde{\mathcal{R}}(t, P)$ is represented as a polynomial zonotope, there are dependencies between different generators. Since range bounding of polynomials is NP-hard in general [33], we compute a zonotope outer approximation $\tilde{\mathcal{Z}}(t, P) = \hat{\mathcal{Z}}\left(\tilde{\mathcal{R}}(t, P)\right) = \left\langle \tilde{c}(t, P), \tilde{G}(t, P) \right\rangle_Z$ of the approximated parameterized reachable set to be able to efficiently bound the maximization in (5.1a). We note that this zonotope outer approximation only operates on β, i.e., $\tilde{\mathcal{Z}}(t, P)$ still captures the nonlinearity of its center and generators on P.

Objective Function: It is clear that the parameterized zonotope $\tilde{\mathcal{Z}}(t, P)$ is in structure identical to the parameterized reachable set of the GSC approach – with the exception that it is now nonlinearly parameterized in P – which yields an upper bound on the objective function analogously to the derivation in (5.8).

Constraints: Using the approximated, parameterized reachable set instead of an outer approximation, the constraints in (5.1b) to (5.1d) for the extended horizon $t \in [0, ht_\mathrm{s}]$

can be written as

$$\forall i \in \{0, ..., m_\mathrm{s} - 1\} \, \forall j \in \{0, ..., m_\mathrm{c} - 1\} : \left\{ \bar{u} \left(\beta, P^{(i,j)} \right) \right\} \subseteq \mathcal{U}, \tag{5.26}$$

$$\tilde{\mathcal{R}} \left([0, h t_\mathrm{s}], P \right) \subseteq \mathcal{X}, \tag{5.27}$$

$$\tilde{\mathcal{R}} \left(t_\mathrm{s}, P \right) \subseteq \mathcal{X}_\mathrm{f}. \tag{5.28}$$

To efficiently bound the range of the controller template in β, we use the support function $\rho_u^{(\mathcal{Z})} \left(\cdot, P^{(i,j)} \right)$ of the zonotope outer approximation $\hat{\mathcal{Z}} \left(\left\{ u \left(P^{(i,j)}, \beta \right) \right\}_\beta \right) = \left\langle c_u \left(P^{(i,j)} \right), G_u \left(P^{(i,j)} \right) \right\rangle_Z$ and replace the input constraint in (5.26) using Prop. 4.6 by

$$\max_{\substack{0 \leq i \leq h-1 \\ 0 \leq j \leq m_\mathrm{c}-1}} \rho_u^{(\mathcal{Z})} \left(C_\mathcal{U}, P^{(i,j)} \right) \leq d_\mathcal{U}.$$

For the formulation of state constraints, time-point and time-interval solutions $\tilde{\mathcal{Z}} \left(t, P \right)$ for $t \in [0, h t_\mathrm{s}]$ are available since we use reachability analysis for the computation of $\tilde{\mathcal{R}} \left(t, P \right)$. Let $m_r \in \mathbb{N}_+$ be the number of steps for reachability analysis for the computation of $\tilde{\mathcal{R}} \left(t, P \right)$ with $t \in i t_\mathrm{s} + [j, j+1] \, t_\mathrm{c}$ for $0 \leq i \leq m_\mathrm{s} - 1$ and $0 \leq j \leq m_\mathrm{c} - 1$. Further, denote with $\tilde{\rho}_x^{(\mathcal{Z})} \left(\cdot, t, P \right)$ the corresponding support function of the zonotope outer approximation $\hat{\mathcal{Z}} \left(\tilde{\mathcal{R}} \left(t, P \right) \right)$: By using a zonotope outer approximation of the reachable set, we can conservatively approximate the polynomial zonotope-in-polytope constraints in (5.27) and (5.28) using Prop. 4.6, which yields

$$\max_{0 \leq k \leq m_r h m_\mathrm{c} - 1} \tilde{\rho}_x^{(\mathcal{Z})} \left(C_{\tilde{\mathcal{X}}}, [k, k+1] \, r, P \right) \leq d_{\tilde{\mathcal{X}}},$$

$$\tilde{\rho}_x^{(\mathcal{Z})} \left(C_{\tilde{\mathcal{X}}_\mathrm{f}}, h t_\mathrm{s}, P \right) \leq d_{\tilde{\mathcal{X}}_\mathrm{f}},$$

where $\tilde{\mathcal{X}} = \left\langle C_{\tilde{\mathcal{X}}}, d_{\tilde{\mathcal{X}}} \right\rangle_H \subseteq \mathbb{R}^{n_x}$ with $C_{\tilde{\mathcal{X}}} \in \mathbb{R}^{o_\mathcal{X} \times n_x}$ and $d_{\tilde{\mathcal{X}}} \in \mathbb{R}^{o_\mathcal{X}}$, $\tilde{\mathcal{X}}_\mathrm{f} = \left\langle C_{\tilde{\mathcal{X}}_\mathrm{f}}, d_{\tilde{\mathcal{X}}_\mathrm{f}} \right\rangle_H \subseteq \mathbb{R}^{n_x}$ with $C_{\tilde{\mathcal{X}}_\mathrm{f}} \in \mathbb{R}^{o_{\mathcal{X}_\mathrm{f}} \times n_x}$ and $d_{\tilde{\mathcal{X}}_\mathrm{f}} \in \mathbb{R}^{o_{\mathcal{X}_\mathrm{f}}}$ denote the adapted state and final state constraints required due to the approximate nature of $\tilde{\mathcal{R}} \left(t, P \right)$, and $r = \frac{t_\mathrm{f}}{m_r m_\mathrm{s} m_\mathrm{c}}$. Similarly to Sec. 5.2.4, these adapted constraints are necessary since we do not know how accurate the approximated parameterized reachable set is.

Optimization Problem: Collecting the objective function and the constraints finally yields the synthesis problem

$$\hat{P}^{(0:h-1)} = \arg\min_P \tilde{J} \left(P \right), \tag{5.29a}$$

$$\text{s.t. } \tilde{g} \left(P \right) \leq 0, \tag{5.29b}$$

with

$$\tilde{J}(P) = \sum_{k=1}^{h} \vartheta_k \left(\left\| \begin{bmatrix} \tilde{c}(kt_\mathrm{s}, P) - x_\mathrm{f}^{(k)} \\ \left[\tilde{G}(kt_\mathrm{s}, P)\right]_{(:)} \end{bmatrix} \right\|_1 + \zeta \sum_{j=0}^{m_\mathrm{c}-1} \left\| \begin{bmatrix} c_u\left(P^{(k-1,j)}\right) \\ \left[G_u\left(P^{(k-1,j)}\right)\right]_{(:)} \end{bmatrix} \right\|_1 \right), \tag{5.30}$$

$$\tilde{g}(P) = \begin{bmatrix} \max_{\substack{0 \le j \le m_\mathrm{c}-1 \\ 0 \le i \le h-1}} \rho_u^{(\mathcal{Z})}\left(C_\mathcal{U}, P^{(i,j)}\right) - d_\mathcal{U} \\ \max_{0 \le k \le m_r h m_\mathrm{c}-1} \tilde{\rho}_x^{(\mathcal{Z})}\left(C_{\tilde{\mathcal{X}}}, [k, k+1]\, r, P\right) - d_{\tilde{\mathcal{X}}} \\ \tilde{\rho}_x^{(\mathcal{Z})}\left(C_{\tilde{\mathcal{X}}_\mathrm{f}}, ht_\mathrm{s}, P\right) - d_{\tilde{\mathcal{X}}_\mathrm{f}} \end{bmatrix}, \tag{5.31}$$

where $\vartheta \in \mathbb{R}^h_{\geq 0}$ weighs each term of the objective function and where absolute values and maximization expressions can be reformulated (see Sec. 2.3.6) to yield a differentiable optimization problem. Depending on the chosen abstraction order and controller order, (5.29) is no longer expressible as an LP or even a convex optimization problem (see Sec. 5.3.7 for details).

Strictly speaking, (5.29) cannot be written as the abstracted problem in (3.2) directly. However, let $s \in \mathbb{R}^{o_\mathcal{U}+o_\mathcal{X}+o_{\mathcal{X}_\mathrm{f}}}$ be a slack variable $s \geq 0$ to relax (5.29) to

$$\hat{P}^{(0:h-1)} = \arg\min_P \tilde{J}(P) + \sigma \|s\|_1, \tag{5.32a}$$

$$\text{s.t. } \tilde{g}(P) \leq s, \tag{5.32b}$$

where $\sigma \in \mathbb{R}_+$ is a large enough factor such that minimizing the constraint violation, i.e., avoiding $s > 0$, is always prioritized over minimizing $\tilde{J}(P)$ (also see [50]). Assuming (5.29) has a feasible global optimizer $\hat{P}$, i.e., $\tilde{g}\left(\hat{P}\right) \leq 0$, the feasible point $(\hat{P}, 0)$ of (5.32) is the global optimum of (5.32) since $\tilde{J}\left(\hat{P}\right) < \tilde{J}\left(\hat{P}\right) + \sigma \|s\|$ for any $s > 0$ due to σ being chosen large enough, and since (5.29) and (5.32) are identical for $s = 0$.

5.3.5 Closed-Loop Reachable Set

Since we use nonlinear abstractions, which involve quadratic (or even higher-order) maps, for reachability analysis, it makes sense to use polynomial zonotopes for reachability analysis since they are closed under these maps. To compute the closed-loop reachable set, the procedure in Sec. 5.2.5.2 can be used.

5.3.6 Computational Complexity

We again differentiate between the complexity of the offline controller synthesis and the application of said controller online. We only discuss the complexity of the first iteration starting at $t = 0$ since the total number of iterations m_s is fixed. For simplicity, let $\kappa \geq 1$ for the order of the controller template introduced in Sec. 5.3.2 and assume that the number of monomials a of the controller template grows at least with $O(n)$, i.e., $O(n + a) = O(a)$. Additionally, we again make use of Ass. 5.1.

5.3.6.1 Offline Complexity

Subsequently, we derive an upper bound on the computational complexity of the proposed PGSC approach in n. Due to the similarities between GSC and PGSC (also see Sec. 5.3.7), we only present the complexity analysis for parts of the algorithm that are different from GSC.

Controller Template: Without loss of generality, we inspect the complexity for the controller template parameterized in $P^{(0,0)}$ and set –with slight abuse of notation – $P = P^{(0,0)}$ to avoid unnecessary indices. The computation of

$$\bar{u}\left(\beta, P\right) = \tilde{u}\left(\tilde{\beta}\left(\beta\right), P\right) = c_{\bar{\mathcal{U}}} + G_{\bar{\mathcal{U}}}\left(\sum_{k=1}^{a} P_{(:,k)} \tilde{\beta}\left(\beta\right)^{E_{(:,k)}}\right),$$

from Sec. 5.3.2 requires the polynomial substitution of $\tilde{\beta}\left(\beta\right) = G_{\bar{\mathcal{X}}^{(0)}}^{-1}\left(x^{(0)}\left(\beta\right) - c_{\bar{\mathcal{X}}^{(0)}}\right)$. With the controller template $\left\{\tilde{u}\left(\tilde{\beta}, P\right)\right\}_{\tilde{\beta}}$ given by the user with controller order κ, we compute its composition with $\left\{\tilde{\beta}\left(\beta\right)\right\}_{\beta}$ using Prop. A.2.

The computation of $\left\{\tilde{\beta}\left(\beta\right)\right\}_{\beta}$ has complexity $O\left(n_x^{\omega} + n_u^2 l\right) = O\left(n^3\right)$ due the computation of the inverse $G_{\bar{\mathcal{X}}^{(0)}}^{-1}$ and the matrix multiplication [61, Prop. 8], where $\omega \in \mathbb{R}_+$ is the current complexity exponent for matrix multiplication. The complexity of the composition $\{\bar{u}\left(\beta, P\right)\}_{\beta}$ from $\left\{\tilde{u}\left(\tilde{\beta}, P\right)\right\}_{\tilde{\beta}}$ and $\left\{\tilde{\beta}\left(\beta\right)\right\}_{\beta}$ is then $O\left(\left(\kappa\left(n_x + an\right) + an_u\right) l^{\kappa}\right) = O\left(an^{1+\kappa}\right)$ according to Prop. A.2.

Reachable Sets: We compute reachable sets using the conservative polynomialization approach from [5] as described in Sec. 2.7.3. Thus, the computation of the parameterized reachable set is $O\left(n^5 + n^2a^2 + n^3a\log n\right)$ according to Prop. A.1 and the complexity of the closed-loop reachable set computation can be bounded from above by $O\left(n^5\right)$ [59, Sec. 4.1.4].

Controller Computation: The number of objective and constraint evaluations can be polynomially bounded in $\frac{1}{\epsilon}$ for second-order optimization methods [18], where $\epsilon > 0$ is the specified accuracy of a first-order critical point of (5.29); thus, the number of iterations required is independent of n for a given ϵ. As a result, it is sufficient to derive the complexity of one objective and constraint function evaluation. Since we check a fixed number of different constraint types, i.e., state, input, and final state constraints, and the number of constraints for each grows at most with $O\left(n\right)$ by assumption, we limit the analysis here to final state constraints. Thus, let $C_{\tilde{\mathcal{X}}_{\mathrm{f}}}\tilde{\mathcal{R}}\left(t_{\mathrm{f}}, P\right) = \left\{c + \sum_{k=1}^{l} M_{(:,k)} \prod_{o=1}^{an_u} p_o^{E_{(o,k)}} \prod_{o=1}^{m} \beta_o^{B_{(o,k)}}\right\}_{\beta}$ be the final constraint set obtained from the parameterized reachability analysis, where $c \in \mathbb{R}^{o_{\mathcal{X}_{\mathrm{f}}}}$ is the starting point, $M \in \mathbb{R}^{o_{\mathcal{X}_{\mathrm{f}}} \times l}$ is the generator matrix, $\begin{bmatrix} E \\ B \end{bmatrix} \in \mathbb{N}_{\geq 0}^{(an_u+m)\times l}$ is the exponent matrix, $p = P_{(:)}$, $\beta \in [-1, 1]^m$,

and we have $O(o_{\mathcal{X}_\mathrm{f}}) = O(l) = O(m) = O(n)$ due to Ass. 5.1. First, we compute the collected polynomial zonotope $\langle \cdot, N(P), 0, B \rangle_{PZ}$ with

$$N_{(:,j)}(P) = \sum_{k \in \mathcal{K}^{(j)}} M_{(:,k)} \prod_{o=1}^{an_u} p_o^{E_{(o,k)}},$$
$$\mathcal{K}^{(j)} = \left\{ k \in \{1, \ldots, l\} \;\middle|\; \forall o \in \{1, ..., an_u\} : B_{(o,j)} = B_{(o,k)} \right\},$$

for $1 \leq j \leq \tilde{l}$, where $\tilde{l}$ is the number of unique columns in B with $\tilde{l} < l$ and $\mathcal{K}^{(j)}$ contains all generator/exponent vector column indices that have the same exponents in β, and where it further holds that $\sum_{j=1}^{\tilde{l}} \left|\mathcal{K}^{(j)}\right| = l$ by construction. The complexity of computing this collected polynomial zonotope is identical to the complexity of computing all $\mathcal{K}^{(j)}$, which is at most $O\left(an_u \tilde{l} \log \tilde{l}\right) = O(an^2 \log n)$ [59, Prop 2.5.2]. We then compute its zonotope outer approximation $\hat{\mathcal{Z}}(\langle \cdot, N(P), 0, B \rangle_{PZ}) = \langle \cdot, G(P) \rangle_Z$ with complexity $O(n^2)$ [59, Prop. 3.1.4], where $G(P) \in \mathbb{R}^{o_{\mathcal{X}_\mathrm{f}} \times \tilde{l}}$ denotes its generator matrix and where we ignore the center as it can be included as a generator with an all-zero exponent vector. From Sec. 2.3.6.1, it is clear that the complexity of computing and evaluating the objective and constraint functions of the smooth reformulation of (5.29) for final state constraints only will be dominated by evaluating and computing the $O\left(o_{\mathcal{X}_\mathrm{f}} \tilde{l}\right) = O(n_x^2)$ auxiliary constraints since $G(P) \in \mathbb{R}^{o_{\mathcal{X}_\mathrm{f}} \times \tilde{l}}$. Further, the complexity of computing and then evaluating these auxiliary constraints will be dominated by the computation and evaluation of all Hessian matrices of $G_{ij}(P)$.

To evaluate $\nabla^2_{P_{(:)}} G_{ij}(P)$, we form $\mathcal{G}_{(:,j)} = \left\{ G_{(:,j)}(P) \right\}_P = \left\langle \cdot, \bar{G}^{(j)}, 0, \bar{E} \right\rangle_{PZ}$ for $1 \leq j \leq \tilde{l}$, i.e., a polynomial zonotope that represents $G_{(:,j)}(P)$ with $\bar{G}^{(j)} \in \mathbb{R}^{o_{\mathcal{X}_\mathrm{f}} \times \left|\mathcal{K}^{(j)}\right|}$ and $\bar{E}^{(j)} \in \mathbb{N}^{an_u \times \left|\mathcal{K}^{(j)}\right|}$: Any polynomial zonotope $G_{(:,j)}(P)$ has at most $\left|\mathcal{K}^{(j)}\right|$ generators and any of its $\left|\mathcal{K}^{(j)}\right|$ exponent vectors is one of the at most l unique exponent vectors in $E \in \mathbb{N}_{\geq 0}^{an_u \times l}$. Thus, we first compute the Hessian matrices of all l possible monomials; the Hessian matrix of any particular $\nabla^2_{P_{(:)}} G_{ij}(P)$ then follows as a product of the $\left|\mathcal{K}^{(j)}\right|$ corresponding monomial Hessian matrices with the generator values in $\bar{G}^{(j)}_{(i,:)}$.

To evaluate the Hessian matrix of a monomial $m^{(j)}(P) = \prod_{k=1}^{an_u} p_k^{E_{(k,j)}}$ for a fixed $p = P_{(:)}$, we first evaluate all l monomials $m^{(j)}(P)$, which has complexity $O(ln_u a) = O(an^2)$ since its evaluation requires $n_u a$ exponentiations and $n_u a - 1$ multiplications. Given $m^{(j)}(P)$, any second-order derivative can be found as

$$\frac{\partial^2 m^{(j)}(P)}{\partial p_k \partial p_o} = \begin{cases} \frac{E_{(k,j)} E_{(o,j)}}{p_k p_o} m^{(j)}(P), & E_{(k,j)} \geq 1 \wedge E_{(o,j)} \geq 1 \wedge p_k p_o \neq 0, \\ 0, & \text{otherwise} \end{cases}.$$

Since computing any second-order derivative from a given $m^{(j)}(P)$ has complexity $O(1)$ by inspection of the above equation, computing all Hessian matrices of $m^{(j)}(P)$ for a

given P has complexity $O\left(l\left(n_u a\right)^2\right) \overset{Ass.\ 5.1}{=} O\left(a^2 n^3\right)$ since each matrix has $O\left(\left(n_u a\right)^2\right)$ elements.

Computing $\nabla^2_{P_{(:)}} G_{ij}\left(P\right)$ for a given i and j now requires $\left|\mathcal{K}^{(j)}\right|$ scalar multiplications of the generator values in $\bar{G}^{(j)}_{(i,:)}$ with the pre-computed Hessian matrices of the monomials and a subsequent summation of these $\left|\mathcal{K}^{j}\right|$ matrices, which both have complexity $O\left(\left|\mathcal{K}^{(j)}\right|\left(n_u a\right)^2\right)$. Repeating this for all $o_{\mathcal{X}_\mathrm{f}}$ rows and all $\tilde{l}$ columns of $G\left(P\right)$, the overall complexity is bounded from above by $O\left(o_{\mathcal{X}_\mathrm{f}} l\left(n_u a\right)^2\right) \overset{Ass.\ 5.1}{=} O\left(a^2 n^4\right)$ since $\sum_{j=1}^{\tilde{l}} \left|\mathcal{K}^{(j)}\right| = l$.

Overall Complexity: The overall offline complexity of PGSC is thus at most

$$\begin{aligned} O\left(c^{(\mathrm{off})}_{\mathrm{PGSC}}\left(n\right)\right) &= O\left(c_{\mathrm{ref}}\left(n\right) + a n^{1+\kappa} + n^5 + a^2 n^2 + a n^3 \log n + a^2 n^4\right) \\ &= O\left(c_{\mathrm{ref}}\left(n\right) + a n^{1+\kappa} + a^2 n^4\right). \end{aligned} \tag{5.33}$$

5.3.6.2 Online Complexity

For the evaluation of the controller in (5.20), we first compute $\tilde{\beta}$ using one matrix vector multiplication with complexity $O\left(n_x^2\right)$ and then evaluate the control law in (5.19) over $\tilde{\beta}$, which has complexity $O\left(a n_x\right)$ [62, Sec. 3.3], followed by one last matrix-vector multiplication with complexity $O\left(n_u^2\right)$.

The online complexity of deploying PGSC is thus

$$O\left(c^{(\mathrm{on})}_{\mathrm{PGSC}}\left(n\right)\right) = O\left(a n\right). \tag{5.34}$$

5.3.7 Generalization of Generator-Space Control

In this section, we show that the PGSC approach generalizes the GSC approach under mild technical assumptions.

Proposition 5.1 (PGSC as Generalization of GSC)**.** *Let $\kappa = 1$ (linear controller), $\pi = 1$ (linear abstraction), $r = 1$ (number of reachability steps), omit any reduction operations, let $\mathcal{X}^{(0)}$ be given as a non-degenerate parallelotope, and consider only input and final state constraints. Then the PGSC and GSC approach are identical.*

Proof. Comparing (5.3) and (5.21), it is obvious that both controllers are identical for $\kappa = 1$ since $\mathcal{X}^{(0)} = \bar{\mathcal{X}}^{(0)}$ by assumption, and therefore $\tilde{\beta} = \beta$ and $\tilde{u}_{\mathrm{lin}}\left(\tilde{\beta}, P^{(i,j)}\right) = \tilde{u}\left(\tilde{\beta}, P^{(i,j)}\right) = \bar{u}\left(\beta, P^{(i,j)}\right)$ (see (5.18), (5.19), (5.21)) for all $0 \leq i \leq m_\mathrm{s} - 1$ and $0 \leq j \leq m_\mathrm{c} - 1$ if their parameterized reachable sets are identical. Because (5.12) and (5.29) are structurally identical, it suffices to show that both parameterized reachable sets are equal.

Let $\tilde{\mathcal{R}}(t,P) = \{\tilde{r}(t,P,\beta)\}_\beta$ denote the parameterized reachable set where $\tilde{\mathcal{R}}(0,P) = \mathcal{X}^{(0)}$. The extended generating function at step $t = it_\mathrm{s}$ following Sec. 5.3.3 is

$$x_\mathrm{ext}^{(i)}(P,\beta) = \begin{bmatrix} \tilde{r}(it_\mathrm{s},P,\beta) \\ \bar{u}\left(\beta, P^{(i,0)}\right) \end{bmatrix},$$

and the linear abstraction of the extended flow $\dot{x}_\mathrm{ext} = f_\mathrm{ext}(x_\mathrm{ext})$ with $x_\mathrm{ext} = \begin{bmatrix} x^T, & u^T \end{bmatrix}^T$ while ignoring abstraction errors – projected onto the first n_x dimensions – yields

$$\tilde{r}(it_\mathrm{s}+t_\mathrm{c},P,\beta) = \left(e^{A_\mathrm{ext}^{(0)} t_\mathrm{c}} x_\mathrm{ext}^{(i)}(P,\beta) + \int_0^{t_\mathrm{c}} e^{A_\mathrm{ext}^{(0)}\tau}\,\mathrm{d}\tau \begin{bmatrix} c^{(0)} \\ 0 \end{bmatrix} \right)_{(1:n_x)}, \tag{5.35}$$

where

$$A_\mathrm{ext}^{(0)} = \left.\frac{df_\mathrm{ext}(x_\mathrm{ext})}{dx_\mathrm{ext}}\right|_{x_\mathrm{ext}=\bar{x}_\mathrm{ext}^{(0)}} = \begin{bmatrix} A^{(0)}, & B^{(0)} \\ 0, & 0 \end{bmatrix},$$

with $\bar{x}_\mathrm{ext}^{(0)} = \begin{bmatrix} \bar{x}^{(0)^T}, & \bar{u}^{(0)^T} \end{bmatrix}^T$, where $\bar{x}^{(0)}$ and $\bar{u}^{(0)}$ are given as in (5.24) and (5.25), and $A^{(0)}$, $B^{(0)}$, and $c^{(0)}$ are defined as in Sec. 5.2.3. Further

$$\begin{aligned} e^{A_\mathrm{ext}^{(0)} t} &= \sum_{k=0}^{\infty} \frac{\left(A_\mathrm{ext}^{(0)} t\right)^k}{k!} = I_{n_x+n_u} + \sum_{k=1}^{\infty} \frac{\begin{bmatrix} \left(A^{(0)}t\right)^k, & {A^{(0)}}^{k-1} t^k B^{(0)} \\ 0, & 0 \end{bmatrix}}{k!} \\ &= \begin{bmatrix} \sum_{k=0}^{\infty} \frac{\left(A^{(0)}t\right)^k}{k!}, & \sum_{k=1}^{\infty} \frac{{A^{(0)}}^k r^k B^{(0)}}{k!} \\ 0, & I_{n_u} \end{bmatrix} = \begin{bmatrix} e^{A^{(0)}t}, & \int_0^t e^{A^{(0)}\tau}\,\mathrm{d}\tau \\ 0, & I_{n_u} \end{bmatrix}, \end{aligned}$$

which, substituted into (5.35), yields

$$\tilde{r}(it_\mathrm{s}+t_\mathrm{c},P,\beta) = c_d^{(0)} + A_d^{(0)} \tilde{r}(it_\mathrm{s},P,\beta) + B_d^{(0)} \bar{u}\left(\beta, P^{(i,0)}\right), \tag{5.36}$$

where $A_d^{(0)}$, $B_d^{(0)}$, and $c_d^{(0)}$ are defined as in Sec. 5.2.3. Thus, (5.36) is identical to the parameterized reachable set computation in (5.6), and applying the remaining steps iteratively concludes the proof. □

Prop. 5.1 excludes state constraints since GSC only approximately enforces state constraint satisfaction at discrete points in time while PGSC approximately enforces state constraint satisfaction over time intervals.

5.3.8 Experiments

In this section, we compare the PGSC approach to the GSC approach introduced in Sec. 5.2 from [99]. Since manually guessing the adapted state and final state constraints is in general non-trivial, we choose to omit both state and final state constraints here, but show the final state constraints $\mathcal{X}_\mathrm{f} = \{-c_{\mathcal{X}^{(0)}} + x_\mathrm{f}\} \oplus \mathcal{X}^{(0)}$ with $\mathcal{X}^{(0)} = \langle c_{\mathcal{X}^{(0)}}, G_{\mathcal{X}^{(0)}} \rangle_Z$

for reference (helpful when result is used in a maneuver automaton; see Sec. 3.1); we will enforce state constraints on the benchmarks shown here in the next section. Lastly, we do not penalize input energy used here, i.e., we set $\zeta = 0$, for all benchmarks in this section.

To measure the control performance of each approach, we introduce

$$\operatorname{csize}\left(\mathcal{R}\left(t_\mathrm{f}\right)\right) = \left\| \begin{bmatrix} c - x_\mathrm{f} \\ G_{(:)} \end{bmatrix} \right\|_1, \tag{5.37}$$

for a given outer approximation $\mathcal{R}\left(t_\mathrm{f}\right)$ of the closed-loop reachable set with the zonotope outer approximation $\mathcal{Z} = \hat{\mathcal{Z}}\left(\mathcal{R}\left(t_\mathrm{f}\right)\right) = \langle c, G \rangle_Z$: We measure how much $\mathcal{Z}$ differs from the target state $x_\mathrm{f} \in \mathbb{R}^{n_x}$ by measuring both the deviation of its center from x_f and the size of $\mathcal{Z}$ by measuring the absolute sum of its generators.

5.3.8.1 Bicycle (Single Track)

While kinematic models are simpler, they do not consider tire slip and thus cannot model effects such as oversteering and understeering. We introduce a variant of the single-track model[2] – illustrated in Fig. 5.5 – given by

$$\dot{x}_1 = \frac{\mu}{x_4 l}\left(\left(\frac{c_r\left(u_2\right) l_r - c_f\left(u_2\right) l_f}{x_4} - 1\right) x_3 + \left(c_r\left(u_2\right) + c_f\left(u_2\right)\right) x_1 + c_f\left(u_2\right) u_1\right), \tag{5.38a}$$

$$\dot{x}_2 = x_3, \tag{5.38b}$$

$$\dot{x}_3 = -\frac{\mu m}{I_z\left(l_r + l_f\right)}\left(\frac{l_f^2 c_f\left(u_2\right) + l_r^2 c_r\left(u_2\right)}{x_4} x_3 + \left(l_r c_r\left(u_2\right) - l_f c_f\left(u_2\right)\right) x_1 + l_f c_f\left(u_2\right) u_1\right), \tag{5.38c}$$

$$\dot{x}_4 = u_2, \tag{5.38d}$$

$$\dot{x}_5 = x_4 \cos\left(x_1 + x_2\right), \tag{5.38e}$$

$$\dot{x}_6 = x_4 \sin\left(x_1 + x_2\right), \tag{5.38f}$$

with

$$\begin{aligned} c_f\left(u_2\right) &= C_{S,f}\left(g l_r - u_2 h_{cg}\right), \\ c_r\left(u_2\right) &= C_{S,r}\left(g l_f + u_2 h_{cg}\right), \\ l &= l_r + l_f, \end{aligned}$$

and where $x = \begin{bmatrix} \beta, & \Psi, & \dot{\Psi}, & v, & s_x, & s_y \end{bmatrix}^T$ is the state, β is the slip angle at the vehicle center, Ψ and $\dot{\Psi}$ are the yaw angle and yaw rate, v is the longitudinal velocity of the vehicle, $\begin{bmatrix} s_x, & s_y \end{bmatrix}^T$ are the two spatial coordinates, u_1 is the steering angle of the front

[2] Sec. 7 (https://gitlab.lrz.de/tum-cps/commonroad-vehicle-models/-/blob/master/vehicleModels_commonRoad.pdf)

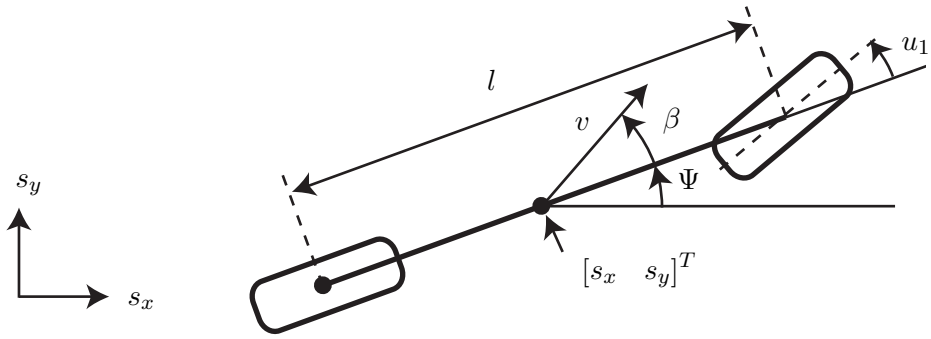


Figure 5.5: States and inputs of the single-track model[4]. Shown are the slip angle β, the yaw angle Ψ, longitudinal velocity v, spatial coordinates $\begin{bmatrix} s_x & s_y \end{bmatrix}^T$, the steering angle u_1, and the wheelbase l.

wheels, and u_2 is the longitudinal acceleration of the vehicle. The benchmark parameters are listed in Tab. 5.1 and have been chosen appropriately[3].

We compute a feedforward controller for the undisturbed system, i.e., we set $m_\mathrm{s} = 1$ and $\mathcal{W} = \{0\}$. Fig. 5.6 shows a comparison between the performance of the GSC approach and the PGSC approach, where computations times were 27.5 s and 36.9 s, respectively, and $\mathrm{csize}\left(\mathcal{R}_\mathrm{GSC}\left(t_\mathrm{f}\right)\right) = 0.92$ and $\mathrm{csize}\left(\mathcal{R}_\mathrm{PGSC}\left(t_\mathrm{f}\right)\right) = 0.65$. Clearly, the PGSC approach outperforms the GSC approach significantly as it reduces the overall cost significantly compared to GSC: While both approaches keep the yaw rate and velocity small (see Fig. 5.6, $\begin{bmatrix} x_3, & x_4 \end{bmatrix}$) and fail to keep the slip angle small (see Fig. 5.6, x_1), the PGSC approach steers the set of initial positions more closely to the target state in the two spatial dimensions ($\begin{bmatrix} x_5, & x_6 \end{bmatrix}$).

5.3.8.2 Controlled Van-der-Pol Oscillator

We present a controlled version of the Van-der-Pol oscillator benchmark [9, Sec. VII]

$$\dot{x}_1 = x_2, \tag{5.39a}$$

$$\dot{x}_2 = \left(1 - x_1^2\right) x_2 - x_1 + u + w. \tag{5.39b}$$

We compute a piecewise constant, discrete-time feedback controller with $m_\mathrm{s} = 2$ and $m_\mathrm{c} = 4$, where all parameters for the benchmark are listed in Tab. 5.2.

Fig. 5.7 shows the reachable set using the GSC approach and the PGSC approach, where the computation took 15.50 s and 20.44 s, respectively. Clearly, PGSC achieves a better control result as the final reachable set is contained within the final reachable

[3]Tab. 4, vehicle identifier 1 (`https://gitlab.lrz.de/tum-cps/commonroad-vehicle-models/-/blob/master/vehicleModels_commonRoad.pdf`)

[4]Fig. 4, vehicle identifier 1 (`https://gitlab.lrz.de/tum-cps/commonroad-vehicle-models/-/blob/master/vehicleModels_commonRoad.pdf`)

Table 5.1: Benchmark parameters for the bicycle model.

Parameter	Value
m_s	1
m_c	6
$\mathcal{X}^{(0)}$	$\left\langle c_{\mathcal{X}^{(0)}}, \mathrm{diag}\left(\begin{bmatrix}0.2, & 0.02, & 0.2, & 0.2, & 0.2, & 0.2\end{bmatrix}\right)\right\rangle_Z$
$c_{\mathcal{X}^{(0)}}$	$\begin{bmatrix}0, & 0, & 0, & 20, & 0, & 0\end{bmatrix}^T$
$\mathcal{U}$	$\left\langle 0, \mathrm{diag}\left(\begin{bmatrix}0.4, & 9.81\end{bmatrix}\right)\right\rangle_Z$
$\mathcal{W}$	$\{0\}$
t_f	1 s
x_f	$\begin{bmatrix}0, & 0, & 0, & 20, & 20, & 0\end{bmatrix}^T$
κ	1

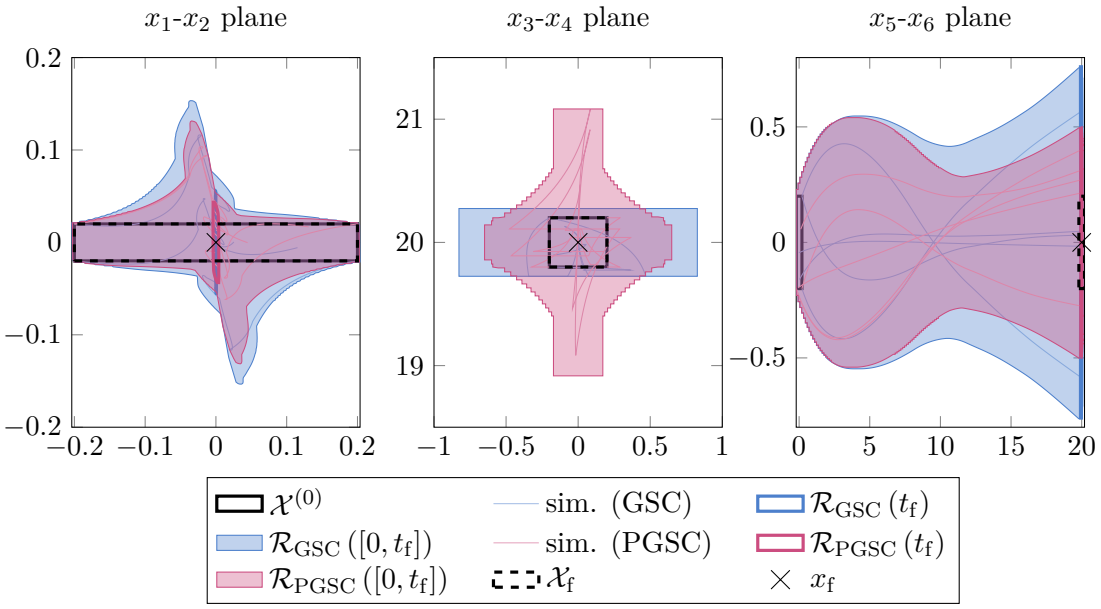


Figure 5.6: Comparison of the GSC approach and the PGSC approach for the single-track (bicycle) model. For a fair comparison, both approaches use a linear controller template.

Table 5.2: Benchmark parameters for the Van-der-Pol model.

Parameter	Value
m_s	2
m_c	4
$\mathcal{X}^{(0)}$	$\left\langle \begin{bmatrix} 1.4, & 2.4 \end{bmatrix}^T, \mathrm{diag}\left(\begin{bmatrix} 0.6, & 0.6 \end{bmatrix}\right) \right\rangle_Z$
$\mathcal{U}$	$\langle 0, 2 \rangle_Z$
t_f	1 s
x_f	$\begin{bmatrix} 1.93, & -0.47 \end{bmatrix}^T$
κ	1

set computed by GSC. The combined deviations from the target state and size of final reachable set are $\mathrm{csize}\left(\hat{\mathcal{Z}}\left(\mathcal{R}_\mathrm{GSC}\left(t_\mathrm{f}\right)\right)\right) = 1.22$ and $\mathrm{csize}\left(\mathcal{R}_\mathrm{PGSC,1}\left(t_\mathrm{f}\right)\right) = 1.09$ (see (5.37)).

Further, Fig. 5.8 shows the improved performance when using a quadratic controller ($\kappa = 2$) over the same linear controller ($\kappa = 1$) for this benchmark using PGSC, where the computation of the quadratic controller took 18.52 s. Here, we have $\mathrm{csize}\left(\mathcal{R}_\mathrm{PGSC,2}\left(t_\mathrm{f}\right)\right) = 0.79$.

5.3.9 Discussion

The numerical experiments in Sec. 5.3.8 demonstrate improved performance of PGSC over GSC, both by using a more accurate parameterized reachable set and thus achieving a better control performance for the same linear controller template as well as by using, e.g., a quadratic controller over a linear one.

While both approaches work reasonably well for input constraints, the fact that they both only approximate the parameterized reachable set during the optimization of the controller – without any regard for the accuracy of this approximation – means that state and final state constraints can seldom be enforced as is; rather, they require the user to specify adapted state and final state constraints. Since this can become quite involved for non-experts, it is a major downside of both approaches. In the next section, we thus propose a novel synthesis algorithm that does not require the manual specification of these adapted state constraints.

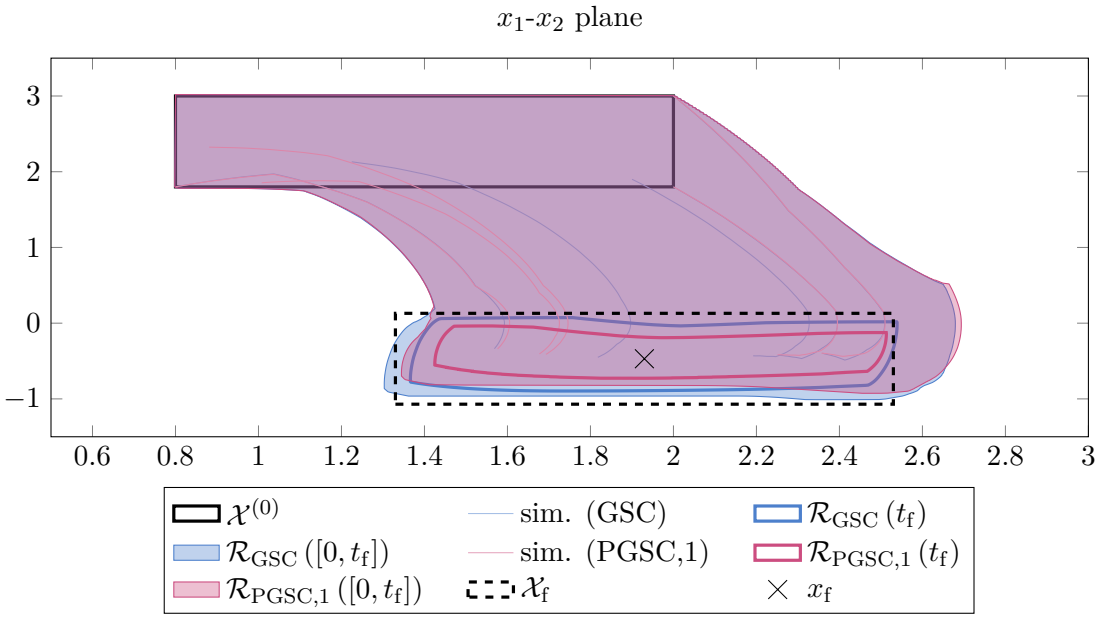


Figure 5.7: Comparison of the GSC approach and the PGSC approach for the Van-der-Pol benchmark. For a fair comparison, both approaches use a linear controller template.

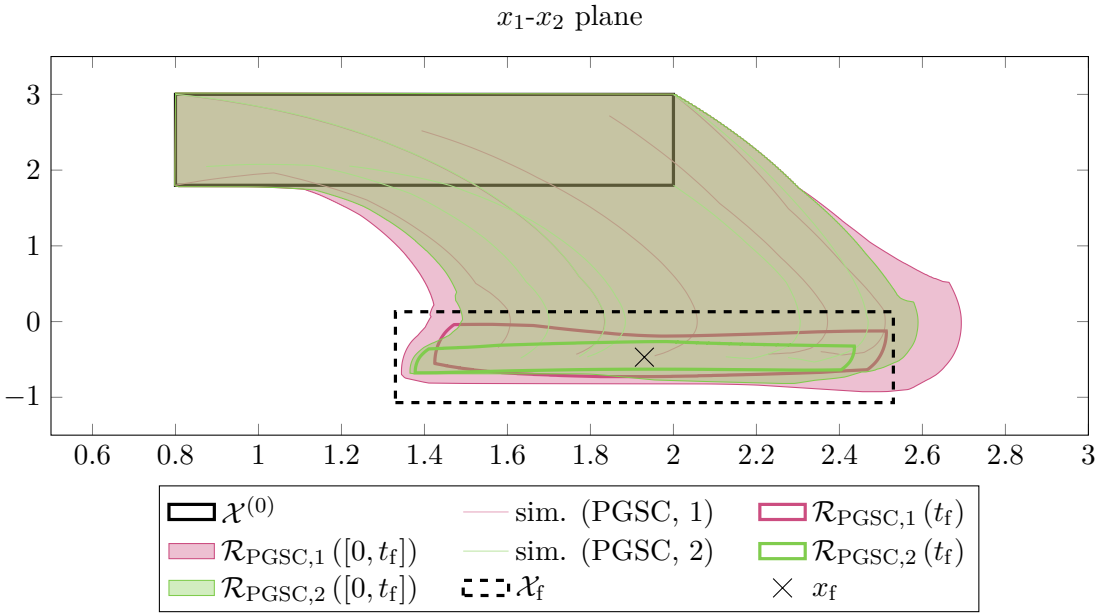


Figure 5.8: Comparison of a linear and quadratic controller template using the PGSC approach for the Van-der-Pol benchmark.

5.4 Iterative Polynomial Generator-Space Control

While the GSC approach used only linear abstractions for the computation of the parameterized reachable set and was limited to linear control laws, the PGSC approach extended this concept to polynomial abstraction orders for reachability analysis and polynomial control laws. While the reachable set can in principle be arbitrarily closely approximated using the conservative polynomialization approach by choosing the abstraction order large enough, in practice, we are often limited to at most quadratic abstractions due to extended computation times or the fact that higher-order abstractions generate even more generators during reachability analysis, which in turn increases the number of generators needing to be reduced; this effect can mostly or even completely negate the benefit of a higher abstraction order to begin with.

However, we still find it necessary to better approximate the parameterized reachable set for strongly nonlinear system dynamics. Therefore, we introduce the iterative polynomial generator-space control (iPGSC) approach, an iterative trust-region approach for the synthesis of formally verified controllers under input and state constraints according to (5.1), which is part of our work in [35]. Since (5.1) is generally not efficiently solvable, we approximate it using a non-differentiable cost function in the controller parameters. By locally approximating this non-differentiable cost function with a differentiable optimization problem, we can iteratively update our controller parameters based on this local approximation, where we control its accuracy by restricting the admissible controller parameters for the optimization problem to a bounded trust-region. In contrast to GSC and PGSC, iPGSC does not require manual tuning of the state constraints. Alg. 3 describes the general solution procedure, where the most important steps are briefly summarized subsequently:

(i) We compute the reference trajectory at the start of each iteration, which yields the intermediate target states necessary for the optimization of the controller parameters (Alg. 3, l. 7).

(ii) We compute an initial guess for the controller parameters using GSC (Alg. 3, l. 8).

(iii) We compute the closed-loop reachable set of the controller, parameterized by the initial guess $\bar{P}$ (Alg. 4, l. 3) and evaluate its initial cost $J\left(\bar{P}\right)$ (Alg. 4, l. 4). The algorithm then iteratively executes the following steps:

  (a) Since the proposed cost is non-differentiable and its evaluation requires the computation of the closed-loop reachable set, we first compute the parameterized reachable set locally for a region of controller parameters $\mathcal{P}_\gamma\left(\bar{P}\right)$ around the current controller parameters $\bar{P}$, whose radius is bounded by the trust-region radius $\gamma \in (0, 1]$ (Alg. 4, l. 6). To form this approximation, we use the available outer approximation of the closed-loop reachable set of the current controller parameterized by $\bar{P}$.

  (b) Using this local parameterized reachable set, we approximate the cost function with a differentiable optimization problem, where the accuracy of this

trust-region subproblem is controlled by the trust-region radius γ (Alg. 4, l. 7). Solving the trust-region subproblem, we obtain the optimal controller parameters $\hat{P}$. Because we use the local parameterized reachable set for the trust-region subproblem, we avoid the recomputation of the closed-loop reachable set in each optimization iteration.

(c) We then evaluate the cost of the new controller parameterized in $\hat{P}$ using an outer approximation of its closed-loop reachable set (Alg. 4, l. 8–9).

(d) By comparing the actual decrease in cost against the decrease in cost predicted by the trust-region subproblem, we adapt the trust-region radius to ensure an accurate approximation of the parameterized reachable set in the next iteration (Alg. 4, l. 10): If the ratio is close enough to one, the approximated trust-region subproblem is accurate and we enlarge γ to allow for faster progress. Otherwise, we shrink the trust-region radius to ensure an accurate approximation in the next iteration.

(e) We then compare the cost of the current controller to the cost of the newly computed controller: If the cost decreased, we accept the step and save these new controller parameters (Alg. 4, l. 11–19). If the algorithm converged (Alg. 4, l. 11), we return the current controller parameters.

(iv) Once the trust-region algorithm converges, we have the optimal controller parameters as well as the corresponding closed-loop reachable set available. If we have not yet reached the final iteration, we again start at (i). Otherwise, the algorithm terminates.

Algorithm 3 Iterative polynomial generator-space control

1: **function** ITERPOLYGENSPACECONTROL($\mathcal{X}^{(0)}, \mathcal{U}, \mathcal{X}, \mathcal{X}_\mathrm{f}, \mathcal{W}, m_\mathrm{s}, m_\mathrm{c}, x_\mathrm{f}, h, \kappa$)
2: $\quad\bar{\mathcal{R}} = \mathcal{X}^{(0)}$
3: $\quad$**for** $i = 0; i < m_\mathrm{s}; i++$ **do**
4: $\quad\quad h = \min(h, m_\mathrm{s} - i)$
5: $\quad\quad\bar{\mathcal{R}} = \bar{\mathcal{R}} \downarrow_o$ ▷ $o \in \mathbb{N}_+$ some sensible order
6: $\quad\quad\bar{\mathcal{X}} = \hat{\mathcal{Z}}\left(\bar{\mathcal{R}}\right) \downarrow_1$
7: $\quad\quad x_\mathrm{f}^{(\cdot)} = \mathrm{REFERENCETRAJECTORY}(\bar{\mathcal{X}}, \mathcal{U}, \mathcal{X}, \mathcal{X}_\mathrm{f}, m_\mathrm{s} - i, m_\mathrm{c}, x_\mathrm{f}, t_\mathrm{f})$ ▷ Sec. 5.3.1
8: $\quad\quad\bar{P}^{(\cdot)} = \mathrm{INITGUESS}(\bar{\mathcal{R}}, \mathcal{U}, \mathcal{X}, \mathcal{X}_\mathrm{f}, \mathcal{W}, m_\mathrm{s} - i, m_\mathrm{c}, x_\mathrm{f}^{(\cdot)}, h, t_\mathrm{f}, \kappa)$ ▷ e.g. Sec. 5.2
9: $\quad\quad\left[\hat{P}^{(i)}, \mathcal{R}([i, i+1]\, t_\mathrm{s})\right] = \mathrm{COMPUTECTRL}(\bar{P}^{(\cdot)}, \bar{\mathcal{R}}, \mathcal{U}, \mathcal{X}, \mathcal{X}_\mathrm{f}, x_\mathrm{f}, u_\mathrm{ref}, \kappa)$ ▷ Alg. 4
10: $\quad\quad\bar{\mathcal{R}} = \mathcal{R}((i+1)\, t_\mathrm{s})$
11: $\quad$**end for**
12: $\quad$**return** $\hat{P} = \left[{\hat{P}^{(0)}}^T, \quad \ldots, \quad {\hat{P}^{(m_\mathrm{s}-1)}}^T\right], \mathcal{R}([0, t_\mathrm{f}])$
13: **end function**

The following example illustrates the solution procedure, which is visualized in Fig. 5.9, where we set $h = m_\mathrm{s} = m_\mathrm{c} = 1$ and only consider input constraints for illustrative purposes.

Algorithm 4 Trust-region controller computation

function COMPUTECTRL($\bar{P}^{(\cdot)}, \bar{\mathcal{X}}, h, x_{\mathrm{f}}^{(\cdot)}$)
  Given: μ, σ, $\gamma = 1$
  $\bar{\mathcal{R}} = \mathrm{REACH}(\bar{\mathcal{X}}, \bar{P}^{(\cdot)}, ht_{\mathrm{s}})$ ▷ Sec. 5.2.5.2
  $\bar{J} = \mathrm{COST}(\bar{\mathcal{R}})$ ▷ Sec. 5.4.1
  for $k = 1; k \leq l_{\max}; k++$ **do**
    $\left[\tilde{\mathcal{R}}, \quad \gamma\right] = \mathrm{PARAMREACH}(\bar{P}^{(\cdot)}, \gamma, \bar{\mathcal{R}})$ ▷ Sec. 5.4.2
    $\hat{P}^{(\cdot)} = \mathrm{SOLVETRSUBPROBLEM}(\tilde{\mathcal{R}}, \bar{P}^{(\cdot)}, \gamma, \bar{\mathcal{R}}, x_{\mathrm{f}}^{(\cdot)})$ ▷ Sec. 5.4.3
    $\hat{\mathcal{R}} = \mathrm{REACH}(\bar{\mathcal{X}}, \hat{P}^{(\cdot)}, ht_{\mathrm{s}})$ ▷ Sec. 5.3.5
    $\hat{J} = \mathrm{COST}(\hat{\mathcal{R}})$ ▷ Sec. 5.4.1
    $\gamma = \mathrm{ADAPTRADIUS}(\hat{J}, \tilde{\mathcal{R}}, \hat{P})$ ▷ Sec. 5.4.4
    if $\hat{J} < \bar{J}$ **then**
      $\nu = \mathrm{tol}\left(\hat{J}, \bar{J}\right)$ ▷ Def. 2.1
      $\bar{J} = \hat{J}$
      $\bar{\mathcal{R}} = \hat{\mathcal{R}}$
      $\bar{P}^{(\cdot)} = \hat{P}^{(\cdot)}$
      if $\nu \leq \mu$ **then**
        break
      end if
    end if
  end for
  return $\left[\bar{P}^{(0)}, \quad \bar{\mathcal{R}}\right]$
end function

Example 5.4 (Solve progress for a 1D system [35, Ex. 1, adapted])**.** Let the final, exact parameterized reachable set at $t = t_\mathrm{f}$ for a 1-dimensional system be given by

$$\mathcal{R}\left(t_\mathrm{f}, P\right) = \left\{-1 + 2p_1^2 + 4p_0^3 + \left(3 + 10p_0p_1^3\right)\beta + \left(2 - 2p_1\right)\beta^2\right\}_\beta,$$

where $\beta \in [-1, 1]$ is the dependent factor of the corresponding initial set, $\bar{u}\left(\beta, P\right) = p_0 + p_1\beta$ with $P = P^{(0)} = P^{(0,0)} = \begin{bmatrix} p_0, & p_1 \end{bmatrix}$ is the controller template as given in Sec. 5.3.2, $x_\mathrm{f} = 0$, and $\mathcal{U} = [-1, 1] = \left\langle \begin{bmatrix} 1, & -1 \end{bmatrix}^T, \begin{bmatrix} 1, & 1 \end{bmatrix}^T \right\rangle_H = \langle C_\mathcal{U}, d_\mathcal{U} \rangle_H$. Using Prop. 4.4, the zonotope outer approximation of the reachable set required to form (5.42) is given by

$$\begin{aligned}
\hat{\mathcal{Z}}\left(\mathcal{R}\left(t_\mathrm{f}, P\right)\right) &= \left\{-1 + 2p_1^2 + 4p_0^3 + \left(3 + 10p_0p_1^3\right)\nu_1 + \left(2 - 2p_1\right)\left(\frac{1}{2} + \frac{1}{2}\nu_2\right)\right\}_\nu \\
&= \left\{-p_1 + 2p_1^2 + 4p_0^3 + \left(3 + 10p_0p_1^3\right)\nu_1 + \left(1 - p_1\right)\nu_2\right\}_\nu \\
&= \left\langle -p_1 + 2p_1^2 + 4p_0^3, \begin{bmatrix} 3 + 10p_0p_1^3, & 1 - p_1 \end{bmatrix} \right\rangle_Z \\
&= \left\langle c\left(t_\mathrm{f}, P\right), G\left(t_\mathrm{f}, P\right) \right\rangle_Z.
\end{aligned}$$

Since we only consider input constraints and because the controller template $\bar{u}\left(P, \beta\right)$ is linear in β, we have $\hat{\mathcal{Z}}\left(\left\{\bar{u}\left(\beta, P\right)\right\}_\beta\right) = \langle p_0, p_1 \rangle_Z$ and thus

$$\rho_u^{(\mathcal{Z})}\left(C_\mathcal{U}, [0, t_\mathrm{f}], P\right) - d_\mathcal{U} = \rho_u^{(\mathcal{Z})}\left(\begin{bmatrix} 1 \\ -1 \end{bmatrix}, [0, t_\mathrm{f}], P\right) - \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} p_0 + |p|_1 - 1 \\ -p_0 + |p|_1 - 1 \end{bmatrix},$$

so that $g\left(P\right)$ in (5.41) reduces to

$$g\left(P\right) = |p|_0 + |p|_1 - 1.$$

Thus, the cost function is given by

$$J\left(P\right) = \left|-p_1 + 2p_1^2 + 4p_0^3\right| + \left|3 + 10p_0p_1^3\right| + \left|1 - p_1\right| + 100\max\left(0, |p_0| + |p_1| - 1\right),$$

for $\sigma = 100$. Fig. 5.9 visualizes the contour lines of $J\left(P\right)$ and its approximation $\tilde{J}\left(P\right)$ over the iterations.

Here, we assume that an approximation of the exact reachable set is available as its quadratic Taylor expansion with the initial trust-region radius $\gamma = \frac{1}{4}$; in practice, the exact reachable set is not available and we thus instead compute an approximation to it (see Sec. 5.4.2). For the initial controller parameter guess $\bar{P} = 0$, the generating function of the parameterized, time-point reachable set for $P \in \mathcal{P}_{\frac{1}{4}}\left(0\right) = \left[-\frac{1}{2}, \frac{1}{2}\right]^{1\times 2}$ (see (5.45))

then is

$$\begin{aligned}\tilde{\mathcal{R}}\left(t_\mathrm{f}, P\right) = \Bigg\{ & \left(-1 + 2\bar{p}_1^2 + 4\bar{p}_0^3 + \left(3 + 10\bar{p}_0\bar{p}_1^3\right)\beta + \left(2 - 2\bar{p}_1\right)\beta^2\right)\Big|_{\bar{P}=0} \\ & + \left(\left[12\bar{p}_0^2 + 10\bar{p}_1^3\beta, \quad 4\bar{p}_1 + 30\bar{p}_0\bar{p}_1^2\beta - 2\beta^2\right]\left(P - \bar{P}\right)\right)\Big|_{\bar{P}=0} \\ & + \frac{1}{2}\left(\left(P - \bar{P}\right)^T \begin{bmatrix} 24\bar{p}_0, & 30\bar{p}_1^2\beta \\ 30\bar{p}_1^2\beta, & 4 + 60\bar{p}_0\bar{p}_1\beta \end{bmatrix}\left(P - \bar{P}\right)\right)\Big|_{\bar{P}=0} \Bigg\}_\beta \\ = \ & \left\{-1 + 3\beta + 2\beta^2 - 2p_1\beta^2 + 2p_1^2\right\}_\beta,\end{aligned}$$

and thus

$$\hat{\mathcal{Z}}\left(\tilde{\mathcal{R}}\left(t_\mathrm{f}, P\right)\right) = \left\langle -p_1 + 2p_1^2, \begin{bmatrix} 3, & 1 - p_1 \end{bmatrix}\right\rangle_Z.$$

The approximated cost function for $P \in \left[-\frac{1}{2}, \frac{1}{2}\right]^{1\times 2}$ for the first step thus is

$$\tilde{J}\left(P\right) = \left|-p_1 + 2p_1^2\right| + \left|1 - p_1\right| + 3 + 100\max\left(0, \left|p_0\right| + \left|p_1\right| - 1\right).$$

For the first iteration, $\tilde{J}\left(P\right)$ clearly does not approximate the exact cost function $J\left(P\right)$ particularly well, even though $J\left(\hat{P}\right) < J\left(\bar{P}\right)$, where $\bar{P}$ and $\hat{P}$ are the initial and optimized controller parameters, respectively (see Fig. 5.9a). Thus, we accept the step but shrink the trust-region radius γ to ensure more accurate approximations in the next steps (see Figs. 5.9b and 5.9c) in order to eventually arrive at the minimum (see Fig. 5.9d). ■

The remainder of this section is structured as follows: We first describe how we approximate (5.1) using a cost function in Sec. 5.4.1 and then derive the computation of the parameterized reachable set in Sec. 5.4.2. To avoid repeated reachable set computations for the cost of the controller and its non-differentiability, we derive the trust-region subproblem – a differentiable optimization problem – as an approximation to this cost in Sec. 5.4.3, whose solution then yields a new controller candidate. To ensure that the trust-region subproblem accurately approximates the cost of the newly computed controller candidate, we describe the tuning of the trust-region radius in Sec. 5.4.4 such that the trust-region subproblem approximates the controller cost arbitrarily closely. We discuss the offline and online complexity of the iPGSC approach in Sec. 5.4.5, demonstrate its applicability using numerical experiments in Sec. 5.4.6, and finally discuss the algorithm in Sec. 5.4.7.

5.4.1 Cost of the Controller

Since we generally cannot solve (5.1) efficiently, we instead propose a cost function as a tractable approximation of (5.1). Inspection of (5.1) yields the following observations: We want to minimize the size of the reachable set, centered at the target state, but also minimize the control effort. Further, input, state, and final state constraints must

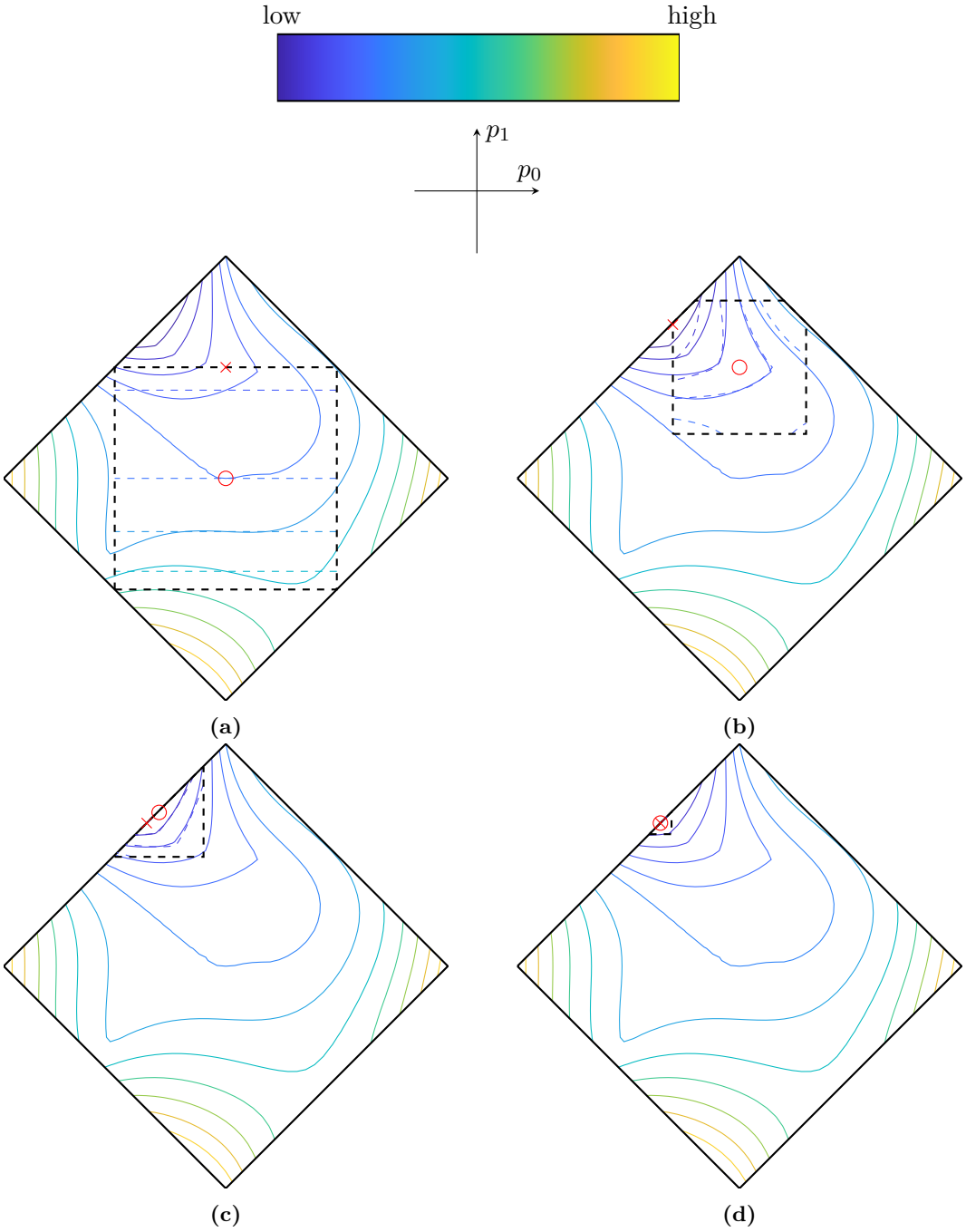


Figure 5.9: Visualization of the iterative controller computation for Ex. 5.4 from [35]. Shown is the set of feasible parameters $\mathcal{P}$ (outer black diamond), the contour lines of the exact cost $J(P)$ (solid) and the current approximation of the cost $\tilde{J}(P)$ (dashed) based on the approximated, parameterized reachable set for $P \in \mathcal{P}_\gamma(\bar{P})$ (dashed black box), the current initial guess of the controller parameters $\bar{P}$ (red circle), and the optimizer $\hat{P} \in \mathcal{P}_\gamma(\bar{P})$ for the current approximated optimization problem (red x).

not be violated. In this section, we thus first transform the objective function and the constraints of (5.1) separately to tractable expressions and then use both to define the cost as an approximation to (5.1). To that end, we first require the following definitions.

Let

$$\hat{\mathcal{Z}}\left(\mathcal{R}\left(t,P\right)\right) = \left\langle c\left(t,P\right), G\left(t,P\right)\right\rangle_Z,$$
$$\hat{\mathcal{Z}}\left(\left\{\bar{u}\left(\beta, P^{(i,j)}\right)\right\}_\beta\right) = \left\langle c_u\left(P^{(i,j)}\right), G_u\left(P^{(i,j)}\right)\right\rangle_Z,$$

where $\mathcal{R}\left(t,P\right)$ is an outer approximation of the closed-loop reachable set for the controller $\bar{u}\left(\beta, P^{(i,j)}\right)$ with $0 \leq i \leq m_\mathrm{s}-1$ and $0 \leq j \leq m_\mathrm{c}-1$. We define the number of reachability steps $m_r \in \mathbb{N}_+$ for the computation of $\mathcal{R}\left(t,P\right)$ in each time interval $\left[k,k+1\right]\delta$ for $0 \leq k \leq m_\mathrm{s}m_\mathrm{c}m_r-1$. Lastly, we define the support functions over any arbitrary time interval $\left[\underline{t},\bar{t}\right]$ as

$$\rho_x^{(\mathcal{Z})}\left(\cdot,\left[\underline{t},\bar{t}\right],P\right) = \max_{k\in\mathcal{I}_{\left[\underline{t},\bar{t}\right]}^{(m_\mathrm{s}m_\mathrm{c}m_r)}} \rho_x^{(\mathcal{Z})}\left(\cdot,\left[k,k+1\right]\frac{t_\mathrm{c}}{m_r},P\right),$$
$$\rho_u^{(\mathcal{Z})}\left(\cdot,\left[\underline{t},\bar{t}\right],P\right) = \max_{k\in\mathcal{I}_{\left[\underline{t},\bar{t}\right]}^{(m_\mathrm{s}m_\mathrm{c})}} \rho_u^{(\mathcal{Z})}\left(\cdot,\left[k,k+1\right]t_\mathrm{c},P\right),$$

with

$$\mathcal{I}_{\left[\underline{t},\bar{t}\right]}^{(n)} = \left\{k\in\left\{0,...,n-1\right\} \;\middle|\; \left[k,k+1\right]\frac{t_\mathrm{f}}{n} \cap \left[\underline{t},\bar{t}\right] \neq \emptyset\right\},$$

where $\rho_x^{(\mathcal{Z})}\left(\cdot,\left[k,k+1\right]\frac{t_\mathrm{c}}{m_r},P\right)$ denotes the support function of $\hat{\mathcal{Z}}\left(\mathcal{R}\left(\left[k,k+1\right]t_\mathrm{c},P\right)\right)$, $\rho_u^{(\mathcal{Z})}\left(\cdot,\left[k,k+1\right]\Delta,P\right)$ denotes the support function of $\hat{\mathcal{Z}}\left(\left\{\bar{u}\left(\beta,P^{(i,j)}\right)\right\}_\beta\right)$ for $k = im_\mathrm{c}+j$ with $0 \leq i \leq m_\mathrm{s}-1$ and $0 \leq j \leq m_\mathrm{c}-1$, and we have $0 \leq \underline{t} \leq \bar{t} \leq t_\mathrm{f}$ for $\underline{t} \in \mathbb{R}_{\geq 0}$ and $\bar{t} \in \mathbb{R}_{\geq 0}$.

Objective: We bound the objective in (5.1a) from above – while considering the extended optimization horizon – by

$$\max_{x(t,P)\in\mathcal{R}(t,P)} \left\{\sum_{k=1}^{h}\left(\left\|x\left(kt_\mathrm{s},P\right)-x_\mathrm{f}^{(k)}\right\|_1 + \zeta\sum_{j=0}^{m_\mathrm{c}-1}\left\|u\left(x\left(kt_\mathrm{s},P\right),P^{(k-1,j)}\right)\right\|_1\right)\right\}$$
$$\leq \sum_{k=1}^{h}\vartheta_k\left(\left\|\begin{bmatrix}c\left(kt_\mathrm{s},P\right)-x_\mathrm{f}^{(k)}\\ \left[G\left(kt_\mathrm{s},P\right)\right]_{(:)}\end{bmatrix}\right\|_1 + \zeta\sum_{j=0}^{m_\mathrm{c}-1}\left\|\begin{bmatrix}c_u\left(P^{(k-1,j)}\right)\\ \left[G_u\left(P^{(k-1,j)}\right)\right]_{(:)}\end{bmatrix}\right\|_1\right), \tag{5.40}$$

where $\vartheta \in \mathbb{R}_{\geq 0}^h$ weighs the contribution of the extended horizon. The inequality follows from a zonotope outer approximation of the reachable set and the input set as well as multiple applications of the triangle inequality.

Constraints: We can conservatively transform the constraints (5.1b) to (5.1d) – again considering the extended optimization horizon – to

$$g(P) = \begin{bmatrix} \rho_u^{(\mathcal{Z})}(C_{\mathcal{U}}, [0, ht_\mathrm{s}], P) - d_{\mathcal{U}} \\ \rho_x^{(\mathcal{Z})}(C_{\mathcal{X}}, [0, ht_\mathrm{s}], P) - d_{\mathcal{X}} \\ \rho_x^{(\mathcal{Z})}(C_{\mathcal{X}_\mathrm{f}}, ht_\mathrm{s}, P) - d_{\mathcal{X}_\mathrm{f}} \end{bmatrix} \leq 0, \tag{5.41}$$

since $\mathcal{U} = \langle C_{\mathcal{U}}, d_{\mathcal{U}} \rangle_H$, $\mathcal{X} = \langle C_{\mathcal{X}}, d_{\mathcal{X}} \rangle_H$, and $\mathcal{X}_\mathrm{f} = \langle C_{\mathcal{X}_\mathrm{f}}, d_{\mathcal{X}_\mathrm{f}} \rangle_H$ are H-polytopes and we can hence apply Prop. 4.6.

Cost: Using (5.40) and (5.41), we propose the cost function

$$\begin{aligned} J(P) = \sum_{k=1}^{h} \vartheta_k \left(\left\| \begin{bmatrix} c(kt_\mathrm{s}, P) - x_\mathrm{f}^{(k)} \\ [G(kt_\mathrm{s}, P)]_{(:)} \end{bmatrix} \right\|_1 + \zeta \sum_{j=0}^{m_\mathrm{c}-1} \left\| \begin{bmatrix} c_u\left(P^{(k-1,j)}\right) \\ \left[G_u\left(P^{(k-15,j)}\right)\right]_{(:)} \end{bmatrix} \right\|_1 \right) \\ + \sigma \left\| \max(0, g(P)) \right\|_1, \end{aligned} \tag{5.42}$$

where $\sigma \in \mathbb{R}_+$: The first term in (5.42) penalizes the deviation of the center from the corresponding target state and the size of the reachable set by penalizing each generator of the reachable set, whereas the second term penalizes the size of the applied input. The third term then penalizes all constraint violations, where σ is an exact penalty multiplier and is chosen such that feasibility is always prioritized over minimizing the first and second term (see [50]).

5.4.2 Parameterized Reachable Set

In Sec. 2.7.3, we remarked that it is in principle possible to represent the exact reachable set arbitrarily closely if the abstraction order and the number of steps during reachability analysis are both large enough. For PGSC, we used this fact as a motivation to compute the parameterized reachable set via the abstraction described in Sec. 2.7.3, where we ignored the abstraction error term since it vanishes for a large enough abstraction order and number of steps for reachability analysis. In practice, however, we are often limited to a quadratic abstraction for the computation of the parameterized reachable set due to the increasing computational effort for a large abstraction order and a large number of time steps during reachability analysis. That said, we would still like to compute the parameterized reachable set accurately. Since computing the reachable set exactly is hard in general [68], it is unreasonable to expect that the parameterized reachable set can become accurate with respect to the exact reachable set. In this section, we thus derive an approximation to the parameterized reachable set which is accurate with respect to a tight enough outer approximation of the exact reachable set. For simplicity, we assume without loss of generality that we start from $t = 0$, and denote $P = P^{(0:h-1)}$ with slight abuse of notation.

We want to find an approximation $\tilde{\mathcal{R}}(t, P)$ that does not need to be recomputed for each P and is ϵ-accurate with respect to a tight outer approximation $\mathcal{R}(t, P)$ of the

reachable set for $\epsilon > 0$, i.e.

$$\forall P \in [-1,1]^{hm_c n_u \times a} \, \forall t \in [0, ht_s] : d_H \left(\mathcal{R}(t,P), \tilde{\mathcal{R}}(t,P) \right) \leq \epsilon, \tag{5.43}$$

where $d_H(\cdot,\cdot)$ denotes the Hausdorff distance between two sets (see Def. 2.16). Because both $\mathcal{R}(t,P)$ and $\tilde{\mathcal{R}}(t,P)$ are reachable sets represented by polynomial zonotopes, they are always closed and thus $d_H \left(\mathcal{R}(t,P), \tilde{\mathcal{R}}(t,P) \right) = 0 \iff \mathcal{R}(t,P) = \mathcal{D}(t,P)$ [103, Chap. 2]. Thus, for a given accuracy ϵ, we want to find an approximation $\tilde{\mathcal{R}}(t,P)$ whose Hausdorff distance to the tight outer approximation $\mathcal{R}(t,P)$ is at most ϵ.

In Ex. 5.3, we saw that the dependency-preserving part $\mathcal{D}(t,P)$ (see (5.23)) used for PGSC often is a reasonable approximation of the parameterized reachable set up to a scaling factor. When only minimizing the size of the final reachable set, a rough approximation of the parameterized reachable set is often sufficient. However, once we also consider state constraints in the synthesis problem, the absolute size of the approximated parameterized reachable set becomes highly relevant. In PGSC and GSC, this was solved by introducing the adapted state constraints $\tilde{\mathcal{X}}$, which have to be provided by the user.

We avoid these adapted constraints by noticing that for the current controller parameters $\bar{P}$ (see Alg. 4), a tight outer approximation $\mathcal{R}\left(t,\bar{P}\right)$ of the reachable set at $\bar{P}$ for all $t \in [0, ht_s]$ is available. Thus, we construct an approximation to the parameterized reachable set as

$$\tilde{\mathcal{R}}(t,P) = \mathcal{R}\left(t,\bar{P}\right) \oplus_e \mathcal{D}(t,P) \oplus_e \left(-\mathcal{D}\left(t,\bar{P}\right) \right), \tag{5.44}$$

where $\oplus_e$ denotes the exact addition of polynomial zonotopes (see Sec. 2.6.6): Since $\mathcal{D}\left(t,\bar{P}\right) \oplus_e \left(-\mathcal{D}\left(t,\bar{P}\right) \right) = 0$ by definition, we recover $\mathcal{R}\left(t,\bar{P}\right)$ by evaluating (5.44) for the current controller parameters $\bar{P}$. For any $P \neq \bar{P}$, we then use $\mathcal{D}(t,P) \oplus_e \left(-\mathcal{D}\left(t,\bar{P}\right) \right)$ to approximate the unknown difference $\mathcal{R}(t,P) \oplus_e \left(-\mathcal{R}\left(t,\bar{P}\right) \right)$. In order to control the approximation quality of (5.44), we limit the controller parameters to the trust region $P \in \mathcal{P}_\gamma\left(\bar{P}\right)$, where

$$\mathcal{P}_\gamma\left(\bar{P}\right) = \left\{ \bar{P} + 2\gamma M \;\middle|\; M \in [-1,1]^{hm_c n_u \times a} \right\}. \tag{5.45}$$

Here, $\gamma \in (0,1]$ is the trust-region radius that restricts the range of possible P around the controller parameters $\bar{P}$. To keep γ within $(0,1]$ for consistency with later chapters, we include the factor 2 in (5.45) so that $\forall \bar{P} \in [-1,1]^{hm_c n_u \times a} : [-1,1]^{hm_c n_u \times a} \subseteq \mathcal{P}_1\left(\bar{P}\right)$, i.e., $\gamma = 1$ means that

$$\forall i \in \{0,...,h-1\} \, \forall j \in \{0,...,m_c - 1\} \, \forall u \in \mathcal{U} \, \exists P \in \mathcal{P}_1\left(\bar{P}\right) \, \exists \beta \in [-1,1]^l : \\ \bar{u}\left(\beta, P^{(i,j)}\right) = u,$$

where β is the dependent factor of the generating function $x^{(0)}(\beta)$ of the initial set $\mathcal{X}^{(0)}$. For $P \in \mathcal{P}_\gamma\left(\bar{P}\right)$, it thus holds that

$$\lim_{\gamma \to 0} \tilde{\mathcal{R}}(t, P) = \mathcal{R}\left(t, \bar{P}\right),$$

and therefore we can ensure (5.43) for $P \in \mathcal{P}_\gamma\left(\bar{P}\right)$, i.e.

$$\exists \gamma \in (0,1] \, \forall P \in \mathcal{P}_\gamma\left(\bar{P}\right) \forall t \in [0, ht_\mathrm{s}] : d_H\left(\mathcal{R}(t, P), \tilde{\mathcal{R}}(t, P)\right) \leq \epsilon, \tag{5.46}$$

for any $\epsilon > 0$ and independent of the choice of $\bar{P}$. We defer the tuning of γ to achieve (5.46) to Sec. 5.4.4.

5.4.3 Trust-Region Subproblem

We now propose the trust-region subproblem as an approximation to (5.42). With a slight abuse of notation, we again set $P = P^{(0:h-1)}$ and propose the trust-region subproblem

$$\hat{P}, \hat{s} = \arg\min_{P,s} \tilde{J}_\mathrm{TR}(P, s), \tag{5.47a}$$

$$\text{s.t. } \tilde{g}(P) \leq s, \tag{5.47b}$$

$$P \in \mathcal{P}_\gamma\left(\bar{P}\right) \cap [-1,1]^{hm_\mathrm{c}n_u \times a}, \tag{5.47c}$$

$$s \geq 0, \tag{5.47d}$$

with

$$\tilde{J}_\mathrm{TR}(P, s) = \sum_{k=1}^{h} \vartheta_k \left(\left\| \begin{bmatrix} \tilde{c}(kt_\mathrm{s}, P) - x_\mathrm{f}^{(k)} \\ \left[\tilde{G}(kt_\mathrm{s}, P)\right]_{(:)} \end{bmatrix} \right\|_1 + \zeta \sum_{j=0}^{m_\mathrm{c}-1} \left\| \begin{bmatrix} c_u\left(P^{(k-1,j)}\right) \\ \left[G_u\left(P^{(k-1,j)}\right)\right]_{(:)} \end{bmatrix} \right\|_1 \right) + \sigma \|s\|_1, \tag{5.48}$$

as an approximation to the cost in (5.42), where $\tilde{g}(P)$ is defined as in (5.41) but with $\rho_x^{(\mathcal{Z})}(\cdot, t, P)$ replaced by the support function $\tilde{\rho}_x^{(\mathcal{Z})}(\cdot, t, P)$ of the zonotope outer approximation $\tilde{\mathcal{Z}}(t, P) = \hat{\mathcal{Z}}\left(\tilde{\mathcal{R}}(t, P)\right)$:

- (5.47a) and (5.48): We avoid the need for an outer approximation of the reachable set in (5.42) by replacing it with the approximated parameterized reachable set $\tilde{\mathcal{Z}}(t, P)$ (see Sec. 5.4.2). Further, we replace the constraint violation in (5.42) with the slack variable s (see next bullet point for more details).

- (5.47b): We replace the support function $\rho_x^{(\mathcal{Z})}$ of the zonotope outer approximation with the support function $\tilde{\rho}_x^{(\mathcal{Z})}$ of $\hat{\mathcal{Z}}\left(\tilde{\mathcal{R}}(t, P)\right)$, i.e., the zonotope outer approximation of the parameterized reachable set. Further, we include input, state, and final state constraints in (5.47b) but relax them with $s \in \mathbb{R}_{\geq 0}^{o_\mathcal{U} + o_\mathcal{X} + o_{\mathcal{X}_\mathrm{f}}}$ so that (5.47)

always has a feasible solution even if $\nexists P \in \mathcal{P}_\gamma\left(\bar{P}\right)\ \forall k \in \{1, ..., o_\mathcal{U} + o_\mathcal{X} + o_{\mathcal{X}_\mathrm{f}}\}$: $\tilde{g}_k\left(P\right) \leq 0$ holds true, i.e., no feasible solution of (5.47) for $s = 0$ exists. We force a reduction in constraint violation for infeasible P by penalizing s in the objective function, where the exact penalty multiplier $\sigma \gg 1$ is chosen large enough (see, e.g., [50] for a discussion on penalty multipliers) so that the minimization of constraint violation is always prioritized over minimizing the objective function for $s = 0$.

- (5.47c): To control the accuracy of the approximated, parameterized reachable set, we restrict P to its trust region, i.e, $P \in \mathcal{P}_\gamma\left(\bar{P}\right) \cap [-1, 1]^{hm_\mathrm{c}n_u \times a}$ (see Sec. 5.4.2 for details).

The trust-region subproblem in (5.47) can be reformulated as a smooth optimization problem (see Sec. 3.2.2) and thus allows us to use off-the-shelf optimization codes, such as the interior point optimizer[5] [111] (IPOPT). Further, regularity of (5.47) follows from Prop. 3.2. As it turns out, we can express the objective value (5.48) at a critical point in $\hat{P}$ only, which allows the definition of the approximated cost $\tilde{J}\left(P\right)$.

Proposition 5.2 (Stationary Objective Value)**.** *At a critical point of the smooth reformulation of* (5.47) *with optimizer $\hat{P}$, the optimal objective value – only dependent on $\hat{P}$ – is given by*

$$\tilde{J}\left(\hat{P}\right) = \tilde{J}_\mathrm{TR}\left(\hat{P}, \max\left(0, \tilde{g}\left(\hat{P}\right)\right)\right). \tag{5.49}$$

Proof. Follows from Prop. 3.1. □

Remark: Because $\tilde{\mathcal{R}}\left(t, \bar{P}\right) = \mathcal{R}\left(t, \bar{P}\right)$ (see (5.44)), the cost $J\left(P\right)$ from (5.42) and the approximated cost $\tilde{J}\left(P\right)$ as defined in (5.49) are equal at $P = \bar{P}$, i.e., $\tilde{J}\left(\bar{P}\right) = J\left(\bar{P}\right)$. Thus, choosing $\bar{s} = \max\left(0, \tilde{g}\left(\bar{P}\right)\right)$ as in Prop. 5.2, the initial guess $\left(\bar{P}, \bar{s}\right)$ for solving the smooth reformulation of (5.47) ensures that we start optimizing from the actual controller cost.

5.4.4 Tuning of the Trust-Region Radius

Previously, we derived an approximation to the controller cost $J\left(\hat{P}\right)$ in (5.42), where $\hat{P}$ is a critical point of (5.47). In this section, we show how the value of the trust-region radius can be adapted after each iteration of Alg. 4 so that the objective value $\tilde{J}\left(\hat{P}\right)$ of (5.47) at a critical point $\hat{P}$ approximates $J\left(\hat{P}\right)$ arbitrarily closely.

After the trust-region subproblem in (5.47) is solved to a critical point $\hat{P}$, we have both the approximated cost $\tilde{J}\left(\hat{P}\right)$ (see Prop. 5.2) and the cost $J\left(\hat{P}\right)$, calculated from the outer approximation of the closed-loop reachable set, available (see Alg. 4). This inspires the following tuning rule.

[5] `https://github.com/coin-or/Ipopt`

Theorem 5.1 (Tuning of Trust-Region Radius)**.** *Let* $J\left(\hat{P}\right)$ *and* $\tilde{J}\left(\hat{P}\right)$ *be defined as in* (5.42) *and* (5.49)*, respectively, where* $\hat{P}$ *is a critical point of the smooth reformulation of the trust-region subproblem* (5.47)*. If we adapt the trust-region radius* γ *according to*

$$\gamma \leftarrow \min\left(1, v\left(\left|\tilde{J}\left(\hat{P}\right) - J\left(\hat{P}\right)\right|\right)\gamma\right), \tag{5.50}$$

where $v : [0; \infty) \mapsto \mathbb{R}_+$ *is an arbitrary, monotonically decreasing function with* $v(0) = \bar{c}$*,* $v(\psi) = 1$*, and* $\forall r \geq \bar{\psi} : v(r) = \underline{c}$*, where* $\frac{1}{\underline{c}} > \bar{c} > 1$ *and* $0 \leq \psi < \epsilon$*, then*

$$\left|\tilde{J}\left(\hat{P}\right) - J\left(\hat{P}\right)\right| \leq \epsilon,$$

is achieved after a finite number of iterations for $\epsilon > 0$*.*

Proof. Comparing (5.42) with (5.49), it is clear that $\lim_{\gamma\to 0}\left|\tilde{J}\left(\hat{P}\right) - J\left(\hat{P}\right)\right| = 0$ since $\lim_{\gamma\to 0} d_H\left(\hat{\mathcal{Z}}\left(\mathcal{R}\left(t, \hat{P}\right)\right), \hat{\mathcal{Z}}\left(\tilde{\mathcal{R}}\left(t, \hat{P}\right)\right)\right) = 0$ due to $\lim_{\gamma\to 0}\hat{P} = \bar{P}$ and $\tilde{\mathcal{R}}\left(t, \bar{P}\right) = \mathcal{R}\left(t, \bar{P}\right)$. Therefore, following (5.50) for a finite number of steps yields a trust-region radius $\gamma > 0$ such that $\left|\tilde{J}\left(\hat{P}\right) - J\left(\hat{P}\right)\right| \leq \epsilon$, which concludes the proof. □

5.4.5 Computational Complexity

We again differentiate between the complexity of the offline controller synthesis and the complexity of applying said controller online. We again make use of Ass. 5.1. For simplicity, let $\kappa \geq 1$ for the order of the controller template introduced in Sec. 5.3.2 and assume that the number of monomials a of the controller template grows at least with $O(n)$, i.e., $O(n + a) = O(a)$.

5.4.5.1 Offline Complexity

We first inspect the complexity of the different components of the iPGSC approach. Since m_s is fixed, it suffices to inspect the algorithm for $m_\mathrm{s} = 1$.

Controller Template: The complexity of the polynomial composition dominates with $O\left(an^{1+\kappa}\right)$ (see Sec. 5.3.6.1).

Reachable Set Computations: The combined complexity for the closed-loop reachable set computation and the parameterized reachable set computation is bounded from above by $O\left(n^5 + n^2a^2 + n^3a\log n\right)$ (see Sec. 5.3.6.1).

Controller Computation: Since the maximum number of steps $l_\mathrm{max} \in \mathbb{N}_+$ in Alg. 4 is fixed, it suffices to check the complexity of one trust-region iteration. As we already considered the complexity for the reachable set computations above and because the trust-region tuning has lower (or equal) computational complexity by visual inspection, it suffices to derive the solution complexity of the trust-region subproblem in (5.47).

Computing the optimal controller parameters $\hat{P}$ requires the solution of the smooth reformulation of the non-convex optimization problem in (5.47). Following the same argument as for the complexity of the controller computation in Sec. 5.3.6.1, the computational complexity of solving the trust-region subproblem is at most $O\left(a^2 n^4\right)$.

Overall Complexity: The offline complexity of the iPGSC approach is identical to the PGSC approach, i.e.

$$O\left(c_{\text{iPGSC}}^{(\text{off})}(n)\right) = O\left(c_{\text{PGSC}}^{(\text{off})}(n)\right) \overset{(5.33)}{=} O\left(c_{\text{ref}}(n) + an^{1+\kappa} + a^2 n^4\right). \tag{5.51}$$

5.4.5.2 Online Complexity

The online complexity of iPGSC is identical to that of PGSC, i.e.

$$O\left(c_{\text{iPGSC}}^{(\text{on})}(n)\right) = O\left(c_{\text{PGSC}}^{(\text{on})}(n)\right) \overset{(5.34)}{=} O(an). \tag{5.52}$$

5.4.6 Experiments

In this section, we demonstrate the applicability of the proposed iPGSC approach by comparing it against the PGSC approach introduced in Sec. 5.3. For that comparison, we revisit both the controlled Van-der-Pol oscillator and the bicycle benchmark described in Sec. 5.3.8. We set $\zeta = 0$ for all benchmarks in this section.

5.4.6.1 Bicycle (Single Track)

We consider the bicycle model with the dynamics given in (5.38) and parameters given as in Tab. 5.1. Additionally, we explicitly enforce the final state constraint $\mathcal{X}_{\text{f}} = \{-c_{\mathcal{X}^{(0)}} + x_{\text{f}}\} \oplus \mathcal{X}^{(0)}$ for the iPGSC approach now (also see Sec. 5.3.9 for why we only enforce it here).

Fig. 5.10 compares the performance of PGSC against iPGSC, where runtimes were 34.7 s and 196 s, respectively. The deviation and size of the final reachable sets is $\text{csize}\left(\mathcal{R}_{\text{PGSC}}(t_{\text{f}})\right) = 0.65$ and $\text{csize}\left(\mathcal{R}_{\text{iPGSC}}(t_{\text{f}})\right) = 0.17$. Clearly, the PGSC approach is outperformed by iPGSC even with $\mathcal{X}_{\text{f}}$ enforced, albeit at the cost of significantly increased computation times, which are mainly due to the more frequent computation of reachable set outer approximations.

In Fig. 5.11, the progress of the iPGSC trust-region algorithm is visualized over the iterations. First, as remarked at the end of Sec. 5.4.3, the approximated cost equals the formally correct controller cost at the current controller guess $\bar{P}^{(i-1)}$. Second, we see how the trust-region radius γ is decreased as the iterations at $i \in \{1, 3\}$ are rejected to recover a "good enough" approximation of the parameterized reachable set for the next iteration.

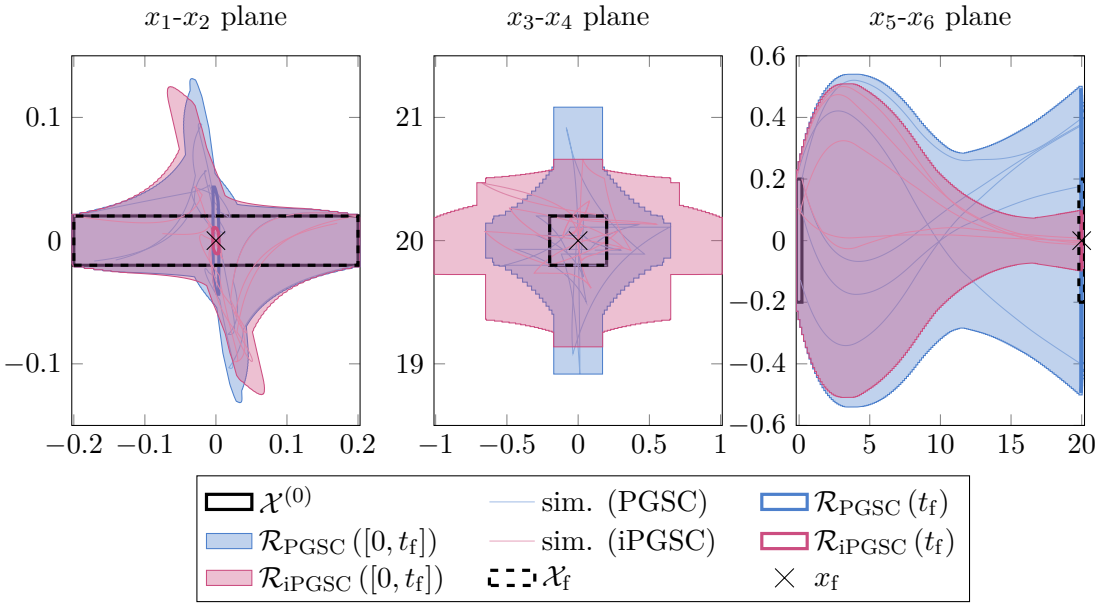


Figure 5.10: Comparison of the PGSC approach and the iPGSC approach for the single-track (bicycle) model. A linear controller template is used.

5.4.6.2 Controlled Van-der-Pol Oscillator

We consider the Van-der-Pol oscillator model described in (5.39) with benchmark parameters given as in Tab. 5.2 but with a quadratic controller, i.e., $\kappa = 2$.

Fig. 5.12 compares the control performance of PGSC against the proposed iPGSC approach, where PGSC and iPGSC took 19.68 s and 146.66 s, respectively, and further $\mathrm{csize}(\mathcal{R}_{\mathrm{PGSC}}(t_\mathrm{f})) = 0.79$ and $\mathrm{csize}(\mathcal{R}_{\mathrm{iPGSC}}(t_\mathrm{f})) = 0.67$. Clearly, iPGSC achieves a better control result, even though we additionally enforce final state constraints $\mathcal{X}_\mathrm{f} = \{-c_{\mathcal{X}^{(0)}} + x_\mathrm{f}\} \oplus \mathcal{X}^{(0)}$ for the iPGSC approach: By approximating the parameterized reachable set only locally around the current controller parameters, the iPGSC uses a more accurate approximation of the closed-loop reachable set during the optimization of the controller parameters. In contrast, the PGSC approach computes its parameterized reachable set only as a global approximation of the closed-loop reachable set.

5.4.7 Discussion

As the numerical experiments indicate, iPGSC synthesizes controllers with better performance compared to the PGSC approach (and thus the GSC approach by extension). Further, iPGSC uses state constraints directly as provided by the user without the need for manual constraint tightening or bloating since we control the accuracy of the parameterized reachable set automatically using the trust-region radius. Furthermore, its design naturally allows to improve existing controllers by providing them as an initial guess to the algorithm. Because we only accept iterations that improve the control per-

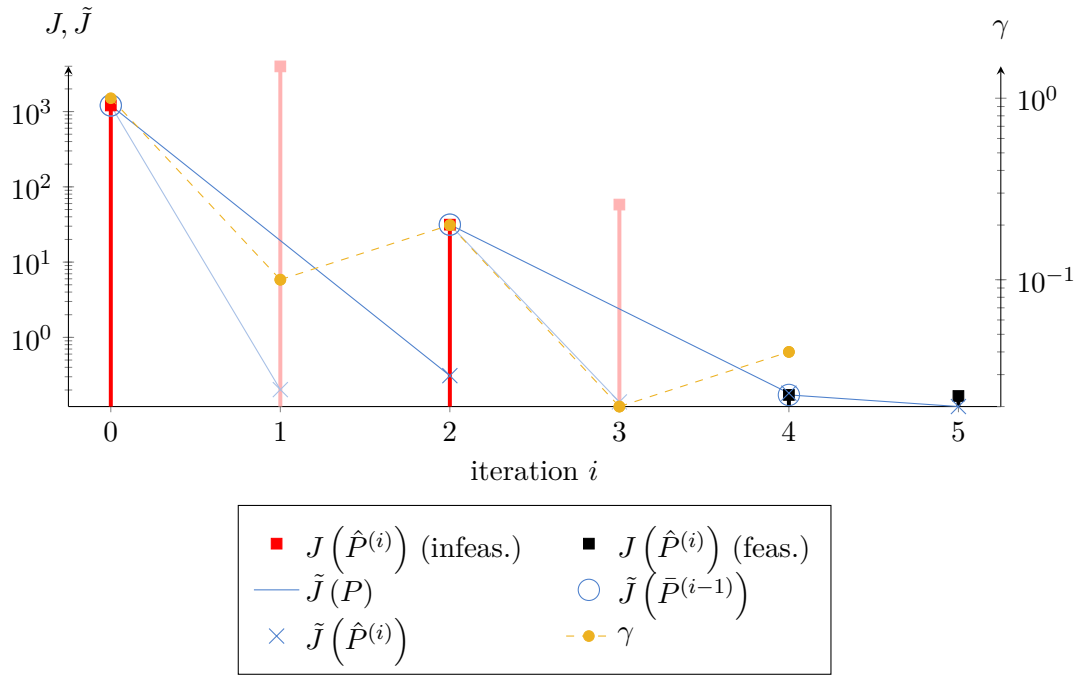


Figure 5.11: Visualization of the solver progress for the PGSC approach (bicycle benchmark). Values between $\tilde{J}\left(\bar{P}^{(i-1)}\right)$ and $\tilde{J}\left(\hat{P}^{(i)}\right)$ for $1 \leq i \leq 5$ – where $\bar{P}^{(i-1)}$ and $\hat{P}^{(i)}$ are the initial and critical points of the trust-region subproblem – are linearly interpolated ($\hat{P}^{(0)} = \bar{P}^{(0)}$). The opaque red vertical bars at $i \in \{1, 3\}$ visualize a rejected iteration. As one might expect, the trust-region radius γ is shrunk at each rejected iteration to ensure a more accurate approximation for the next step.

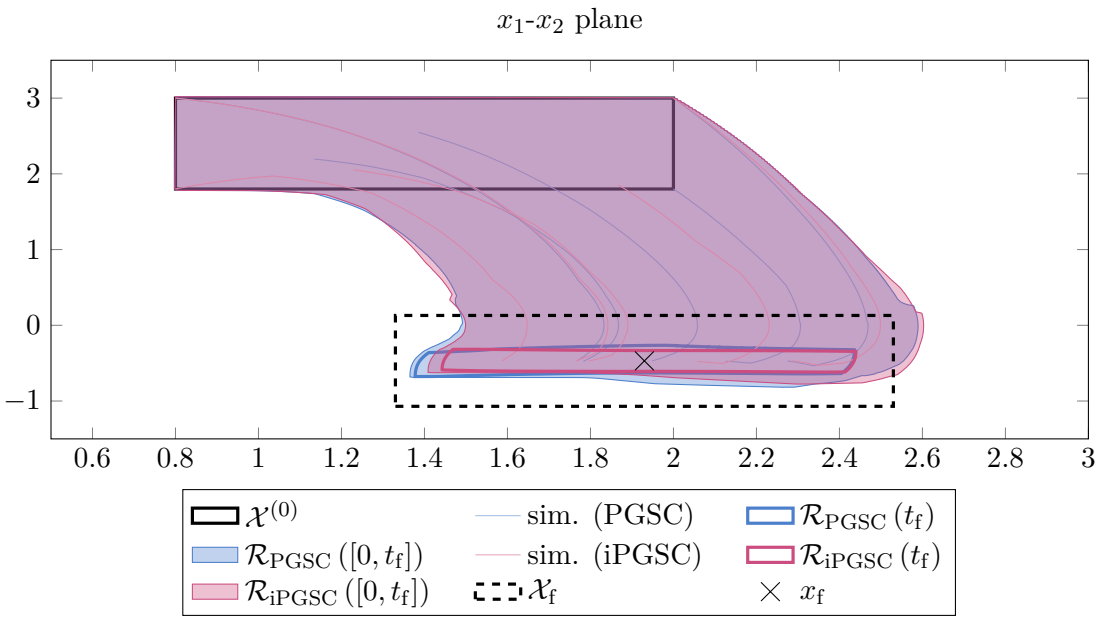


Figure 5.12: Comparison of the PGSC approach and the iPGSC approach for the Van-der-Pol model. Both approaches use a quadratic controller template.

formance, the iPGSC algorithm can guarantee that the control performance of the newly computed controller is at least as good as the one provided as an initial guess.

That said, all these advantages come at the cost of increased computational effort, which is mainly caused by the need to compute outer approximations of the reachable sets. With that in mind, the increased computational effort for iPGSC only occurs offline and still has polynomial complexity (see Sec. 5.4.5.1) so that it should always be the first piecewise constant synthesis algorithm to try on any system if very fast offline computation times are not required.

5.5 Summary

Starting from the problem statement in Sec. 5.1, we reviewed existing approaches for the piecewise constant controller synthesis for disturbed nonlinear systems in Sec. 5.2 and extended existing work to include polynomial controllers and allow parameterized reachable sets of arbitrary abstraction order in Sec. 5.3 to improve the performance of the synthesized controllers. Furthermore, we introduced a novel trust-region algorithm for piecewise constant controller synthesis in Sec. 5.4 that can directly consider state constraints without the need for manual tuning.

Generator-space control (GSC) introduced in [99] synthesizes piecewise constant feedforward controllers that are linear in the initial state under input and state constraints. With the initial set given as a zonotope, a zonotope is used to represent the controller, where the center and generator matrix are parameters that need to be computed. An

approximation to the closed-loop reachable set using that controller is computed by linearizing the undisturbed nonlinear dynamics along a previously computed reference trajectory. Because of this linearization, we obtained an analytic expression for the parameterized reachable set in the controller parameters. Using this parameterized reachable set, a linear program (LP) is formulated which minimizes the distance of this approximated reachable set from the target state and further minimizes the size of the final, approximated reachable set as well as the control energy. Due to the parameterization of the controller, input constraints can be exactly represented as linear constraints. Since the parameterized reachable set is only approximately computed, state constraints can only be considered approximately and require additional user input. Sub-optimal choices of this input may require additional runs of the algorithm.

To better approximate the reachable set, we described the polynomial generator-space control (PGSC) approach from our work in [37]: Here, we computed the parameterized reachable set using a slightly modified version of reachability analysis from [5] with a polynomial abstraction order to achieve better results. In contrast to the GSC approach [99] where only linear control laws are synthesized, we proposed a controller which is polynomial in the initial state and showed that PGSC generalizes the GSC approach. Numerical experiments demonstrated the improved control performance due to the increased abstraction order for the parameterized reachable set and the polynomial controller compared to the GSC approach from previous work [99].

To avoid the need for the manual tuning of state constraints by the user as required for GSC and PGSC, we introduced the iterative polynomial generator-space control (iPGSC) approach as part of our work in [35] to solve the synthesis problem. To that end, we first defined a cost function in the controller parameters as an approximation to the general synthesis problem, penalizing the deviation from the target state, the size of the reachable set, the input energy, and the constraint violations. Since this cost function is non-differentiable and because its evaluation requires the computation of the closed-loop reachable set, we optimized over this cost iteratively: We approximate the non-differentiable cost in the neighborhood of the current controller parameters by the differentiable trust-region subproblem, whose accuracy is ensured by restricting its domain to a bounded trust region around the current controller parameters. The size of the trust region is then updated by comparing the change in cost between the newly synthesized controller and the current controller to the change predicted by the trust-region subproblem. If the new controller improves the controller cost, we accept the step and update the current controller parameters accordingly. The advantages of this novel synthesis algorithm were demonstrated in numerical experiments.

6 Piecewise Constant Controller Synthesis with Continuous State Feedback

Previously, we considered the synthesis of controllers which are piecewise constant in time. Due to their simple structure, they are easy to implement and can be applied efficiently online. Additionally, they also allow the incorporation of state feedback at discrete points in time. In practice, however, it is often necessary to shrink the time between input updates, i.e., increase the number of piecewise constant controllers that need to be computed, e.g., when a system is dominated by disturbances. Thus, larger disturbance sets may require a larger number of piecewise constant controllers, increasing the computational effort required, both offline and online. In this chapter, we combine piecewise constant control with continuous state feedback in order to continuously counteract disturbances without the need for a larger number of piecewise constant controllers.

To that end, we first introduce the synthesis problem for this combined controller in Sec. 6.1 and present a solution approach from previous work [101],[98, Sec. 3.6] in Sec. 6.2. We then propose a novel synthesis approach for the combined controller in Sec. 6.3 which – for the first time – synthesizes the feedforward and feedback controller simultaneously.

6.1 Problem Statement

In this chapter, we synthesize controllers for $t \in \tau^{(j)} = [j, j+1]\,\frac{t_\mathrm{f}}{m_\mathrm{c}}$ with $0 \leq j \leq m_\mathrm{c} - 1$ of the form (as proposed in [98, Eq. (3.43)])

$$u\left(t, x\left(t\right), P, K\right) = u_\mathrm{ff}\left(x\left(0\right), P^{(0,j)}\right) + K^{(j)}\left(x\left(t, P, K\right) - x_\mathrm{ff}\left(t, P\right)\right), \tag{6.1}$$

where $x\left(0\right) \in \mathcal{X}^{(0)}$, $K\left(t\right) = K^{(j)} \in \mathbb{R}^{n_u \times n_x}$ for $t \in \tau^{(j)}$, the controller parameters of the piecewise constant feedforward controller $u_\mathrm{ff}\left(x\left(0\right), P^{(0,j)}\right)$ (see Chap. 5) are given by $P = \left[{P^{(0,0)}}^T, \quad {P^{(0,1)}}^T, \quad \ldots, \quad {P^{(0,m_\mathrm{c}-1)}}^T\right] \in [-1, 1]^{m_\mathrm{c} n_u \times a}$, and

$$x_\mathrm{ff}\left(t, P\right) = \xi\left(t, x\left(0\right), u_\mathrm{ff}\left(x\left(0\right), P^{(0,\cdot)}\right), 0\right), \tag{6.2}$$

is the feedforward trajectory that one obtains by starting from the initial state $x\left(0\right)$ and applying the m_c piecewise constant feedforward controllers: Piecewise constant controllers are able to solve a given reach-avoid problem efficiently when little or no disturbance is involved (see Chap. 5). Thus, to steer the undisturbed system close to the target state, the feedforward controller is applied. Since the actual system is affected by

disturbances, the continuous feedback controller is applied on the difference between the nominal feedforward state and the actual state – caused by disturbances – to minimize this difference. We sometimes omit time indices and dependencies for readability, e.g., we write K instead of $K(t)$. The feedforward part is a piecewise constant controller as described in Chap. 5 with $m_\mathrm{s} = 1$ and therefore $P = P^{(0)}$. To simplify notation, we further set $P^{(j)} = P^{(0,j)}$ for the remainder of this chapter. We again assume that the bounded input constraints $\mathcal{U} = \langle C_\mathcal{U}, d_\mathcal{U} \rangle_H$ with $C_\mathcal{U} \in \mathbb{R}^{o_\mathcal{U} \times n_u}$ and $d_\mathcal{U} \in \mathbb{R}^{o_\mathcal{U}}$, the state constraints $\mathcal{X} = \langle C_\mathcal{X}, d_\mathcal{X} \rangle_H$ with $C_\mathcal{X} \in \mathbb{R}^{o_\mathcal{X} \times n_x}$ and $d_\mathcal{X} \in \mathbb{R}^{o_\mathcal{X}}$, and the final state constraints $\mathcal{X}_\mathrm{f} = \langle C_{\mathcal{X}_\mathrm{f}}, d_{\mathcal{X}_\mathrm{f}} \rangle_H$ with $C_{\mathcal{X}_\mathrm{f}} \in \mathbb{R}^{o_{\mathcal{X}_\mathrm{f}} \times n_x}$ and $d_{\mathcal{X}_\mathrm{f}} \in \mathbb{R}^{o_{\mathcal{X}_\mathrm{f}}}$ are given as H-polytopes.

Let $\mathcal{R}(t, P, K)$ denote an outer approximation of the closed-loop reachable set of the combined controller $u(t, x(t), P, K)$ from (6.1). Substitution of (6.1) into the general synthesis problem in (3.1) then yields an optimization problem for the combined synthesis of a piecewise constant feedforward controller with continuous state feedback, i.e. (also compare to)

$$\hat{P}, \hat{K} = \arg\min_{P,K} \max_{x(t,P,K) \in \mathcal{R}(t,P,K)} \left\{ \left\| x(t_\mathrm{f}, P, K) - x_\mathrm{f} \right\|_1 + \zeta \int_0^{t_\mathrm{f}} \left\| u(\tau, x(\tau), P, K) \right\|_1 \mathrm{d}\tau \right\} \tag{6.3a}$$

$$\text{s.t. } \forall t \in [0, t_\mathrm{f}] : \mathcal{S}_u(t, P, K) \subseteq \mathcal{U}, \tag{6.3b}$$

$$\mathcal{R}([0, t_\mathrm{f}], P, K) \subseteq \mathcal{X}, \tag{6.3c}$$

$$\mathcal{R}(t_\mathrm{f}, P, K) \subseteq \mathcal{X}_\mathrm{f}, \tag{6.3d}$$

where

$$\mathcal{S}_u(t, P, K) = \mathcal{S}_{u_\mathrm{ff}}(t, P) \oplus_e \mathcal{S}_{\Delta u}(t, P, K), \tag{6.4}$$

follows from (6.1) with

$$\mathcal{S}_{u_\mathrm{ff}}(t, P) = \left\{ u_\mathrm{ff}\left(x(0), P^{(0,j)}\right) \,\middle|\, x(0) \in \mathcal{X}^{(0)} \right\}, \tag{6.5}$$

$$\mathcal{S}_{\Delta u}(t, P, K) = K^{(j)}\left(\mathcal{R}(t, P, K) \oplus_e \left(-\mathcal{R}_{x_\mathrm{ff}}(t, P)\right)\right), \tag{6.6}$$

for $t \in \tau^{(j)}$ with $0 \leq j \leq m_\mathrm{c} - 1$. Here, $\mathcal{R}_{x_\mathrm{ff}}(t, P)$ denotes the reachable set of the feedforward state from (6.2).

6.2 Reachset Optimal Control

We already established that continuous state feedback is beneficial to quickly counteract disturbances for systems where the disturbances are large or the dynamics of the system are changing quickly. That said, the control goal is not only rejecting disturbances as efficiently as possible but also steering the system close to the target state. Therefore, the authors in [101] combine the GSC feedforward synthesis approach with a subsequent optimization over the feedback matrices in order to minimize the effect of disturbances, which we review in this section. The general structure of the reachset optimal control (ROC) approach is detailed in Alg. 5 and visualized in Fig. 6.1. The most important steps can be summarized as follows:

(i) We start by computing a reference trajectory steering the center of the initial set close to the target state (l. 2).

(ii) Since the feedforward and the feedback synthesis problems are solved separately, we compute the feedforward controller using GSC with adapted state and input constraints (l. 3).

(iii) Next, we compute an initial guess of the feedback matrices for the feedback synthesis (l. 4).

(iv) Given the feedforward controller from (ii), we then optimize the feedback matrices to obtain a feedback control law that minimizes the deviation of the disturbed, closed-loop state from the undisturbed feedforward state. However, we do not directly optimize over the feedback matrices; rather, we parameterize all feedback matrices using LQR control, i.e., the feedback matrices become functions of the LQR weighting matrices $Q \in \mathbb{S}_{++}^{n_x \times n_x}$ and $R \in \mathbb{S}_{++}^{n_u \times n_u}$ (also see Sec. 6.2.1); here, Q and R are kept constant over all time steps. This reduces the feedback synthesis to an optimization over the weighting matrices Q and R. In each optimization iteration, we compute a formally verified outer approximation to the closed-loop reachable set for the combined controller (see Sec. 6.2.2) and then use this reachable set to evaluate the objective function and check constraint satisfaction (l. 5).

Algorithm 5 Reachset Optimal Control

1: **function** REACHSETOPTIMALCONTROL($\mathcal{X}^{(0)}, \mathcal{U}, \tilde{\mathcal{U}}, \mathcal{X}, \tilde{\mathcal{X}}, \mathcal{X}_\mathrm{f}, \tilde{\mathcal{X}}_\mathrm{f}, \mathcal{W}, m_\mathrm{c}, x_\mathrm{f}$)
2: $\quad [x_\mathrm{ref}, u_\mathrm{ref}] = \text{REFERENCETRAJECTORY}(\mathcal{X}^{(0)}, \mathcal{U}, \mathcal{X}, \mathcal{X}_\mathrm{f}, 1, m_\mathrm{c}, x_\mathrm{f}, t_\mathrm{f})$ ▷ Sec. 5.2.1
3: $\quad \hat{P} = \text{GENERATORSPACECONTROL}(\mathcal{X}^{(0)}, \tilde{\mathcal{U}}, \tilde{\mathcal{X}}, \tilde{\mathcal{X}}_\mathrm{f}, \{0\}, 1, m_\mathrm{c}, x_\mathrm{f}, 1)$ ▷ Sec. 6.2.3
4: $\quad \left[\bar{Q}, \bar{R}\right] = \text{INITGUESS}(\dots)$ ▷ Sec. 6.2.5
5: $\quad \left[\hat{K}, \mathcal{R}([0, t_\mathrm{f}])\right] = \text{OPTPROB}(\hat{P}, \bar{Q}, \bar{R}, \mathcal{X}^{(0)}, \mathcal{U}, \mathcal{X}, \mathcal{X}_\mathrm{f}, \mathcal{W}, m_\mathrm{c}, t_\mathrm{f}, x_\mathrm{ref}, u_\mathrm{ref})$ ▷ (6.16)
6: $\quad$ **return** $\left[\hat{P}, \hat{K}, \mathcal{R}([0, t_\mathrm{f}])\right]$
7: **end function**

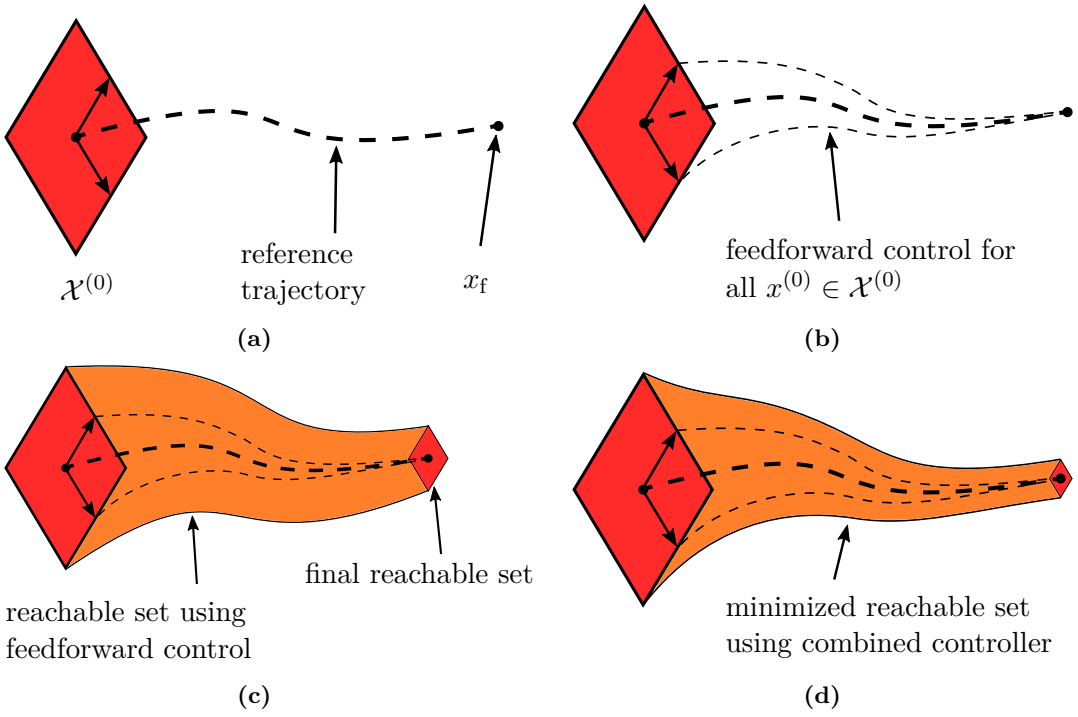


Figure 6.1: Overview of the ROC approach [101, adapted]. First, a reference trajectory is computed (Fig. 6.1a). For the adapted input and state constraints, the feedforward synthesis problem for the target state x_f is solved using the GSC approach as described in Sec. 5.2 (Fig. 6.1b). Since the feedforward synthesis problem does not consider disturbances, the resulting closed-loop reachable set under disturbances might be too large (Fig. 6.1c). Thus, we optimize over the feedback controller to minimize the effect of the disturbance (Fig. 6.1d).

We structure this section as follows: In Sec. 6.2.1, we briefly introduce the parameterization of the feedback matrices using LQR control, and then discuss the closed-loop reachable set computation for a given combined controller in Sec. 6.2.2. In Sec. 6.2.3, we discuss how the GSC approach can be used as a feedforward controller and then describe the feedback controller synthesis in Sec. 6.2.4. Next, we briefly comment on the role of initial guesses in Sec. 6.2.5. In Sec. 6.2.6, we then analyze the computational complexity of ROC and finally discuss important properties of the algorithm in Sec. 6.2.7. Note that we defer numerical experiments of ROC to Sec. 6.3, where we compare ROC against our novel combined synthesis approach. We refer the reader to [101, Sec. VI] for a comparison of ROC to other state-of-the-art approaches.

6.2.1 Feedback Matrix Parameterization

Directly optimizing over $K^{(j)}$ for $0 \leq j \leq m_\mathrm{c} - 1$ requires $m_\mathrm{c} n_x n_u$ variables since $K^{(j)} \in \mathbb{R}^{n_u \times n_x}$. In this section, we therefore introduce a parameterization for the feedback matrices using LQR control (see Sec. 2.4) as proposed in [101, Sec. IV.B.], [98, Sec. 3.6.3] to reduce the number of optimization variables.

Since LQR computes a control law for an LTI system, we first linearize the nonlinear dynamics f along the reference trajectory to obtain m_c LTI systems. For $t \in \tau^{(j)} = [j, j+1] \frac{t_\mathrm{f}}{m_\mathrm{c}}$, we obtain the LTI system $\left(A^{(j)}, B^{(j)}\right)$ as

$$A^{(j)} = \left.\frac{\partial f(x,u,w)}{\partial x}\right|_{\substack{x=x_\mathrm{lin}^{(j)} \\ u=u_\mathrm{lin}^{(j)} \\ w=0}},$$
$$B^{(j)} = \left.\frac{\partial f(x,u,w)}{\partial u}\right|_{\substack{x=x_\mathrm{lin}^{(j)} \\ u=u_\mathrm{lin}^{(j)} \\ w=0}},$$

where

$$x_\mathrm{lin}^{(j)} = \frac{1}{2}\left(x_\mathrm{ref}^{(j)} + x_\mathrm{ref}^{(j+1)}\right),$$
$$u_\mathrm{lin}^{(j)} = u_\mathrm{ref}^{(j)},$$

and the reference inputs and reference trajectory are computed as described in Sec. 5.2.1 with $m_\mathrm{s} = 1$. To the best of our knowledge, the reason why the LTI systems $\left(A^{(j)}, B^{(j)}\right)$ are an appropriate choice for the computation of the LQR feedback matrix is not given by the original authors; we provide more details in Sec. 6.3.1 as we use this parameterization in a later algorithm. Assume that $\left(A^{(j)}, B^{(j)}\right)$ is controllable per Corr. 2.1; the corresponding LQR matrix $K^{(j)}$ of the LTI system $\left(A^{(j)}, B^{(j)}\right)$ with given Q and R can then be computed as described in Sec. 2.4. Intuitively, the assumption of controllability of the linearized dynamics may seem quite restrictive; however, trying to control the nonlinear dynamics when even the linearized dynamics (which are only valid approximations in a small neighbor around $x_\mathrm{lin}^{(j)}$ and $u_\mathrm{lin}^{(j)}$) are not controllable seems rather

ambitious. While keeping Q and R constant over the entire horizon reduces the number of optimization variables, it may also reduce the overall performance of the controller as we limit the range of possible feedback matrices. If this extra performance is required, one may introduce separate weighting matrices Q and R for each of the m_c LTI systems. Alternatively, one can also directly optimize over $K^{(j)}$, in which case controllability is not required; however, this means that one may also generate non-stabilizing feedback matrices.

6.2.2 Closed-Loop Reachable Set

Since ROC optimizes directly over outer approximations of the closed-loop reachable set, we discuss the closed-loop reachable set computation for a combined controller of the form (6.1) for the conservative linearization approach (see Sec. 2.7.2) for a given $K^{(j)}$ (also compare to [98, Sec. 3.6.4]). The reachable set computation using conservative polynomialization approach follows analogously and – since CORA implements dependency preservation for polynomial zonotopes – does not require the dependent factors β in the extended state definition.

To retain the dependency between each initial state and its corresponding control input, we define the extended state $x_\mathrm{ext} = \left[x^T, \quad x_\mathrm{ref}^T, \quad \Delta\tilde{x}_\mathrm{ff}^T, \quad \beta^T\right]^T$ so that the extended dynamics for $t \in \tau^{(j)}$ with $0 \leq j \leq m_\mathrm{c} - 1$ are given by

$$f_\mathrm{ext}(x,u,w) = \begin{bmatrix} f\left(x, \bar{u}\left(\beta, \hat{P}^{(j)}\right) + K^{(j)}(Q,R)\left(x - (x_\mathrm{ref} + \Delta\tilde{x}_\mathrm{ff})\right), w\right) \\ f(x_\mathrm{ref}, u_\mathrm{ref}, 0) \\ A\Delta\tilde{x}_\mathrm{ff} + B\left(\bar{u}\left(\beta, \hat{P}^{(j)}\right) - u_\mathrm{lin}\right) \\ 0 \end{bmatrix}, \tag{6.7}$$

where the feedforward controller $\bar{u}\left(\beta, \hat{P}^{(j)}\right)$, parameterized in β, is defined as in (5.3). Here, $\hat{P}$ denotes the optimal controller parameters that define the feedforward controller as synthesized by GSC. The generating function of the extended initial set then is

$$x_\mathrm{ext}^{(0)}(\beta) = \begin{bmatrix} x^{(0)}(\beta) \\ c_{\mathcal{X}^{(0)}} \\ G_{\mathcal{X}^{(0)}}\beta \\ \beta \end{bmatrix},$$

where $\mathcal{X}^{(0)} = \left\{x^{(0)}(\beta)\right\}_\beta = \langle c_{\mathcal{X}^{(0)}}, G_{\mathcal{X}^{(0)}}\rangle_Z$. Note that we do not use the feedforward state

$$x_\mathrm{ff}(t) = \xi\left(t, x(0), u_\mathrm{ff}\left(x(0), \hat{P}^{(\cdot)}\right), 0\right),$$

directly, but rather the same linearized approximation $\tilde{x}_\mathrm{ff} = x_\mathrm{ref} + \Delta\tilde{x}_\mathrm{ff}$ that is used to compute the parameterized reachable set for the GSC approach. By doing so, we try to bring the closed-loop reachable set of the combined controller as close as possible to the parameterized reachable set, evaluated at the optimal controller parameter $\hat{P}$. Since we

will replace x_{ff} by $\tilde{x}_{\mathrm{ff}}$ during the feedback controller synthesis (see Sec. 6.2.4) and during the online deployment of GSC (see Sec. 6.2.6.2), all formal guarantees still hold.

Applying the reachability algorithm from Sec. 2.7.2 to the extended system dynamics in (6.7) for $t \in [0, t_{\mathrm{c}}]$ yields an outer approximation of the closed-loop reachable set, given by

$$\mathcal{R}_{\mathrm{ext}}\left(t, Q, R\right) = \left\{ \begin{bmatrix} r_x\left(t, Q, R, \tilde{\beta}\right) \\ r_{x_{\mathrm{ref}}}\left(t, \tilde{\beta}\right) \\ r_{\Delta\tilde{x}_{\mathrm{ff}}}\left(t, \tilde{\beta}\right) \\ \beta\left(\tilde{\beta}\right) \end{bmatrix} \right\}_{\tilde{\beta}},$$

where its generating function is no longer dependent on β (e.g. due to reduction operations during reachability analysis) but $\tilde{\beta}$. Since this dependency is required for the evaluation of the dynamics in (6.7), we extended the state with β, making $\beta\left(\tilde{\beta}\right)$ available. The next extended initial set at $t = t_{\mathrm{c}}$ is then given by

$$x_{\mathrm{ext}}^{(1)}\left(\tilde{\beta}, Q, R\right) = \begin{bmatrix} r_x\left(\frac{t_{\mathrm{f}}}{m_{\mathrm{c}}}, Q, R, \tilde{\beta}\right) \\ r_{x_{\mathrm{ref}}}\left(\frac{t_{\mathrm{f}}}{m_{\mathrm{c}}}, Q, R, \tilde{\beta}\right) \\ r_{\Delta\tilde{x}_{\mathrm{ff}}}\left(\frac{t_{\mathrm{f}}}{m_{\mathrm{c}}}, \tilde{\beta}\right) \\ \beta\left(\tilde{\beta}\right) \end{bmatrix},$$

and hence $\mathcal{R}\left(t, Q, R\right) = \left\{ r_x\left(t, Q, R, \tilde{\beta}\right) \right\}_{\tilde{\beta}}$ for $t \in [0, t_{\mathrm{f}}]$ follows by the iterative application of the above steps.

6.2.3 Feedforward Controller Computation

Because the feedforward and feedback synthesis are executed sequentially, one can simply use the GSC approach described in Sec. 5.2 to compute the feedforward controller parameters. However, we cannot directly use the input constraint set $\mathcal{U}$ as is: If we compute the feedforward controller for the input constraint set $\mathcal{U}$, the feedforward controller may use up all of the input capacity, leaving no available input capacity for the continuous feedback controller. As a result, the user needs to specify a tightened input constraint set $\tilde{\mathcal{U}} \subseteq \mathcal{U}$ in addition to the adapted state and final state constraint sets $\tilde{\mathcal{X}}$ and $\tilde{\mathcal{X}}_{\mathrm{f}}$, which are necessary for non-conservative constraint checking.

6.2.4 Feedback Controller Computation

With the feedfoward controller fixed, we only describe the synthesis of the feedback matrices here. To pose an approximation of (6.3), we subsequently introduce an approximation of its objective function and an approximation of its constraints as proposed in [98]. To that end, we first introduce all required generating functions.

From Sec. 6.2.2, the generating function of the extended reachable set is

$$\mathcal{R}_{\text{ext}}(t,Q,R) = \left\{ \begin{bmatrix} r_x\left(t,Q,R,\tilde{\beta}\right) \\ r_{x_{\text{ref}}}\left(t,\tilde{\beta}\right) \\ r_{\Delta\tilde{x}_{\text{ff}}}\left(t,\tilde{\beta}\right) \\ \beta\left(\tilde{\beta}\right) \end{bmatrix} \right\}_{\tilde{\beta}} . \tag{6.8}$$

Let $m_r \in \mathbb{N}_+$ denote the number of steps during reachability analysis per interval $\tau^{(j)}$ for $0 \leq j \leq m_\text{c} - 1$, and let $\tau^{(k,j)} = j\frac{t_\text{f}}{m_\text{c}} + [k, k+1]\,\Delta t$ for $0 \leq k \leq m_r - 1$ with $\Delta t = \frac{t_\text{f}}{m_\text{c} m_r}$. Using (6.4) and (6.8), the generating function for the set of applied inputs $\hat{\mathcal{S}}_u(t,Q,R) = \left\{\hat{s}_u\left(t,Q,R,\tilde{\beta}\right)\right\}_{\tilde{\beta}} = \mathcal{S}_u\left(t,\hat{P},Q,R\right)$ for a fixed $\hat{P}$ and $t \in \tau^{(k,j)}$ (or $t = \tau^{(k,j)}$ since reachability analysis returns both time-point and time-interval solutions) can be computed via

$$\hat{s}_u\left(t,Q,R,\tilde{\beta}\right) = \bar{u}\left(\beta\left(\tilde{\beta}\right), \hat{P}^{(j)}\right) + K^{(j)}(Q,R)\left(r_x\left(t,Q,R,\tilde{\beta}\right) - r_{\tilde{x}_{\text{ff}}}\left(t,Q,R,\tilde{\beta}\right)\right),$$

where

$$r_{\tilde{x}_{\text{ff}}}\left(t,Q,R,\tilde{\beta}\right) = r_{x_{\text{ref}}}\left(t,\tilde{\beta}\right) + r_{\Delta\tilde{x}_{\text{ff}}}\left(t,Q,R,\tilde{\beta}\right).$$

We define

$$\begin{aligned} \hat{\mathcal{Z}}\left(\hat{\mathcal{S}}_u(t,Q,R)\right) &= \langle c_u(t,Q,R), G_u(t,Q,R)\rangle_Z, \\ \hat{\mathcal{Z}}\left(\left\{r_x\left(t,Q,R,\tilde{\beta}\right)\right\}_{\tilde{\beta}}\right) &= \langle c(t,Q,R), G(t,Q,R)\rangle_Z. \end{aligned}$$

Objective Function: Let $\hat{u}(t,x,Q,R) = u\left(t,x,\hat{P},Q,R\right)$. The objective function in (6.3a) can be approximated as (also see [98, Sec. 3.5.3])

$$\begin{aligned} &\max_{x(t,Q,R)\in\mathcal{R}(t,Q,R)} \left\{ \|x(t_\text{f},Q,R) - x_\text{f}\|_1 + \zeta \int_0^{t_\text{f}} \|\hat{u}(\xi,x,Q,R)\|_1 \,\mathrm{d}\xi \right\} \\ &\leq \|c(t_\text{f},Q,R) - x_\text{f}\|_1 + \left\| [G(t_\text{f},Q,R)]_{(:)} \right\|_1 + \max_{x(t,Q,R)\in\mathcal{R}(t,Q,R)} \zeta \int_0^{t_\text{f}} \|\hat{u}(\xi,x,Q,R)\|_1 \,\mathrm{d}\xi \\ &\leq \|c(t_\text{f},Q,R) - x_\text{f}\|_1 + \left\| [G(t_\text{f},Q,R)]_{(:)} \right\|_1 + \max_{\|\tilde{\beta}\|_\infty \leq 1} \sum_{k=0}^{m_r-1} \sum_{j=0}^{m_\text{c}-1} \Delta t \zeta \left\| \hat{s}_u\left(\tau^{(k,j)},Q,R,\tilde{\beta}\right) \right\|_1 \\ &\leq \left\| \begin{bmatrix} c(t_\text{f},Q,R) - x_\text{f} \\ [G(t_\text{f},Q,R)]_{(:)} \end{bmatrix} \right\|_1 + \Delta t \zeta \sum_{k=0}^{m_r-1} \sum_{j=0}^{m_\text{c}-1} \left\| \begin{bmatrix} c_u\left(\tau^{(k,j)},Q,R\right) \\ \left[G_u\left(\tau^{(k,j)},Q,R\right)\right]_{(:)} \end{bmatrix} \right\|_1, \end{aligned} \tag{6.9}$$

where the first inequality follows from the triangle inequality, the second inequality follows from applying the worst-case value during the entire interval of length Δt, and the last inequality again follows from the triangle inequality.

Constraints: Denote with $\rho_u^{(\mathcal{Z})}\left(\cdot, \tau^{(k,j)}, Q, R\right)$ the support function of $\hat{\mathcal{Z}}\left(\hat{\mathcal{S}}_u\left(t, Q, R\right)\right)$. For the input constraint from (6.3b) with $\mathcal{U} = \langle C_\mathcal{U}, d_\mathcal{U} \rangle_H$, it holds that

$$\begin{aligned}
& \forall t \in [0, t_\mathrm{f}] : \hat{\mathcal{S}}_u\left(t, Q, R\right) \subseteq \mathcal{U} && (6.10)\\
\overset{Prop.\ 4.4}{\Longleftarrow} \quad & \forall t \in [0, t_\mathrm{f}] : \hat{\mathcal{Z}}\left(\hat{\mathcal{S}}_u\left(t, Q, R\right)\right) \subseteq \mathcal{U} && (6.11)\\
\Longleftrightarrow \quad & \bigcup_{\substack{0 \le k \le m_r \\ 0 \le j \le m_\mathrm{c}-1}} \hat{\mathcal{Z}}\left(\hat{\mathcal{S}}_u\left(\tau^{(k,j)}, Q, R\right)\right) \subseteq \mathcal{U} && (6.12)\\
\overset{Prop.\ 4.6}{\Longleftrightarrow} \quad & \rho_u^{(\mathcal{Z})}\left(C_\mathcal{U}, [0, t_\mathrm{f}], Q, R\right) \le d_\mathcal{U}, && (6.13)
\end{aligned}$$

where

$$\rho_u^{(\mathcal{Z})}\left(C_\mathcal{U}, [0, t_\mathrm{f}], Q, R\right) = \max_{\substack{0 \le k \le m_r \\ 0 \le j \le m_\mathrm{c}-1}} \rho_u^{(\mathcal{Z})}\left(C_\mathcal{U}, \tau^{(k,j)}, Q, R\right).$$

Further, denote with $\rho_x^{(\mathcal{Z})}\left(\cdot, t, Q, R\right)$ the support function of $\hat{\mathcal{Z}}\left(\left\{r_x\left(t, Q, R, \tilde{\beta}\right)\right\}_{\tilde{\beta}}\right)$. Then

$$\begin{aligned}
& \mathcal{R}\left([0, t_\mathrm{f}], \hat{P}, Q, R\right) \subseteq \mathcal{X}\\
\overset{Prop.\ 4.4}{\Longleftarrow} \quad & \bigcup_{\substack{0 \le k \le m_r \\ 0 \le j \le m_\mathrm{c}-1}} \hat{\mathcal{Z}}\left(\left\{r_x\left(\tau^{(k,j)}, Q, R, \tilde{\beta}\right)\right\}_{\tilde{\beta}}\right) \subseteq \mathcal{X}\\
\overset{Prop.\ 4.6}{\Longleftrightarrow} \quad & \rho_x^{(\mathcal{Z})}\left(C_\mathcal{X}, [0, t_\mathrm{f}], Q, R\right) \le d_\mathcal{X}, && (6.14)
\end{aligned}$$

where

$$\rho_x^{(\mathcal{Z})}\left(C_\mathcal{X}, [0, t_\mathrm{f}], Q, R\right) = \max_{\substack{0 \le k \le m_r-1 \\ 0 \le j \le m_\mathrm{c}-1}} \rho_x^{(\mathcal{Z})}\left(C_\mathcal{X}, \tau^{(k,j)}, Q, R\right),$$

and

$$\begin{aligned}
& \mathcal{R}\left(t_\mathrm{f}, Q, R\right) \subseteq \mathcal{X}_\mathrm{f}\\
\overset{Prop.\ 4.4}{\Longleftarrow} \quad & \hat{\mathcal{Z}}\left(\left\{r_x\left(t_\mathrm{f}, Q, R, \tilde{\beta}\right)\right\}_{\tilde{\beta}}\right) \subseteq \mathcal{X}_\mathrm{f}\\
\overset{Prop.\ 4.6}{\Longleftrightarrow} \quad & \rho_x^{(\mathcal{Z})}\left(C_{\mathcal{X}_\mathrm{f}}, t_\mathrm{f}, Q, R\right) \le d_{\mathcal{X}_\mathrm{f}}. && (6.15)
\end{aligned}$$

Optimization Problem: Collecting (6.9) and (6.13) to (6.15) yields the optimization problem

$$\min_{Q,R} \left\| \begin{bmatrix} c\left(t_\mathrm{f}, Q, R\right) - x_\mathrm{f} \\ \left[G\left(t_\mathrm{f}, Q, R\right)\right]_{(:)} \end{bmatrix} \right\|_1 + \Delta t \zeta \sum_{k=0}^{m_r-1} \sum_{j=0}^{m_\mathrm{c}-1} \left\| \begin{bmatrix} c_u\left(\tau^{(k,j)}, Q, R\right) \\ \left[G_u\left(\tau^{(k,j)}, Q, R\right)\right]_{(:)} \end{bmatrix} \right\|_1, \tag{6.16a}$$

$$\text{s.t. } \rho_u^{(\mathcal{Z})}\left(C_\mathcal{U}, [0, t_\mathrm{f}], Q, R\right) \leq d_\mathcal{U}, \tag{6.16b}$$

$$\rho_x^{(\mathcal{Z})}\left(C_\mathcal{X}, [0, t_\mathrm{f}], Q, R\right) \leq d_\mathcal{X}, \tag{6.16c}$$

$$\rho_x^{(\mathcal{Z})}\left(C_{\mathcal{X}_\mathrm{f}}, t_\mathrm{f}, Q, R\right) \leq d_{\mathcal{X}_\mathrm{f}}, \tag{6.16d}$$

$$Q \in \left\{ \mathrm{diag}\left(q\right) \,\middle|\, q \in \left[\underline{q}, \bar{q}\right] \right\}, \tag{6.16e}$$

$$R \in \left\{ \mathrm{diag}\left(r\right) \mid r \in [\underline{r}, \bar{r}] \right\}, \tag{6.16f}$$

where we start from $Q = \mathrm{diag}\left(\bar{q}\right)$ and $R = \mathrm{diag}\left(\bar{r}\right)$ with $\left[\underline{q}, \bar{q}\right] \subseteq \mathbb{R}_+^{n_x}$ and $[\underline{r}, \bar{r}] \subseteq \mathbb{R}_+^{n_u}$: To avoid semi-definite constraints for Q and R, the authors in [101] restrict the weighting matrices to positive diagonal matrices. Since the computation of $K^{(j)}\left(Q, R\right)$ for $0 \leq m_\mathrm{c} - 1$ in dependence of Q and R is invariant when multiplying Q and R with some positive scalar, i.e., $K^{(j)}\left(Q, R\right) = K^{(j)}\left(sQ, sR\right)$ for $s > 0$, the authors in [101] further propose to set $\underline{q}_1 = \bar{q}_1 = 1$.

6.2.5 Initial Guess For Feedback Synthesis

Since the feedforward controller is fixed and $K^{(j)}\left(Q, R\right)$ for $0 \leq j \leq m_\mathrm{c} - 1$ is a function of Q and R, it suffices to find an initial guess for Q and R. Because solving (6.16) requires the evaluation of the closed-loop reachable set for each optimization iteration (see Sec. 6.2.4 for details), providing good initial guesses $\bar{Q} \in \mathbb{S}_{++}^{n_x \times n_x}$ and $\bar{R} \in \mathbb{S}_{++}^{n_u \times n_u}$ can save a lot of computation time. Such an initial guess computation is outside of the scope of this thesis; that said, a possible initial guess – an idea from the code of the authors of [101] – can be obtained by solving (6.16), but replacing the outer approximations of the closed-loop reachable set for the nonlinear dynamics with approximations using reachability analysis of a linearized sequence of LTI systems. This has the advantage that linear reachability analysis is much more efficient than reachability analysis for nonlinear systems.

6.2.6 Computational Complexity

We again use Ass. 5.1. We distinguish between the complexity of the offline controller synthesis and the complexity of applying the controller online, both in n.

6.2.6.1 Offline Complexity

Since we already discussed the offline complexity of GSC (see Sec. 5.2.6.1), we focus on the complexity of computing the continuous feedback controller.

Closed-loop Reachability Analysis: Computing the reachable set for a given combined controller has a complexity of at most $O\left(n^5\right)$ [59, Sec. 4.1.4].

Feedback Optimization Problem: The computation of the closed-loop reachable set in each optimization iteration when solving (6.16) naturally dominates the computational complexity with $O\left(n^5\right)$ (if the conservative polynomialization algorithm is used; see Sec. 2.7.3). Since (6.16) cannot be reformulated as a smooth optimization problem (see Sec. 6.2.7 for details), there is no straightforward way to bound the number of optimization iterations.

Overall Complexity: Since the optimization problem to solve is not differentiable, we cannot make a statement about the overall complexity of ROC.

6.2.6.2 Online Complexity

Evaluation of the combined controller requires the evaluation of the GSC feedforward controller, one matrix vector multiplication between the feedback matrix and $x-\tilde{x}_{\mathrm{ff}}$, and the forward simulation of $\tilde{x}_{\mathrm{ff}}$. If a fixed-step ODE solver is used, the online computational complexity for the forward simulation of $\tilde{x}_{\mathrm{ff}}$ is at most $O\left(b\left(n_x e + c_{\mathrm{GSC}}^{(\mathrm{on})}(n)\right)\right) \overset{Ass.\ 5.1}{=} O\left(n^2 + c_{\mathrm{GSC}}^{(\mathrm{on})}(n)\right)$, where $b \in \mathbb{N}_+$ collects both the number of fixed steps as well as the number of fixed function evaluations per step: We have n_x dimensions, the evaluation of f in each dimension requires e elementary operations, and we need to evaluate the feedforward controller b times for the computation of the feedforward input.

With the complexity of the matrix-vector multiplication of at most $O\left(n_u n_x\right) = O\left(n^2\right)$, the online complexity of the ROC approach is

$$\begin{aligned} O\left(c_{\mathrm{ROC}}^{(\mathrm{on})}(n)\right) &= O\left(c_{\mathrm{GSC}}^{(\mathrm{on})}(n) + n^2\right) \\ &\overset{(5.15)}{=} \begin{cases} O\left(n^2\right), & \mathcal{X}^{(0)} = \mathcal{X}^{(0)} \downarrow_1 \\ O\left(c_{\mathrm{LP}}(n) + n^2\right), & \text{otherwise} \end{cases}. \end{aligned} \tag{6.17}$$

6.2.7 Discussion

Using the GSC approach for the feedforward synthesis, ROC is conceptually an intuitive extension for the inclusion of a continuous feedback term: By directly optimizing over outer approximations of the reachable set, constraints can be easily checked using Prop. 4.6. With an appropriate choice of both the tightened feedforward input constraint set as well as the adapted state and final state constraint sets, ROC significantly improves the control result compared to GSC when the disturbance sets get larger [101, Sec. VI].

However, this direct optimization over the reachable set in (6.16) requires the recomputation of an outer approximation of the extended reachable set in each optimization iteration. Further, the absolute values in (6.16) cannot be resolved as done previously: Such a reformulation requires that the same generator for different values of Q and R is

used for each resulting inequality. Since we have no way of identifying the same generator for different values of Q and R, this approach cannot be used. Since a reformulation of absolute values is not possible, differentiability of (6.16) is strictly speaking not given. Thus, applying most gradient-based solvers to (6.16) will yield no theoretical convergence guarantee. Furthermore, gradients for (6.16) cannot be computed analytically and thus gradient information has to be approximated using finite differences (see [53, Chap. 5] for details on finite differences) if a gradient-based solver is used. Additionally, ROC requires the user to choose the adapted constraint sets – such as adapted state constraints and tightened feedfoward input constraints – appropriately, which makes the approach non-trivial to apply for non-experts. Lastly, ROC first computes the feedforward controller and only then synthesizes the continuous feedback controller. Since the choice of the feedforward controller directly influences the feedback controller, this separation into two synthesis steps might result in a sub-optimal controller (also see Sec. 6.3.10).

6.3 Iterative Polynomial Reachset Optimal Control

The ROC approach first synthesizes a feedforward controller and then computes the continuous feedback control law while keeping the feedforward controller fixed. Not only does this generally result in a sub-optimal solution, but it also introduces additional algorithm parameters, such as the adapted input and state constraints, which have to be set appropriately to arrive at a performant controller; this can limit the applicability of ROC by non-experts. To avoid these additional parameters, we present the iterative polynomial reachset optimal control (iPROC) approach from our work in [35], which solves the synthesis problem for the combined controller simultaneously, i.e., without separation of the feedforward and feedback controller synthesis: Since (6.3) is generally not efficiently solvable, we approximate (6.3) – similarly to iPGSC – using a non-differentiable cost function in the combined controller parameters. By locally approximating this non-differentiable cost function with a differentiable optimization problem, we can iteratively update our controller parameters based on this local approximation, where we control its accuracy by restricting the admissible controller parameters for the optimization problem to a bounded trust region.

As it turns out, parameterizing $K^{(j)} = K^{(j)}(z)$ for $0 \leq j \leq m_\mathrm{c} - 1$ in the controller parameters $z = \left[P_{(:)}^T, \quad Q_{(:)}^T, \quad R_{(:)}^T\right]^T$ using LQR control – as used in Sec. 6.2 and originally proposed in [101, Sec. IV.B] – is also beneficial here, where $P \in [-1, 1]^{m_\mathrm{c} n_u \times a}$ are the feedforward controller parameters of the controller template from (5.20), and $Q \in \mathbb{S}_{++}^{n_x \times n_x}$ and $R \in \mathbb{S}_{++}^{n_u \times n_u}$ are the feedback controller parameters. For now, we simply take this parameterization as a given and defer its motivation and introduction to Sec. 6.3.1. Fig. 6.2 visualizes the general idea of iPROC, where each step of Alg. 6 is briefly described subsequently.

(i) Starting from an initial guess for both the feedforward controller parameters $\bar{P}$ and feedback controller parameters $\bar{Q}$ and $\bar{R}$ (l. 3), we first compute an outer approximation to the undisturbed feedforward reachable set for the current feedforward controller (l. 5).

(ii) Using this feedforward outer approximation, we then construct a locally accurate approximation of the undisturbed feedforward reachable set around the current controller parameters (l. 6).

(iii) We evaluate the cost of the initial controller, parameterized by the initial controller parameters, using an outer approximation of the closed-loop reachable set of this initial controller (l. 7–8). We then iteratively update the current controller parameters by executing the following steps:

  (a) Starting from the current controller parameters, we solve the trust-region subproblem – which is a locally accurate, differentiable approximation of the controller cost – to obtain a new controller candidate (l. 10): To ensure accuracy, we restrict the domain of controller parameters z to a bounded trust region, whose size is determined by the feedforward and feedback trust-region radii γ and $\eta \in (0, 1]$.

(b) We try to compute an outer approximation to the closed-loop reachable set of the new controller candidate (l. 13–18): For very large trust-region radii, the inaccurate approximation of the reachable set may lead to a controller for which the closed-loop reachable set computation encounters numerical errors. In that case, we shrink the trust-region radii and restart the iteration.

(c) We then compute the cost of the newly synthesized controller (l. 19). (l. 20).

(d) To ensure the accuracy of the trust-region subproblem, we compare the difference in the controller cost to the difference predicted by the trust-region subproblem and tune the trust-region radii accordingly (l. 20).

(e) If the newly synthesized controller has a lower cost than the current controller candidate, the step is accepted and we update all necessary values (l. 23–24). We terminate the algorithm if the change from the previous best parameters to the current best parameters is small enough (l. 26). Otherwise, we recompute an updated parameterized feedforward reachable set (l. 29) and continue with (iii-a).

The remainder of this section explains iPROC in more detail: We start by introducing the feedback matrix parameterization using LQR control in Sec. 6.3.1 and describe the closed-loop reachable set computation of the combined controller in Sec. 6.3.2. We then derive the controller cost as an approximation to (6.3) in Sec. 6.3.3. Because the evaluation of the cost is computationally expensive, we compute a locally accurate approximation to the parameterized feedforward reachable set in Sec. 6.3.4, which is then used in Sec. 6.3.5 to construct the trust-region subproblem in each iteration to locally approximate the controller cost. In Sec. 6.3.6, we then derive tuning rules for the trust-region radii, which limit the domain of the controller parameters to a local neighborhood around the current controller parameters, to guarantee an accurate approximation of the controller cost. Since the trust-region subproblem is a differentiable optimization problem, we demonstrate the computation of necessary derivatives using differentials in Sec. 6.3.7. Since the proposed algorithm strongly benefits from good initial guesses for the controller parameters, we briefly discuss possible initial guesses in Sec. 6.3.8 and then derive the computational complexity, both online and offline, for the iPROC approach in Sec. 6.3.9. Finally, we demonstrate the applicability of iPROC with a comparison to the ROC approach in numerical experiments in Sec. 6.3.10, and discuss advantages of the algorithm in Sec. 6.3.11.

6.3.1 Feedback Matrix Parameterization

Directly optimizing over $K^{(j)}$ for $0 \leq j \leq m_\mathrm{c} - 1$ requires $m_\mathrm{c} n_x n_u$ variables since $K^{(j)} \in \mathbb{R}^{n_u \times n_x}$. In Sec. 6.2.1, this same parameterization has already been introduced; in this section, we focus on the necessary, additional requirements to use this parameterization for iPROC. First, we briefly motivate the choice of LTI systems for the LQR control approach. Then, we further show that the LQR feedback matrix is continuously differentiable in the controller parameters z, which is a necessary requirement to later optimize over them (see Sec. 6.3.5).

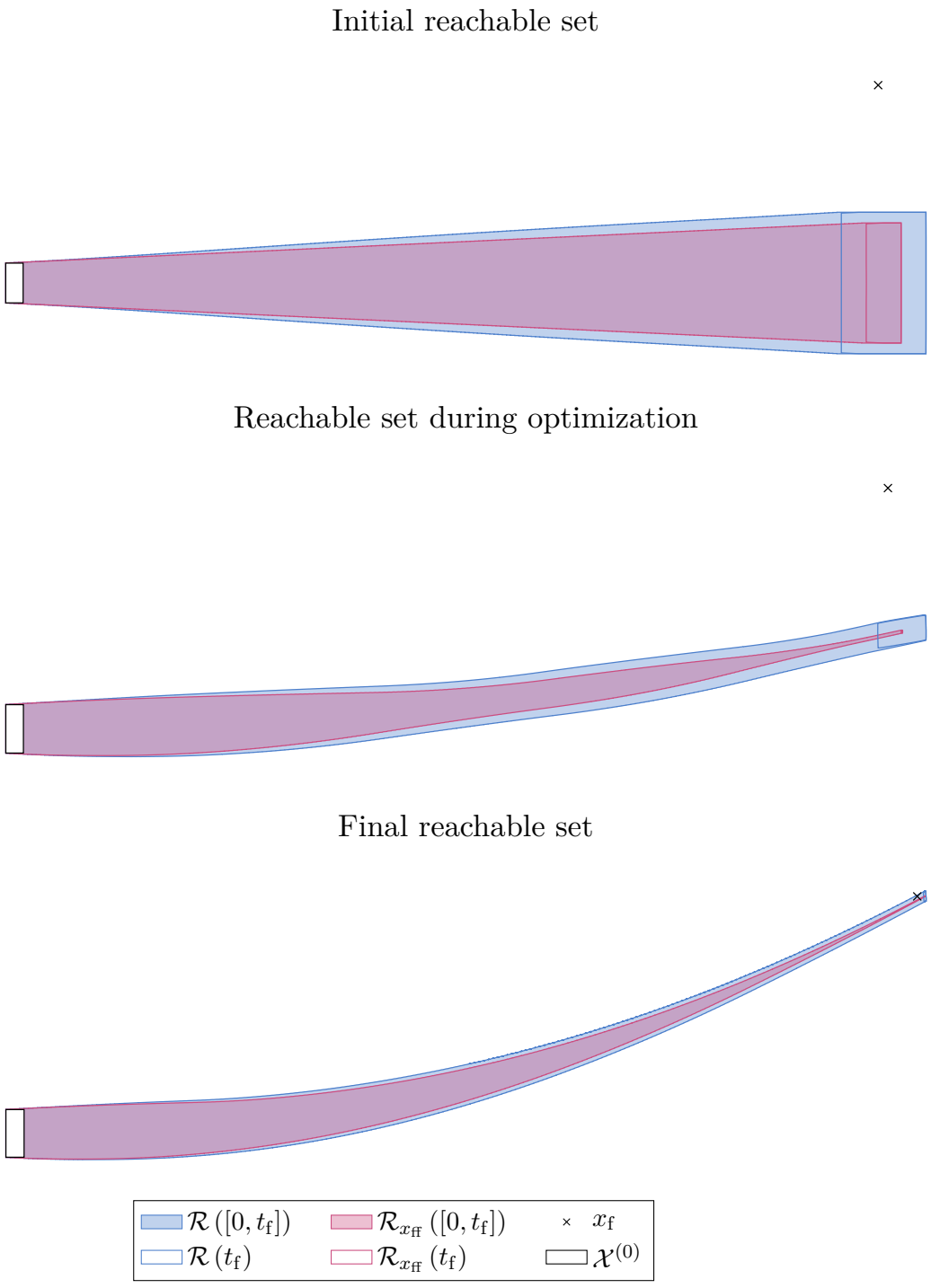


Figure 6.2: Visualization of the iPROC approach. Starting from an initial guess for both the feedforward controller parameters as well as state and input weighting matrices of the LQR formulation, the feedforward controller steers the reachable set closer to the target state while the feedback controller simultaneously minimizes the effect of the disturbance.

Algorithm 6 Iterative Polynomial Reachset Optimal Control

function ITERPOLYREACHSETOPTIMALCONTROL($\mathcal{X}^{(0)}, \mathcal{U}, \mathcal{X}, \mathcal{X}_\mathrm{f}, \mathcal{W}, m_\mathrm{c}, x_\mathrm{f}, \kappa$)
  $\gamma = \eta = 1$
  $\left[\bar{P}, \bar{Q}, \bar{R}\right] =$ INITGUESS(...) ▷ Sec. 6.3.8
  $\bar{z} = \left[\bar{P}_{(:)}^T, \quad \bar{Q}_{(:)}^T, \quad \bar{R}_{(:)}^T\right]^T$
  $\bar{\mathcal{R}}_{x_\mathrm{ff}} =$ REACHFF($\mathcal{X}^{(0)}, \bar{P}, t_\mathrm{f}, \kappa$) ▷ see Sec. 5.2.5.2 with $m_\mathrm{s} = h = 1$
  $\left[\tilde{\mathcal{R}}_{x_\mathrm{ff}}, \gamma\right] =$ REACHPARAMFF($\mathcal{X}^{(0)}, \bar{P}, \gamma, \bar{\mathcal{R}}_{x_\mathrm{ff}}, t_\mathrm{f}, \kappa$) ▷ Sec. 5.4.2
  $\left[\bar{\mathcal{R}}_\mathrm{ext}, \bar{K}\right] =$ REACH($\mathcal{X}^{(0)}, \bar{z}, \tilde{\mathcal{R}}_{x_\mathrm{ff}}, t_\mathrm{f}, \kappa$) ▷ Sec. 6.3.2
  $\bar{J} =$ COST($\bar{\mathcal{R}}_\mathrm{ext}$) ▷ Sec. 6.3.3
  for $k = 1; k \leq l_\mathrm{max}; k{+}{+}$ **do**
    $\hat{z} =$ TRSUBPROBLEM($\bar{z}, \gamma, \eta, \tilde{\mathcal{R}}_{x_\mathrm{ff}}, \bar{\mathcal{R}}_\mathrm{ext}, \mathcal{U}, \mathcal{X}, \mathcal{X}_\mathrm{f}$) ▷ Sec. 6.3.5
    try
      $\left[\hat{\mathcal{R}}_\mathrm{ext}, \hat{K}\right] =$ REACH($\mathcal{X}^{(0)}, \hat{z}, \tilde{\mathcal{R}}_{x_\mathrm{ff}}, t_\mathrm{f}, \kappa$) ▷ Sec. 6.3.2
    catch
      $\gamma = \frac{1}{2}\gamma$
      $\eta = \frac{1}{2}\eta$
      $\left[\tilde{\mathcal{R}}_{x_\mathrm{ff}}(t, P), \gamma\right] =$ REACHPARAMFF($\mathcal{X}^{(0)}, \bar{P}, \gamma, \bar{\mathcal{R}}_{x_\mathrm{ff}}, t_\mathrm{f}, \kappa$) ▷ Sec. 5.4.2
      continue
    end try
    $\hat{J} =$ COST($\hat{\mathcal{R}}_\mathrm{ext}$) ▷ Sec. 6.3.3
    $[\gamma, \eta] =$ TUNERADII($\gamma, \eta, \tilde{\mathcal{R}}_{x_\mathrm{ff}}, \bar{\mathcal{R}}_\mathrm{ext}, \hat{\mathcal{R}}_\mathrm{ext}, \bar{J}, \hat{J}$) ▷ Sec. 6.3.6
    if $\hat{J} < \bar{J}$ **then**
      $\nu = \mathrm{tol}\left(\hat{J}, \bar{J}\right)$
      $\bar{z} = \hat{z}$, $\bar{J} = \hat{J}$, $\bar{\mathcal{R}}_\mathrm{ext} = \hat{\mathcal{R}}_\mathrm{ext}$, $\bar{K} = \hat{K}$
      $\bar{\mathcal{R}}_{x_\mathrm{ff}} = \left[\bar{\mathcal{R}}_\mathrm{ext}\right]_{(n_x + (1:n_x))}$
      if $\nu < \mu$ **then**
        break
      end if
    end if
    $\left[\tilde{\mathcal{R}}_{x_\mathrm{ff}}(t, P), \gamma\right] =$ REACHPARAMFF($\mathcal{X}^{(0)}, \bar{P}, \gamma, \bar{\mathcal{R}}_{x_\mathrm{ff}}, t_\mathrm{f}, \kappa$) ▷ Sec. 5.4.2
  end for
  return $\bar{P}$, $\bar{K}$, $\left[\bar{\mathcal{R}}_\mathrm{ext}\right]_{(1:n_x)}$
end function

LTI systems: We now derive the LTI systems required for the computation of the feedback matrices using LQR control. We split the state

$$x\left(t,z\right)=x_{\mathrm{ff}}\left(t,P\right)+\Delta x\left(t,z\right), \tag{6.18}$$

where P are the feedforward parameters, and $z=\left[P_{(:)}^{T},\quad Q_{(:)}^{T},\quad R_{(:)}^{T}\right]^{T}$ are the combined controller parameters with LQR weighting matrices $Q\in\mathbb{S}_{++}^{n_x\times n_x}$ and $R\in\mathbb{S}_{++}^{n_u\times n_u}$. Since the feedback controller tries to follow the undisturbed feedforward state x_{ff} and $x=x_{\mathrm{ff}}+\Delta x$, we ideally try to steer Δx to the origin. For the m_{c} feedback matrices $K\left(t,z\right)=K^{(j)}\left(z\right)$ with $t\in\tau^{(j)}$ and $0\leq j\leq m_{\mathrm{c}}-1$ that need to be computed, we can thus compute the m_{c} LTI systems required for LQR control by performing a first-order Taylor expansion of $\Delta\dot{x}=f\left(x,u,w\right)-f\left(x_{\mathrm{ff}},u_{\mathrm{ff}},0\right)$ around $x=x_{\mathrm{ff}}$, $u=u_{\mathrm{ff}}$ and $w=0$, which yields (arguments omitted where clear from the context for readability)

$$\begin{aligned}\Delta\dot{x}=\dot{x}-\dot{x}_{\mathrm{ff}}&=f\left(x,u,w\right)-f\left(x_{\mathrm{ff}},u_{\mathrm{ff}},0\right)\\&\approx f\left(x_{\mathrm{ff}},u_{\mathrm{ff}},0\right)+\left.\frac{\partial f\left(x,u,w\right)}{\partial x}\right|_{\substack{x=x_{\mathrm{ff}}\\u=u_{\mathrm{ff}}\\w=0}}\left(x-x_{\mathrm{ff}}\right)\\&\quad+\left.\frac{\partial f\left(x,u,w\right)}{\partial u}\right|_{\substack{x=x_{\mathrm{ff}}\\u=u_{\mathrm{ff}}\\w=0}}\left(u-u_{\mathrm{ff}}\right)+\left.\frac{\partial f\left(x,u,w\right)}{\partial w}\right|_{\substack{x=x_{\mathrm{ff}}\\u=u_{\mathrm{ff}}\\w=0}}w\\&\quad-f\left(x_{\mathrm{ff}},u_{\mathrm{ff}},0\right)\\&\approx A_{\mathrm{cl}}^{(j)}\left(z\right)\Delta x+E^{(j)}\left(P\right)w,\end{aligned} \tag{6.19}$$

since $u\left(t,x\left(0\right),z\right)=u_{\mathrm{ff}}\left(x\left(0\right),P^{(j)}\right)+K^{(j)}\left(z\right)\Delta x\left(t,z\right)$ for $t\in\tau^{(j)}$ with $0\leq j\leq m_{\mathrm{c}}-1$ (see (6.1)) and where

$$A^{(j)}\left(P\right)=\left.\frac{\partial f\left(x,u,w\right)}{\partial x}\right|_{\substack{x=x_{\mathrm{lin}}^{(j)}(P)\\u=u_{\mathrm{lin}}^{(j)}(P)\\w=0}}, \tag{6.20}$$

$$B^{(j)}\left(P\right)=\left.\frac{\partial f\left(x,u,w\right)}{\partial u}\right|_{\substack{x=x_{\mathrm{lin}}^{(j)}(P)\\u=u_{\mathrm{lin}}^{(j)}(P)\\w=0}}, \tag{6.21}$$

$$E^{(j)}\left(P\right)=\left.\frac{\partial f\left(x,u,w\right)}{\partial w}\right|_{\substack{x=x_{\mathrm{lin}}^{(j)}(P)\\u=u_{\mathrm{lin}}^{(j)}(P)\\w=0}}, \tag{6.22}$$

$$A_{\mathrm{cl}}^{(j)}\left(z\right)=A^{(j)}\left(P\right)+B^{(j)}\left(P\right)K^{(j)}\left(z\right), \tag{6.23}$$

with

$$\begin{aligned}x_{\mathrm{lin}}^{(j)}\left(P\right)&=\frac{1}{2}\left(\tilde{c}_{x_{\mathrm{ff}}}\left(jt_{\mathrm{c}},P\right)+\tilde{c}_{x_{\mathrm{ff}}}\left(\left(j+1\right)t_{\mathrm{c}},P\right)\right),\\u_{\mathrm{lin}}^{(j)}\left(P\right)&=c_{u_{\mathrm{ff}}}\left(P^{(j)}\right).\end{aligned}$$

Here,

$$\hat{\mathcal{Z}}\left(\left\{\bar{u}\left(\beta, P^{(j)}\right)\right\}_{\beta}\right) = \left\langle c_{u_{\mathrm{ff}}}\left(P^{(j)}\right), G_{u_{\mathrm{ff}}}\left(P^{(j)}\right)\right\rangle_{Z},\ 0 \leq j \leq m_{\mathrm{c}} - 1,$$
$$\hat{\mathcal{Z}}\left(\tilde{\mathcal{R}}_{x_{\mathrm{ff}}}(t, P)\right) = \left\langle \tilde{c}_{x_{\mathrm{ff}}}(t, P), \tilde{G}_{x_{\mathrm{ff}}}(t, P)\right\rangle_{Z},$$

are the zonotope outer approximations of $\left\{\bar{u}\left(\beta, P^{(j)}\right)\right\}_{\beta}$ and $\tilde{\mathcal{R}}_{x_{\mathrm{ff}}}(t, P)$ from (5.21) and (5.44) for $m_{\mathrm{s}} = 1$, whose centers $\tilde{c}_{x_{\mathrm{ff}}}(t, P)$ and $c_{u_{\mathrm{ff}}}\left(P^{(j)}\right)$ are used to define the linearization points for the linearized matrices in (6.20) to (6.22). As a result, we use the feedforward input and the parameterized feedforward reachable set to extract their centers as approximations for a reference trajectory parameterized in P; the zonotope outer approximation is a preferred approximation of the geometric center (see Ex. 5.2). Note that in (6.19), we replaced the implicit dependence of the linearized dynamics on the initial state $x_{\mathrm{ff}}(0) = x(0)$ with $\tilde{c}_{x_{\mathrm{ff}}}(0, P)$: This approximation is motivated by the fact that the feedforward controller generally minimizes the size of the undisturbed reachable set and the feedback controller minimizes the effect of the disturbance.

If controllability for all P can be assumed, the LQR parameterization $K(t, z) = K^{(j)}(z)$ using the weighting matrices $Q \in \mathbb{S}_{++}^{n_x \times n_x}$ and $R \in \mathbb{S}_{++}^{n_u \times n_u}$ – which are constant for the entire time horizon – and the system matrices $A^{(j)}(P)$ and input matrices $B^{(j)}(P)$ is then well-defined and motivated as follows: By design, LQR then yields the stable closed-loop system matrices $A_{\mathrm{cl}}^{(j)}(z)$ (see (6.23)) and thus tries to bring Δx, which is approximately described by the m_{c} systems $\left(A_{\mathrm{cl}}^{(j)}(z), E^{(j)}(P)\right)$, to the origin. While keeping Q and R constant over the entire horizon reduces the number of optimization variables, it may also reduce the overall performance of the controller as we limit the range of possible feedback matrices. If this extra performance is required, one may introduce separate weighting matrices for each of the m_{c} LTI systems. Alternatively, one can also directly optimize over $K^{(j)}$, in which case controllability is not required; however, this means that one may also generate non-stabilizing feedback matrices.

Differentiability: In Prop. 3.1, we showed that the abstracted synthesis problem can be replaced by a smooth optimization problem which shares its minimum. Since Prop. 3.1 requires all involved functions to be sufficiently smooth, we must show that the LQR feedback matrix $K^{(j)}(z)$ is continuously differentiable with respect to z.

Lemma 6.1 (Continuous Differentiability of LQR Matrix)**.** *Let $(A(P), B(P))$ for $P \in [-1, 1]^{m_{\mathrm{c}} n_u \times a}$ denote an n_x-dimensional controllable LTI system, where $A(P)$ and $B(P)$ are k-times continuously differentiable in each matrix element. Then the LQR gain matrix $K(z)$ for the system $(A(P), B(P))$ and weight matrices $Q \in \mathbb{S}_{++}^{n_x \times n_x}$ and $R \in \mathbb{S}_{++}^{n_u \times n_u}$ is k-times continuously differentiable with respect to the collected variables $z = \left[P_{(:)}^T,\ Q_{(:)}^T,\ R_{(:)}^T\right]^T$.*

Proof. We first prove the differentiability of X with respect to z. The algebraic Riccati equation (see (2.25)) is

$$F(z) = A(P)^T X(z) + X(z) A(P) - X(z) B(z) R^{-1} B(P)^T X(z) + Q = 0.$$

From the implicit function theorem [27, Th. 1B.1, Prop. 1B.5], we know that $X_{(:)}(z)$ exists, is unique and k-times differentiable in a neighborhood around z if $\det\left(\frac{\mathrm{d}F_{(:)}}{\mathrm{d}X_{(:)}}\right) \neq 0$. Since $K = -R^{-1}B^T X$ (see (2.24)), we have

$$\begin{aligned}
\mathrm{d}F &= A^T\,\mathrm{d}X + \mathrm{d}X A - \mathrm{d}X B R^{-1} B^T X - X B R^{-1} B^T\,\mathrm{d}X \\
&= (A+BK)^T\,\mathrm{d}X + \mathrm{d}X\,(A+BK) \\
\overset{(2.1)}{\Longleftrightarrow}\ \mathrm{d}F_{(:)} &= \left(I_{n_x} \boxtimes (A+BK)^T + (A+BK)^T \boxtimes I_{n_x}\right)\mathrm{d}X_{(:)},
\end{aligned}$$

and thus

$$\frac{\mathrm{d}F_{(:)}}{\mathrm{d}X_{(:)}} = A_{\mathrm{cl}}^T \boxplus A_{\mathrm{cl}}^T,$$

where we used the definition of the Kronecker sum and $A_{\mathrm{cl}} = A + BK$. Since A_{cl} is stable by design, i.e., $\mathrm{real}\left(\lambda\left(A_{\mathrm{cl}}\right)\right) < 0$, it follows that $\mathrm{real}\left(\lambda\left(A_{\mathrm{cl}}^T \boxplus A_{\mathrm{cl}}^T\right)\right) < 0$ [70, Th. 13.16] and thus $\det\left(\frac{\mathrm{d}F_{(:)}}{\mathrm{d}X_{(:)}}\right) = \det\left(A_{\mathrm{cl}}^T \boxplus A_{\mathrm{cl}}^T\right) \neq 0$. As a result, $X(z)$ exists, is unique, and k-times continuously differentiable in a neighborhood around z, and since we made no assumptions about z, the result follows for all $P \in [-1,1]^{m_{\mathrm{c}} m_{\mathrm{s}} n_u \times a}$, $Q \in \mathbb{S}_{++}^{n_x \times n_x}$, and $R \in \mathbb{S}_{++}^{n_u \times n_u}$.

The differentiability of K then directly follows from (2.24) and the fact that $R \in \mathbb{S}_{++}^{n_u \times n_u}$. □

Remark: As it turns out, one can use Lem. 6.1 to derive an analytic approximation of the LQR matrix with arbitrary precision, which we now briefly sketch. The second-order Taylor expansion of $K_{ij}(z)$ for $1 \leq i \leq n_u$ and $1 \leq j \leq n_x$ around some expansion point $\bar{z} \in \mathbb{R}^{n_z}$ is given by

$$K_{ij}(z) = K_{ij}(\bar{z}) + J_{K_{ij}}(\bar{z})(z-\bar{z}) + \frac{1}{2}(z-\bar{z})^T H_{K_{ij}}(\bar{z})(z-\bar{z}) + R_{ij}^{(3)}(z-\bar{z}), \quad (6.24)$$

where $R_{ij}^{(3)}(z-\bar{z})$ is the remainder of the second-order Taylor expansion in (6.24) and [29, Corr. 7.2]

$$\lim_{z-\bar{z}\to 0} \frac{R_{ij}^{(3)}(z-\bar{z})}{\|z-\bar{z}\|_2^3} = 0. \quad (6.25)$$

Thus, if we want to compute $K(\hat{z})$ for some $\hat{z} \in \mathbb{R}^{n_z}$ starting from $\bar{z}$, we apply (6.24) recursively $n \in \mathbb{N}_+$ times, yielding

$$\begin{aligned}
K_{ij}\left(z^{(k+1)}\right) &= K_{ij}\left(z^{(k)}\right) + J_{K_{ij}}\left(z^{(k)}\right)\left(z^{(k+1)} - z^{(k)}\right) \\
&\quad + \frac{1}{2}\left(z^{(k+1)} - z^{(k)}\right)^T H_{K_{ij}}\left(z^{(k)}\right)\left(z^{(k+1)} - z^{(k)}\right) + R_{ij}^{(3)}\left(z^{(k+1)} - z^{(k)}\right) \\
&= K_{ij}\left(z^{(k)}\right) + J_{K_{ij}}\left(z^{(k)}\right)\Delta z + \frac{1}{2}\Delta z^T H_{K_{ij}}\left(z^{(k)}\right)\Delta z + R_{ij}^{(3)}(\Delta z),
\end{aligned}$$

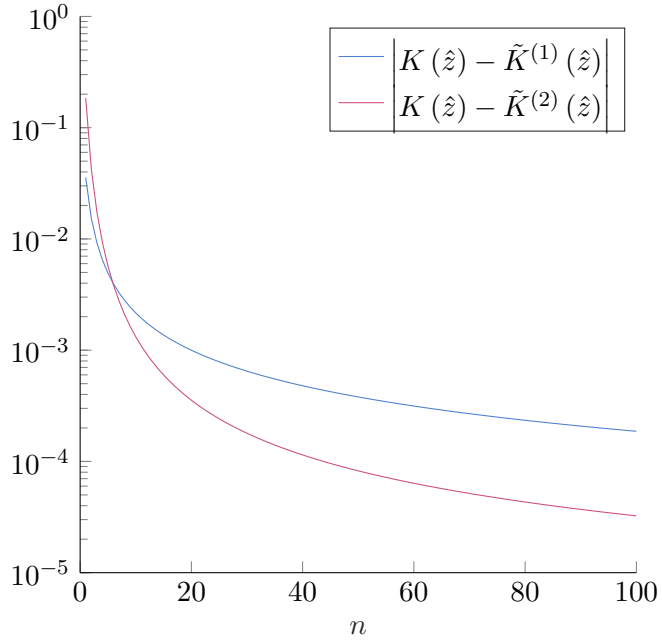


Figure 6.3: Visualization of the decrease in error for the recursive approximation of the LQR feedback matrix as defined in (6.26) and (6.27).

where $\forall k \in \{1, \ldots, n-1\} : \Delta z = z^{(k+1)} - z^{(k)} = \frac{\hat{z}-\bar{z}}{n}$, $\hat{z} = z^{(n)}$ and $J_{K_{ij}}\left(z^{(k)}\right)$ and $H_{K_{ij}}\left(z^{(k)}\right)$ can be computed using Lem. 6.1. Since $R_{ij}^{(3)}\left(\Delta z\right)$ tends to 0 at least as fast as $\|\Delta z\|_2^3$ for $\Delta z \to 0$ (see (6.25)) and increasing n dercreases $\|\Delta z\|_2$, the total error $nR_{ij}^{(3)}\left(\Delta z\right)$ shrinks for increasing n. We define

$$K_{ij}^{(1)}\left(z^{(k+1)}\right) = K_{ij}\left(z^{(k)}\right) + J_{K_{ij}}\left(z^{(k)}\right)\Delta z, \tag{6.26}$$

$$K_{ij}^{(2)}\left(z^{(k+1)}\right) = K_{ij}\left(z^{(k)}\right) + J_{K_{ij}}\left(z^{(k)}\right)\Delta z + \frac{1}{2}\Delta z^T H_{K_{ij}}\left(z^{(k)}\right)\Delta z. \tag{6.27}$$

The following example demonstrates the expected decrease in error with increasing n.

Example 6.1. Let an LTI system (A, B) with $A = 1$ and $B = 1$, where $n_x = 1$ and $n_u = 1$, be given. Further, we set $\bar{Q} = \bar{R} = 1$ so that $\bar{z} = \begin{bmatrix}1, & 1\end{bmatrix}^T$ since we assume that A and B are constant and thus omit the feedforward parameters in z. Fig. 6.3 visualizes the error of the first-order and second-order approximations in (6.26) and (6.27) with $\hat{z} = \begin{bmatrix}10, & 5\end{bmatrix}^T$ for increasing n. As expected, the error decreases for increasing n and the second-order approximation achieves a smaller error for the same n and large enough n.

■

6.3.2 Closed-Loop Reachable Set

During each iteration of Alg. 6, an outer approximation to the extended closed-loop reachable set needs to be computed. The combined controller for $t \in \tau^{(j)}$ with $0 \leq j \leq m_\mathrm{c} - 1$ is given by (see (6.1))

$$u\left(t, x, z\right) = u_\mathrm{ff}\left(x\left(0\right), P^{(j)}\right) + K^{(j)}\left(z\right)\left(x\left(t\right) - x_\mathrm{ff}\left(t\right)\right),$$

where $u_\mathrm{ff}\left(x\left(0\right), P^{(j)}\right)$ is defined as in (5.20).

To retain dependencies of the combined state, feedforward state, and feedforward input on the initial state during the closed-loop reachable set computation, we define the extended state $x_\mathrm{ext} = \left[x^T, \quad x_\mathrm{ff}^T, \quad u_\mathrm{ff}^T\right]^T$ with the extended flow given by

$$f_\mathrm{ext}\left(x_\mathrm{ext}, w\right) = \begin{bmatrix} f\left(x, u_\mathrm{ff} + K\left(t, z\right)\left(x - x_\mathrm{ff}\right), w\right) \\ f\left(x_\mathrm{ff}, u_\mathrm{ff}, 0\right) \\ 0 \end{bmatrix},$$

and the extended initial set

$$\mathcal{X}_\mathrm{ext}^{(0)}\left(P\right) = \left\{ \begin{bmatrix} x^{(0)}\left(\beta\right) \\ x^{(0)}\left(\beta\right) \\ \bar{u}\left(\beta, P^{(0)}\right) \end{bmatrix} \right\}_\beta.$$

Reachability analysis then yields

$$\mathcal{R}_\mathrm{ext}\left(t, z\right) = \left\{ \begin{bmatrix} r_x\left(t, \beta, \xi, z\right) \\ r_{x_\mathrm{ff}}\left(t, \beta, \xi, P\right) \\ \bar{u}\left(\beta, P^{(0)}\right) \end{bmatrix} \right\}_{\beta,\xi},$$

for $t \in \left[0, 1\right] t_\mathrm{c}$, where r_x and r_{x_ff} denote the respective generating functions and the dependent factor ξ is caused by the abstraction and reduction errors as well as the convex combination required for the computation of the time-interval reachable sets. The extended initial set for $t \in \left[1, 2\right] t_\mathrm{c}$ then is

$$\mathcal{X}_\mathrm{ext}^{(1)}\left(z\right) = \left\{ \begin{bmatrix} r_x\left(t_\mathrm{c}, \beta, \xi, z\right) \\ r_{x_\mathrm{ff}}\left(t_\mathrm{c}, \beta, \xi, P\right) \\ \bar{u}\left(\beta, P^{(1)}\right) \end{bmatrix} \right\}_{\beta,\xi},$$

from which the remaining $m_\mathrm{c} - 1$ steps until $t = t_\mathrm{f}$ follow analogously.

6.3.3 Cost of the Controller

In this section, we approximate (6.3) with a cost function. To that end, let the extended closed-loop reachable set $\mathcal{R}_\mathrm{ext}\left(t, z\right)$ at z (see Sec. 6.3.2), containing an outer approximation of the closed-loop reachable set $\mathcal{R}\left(t, z\right)$ and an outer approximation of

the closed-loop feedforward reachable set $\mathcal{R}_{x_{\text{ff}}}(t, P)$, be available for $m_r \in \mathbb{N}_+$ reachability steps, i.e., interval sets for $t \in [k, k+1]\frac{t_c}{m_r}$ and time point sets for $t = k\frac{t_c}{m_r}$ with $0 \leq k \leq m_c m_r - 1$ are given.

The remainder of this section is structured as follows: First, we review the split of the state and the input, which simplifies all subsequent computations. Then, we approximate the objective function and constraints of (6.3) and form the cost as an approximation to (6.3).

State and input separation: Let the split of the state be defined as in (6.18), where P are the feedforward parameters, and $z = \left[P_{(:)}^T, \quad Q_{(:)}^T, \quad R_{(:)}^T\right]^T$ are the combined controller parameters (for details see Sec. 6.3.1)). Further, let

$$\Delta x(t, z) \in \mathcal{S}_{\Delta x}(t, z) = \mathcal{R}(t, z) \oplus_e \left(-\mathcal{R}_{x_{\text{ff}}}(t, P)\right), \tag{6.28}$$

collect all Δx as defined in (6.18) (we use $\mathcal{S}_{\Delta x}$ instead of $\mathcal{R}_{\Delta x}$ since a reachable set requires a flow equation according to Def. 2.25). By definition of the exact sum, we have

$$x(t, z) \in \mathcal{R}(t, z) = \mathcal{R}_{x_{\text{ff}}}(t, P) \oplus_e \mathcal{S}_{\Delta x}(t, z) \subseteq \mathcal{R}_{x_{\text{ff}}}(t, P) \oplus \mathcal{S}_{\Delta x}(t, z), \tag{6.29}$$

where the last inclusion follows from the fact that ignoring dependencies always results in an outer approximation. By removing these dependencies, $\mathcal{R}_{x_{\text{ff}}}(t, P)$ can be computed independently of $\mathcal{S}_{\Delta x}(t, z)$. Naturally, the removal of these dependencies introduces conservatism but we argue that this does not cause a large outer approximation: Without any disturbances, $\mathcal{S}_{\Delta x}(t, z) = \emptyset$ and thus $\mathcal{R}(t, z) = \mathcal{R}_{x_{\text{ff}}}(t, P)$. As the disturbance set grows, the size of $\mathcal{S}_{\Delta x}(t, z)$ is mainly affected by the size of the disturbance set (see Sec. 6.3.4 for the derivation of the approximated flow of Δx).

Similarly, one can define an input separation following (6.4) as

$$\mathcal{S}_u(t, z) = \mathcal{S}_{u_{\text{ff}}}(t, P) \oplus_e S_{\Delta u}(t, z) \subseteq \mathcal{S}_{u_{\text{ff}}}(t, P) \oplus S_{\Delta u}(t, z), \tag{6.30}$$

where the set of feedforward inputs is given by $\mathcal{S}_{u_{\text{ff}}}(t, z) = \left\{\bar{u}\left(\beta, P^{(j)}\right)\right\}$ for $t \in \tau^{(j)}$ with $0 \leq j \leq m_c - 1$ and $\mathcal{S}_{\Delta u}(t, z)$ contains all input deviations $\Delta u(t, z) = K(t, z)\Delta x(t, z)$ as defined in (6.5) and (6.6).

Objective: Based on the separation of the reachable set into the feedforward reachable set $\mathcal{R}_{x_{\text{ff}}}(t, P)$ and the set of deviation states $\mathcal{S}_{\Delta x}(t, z)$ from (6.29) as well as the corresponding separation of the combined input into the feedforward input set $\mathcal{S}_{u_{\text{ff}}}(t, P)$ and the set of deviation inputs $\mathcal{S}_{\Delta u}(t, z)$ from (6.30), we bound the objective function

in (6.3a) from above by

$$
\begin{aligned}
&\max_{x(t,z)\in\mathcal{R}(t,z)} \left\{ \left\| x\left(t_\mathrm{f},z\right) - x_\mathrm{f} \right\|_1 + \zeta \int_0^{t_\mathrm{f}} \left\| u\left(\tau,x,z\right) \right\|_1 \mathrm{d}\tau \right\} \\
= &\max_{\substack{x_\mathrm{ff}(t,P)\in\mathcal{R}_{x_\mathrm{ff}}(t,P) \\ \Delta x(t,z)\in\mathcal{S}_{\Delta x}(t,z) \\ \Delta u(t,z)\in\mathcal{S}_{\Delta u}(t,z) \\ \|\beta\|_\infty\leq 1}} \Bigg\{ \left\| x_\mathrm{ff}\left(t_\mathrm{f},P\right) + \Delta x\left(t,z\right) - x_\mathrm{f} \right\|_1 \\
&+ \zeta \sum_{j=0}^{m_\mathrm{c}-1} \sum_{k=0}^{m_r-1} \int_{jt_\mathrm{c}+kr}^{jt_\mathrm{c}+(k+1)\delta} \left\| \bar{u}\left(\beta,P^{(j)}\right) + \Delta u\left(\tau,z\right) \right\|_1 \mathrm{d}\tau \Bigg\} \\
\leq &\max_{\|\xi\|_\infty\leq 1} \left\| c_{x_\mathrm{ff}}\left(t_\mathrm{f},P\right) - x_\mathrm{f} + G_{x_\mathrm{ff}}\left(t_\mathrm{f},P\right)\xi \right\|_1 + \max_{\Delta x(t_\mathrm{f},z)\in\mathcal{Z}_{\Delta x}(t_\mathrm{f},z)} \left\| \Delta x\left(t_\mathrm{f},z\right) \right\|_1 \\
&+ \zeta \sum_{j=0}^{m_\mathrm{c}-1} \sum_{k=0}^{m_r-1} \int_{jt_\mathrm{c}+kr}^{jt_\mathrm{c}+(k+1)\delta} \Big(\max_{\|\xi\|_\infty\leq 1} \left\| c_{u_\mathrm{ff}}\left(P^{(j)}\right) + G_{u_\mathrm{ff}}\left(P^{(j)}\right)\xi \right\|_1 \\
&+ \max_{\Delta u(\tau,z)\in\mathcal{Z}_{\Delta u}(\tau,z)} \left\| \Delta u\left(\tau,z\right) \right\|_1 \Big) \mathrm{d}\tau \\
\leq &\left\| \begin{bmatrix} c_{x_\mathrm{ff}}\left(t_\mathrm{f},P\right) - x_\mathrm{f} \\ \left[G_{x_\mathrm{ff}}\left(t_\mathrm{f},P\right)\right]_{(:)} \end{bmatrix} \right\|_1 + 1_{n_x}^T \max\left(\rho_{\Delta x}^{(\mathcal{Z})}\left(-I_{n_x},t_\mathrm{f},z\right), \rho_{\Delta x}^{(\mathcal{Z})}\left(I_{n_x},t_\mathrm{f},z\right) \right) \\
&+ \zeta\delta \sum_{j=0}^{m_\mathrm{c}-1} \sum_{k=0}^{m_r-1} \Bigg(\left\| \begin{bmatrix} c_{u_\mathrm{ff}}\left(P^{(j)}\right) \\ G_{u_\mathrm{ff}}\left(P^{(j)}\right)_{(:)} \end{bmatrix} \right\|_1 \\
&+ 1_{n_u}^T \max\left(\rho_{\Delta u}^{(\mathcal{Z})}\left(-I_{n_u},\tau^{(j,k)},z\right), \rho_{\Delta u}^{(\mathcal{Z})}\left(I_{n_u},\tau^{(j,k)},z\right) \right) \Bigg),
\end{aligned}
$$

where

$$\mathcal{Z}_{x_\mathrm{ff}}\left(t,P\right) = \hat{\mathcal{Z}}\left(\mathcal{R}_{x_\mathrm{ff}}\left(t,P\right)\right) = \left\langle c_{x_\mathrm{ff}}\left(t,P\right), G_{x_\mathrm{ff}}\left(t,P\right) \right\rangle_Z, \tag{6.31a}$$

$$\mathcal{Z}_{\Delta x}\left(t,z\right) = \hat{\mathcal{Z}}\left(\mathcal{R}\left(t,z\right) \oplus_e \left(-\mathcal{R}_{x_\mathrm{ff}}\left(t,P\right)\right)\right), \tag{6.31b}$$

$$\mathcal{Z}_{u_\mathrm{ff}}\left(t,P\right) = \hat{\mathcal{Z}}\left(\left\{ \bar{u}\left(\beta,P^{(j)}\right) \right\}_\beta\right) = \left\langle c_{u_\mathrm{ff}}\left(P^{(j)}\right), G_{u_\mathrm{ff}}\left(P^{(j)}\right) \right\rangle_Z, \tag{6.31c}$$

$$\mathcal{Z}_{\Delta u}\left(t,z\right) = K\left(t,z\right)\mathcal{Z}_{\Delta x}\left(t,z\right), \tag{6.31d}$$

are the zonotope outer approximations of $\mathcal{R}_{x_\mathrm{ff}}\left(t,P\right)$, $\mathcal{S}_{u_\mathrm{ff}}\left(t,P\right)$, $\mathcal{S}_{\Delta x}\left(t,z\right)$, and $\mathcal{S}_{\Delta u}\left(t,z\right)$ with $\rho_{x_\mathrm{ff}}^{(\mathcal{Z})}\left(\cdot,t,z\right)$, $\rho_{\Delta x}^{(\mathcal{Z})}\left(\cdot,t,z\right)$, $\rho_{u_\mathrm{ff}}^{(\mathcal{Z})}\left(\cdot,t,z\right)$, and $\rho_{\Delta u}^{(\mathcal{Z})}\left(\cdot,t,z\right)$ denoting the corresponding support functions: The first inequality follows from the separation of x_ff and Δx as well as u_ff and Δu using the triangle inequality, where we replaced all sets with their respective zonotope outer approximations as defined in (6.31). The second inequality then follows from applying the triangle inequality again to separate the center and the generators for states and inputs (also see Sec. 5.2.4). Furthermore, it holds that

$$\max_{\Delta x(t,z)\in\mathcal{Z}_{\Delta x}(t,z)} \left\| \Delta x\left(t_\mathrm{f},z\right) \right\|_1 = 1_{n_x}^T \max\left(\rho_{\Delta x}^{(\mathcal{Z})}\left(-I_{n_x},t_\mathrm{f},z\right), \rho_{\Delta x}^{(\mathcal{Z})}\left(I_{n_x},t_\mathrm{f},z\right) \right),$$

which follows from $0 \in \mathcal{Z}_{\Delta x}(t,z)$ and the central symmetry of $\mathcal{Z}_{\Delta x}(t,z)$ since it is a zonotope. Similarly, it holds that

$$\max_{\Delta u(\tau,z)\in\mathcal{Z}_{\Delta u}(\tau,z)} \|\Delta u(\tau,z)\|_1 = 1_{n_u}^T \max\left(\rho_{\Delta u}^{(\mathcal{Z})}(-I_{n_u},\tau,z), \rho_{\Delta u}^{(\mathcal{Z})}(I_{n_u},\tau,z)\right).$$

Constraints: We conservatively encode the constraints of (6.3) as

$$\forall t \in [0,t_\mathrm{f}] : \mathcal{Z}_{u_\mathrm{ff}}(t,P) \oplus \mathcal{Z}_{\Delta u}(t,z) \subseteq \mathcal{U}, \tag{6.32}$$

$$\mathcal{Z}_{x_\mathrm{ff}}([0,t_\mathrm{f}],z) \oplus \mathcal{Z}_{\Delta x}([0,t_\mathrm{f}],z) \subseteq \mathcal{X}, \tag{6.33}$$

$$\mathcal{Z}_{x_\mathrm{ff}}(t_\mathrm{f},z) \oplus \mathcal{Z}_{\Delta x}(t_\mathrm{f},z) \subseteq \mathcal{X}_\mathrm{f}, \tag{6.34}$$

where we used (6.29) and (6.30) and replaced all sets with their respective zonotope outer approximations from (6.31). Let

$$\rho_x^{(\mathcal{Z})}(\cdot,\tau,z) = \max_{\bar\tau\in\mathcal{T}_\tau(m_\mathrm{c}m_r)} \left(\rho_{x_\mathrm{ff}}^{(\mathcal{Z})}(\cdot,\bar\tau,z) + \rho_{\Delta x}^{(\mathcal{Z})}(\cdot,\bar\tau,z)\right),$$
$$\rho_u^{(\mathcal{Z})}(\cdot,\tau,z) = \max_{\bar\tau\in\mathcal{T}_\tau(m_\mathrm{c}m_r)} \left(\rho_{u_\mathrm{ff}}^{(\mathcal{Z})}(\cdot,\bar\tau,z) + \rho_{\Delta u}^{(\mathcal{Z})}(\cdot,\bar\tau,z)\right),$$

with

$$\mathcal{T}_\tau(n) = \left\{\bar\tau = [o,o+1]\frac{t_\mathrm{f}}{n} \;\middle|\; 0 \le o \le n-1,\ \tau\cap\bar\tau \neq \emptyset\right\}, \tag{6.35}$$

for some interval $\tau \in [0,t_\mathrm{f}]$. Following Prop. 4.6, the set containment constraints in (6.32) to (6.34) can thus be encoded by

$$g(z) = \begin{bmatrix} \rho_u^{(\mathcal{Z})}(C_\mathcal{U},[0,t_\mathrm{f}],z) - d_\mathcal{U} \\ \rho_x^{(\mathcal{Z})}(C_\mathcal{X},[0,t_\mathrm{f}],z) - d_\mathcal{X} \\ \rho_x^{(\mathcal{Z})}(C_{\mathcal{X}_\mathrm{f}},t_\mathrm{f},z) - d_{\mathcal{X}_\mathrm{f}} \end{bmatrix} \le 0, \tag{6.36}$$

since the bounded input constraints $\mathcal{U} = \langle C_\mathcal{U}, d_\mathcal{U}\rangle_H$, state constraints $\mathcal{X} = \langle C_\mathcal{X}, d_\mathcal{X}\rangle_H$, and final state constraints $\mathcal{X}_\mathrm{f} = \langle C_{\mathcal{X}_\mathrm{f}}, d_{\mathcal{X}_\mathrm{f}}\rangle_H$ are all given as H-polytopes.

Cost: Thus, we propose the cost function (also see (5.42))

$$\begin{aligned} J(z) = {} & \|c_{x_\mathrm{ff}}(t_\mathrm{f},P) - x_\mathrm{f}\|_1 + \left\|[G_{x_\mathrm{ff}}(t_\mathrm{f},P)]_{(:)}\right\|_1 \\ & + 1_{n_x}^T \max\left(\rho_{\Delta x}^{(\mathcal{Z})}(-I_{n_x},t_\mathrm{f},z), \rho_{\Delta x}^{(\mathcal{Z})}(I_{n_x},t_\mathrm{f},z)\right) \\ & + \zeta\delta \sum_{j=0}^{m_\mathrm{c}-1}\sum_{k=0}^{m_r-1} \left(\left\| \begin{bmatrix} c_{u_\mathrm{ff}}\left(P^{(j)}\right) \\ G_{u_\mathrm{ff}}\left(P^{(j)}\right)_{(:)} \end{bmatrix} \right\|_1 \right. \\ & \left. + 1_{n_u}^T \max\left(\rho_{\Delta u}^{(\mathcal{Z})}\left(-I_{n_u},\tau^{(j,k)},z\right), \rho_{\Delta u}^{(\mathcal{Z})}\left(I_{n_u},\tau^{(j,k)},z\right)\right) \right) \\ & + \sigma \|\max(0,g(z))\|_1, \end{aligned} \tag{6.37}$$

where we penalize the constraint violation by choosing the exact penalty multiplier $\sigma \in \mathbb{R}_+$ large enough such that constraint violations always dominate all other terms in (6.37) (see Sec. 5.4.1).

6.3.4 Parameterized Reachable Set

In Sec. 5.4.2, we described how the feedforward trust-region radius γ can be used to construct an approximation of the feedforward reachable set for the iPGSC approach that can be brought arbitrarily close to the tight outer approximation of the closed-loop feedforward reachable set. In this section, we derive how this approximation can be used to approximate the closed-loop reachable set for the combined controller – which is required to solve (6.3) – without recomputing outer approximations of the closed loop reachable set during each optimization iteration.

Using the state separation from (6.29) allows us to compute the parameterized feedforward reachable set of x_{ff} independently of Δx. Since the parameterized feedforward reachable set can be computed as described in Sec. 5.4.2, we first derive an approximated flow for Δx, and then discuss how this flow can be used to find an efficient approximation for the set of deviation vectors $\mathcal{S}_{\Delta x}(t, z)$. That said, we only need the support function of this approximation in all directions which are normal to the half-spaces of the polytopic input, state, and final state constraints (see (6.37)); since $\mathcal{R}_{\mathrm{ext}}(t, \bar{z})$ is available for the current controller parameters $\bar{z}$ (see Alg. 6) and thus $\rho^{(\mathcal{Z})}_{\Delta x}(\cdot, t, \bar{z})$ can be efficiently computed, we next construct an approximation for the support function of Δx that only approximates the unknown difference from $\rho^{(\mathcal{Z})}_{\Delta x}(\cdot, t, \bar{z})$ to $\rho^{(\mathcal{Z})}_{\Delta x}(\cdot, t, z)$.

Approximated reachable set: Let the sequence of LTI systems for Δx as derived in Sec. 6.3.1 with $A_{\mathrm{cl}}(t, z) = A^{(k)}_{\mathrm{cl}}(z)$ and $E(t, P) = E^{(k)}(P)$ for $0 \leq k \leq m_{\mathrm{c}} m_{\mathrm{d}}$ be given, where instead of m_{c} LTI systems, we linearize $m_{\mathrm{c}} m_{\mathrm{d}}$ times: One may introduce $m_{\mathrm{d}} \in \mathbb{N}_+$ to obtain a better approximation of the original nonlinear dynamics. Since these LTI systems only approximate the flow for Δx, we can only compute an approximation of $\mathcal{S}_{\Delta x}(t, z)$. To that end, we compute the reachable set of the linearized flow in (6.19) using reachability analysis for linear systems (see Sec. 2.7.1)

$$\tilde{\mathcal{R}}^{(\mathcal{Z})}_{\Delta x}(t + \delta, z) = e^{A_{\mathrm{cl}}(t,z)\delta} \tilde{\mathcal{R}}^{(\mathcal{Z})}_{\Delta x}(t, z) \oplus \int_0^{\delta} e^{A_{\mathrm{cl}}(t,z)(\delta-\tau)} E(t, P) \mathcal{W} \, \mathrm{d}\tau, \tag{6.38}$$

for $t + \delta \leq t_{\mathrm{f}}$ with $\tilde{\mathcal{R}}^{(\mathcal{Z})}_{\Delta x}(0, z) = \{0\}$. In contrast to $\mathcal{S}_{\Delta x}(t, z)$, its approximation $\tilde{\mathcal{R}}^{(\mathcal{Z})}_{\Delta x}(t, z)$ has a flow and thus is a reachable set. Since $\mathcal{W}$ is given as a zonotope and the dynamics consist of $m_{\mathrm{c}} m_{\mathrm{d}}$ LTI systems, one can compute an arbitrarily tight outer approximation of the reachable set in (6.38) [114, Sec. IV.A]. However, since $\tilde{\mathcal{R}}^{(\mathcal{Z})}_{\Delta x}(t, z)$ will be formed in each optimization iteration of the trust-region subproblem problem later, its growing representation size due to the repeated Minkowski sums (see Sec. 2.6.5) when increasing m_{c} or m_{d} means additional computational effort, especially since we want to compute both Jacobian and Hessian matrices for the optimization of the trust-region subproblem. To maintain a constant representation size, we thus propose to evaluate (6.38) using ellipsoids. With $\bar{\mathcal{W}} = \hat{\mathcal{E}}(\mathcal{W})$ being an ellipsoid outer approximation of $\mathcal{W}$ according to Th. 4.1, we compute the ellipsoidal reachable set of

the linearized flow of the $m_c m_d$ LTI systems with [65, Lem. 2.2]

$$\tilde{\mathcal{R}}_{\Delta x}^{(\mathcal{E})}(t+\delta, z) = e^{A_{\mathrm{cl}}(t,z)\delta}\tilde{\mathcal{R}}_{\Delta x}^{(\mathcal{E})}(t, z) \oplus \int_0^{\delta} e^{A_{\mathrm{cl}}(t,z)\phi} E(t, P)\, \mathcal{E}(0, Q_{\mathcal{W}})\, d\phi. \tag{6.39}$$

Since the representation size of ellipsoids remains constant, there is no need to manually approximate the integral in (6.39) and we can directly use numerical integration methods (see Sec. 6.3.7 for more details on computing (6.39)). However, while ellipsoids are closed under linear maps, the Minkowski sum of two ellipsoids is not an ellipsoid in general (see Sec. 2.6.3). While efficient methods for the outer approximation of the Minkowski sum exist [47], finding the minimum-volume ellipsoid is not required for our purposes since we only need a good enough approximation. To save computation time, we instead propose to approximate this Minkowski sum by simply adding their respective shape matrices. While this does not result in an outer approximation, we argue that it is a good approximation of the Minkowski sum up to a scaling factor: First, for two ellipsoids $\langle 0, U\rangle_E$ with $U \in \mathbb{S}_{++}^{n\times n}$ and $\langle 0, V\rangle_E$ with $V \in \mathbb{S}_{++}^{n\times n}$, it holds that $\langle 0, U\rangle_E \oplus \langle 0, V\rangle_E \subseteq \left\langle 0, \frac{1}{\alpha_1}U + \frac{1}{\alpha_2}V\right\rangle_E$ for $\alpha_1 + \alpha_2 = 1$ and $\alpha_1, \alpha_2 > 0$ [28, Th. 4.2]. Choosing $\alpha_1 = \alpha_2 = \frac{1}{2}$ thus means that an outer approximation of the Minkowski sum is given by the sum of their shape matrices, scaled by a scalar factor (here 2). Second, we can interpret the two ellipsoids $\langle 0, U\rangle_E$ and $\langle 0, V\rangle_E$ as confidence ellipsoids of two zero-mean, independently distributed Gaussian random variables with covariance matrices Q and U, respectively [81]. Then $\langle 0, U\rangle_E$ and $\langle 0, V\rangle_E$ describe an area of the support of the two underlying random variables such that any realization of these random variables lies within $\langle 0, U\rangle_E$ and $\langle 0, V\rangle_E$ with some probability. It follows that the covariance matrix of the sum of these two random variables is given by the sum of the covariance matrices of each random variable.

Approximated support function: So denote with $\tilde{\mathcal{R}}_{\Delta x}^{(\mathcal{E})}(t, z) = \mathcal{E}(0, D_{\Delta x}(t, z))$ the propagation of $\bar{\mathcal{W}}$ through (6.39), where we replace the Minkowski sum of two ellipsoids with the sum of their respective shape matrices. We now derive the approximated support function $\tilde{\rho}_{\Delta x}(\cdot, t, z)$ of $\rho_{\Delta x}^{(\mathcal{Z})}(\cdot, t, z)$.

We start with the support function of $\tilde{\mathcal{R}}_{\Delta x}^{(\mathcal{E})}(t, z)$, which is given by

$$\tilde{\rho}_{\Delta x}^{(\mathcal{E})}(l, t, z) = \sqrt{l^T D_{\Delta x}(t, z)\, l + \epsilon}, \tag{6.40}$$

where $l \in \mathbb{R}^{n_x}$ and $\epsilon \ll 1$ to achieve differentiability for all z since

$$\forall l \in \mathbb{R}^{n_x} : l^T D_{\Delta x}(t, z)\, l \geq 0,$$

due to $D_{\Delta x}(t, z) \succeq 0$. Using the support function $\tilde{\rho}_{\Delta x}^{(\mathcal{E})}(l, t, z)$ of $\tilde{\mathcal{R}}_{\Delta x}^{(\mathcal{E})}(t, z)$ and the known support function value $\rho_{\Delta x}^{(\mathcal{Z})}(l, t, \bar{z})$ at the current controller parameters $\bar{z}$, we are now ready to define the approximation of the support function $\rho_{\Delta x}^{(\mathcal{Z})}(l, t, z)$ using the support function $\tilde{\rho}_{\Delta x}^{(\mathcal{E})}(l, t, z)$ of the ellipsoidal reachable set $\tilde{\mathcal{R}}_{\Delta x}^{(\mathcal{E})}(t, z)$ as

$$\tilde{\rho}_{\Delta x}(l, t, z) = \rho_{\Delta x}^{(\mathcal{Z})}(l, t, \bar{z}) + \chi_{\Delta x}(l, t)\left(\tilde{\rho}_{\Delta x}^{(\mathcal{E})}(l, t, z) - \tilde{\rho}_{\Delta x}^{(\mathcal{E})}(l, t, \bar{z})\right), \tag{6.41}$$

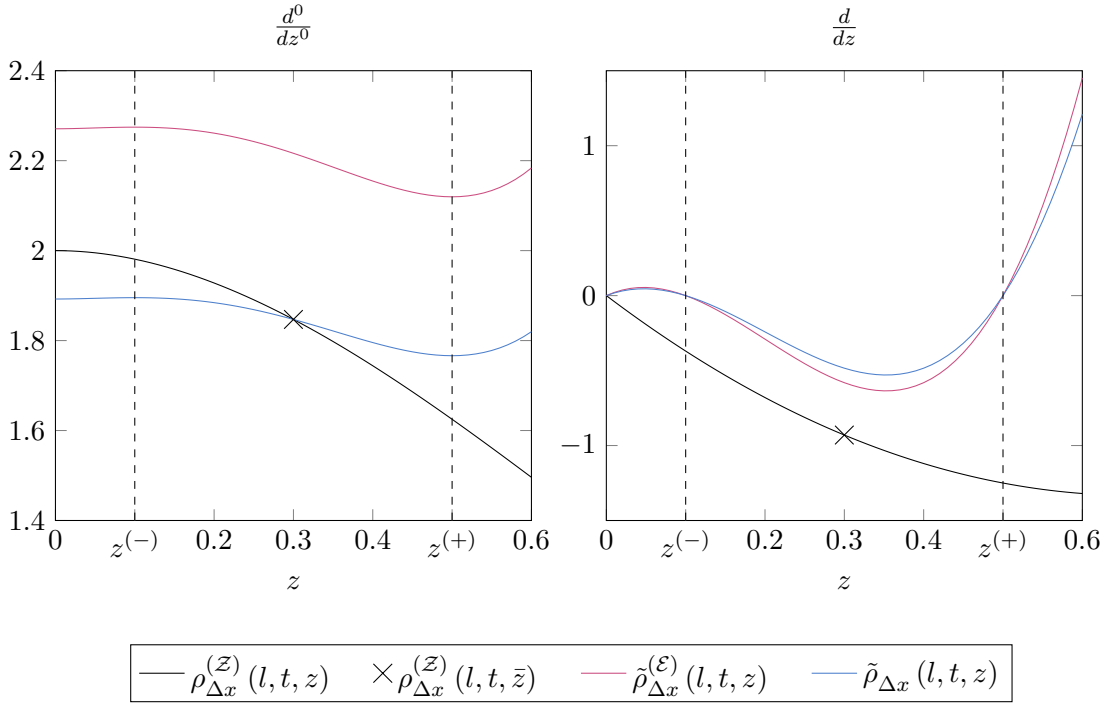


Figure 6.4: Visualization (1D) of the approximation of the support function $\rho_{\Delta x}^{(\mathcal{Z})}(l,t,z)$: Generally, we assume that $\frac{d}{dz}\tilde{\rho}_{\Delta x}^{(\mathcal{E})}(l,t,z)$ approximates the $\frac{d}{dz}\rho_{\Delta x}^{(\mathcal{Z})}(l,t,\bar{z})$ reasonably well for $z \in \left[z^{(-)}, z^{(+)}\right] = [\bar{z}-r, \bar{z}+r]$ with a given radius $r \in \mathbb{R}_+$; for illustrative purposes, we here thus assume $\operatorname{sign}\left(\tilde{\rho}_{\Delta x}^{(\mathcal{E})}(l,t,z)\right) = \operatorname{sign}\left(\rho_{\Delta x}^{(\mathcal{Z})}(l,t,z)\right)$ for $z \in \left[z^{(-)}, z^{(+)}\right]$ (see right plot). With appropriate scaling, we obtain the reasonable approximation $\rho_{\Delta x}^{(\mathcal{Z})}(l,t,\bar{z})$ with $\tilde{\rho}_{\Delta x}(l,t,\bar{z}) = \rho_{\Delta x}^{(\mathcal{Z})}(l,t,\bar{z})$ (see left plot).

where $\chi_{\Delta x}(l,t) = \frac{\rho_{\Delta x}^{(\mathcal{Z})}(l,t,\bar{z})}{\tilde{\rho}_{\Delta x}^{(\mathcal{E})}(l,t,\bar{z})}$: Since $\tilde{\rho}_{\Delta x}^{(\mathcal{E})}(l,t,z)$ is calculated based on various approximations we made previously, we do not assume that its magnitude is accurate. Instead, we define the scaling function $\chi_{\Delta x}(l,t)$ which is chosen such that the scaled approximation is identical to the support function of the outer approximation, i.e.

$$\rho_{\Delta x}^{(\mathcal{Z})}(l,t,\bar{z}) = \chi_{\Delta x}(l,t)\,\tilde{\rho}_{\Delta x}^{(\mathcal{E})}(l,t,\bar{z}),$$

holds for $z = \bar{z}$. Using the scaled approximation $\chi_{\Delta x}(l,t)\,\tilde{\rho}_{\Delta x}^{(\mathcal{E})}(l,t,z)$, we then only approximate the unknown difference

$$\rho_{\Delta x}^{(\mathcal{Z})}(l,t,z) - \rho_{\Delta x}^{(\mathcal{Z})}(l,t,\bar{z}) \approx \chi_{\Delta x}(l,t)\left(\tilde{\rho}_{\Delta x}^{(\mathcal{E})}(l,t,z) - \tilde{\rho}_{\Delta x}^{(\mathcal{E})}(l,t,\bar{z})\right),$$

so that we recover $\tilde{\rho}_{\Delta x}(l,t,\bar{z}) = \rho_{\Delta x}^{(\mathcal{Z})}(l,t,\bar{z})$ by substitution of $z = \bar{z}$ into (6.41). Fig. 6.4 visualizes this approximation using a 1D example.

An approximation of the support function for the set of input deviations $S_{\Delta u}(t,z)$ can be derived analogously. Since $\mathcal{S}_{u_{\text{ff}}}(t,z)$ is known, it only remains to compute an

approximation of the support function of $\mathcal{S}_{\Delta u}(t,z)$ to approximate the input constraint in (6.3b). It follows from (6.6) that $S_{\Delta u}(t,z) = K(t,z)\, S_{\Delta x}(t,z)$, and thus

$$\tilde{\mathcal{S}}^{(\mathcal{E})}_{\Delta u}(t,z) = K(t,z)\, \tilde{\mathcal{R}}^{(\mathcal{E})}_{\Delta x}(t,z) = \left\langle 0, K(t,z)\, D_{\Delta x}(t,z)\, K(t,z)^T \right\rangle_E .$$

With

$$\tilde{\rho}^{(\mathcal{E})}_{\Delta u}(l,t,z) = \sqrt{K(t,z)\, D_{\Delta x}(t,z)\, K(t,z)^T + \epsilon}, \tag{6.42}$$

given analogously to (6.40), we can – equivalently to (6.41) – define

$$\tilde{\rho}_{\Delta u}(l,t,z) = \rho^{(\mathcal{Z})}_{\Delta u}(l,t,\bar{z}) + \chi_{\Delta u}(l,t)\left(\tilde{\rho}^{(\mathcal{E})}_{\Delta u}(l,t,z) - \tilde{\rho}^{(\mathcal{E})}_{\Delta u}(l,t,\bar{z})\right), \tag{6.43}$$

as an approximation to the support function $\rho^{(\mathcal{Z})}_{\Delta u}(\cdot,t,z)$ of $\mathcal{Z}_{\Delta u}(t,z)$ as defined in (6.31d) with $\chi_{\Delta u}(l,t) = \frac{\rho^{(\mathcal{Z})}_{\Delta u}(l,t,\bar{z})}{\tilde{\rho}^{(\mathcal{E})}_{\Delta u}(l,t,\bar{z})}$.

Combining (6.29) with (6.41) and (6.30) with (6.43) finally yields

$$\tilde{\rho}_x(l,t,z) = \tilde{\rho}^{(\mathcal{Z})}_{x_{\mathrm{ff}}}(l,t,P) + \tilde{\rho}_{\Delta x}(l,t,z), \tag{6.44}$$

$$\tilde{\rho}_u(l,t,z) = \rho^{(\mathcal{Z})}_{u_{\mathrm{ff}}}(l,t,P) + \tilde{\rho}_{\Delta u}(l,t,z), \tag{6.45}$$

where we used (2.26), $\tilde{\rho}^{(\mathcal{Z})}_{x_{\mathrm{ff}}}(l,t,P)$ denotes the support function of the feedforward reachable set approximation $\hat{\mathcal{Z}}\left(\tilde{\mathcal{R}}_{x_{\mathrm{ff}}}(t,P)\right)$ according to Sec. 5.4.2, and $\rho^{(\mathcal{Z})}_{u_{\mathrm{ff}}}(l,t,P)$ denotes the support function of $\mathcal{Z}_{u_{\mathrm{ff}}}(t,z)$ as defined in (6.31c). Choosing m_{d} large enough further motivates the use of (6.44) and (6.45) to obtain the approximated support functions for time intervals (instead of time points in (6.44) and (6.45))

$$\tilde{\rho}_{\Delta x}(l,[k,k+1]\,\delta,z) \approx \tilde{\rho}_{\Delta x}(l,(k+1)\,\delta,z),$$
$$\tilde{\rho}_{\Delta u}(l,[k,k+1]\,\delta,z) \approx \tilde{\rho}_{\Delta u}(l,(k+1)\,\delta,z).$$

For later convenience, we introduce the definition of the support function approximations over any time interval $\tau \subseteq [0,t_{\mathrm{f}}]$ as

$$\tilde{\rho}_x(l,\tau,z) = \max_{\bar{\tau}\in\mathcal{T}_\tau(m_{\mathrm{c}}m_{\mathrm{d}})}\left(\max_{\tilde{\tau}\in\mathcal{T}_{\bar{\tau}}(m_{\mathrm{c}}m_r)} \tilde{\rho}^{(\mathcal{Z})}_{x_{\mathrm{ff}}}(l,\tilde{\tau},P) + \tilde{\rho}_{\Delta x}(l,\bar{\tau},z)\right), \tag{6.46}$$

$$\tilde{\rho}_u(l,\tau,z) = \max_{\bar{\tau}\in\mathcal{T}_\tau(m_{\mathrm{c}})}\left(\rho^{(\mathcal{Z})}_{u_{\mathrm{ff}}}(l,\tilde{\tau},P) + \max_{\tilde{\tau}\in\mathcal{T}_{\bar{\tau}}(m_{\mathrm{c}}m_{\mathrm{d}})} \tilde{\rho}_{\Delta u}(l,\bar{\tau},z)\right), \tag{6.47}$$

where $\mathcal{T}_\tau(\cdot)$ is defined in (6.35), we assumed that $m_r \geq m_{\mathrm{d}}$ with m_r being the number of reachability steps per duration t_{c} and m_{d} being the number of additional linearization points for (6.20) to (6.22) per duration t_{c}.

6.3.5 Controller Computation

Ideally, we want to optimize over the controller cost in (6.37). However, its evaluation requires the computation of an outer approximation of the closed-loop extended reachable set for each controller parameter value z, so that direct optimization over (6.37) is computationally expensive. For the efficient, simultaneous synthesis of the combined controller, we thus need to find a smooth approximation to (6.37). In Sec. 6.3.4, we already derived expressions for the approximated support function for both the combined state and input set. In this section, we use these expressions to approximate (6.37).

Trust Region: As demonstrated in Sec. 5.4, the trust-region radius γ is required to restrict P to a local, compact neighborhood around the current controller guess $\bar{P}$. Similarly, we now introduce the trust-region radius $\eta \in (0, 1]$ to restrict Q and R to local neighborhoods of $\bar{Q} \in \mathcal{M}^{(Q)}$ and $\bar{R} \in \mathcal{M}^{(R)}$, respectively, i.e.

$$\begin{aligned} Q &\in \mathcal{M}_\eta^{(Q)}\left(\bar{Q}\right) \cap \mathcal{M}^{(Q)}, \\ R &\in \mathcal{M}_\eta^{(R)}\left(\bar{R}\right) \cap \mathcal{M}^{(R)}, \end{aligned}$$

where $\mathcal{M}^{(Q)} \subset \mathbb{S}_{++}^{n_x \times n_x}$ and $\mathcal{M}^{(R)} \subset \mathbb{S}_{++}^{n_u \times n_u}$ are bounded, given sets from which Q and R are chosen, and where

$$\mathcal{M}_\eta^{(Q)}\left(\bar{Q}\right) = \left\{ \bar{Q} + h_Q\left(\eta M\right) \;\middle|\; \|M\|_\infty \leq 1 \right\}, \tag{6.48}$$

$$\mathcal{M}_\eta^{(R)}\left(\bar{R}\right) = \left\{ \bar{R} + h_R\left(\eta N\right) \;\middle|\; \|N\|_\infty \leq 1 \right\}, \tag{6.49}$$

are bounded sets defined by the trust region η. Here, $h_Q : [-1, 1]^{p \times q} \mapsto \mathbb{S}^{n_x \times n_x}$ and $h_R : [-1, 1]^{r \times s} \mapsto \mathbb{S}^{n_u \times n_u}$ can be any sufficiently smooth functions which generate symmetric matrices. That said, this creates semi-definite constraints in the final optimization problem which can significantly impact solver times. Therefore, we restrict Q and R to diagonal matrices, i.e., we set

$$\begin{aligned} h_Q\left(\omega\right) &= \operatorname{diag}\left(\bar{q} - \underline{q}\right)\omega, \\ h_R\left(\xi\right) &= \operatorname{diag}\left(\bar{r} - \underline{r}\right)\xi, \\ \mathcal{M}^{(Q)} &= \left\{ \operatorname{diag}\left(\tilde{q}\right) \;\middle|\; \tilde{q} \in \left[\underline{q}, \bar{q}\right] \right\}, \\ \mathcal{M}^{(R)} &= \left\{ \operatorname{diag}\left(\tilde{r}\right) \mid \tilde{r} \in [\underline{r}, \bar{r}] \right\}, \end{aligned}$$

with $\bar{q} \geq \underline{q} \in \mathbb{R}_+^{n_x}$, $\bar{r} \geq \underline{r} \in \mathbb{R}_+^{n_u}$, and dependent factors $\omega \in [-1, 1]^{n_x}$ and $\xi \in [-1, 1]^{n_u}$. Additionally, we set $\underline{q}_1 = \bar{q}_1 = 1$ to eliminate the invariance of the LQR computation under equal scaling of Q and R. We remark that more general choices are possible while still avoiding positive definite constraints, e.g., one can generate symmetric, strictly diagonally dominant matrices with real positive entries, which is sufficient to ensure positive definiteness (follows from the Gershgorin Circle Theorem [11, Th. 0]).

Optimization Problem: Starting from the current controller parameters collected in $\bar{z} = \left[\bar{P}_{(:)}^T, \quad \bar{Q}_{(:)}^T, \quad \bar{R}_{(:)}^T\right]^T$, an approximation to the controller cost in (6.37) is given by

$$\min_{z,s} \tilde{J}_{\mathrm{TR}}(z,s), \tag{6.50a}$$

$$\text{s.t. } \tilde{g}(z) \leq s, \tag{6.50b}$$

$$P \in \mathcal{P}_\gamma\left(\bar{P}\right) \cap [-1,1]^{m_\mathrm{c} n_u \times a}, \tag{6.50c}$$

$$Q \in \mathcal{M}_\eta^{(Q)}\left(\bar{Q}\right) \cap \mathcal{M}^{(Q)}, \tag{6.50d}$$

$$R \in \mathcal{M}_\eta^{(R)}\left(\bar{R}\right) \cap \mathcal{M}^{(R)}, \tag{6.50e}$$

$$s = \left[s_\mathcal{U}^T, \quad s_\mathcal{X}^T, \quad s_{\mathcal{X}_\mathrm{f}}^T\right]^T \geq 0, \tag{6.50f}$$

with

$$\begin{aligned}
\tilde{J}_{\mathrm{TR}}(z,s) = \; & \left\|\tilde{c}_{x_\mathrm{ff}}(t_\mathrm{f},P) - x_\mathrm{f}\right\|_1 + \left\|\left[\tilde{G}_{x_\mathrm{ff}}(t_\mathrm{f},P)\right]_{(:)}\right\|_1 \\
& + 1_{n_x}^T \max\left(\tilde{\rho}_{\Delta x}(-I_{n_x}, t_\mathrm{f}, z), \tilde{\rho}_{\Delta x}(I_{n_x}, t_\mathrm{f}, z)\right) \\
& + \zeta\tilde{\delta} \sum_{j=0}^{m_\mathrm{c}-1} \sum_{k=0}^{m_\mathrm{d}-1} \left(\left\| \begin{bmatrix} c_{u_\mathrm{ff}}\left(P^{(j)}\right) \\ G_{u_\mathrm{ff}}\left(P^{(j)}\right)_{(:)} \end{bmatrix} \right\|_1 \right. \\
& \left. + 1_{n_u}^T \max\left(\tilde{\rho}_{\Delta u}\left(-I_{n_u}, \tau^{(j,k)}, z\right), \tilde{\rho}_{\Delta u}\left(I_{n_u}, \tau^{(j,k)}, z\right)\right) \right) \\
& + \sigma \|s\|_1,
\end{aligned} \tag{6.51}$$

and

$$\tilde{g}(z) = \begin{bmatrix} \tilde{\rho}_u(C_\mathcal{U}, [0, t_\mathrm{f}], z) - d_\mathcal{U} \\ \tilde{\rho}_x(C_\mathcal{X}, [0, t_\mathrm{f}], z) - d_\mathcal{X} \\ \tilde{\rho}_x(C_{\mathcal{X}_\mathrm{f}}, t_\mathrm{f}, z) - d_{\mathcal{X}_\mathrm{f}} \end{bmatrix}, \tag{6.52}$$

where $\sigma \in \mathbb{R}_+$ and $\tilde{\delta} = \frac{t_\mathrm{f}}{m_\mathrm{c} m_\mathrm{d}}$:

- (6.50a) and (6.51): We replace the outer approximations used in (6.37) with their respective approximations derived previously. Additionally, we have $m_\mathrm{c} m_\mathrm{d}$ time intervals instead of $m_\mathrm{c} m_r$ as in Sec. 5.4.1 and thus adapt the sum accordingly. Lastly, we replace the constraint violation with a slack variable (see next bullet point for more details).

- (6.50b): Instead of incorporating the constraints in the objective function, we enforce input, state, and final state constraints with $\tilde{g}(z) \leq s$ directly; however, we relax these constraints with $s \geq 0$ so that (6.50) always has a feasible solution, even if there is no feasible solution for $s = 0$. To still ensure feasibility, we penalize s in the objective function with $\sigma \in \mathbb{R}_+$.

- (6.50c) to (6.50e): Because we replaced tight outer approximations of the closed-loop reachable sets and input sets with their respective approximations, (6.50) is only accurate in a neighborhood of $\bar{z}$. We therefore restrict z to a bounded set around $\bar{z}$.

Since (6.50) can be written as the abstracted synthesis problem, it can be reformulated as a smooth non-convex optimization problem according to Prop. 3.1 and the KKT conditions are always necessary for this smooth reformulation due to Prop. 3.2. We show in Sec. 6.3.6 that (6.50) can approximate (6.37) arbitrarily closely. For this proof, we require a cost value of the trust-region solution only dependent on $\hat{z}$, which we introduce subsequently.

Proposition 6.1 (Stationary Objective Value)**.** *At a first-order critical point of the smooth reformulation of* (6.50) *with optimizer* $\hat{z}$*, the optimal objective value is given by*

$$\tilde{J}(\hat{z}) = \tilde{J}_{\mathrm{TR}}(\hat{z}, \max(0, \tilde{g}(\hat{z}))), \tag{6.53}$$

where $\tilde{g}(\hat{z})$ *is given by* (6.52)*.*

Proof. Follows from Prop. 3.1. □

6.3.6 Tuning of Trust-Region Radii

Since the trust-region subproblem approximates the cost to find a new controller candidate, it is paramount to analyze how close this approximation is to the real cost; otherwise, optimizing the trust-region problem might not decrease the actual cost of the controller. However, if the trust-region subproblem and thus its approximated objective value (see Prop. 6.1) is close enough to the real cost, optimizing (6.50) – and thus reducing the approximated cost in (6.53) – also reduces the real cost in (6.37). In this section, we therefore show that the trust-region radii γ and η can be tuned so that the approximated cost described in Prop. 5.2 stays arbitrarily close to the actual controller cost from (6.37).

To that end, we first define $\tilde{J}_{\Delta x}(z)$ equivalently to $\tilde{J}(z)$ but replace all approximated feedforward sets and corresponding feedforward support functions with their respective formally correct versions using the closed-loop feedforward reachable set available at $\bar{z}$. Intuitively, $\tilde{J}_{\Delta x}(z)$ then contains no inaccuracies (compared to $J(z)$) from approximating the feedforward reachable set and hence is only inaccurate due to inaccurate support function approximations for Δx and Δu. We are now ready to state the main result of this section.

Theorem 6.1 (Accurate Trust-Region Subproblem)**.** *Let* $\hat{z}$ *denote a first-order critical point of the reformulation of the trust-region subproblem in* (6.50)*. Then*

$$\left| J(\hat{z}) - \tilde{J}(\hat{z}) \right| \leq \epsilon, \tag{6.54}$$

is achieved after a finite number of trust-region iterations with given tolerance $\epsilon > 0$ *if we adapt the trust-region radii according to*

$$\gamma \leftarrow \min\left(1, v\left(\max\left(e_{\mathrm{ff},\gamma}, e_{\Delta,\gamma}\right)\right)\gamma\right), \tag{6.55}$$

$$\eta \leftarrow \min\left(1, v\left(e_{\Delta,\eta}\right)\eta\right), \tag{6.56}$$

with

$$e_{\mathrm{ff},\gamma} = \left|\tilde{J}\left(\hat{z}\right) - \tilde{J}_{\Delta x}\left(\hat{z}\right)\right|, \tag{6.57}$$

$$e_{\Delta,\gamma} = \left|\tilde{J}_{\Delta x}\left(\hat{P},\bar{Q},\bar{R}\right) - J\left(\hat{P},\bar{Q},\bar{R}\right)\right|, \tag{6.58}$$

$$e_{\Delta,\eta} = \left|\tilde{J}_{\Delta x}\left(\hat{z}\right) - \tilde{J}_{\Delta x}\left(\hat{P},\bar{Q},\bar{R}\right) - \left(J\left(\hat{z}\right) - J\left(\hat{P},\bar{Q},\bar{R}\right)\right)\right|, \tag{6.59}$$

where $v : (0;\infty) \mapsto \mathbb{R}_+$*, is an arbitrary, monotonically decreasing function with* $v(0) = \bar{c}$*,* $v(\psi) = 1$*,* $\lim_{r\to\infty} v(r) = \underline{c}$*,* $\frac{1}{\underline{c}} > \bar{c} > 1$*, and* $0 \leq \psi < \epsilon$*.*

Proof. The proof is structured as follows: We bound (6.54) from above by using (6.57) to (6.59), and then show that all latter expressions can be made arbitrarily small by following the tuning rules in (6.55) and (6.56).

It follows from the triangle inequality that

$$\begin{aligned}
&\left|\tilde{J}\left(\hat{z}\right) - J\left(\hat{z}\right)\right| \\
&= \Big|\tilde{J}\left(\hat{z}\right) - \tilde{J}_{\Delta x}\left(\hat{z}\right) + \tilde{J}_{\Delta x}\left(\hat{P},\bar{Q},\bar{R}\right) - J\left(\hat{P},\bar{Q},\bar{R}\right) \\
&\quad + \tilde{J}_{\Delta x}\left(\hat{z}\right) - \tilde{J}_{\Delta x}\left(\hat{P},\bar{Q},\bar{R}\right) - \left(J\left(\hat{z}\right) - J\left(\hat{P},\bar{Q},\bar{R}\right)\right)\Big| \\
&\leq e_{\mathrm{ff},\gamma} + e_{\Delta,\gamma} + e_{\Delta,\eta}.
\end{aligned} \tag{6.60}$$

Since $\tilde{J}_{\Delta x}\left(\hat{z}\right)$ uses the outer approximation of the feedforward reachable set, it follows that $\lim_{\gamma\to 0} e_{\mathrm{ff},\gamma} = 0$ since $\lim_{\gamma\to 0} \tilde{\mathcal{R}}_{x_{\mathrm{ff}}}\left(t,\hat{P}\right) = \tilde{\mathcal{R}}_{x_{\mathrm{ff}}}\left(t,\bar{P}\right) = \mathcal{R}_{x_{\mathrm{ff}}}\left(t,\bar{P}\right)$ (see (5.44)). Since $\lim_{\gamma\to 0} \tilde{J}_{\Delta x}\left(\hat{P},\bar{Q},\bar{R}\right) = \tilde{J}_{\Delta x}\left(\bar{z}\right)$, $\tilde{J}_{\Delta x}\left(\bar{z}\right) = J\left(\bar{z}\right)$ by construction, and $\lim_{\gamma\to 0} J\left(\hat{P},\bar{Q},\bar{R}\right) = J\left(\bar{z}\right)$, we further have $\lim_{\gamma\to 0} e_{\Delta,\gamma} = 0$. Lastly, it holds that $\lim_{\eta\to 0} \tilde{J}_{\Delta x}\left(\hat{z}\right) = \tilde{J}_{\Delta x}\left(\hat{P},\bar{Q},\bar{R}\right)$ and $\lim_{\eta\to 0} J\left(\hat{z}\right) = J\left(\hat{P},\bar{Q},\bar{R}\right)$ because $\lim_{\eta\to 0} \hat{z} = \left[\hat{P}_{(:)}^T,\ \bar{Q}_{(:)}^T,\ \bar{R}_{(:)}^T\right]^T$, and therefore it holds that $\lim_{\eta\to 0} e_{\Delta,\eta} = 0$.

Hence we can always achieve $e_{\mathrm{ff},\gamma} \leq \delta$, $e_{\Delta,\gamma} \leq \delta$, and $e_{\Delta,\eta} \leq \delta$ for $\delta > 0$ by following (6.55) and (6.56) for a finite number of steps, yielding $\left|\tilde{J}\left(\hat{z}\right) - J\left(\hat{z}\right)\right| \leq 3\delta$, which concludes the proof since δ is arbitrary. □

Evaluating (6.57) to (6.59) requires – in addition to the extended closed-loop reachable set already available at $z = \bar{z}$ and $z = \hat{z}$ – the extended closed-loop reachable set at $z = \left[\hat{P}_{(:)}^T,\ \bar{Q}_{(:)}^T,\ \bar{R}_{(:)}^T\right]^T$: While the feedforward trust-region radius γ ensures that the feedforward approximation remains accurate, it also indirectly contributes to

inaccuracies in the approximated support functions of Δx and Δu. Thus, even if γ is small enough so that the approximation of the feedforward reachable set is accurate, the approximated support function for Δx and Δu might still be too inaccurate in P. As a result, we consider both the accuracy of the feedforward reachable set as well as the accuracy of the feedforward parameters in the disturbance support function approximation (see (6.55)) for the tuning of γ. Put differently, we bound

$$e_\Delta = \left| \tilde{J}_{\Delta x}(\hat{z}) - J(\hat{z}) \right| \leq e_{\Delta,\gamma} + e_{\Delta,\eta},$$

from above, which allows us to assess whether the error $e_{\Delta,\gamma}$ due to γ is too large or the error $e_{\Delta,\eta}$ due to η is too large. Alternatively, one can skip this separation by setting $e_{\mathrm{ff},\gamma} = e_{\Delta,\eta} = e_\Delta$ and then apply (6.55) and (6.56), albeit at the cost of possibly unnecessarily shrinking either γ or η. In this thesis, we choose v from Th. 6.1 as a linear function interpolating between $\underline{c}$ and $\bar{c}$; that said, a better choice for v may be possible.

6.3.7 Feedback Derivative Information

For the efficient implementation of the trust-region subproblem in Sec. 6.3.5, it is beneficial to provide derivative information for the objective function and the constraints. Since the parameterized feedforward input sets are generated by the polynomial controller template and the approximations to the parameterized feedforward reachable sets are computed according to Sec. 5.4.2, both can be expressed as polynomial zonotopes and thus their derivatives are straightforward to compute. Thus, we focus on the computation of the first-order and second-order derivatives for the approximated support function $\tilde{\rho}_{\Delta u}(l,t,z)$ and $\tilde{\rho}_{\Delta x}(l,t,z)$ with respect to $z \in \mathbb{R}^{n_z}$.

Deviation state: For convenience, we restate (6.40) and (6.41) here:

$$\begin{aligned}
\tilde{\rho}_{\Delta x}(l,t,z) &= \rho^{(\mathcal{Z})}_{\Delta x}(l,t,\bar{z}) + \chi_{\Delta x}(l,t)\left(\tilde{\rho}^{(\mathcal{E})}_{\Delta x}(l,t,z) - \tilde{\rho}^{(\mathcal{E})}_{\Delta x}(l,t,\bar{z})\right), \\
\tilde{\rho}^{(\mathcal{E})}_{\Delta x}(l,t,z) &= \sqrt{l^T D_{\Delta x}(t,z)\, l + \epsilon}.
\end{aligned}$$

Thus, only the derivative of $\tilde{\rho}^{(\mathcal{E})}_{\Delta x}(l,t,z)$ is required since $\rho^{(\mathcal{Z})}_{\Delta x}(l,t,\bar{z})$ and $\chi_{\Delta x}(l,t)$ are not dependent on z. For the derivative of $\tilde{\rho}^{(\mathcal{E})}_{\Delta x}(l,t,z)$, we require the derivative of each matrix element of the positive-definite shape matrix $D_{\Delta x}(t,z)$. The corresponding ellipsoid is given by $\tilde{\mathcal{R}}^{(\mathcal{E})}_{\Delta}(t,z) = \langle 0, D_{\Delta x}(t,z)\rangle_E$ and we repeat the propagation equation for $\bar{\mathcal{W}}$ from (6.38) here for convenience:

$$\tilde{\mathcal{R}}^{(\mathcal{E})}_{\Delta x}(t+\delta,z) = e^{A_{\mathrm{cl}}(t,z)\delta}\mathcal{R}^{(\mathcal{E})}_{\Delta x}(t,z) \oplus \int_0^\delta e^{A_{\mathrm{cl}}(t,z)(\delta-\tau)} E(t,P)\,\bar{\mathcal{W}}\,\mathrm{d}\tau, \tag{6.61}$$

where $A_{\mathrm{cl}}(t,z) = A(t,P) + B(t,P)\,K(t,z)$.

In order to compute the Hessian matrix $H_{\tilde{\rho}^{(\mathcal{E})}_{\Delta x}}(l,t,z)$ of $\tilde{\rho}^{(\mathcal{E})}_{\Delta x}(l,t,z)$, we use differentials as introduced in Sec. 2.2. For the sake of simplicity and because $\tilde{\mathcal{R}}^{(\mathcal{E})}_{\Delta x}(t+\delta,z)$ is defined recursively, we assume that $\tilde{\mathcal{R}}^{(\mathcal{E})}_{\Delta x}(t,z)$ with first-order differential tensor $dD_{\Delta x}(t,z)$

and second-order differential tensor $d^2 D_{\Delta x}(t, z)$ are given and we want to the compute derivative information for $D_{\Delta x}\left(\hat{t}, z\right)$ with $\hat{t} = t + \delta$. From (6.20) to (6.22), it is clear that $A(t, P) = A^{(k)}(P)$, $B(t, P) = B^{(k)}(P)$, and $E(t, P) = E^{(k)}(P)$ for $0 \leq k \leq m_\mathrm{c} m_\mathrm{d} - 1$ can be precomputed, and therefore we assume that (arguments omitted for readability) the first-order differential tensors $dA \in \mathbb{R}^{1 \times n_z \times n_x \times n_x}$, $dB \in \mathbb{R}^{1 \times n_z \times n_x \times n_u}$, and $dE \in \mathbb{R}^{1 \times n_z \times n_x \times n_w}$ as well as second-order differential tensors $d^2 A \in \mathbb{R}^{n_z \times n_z \times n_x \times n_x}$, $d^2 B \in \mathbb{R}^{n_z \times n_z \times n_x \times n_u}$, and $d^2 E \in \mathbb{R}^{n_z \times n_z \times n_x \times n_w}$ are available (see Sec. 2.2). These can be computed by computing the Hessian matrix of each element of A, B, and E. For this example, we directly express all results with differential tensors instead of differentials (see Sec. 2.2 for details).

We start by applying the differential operator to $\tilde{\rho}^{(\mathcal{E})}_{\Delta x}\left(l, \hat{t}, z\right)$, yielding

$$\begin{aligned}
\tilde{\rho}^{(\mathcal{E})}_{\Delta x}\left(l, \hat{t}, z\right) &= \sqrt{l^T D_{\Delta x}\left(\hat{t}, z\right) l + \epsilon}, \\
d\tilde{\rho}^{(\mathcal{E})}_{\Delta x}\left(l, \hat{t}, z\right) &= \frac{l^T dD_{\Delta x}\left(\hat{t}, z\right) l}{2\sqrt{l^T D_{\Delta x}\left(\hat{t}, z\right) l + \epsilon}} \\
&= \frac{l^T dD_{\Delta x}\left(\hat{t}, z\right) l}{2\tilde{\rho}^{(\mathcal{E})}_{\Delta x}\left(l, \hat{t}, z\right)} \qquad (6.62) \\
d^2\tilde{\rho}^{(\mathcal{E})}_{\Delta x}\left(l, \hat{t}, z\right) &= \frac{l^T d^2 D_{\Delta x}\left(\hat{t}, z\right) l}{2\tilde{\rho}^{(\mathcal{E})}_{\Delta x}\left(l, \hat{t}, z\right)} - \frac{l^T dD_{\Delta x}\left(\hat{t}, z\right) l}{2\tilde{\rho}^{(\mathcal{E})}_{\Delta x}\left(l, \hat{t}, z\right)^2} d\tilde{\rho}^{(\mathcal{E})}_{\Delta x}\left(l, \hat{t}, z\right), \\
&= \frac{l^T d^2 D_{\Delta x}\left(\hat{t}, z\right) l}{2\tilde{\rho}^{(\mathcal{E})}_{\Delta x}\left(l, \hat{t}, z\right)} - \frac{1}{\tilde{\rho}^{(\mathcal{E})}_{\Delta x}\left(l, \hat{t}, z\right)} d\tilde{\rho}^{(\mathcal{E})}_{\Delta x}\left(l, \hat{t}, z\right) d\tilde{\rho}^{(\mathcal{E})}_{\Delta x}\left(l, \hat{t}, z\right) \\
&= \frac{l^T d^2 D_{\Delta x}\left(\hat{t}, z\right) l - 2 d\tilde{\rho}^{(\mathcal{E})}_{\Delta x}\left(l, \hat{t}, z\right) d\tilde{\rho}^{(\mathcal{E})}_{\Delta x}\left(l, \hat{t}, z\right)}{2\tilde{\rho}^{(\mathcal{E})}_{\Delta x}\left(l, \hat{t}, z\right)}, \qquad (6.63)
\end{aligned}$$

where $d\tilde{\rho}^{(\mathcal{E})}_{\Delta x}\left(l, \hat{t}, z\right) \in \mathbb{R}^{1 \times n_z \times 1 \times 1}$ and $d^2\tilde{\rho}^{(\mathcal{E})}_{\Delta x}\left(l, \hat{t}, z\right) \in \mathbb{R}^{n_z \times n_z \times 1 \times 1}$. Ignoring trailing singleton dimensions, we thus have

$$J_{\tilde{\rho}^{(\mathcal{E})}_{\Delta x}}\left(l, \hat{t}, z\right) = d\tilde{\rho}^{(\mathcal{E})}_{\Delta x}\left(l, \hat{t}, z\right), \qquad (6.64)$$

$$H_{\tilde{\rho}^{(\mathcal{E})}_{\Delta x}}\left(l, \hat{t}, z\right) = \frac{1}{2}\left(d^2\tilde{\rho}^{(\mathcal{E})}_{\Delta x}\left(l, \hat{t}, z\right) + \left(d^2\tilde{\rho}^{(\mathcal{E})}_{\Delta x}\left(l, \hat{t}, z\right)\right)^T\right), \qquad (6.65)$$

due to Th. 2.4 and Th. 2.5. However, the differential tensors $dD_{\Delta x}\left(\hat{t}, z\right)$ and $d^2 D_{\Delta x}\left(\hat{t}, z\right)$ still need to be computed.

With the shape matrix $D_{\Delta x}(t, z)$ as defined in (6.61), its differential tensors – when using the sum of the shape matrices as an approximation of the Minkowski sum as

discussed in Sec. 6.3.4 and the definition of the linear map of ellipsoids from Sec. 2.6.3 – are then given by

$$\begin{aligned}
D_{\Delta x}\left(\hat{t},z\right) &= e^{A_{\mathrm{cl}}(t,z)\delta} D_{\Delta x}\left(t,z\right)\left(e^{A_{\mathrm{cl}}(t,z)}\right)^T + V\left(t,z\right),\\
dD_{\Delta x}\left(\hat{t},z\right) &= de^{A_{\mathrm{cl}}(t,z)\delta} D_{\Delta x}\left(t,z\right)\left(e^{A_{\mathrm{cl}}(t,z)}\right)^T + e^{A_{\mathrm{cl}}(t,z)\delta} dD_{\Delta x}\left(t,z\right)\left(e^{A_{\mathrm{cl}}(t,z)}\right)^T\\
&\quad + e^{A_{\mathrm{cl}}(t,z)\delta} D_{\Delta x}\left(t,z\right)\left(de^{A_{\mathrm{cl}}(t,z)}\right)^T + dV\left(t,z\right),\\
d^2D_{\Delta x}\left(\hat{t},z\right) &= d^2e^{A_{\mathrm{cl}}(t,z)\delta} D_{\Delta x}\left(t,z\right)\left(e^{A_{\mathrm{cl}}(t,z)}\right)^T + 2de^{A_{\mathrm{cl}}(t,z)\delta} D_{\Delta x}\left(t,z\right)\left(de^{A_{\mathrm{cl}}(t,z)}\right)^T\\
&\quad + e^{A_{\mathrm{cl}}(t,z)\delta} d^2D_{\Delta x}\left(t,z\right)\left(de^{A_{\mathrm{cl}}(t,z)}\right)^T\\
&\quad + e^{A_{\mathrm{cl}}(t,z)\delta} D_{\Delta x}\left(t,z\right)\left(d^2e^{A_{\mathrm{cl}}(t,z)}\right)^T + d^2V\left(t,z\right),
\end{aligned}$$

where

$$\begin{aligned}
V\left(t,z\right) &= \int_0^\delta U\left(\tau,t,z\right)\mathrm{d}\tau, \qquad (6.66)\\
U\left(\tau,t,z\right) &= M\left(\tau,t,z\right)\bar{W}M\left(\tau,t,z\right)^T,\\
M\left(\tau,t,z\right) &= e^{A_{\mathrm{cl}}(t,z)(t-\tau)}E\left(t,P\right),
\end{aligned}$$

such that $\int_0^\delta e^{A_{\mathrm{cl}}(t,z)(\delta-\tau)}E\left(t,P\right)\bar{\mathcal{W}}\,\mathrm{d}\tau = \left\langle 0, V\left(t,z\right)\right\rangle_E$ with $\bar{\mathcal{W}} = \left\langle 0,\bar{W}\right\rangle_E$ and $\bar{W} \in \mathbb{S}_{++}^{n_w\times n_w}$. That said, we still require the differential tensors of $V\left(t,z\right)$.

Using Prop. 2.1, the differential tensors of $V\left(t,z\right)$ are simply

$$\begin{aligned}
dV\left(t,z\right) &= d\int_0^\delta U\left(\tau,t,z\right)\mathrm{d}\tau = \int_0^\delta dU\left(\tau,t,z\right)\mathrm{d}\tau, \qquad (6.67)\\
d^2V\left(t,z\right) &= \int_0^\delta d^2U\left(\tau,t,z\right)\mathrm{d}\tau, \qquad (6.68)\\
dU\left(\tau,t,z\right) &= dM\left(\tau,t,z\right)\bar{W}M\left(\tau,t,z\right)^T + M\left(\tau,t,z\right)\bar{W}dM\left(\tau,t,z\right)^T,\\
d^2U\left(\tau,t,z\right) &= d^2M\left(\tau,t,z\right)\bar{W}M\left(\tau,t,z\right)^T + 2dM\left(\tau,t,z\right)\bar{W}dM\left(\tau,t,z\right)^T\\
&\quad + M\left(\tau,t,z\right)\bar{W}d^2M\left(\tau,t,z\right)^T,\\
dM\left(\tau,t,z\right) &= de^{A_{\mathrm{cl}}(t,z)(t-\tau)}E\left(t,P\right) + e^{A_{\mathrm{cl}}(t,z)(t-\tau)}dE\left(t,P\right),\\
d^2M\left(\tau,t,z\right) &= d^2e^{A_{\mathrm{cl}}(t,z)(t-\tau)}E\left(t,P\right) + 2de^{A_{\mathrm{cl}}(t,z)(t-\tau)}dE\left(t,P\right)\\
&\quad + e^{A_{\mathrm{cl}}(t,z)(t-\tau)}d^2E\left(t,P\right),
\end{aligned}$$

with $d\bar{W} = 0$ and $d^2\bar{W} = 0$. We can now evaluate $V\left(t,z\right)$, $dV\left(t,z\right)$, and $d^2V\left(t,z\right)$ in (6.66) to (6.68) by numerical integration for a given t and z. It now only remains to compute the differential tensors of $e^{A_{\mathrm{cl}}(t,z)t}$.

Since the differential tensors of $e^{A_{\mathrm{cl}}(t,z)t}$ also require the the differential tensors of $A_{\mathrm{cl}}\left(t,z\right)$, we first compute $dA_{\mathrm{cl}} = dA + dBK + BdK$ and $d^2A_{\mathrm{cl}} = d^2A + d^2BK + dBdK + Bd^2K$ by applying the chain rule from Prop. 2.1, where $dK \in \mathbb{R}^{1\times n_z\times n_u\times n_x}$ and

$d^2K \in \mathbb{R}^{n_z \times n_z \times n_u \times n_x}$ are computed according to Prop. A.3 and Prop. A.4, respectively. The differential tensor of the matrix exponential with $\mathrm{d}t = 0$ can then be computed by using its definition, i.e.

$$e^{A_{\mathrm{cl}}t} = \sum_{k=0}^{\infty} \frac{(A_{\mathrm{cl}})^k \, t^k}{k!},$$

so that

$$\begin{aligned} de^{A_{\mathrm{cl}}t} &= \sum_{k=1}^{\infty} \frac{t^k \left(dA_{\mathrm{cl}} A_{\mathrm{cl}}^{k-1} + A_{\mathrm{cl}} d\left(A_{\mathrm{cl}}^{k-1} \right) \right)}{k!}, \\ d^2e^{A_{\mathrm{cl}}t} &= \sum_{k=1}^{\infty} \frac{t^k \left(d^2A_{\mathrm{cl}} A_{\mathrm{cl}}^{k-1} + 2dA_{\mathrm{cl}} d\left(A_{\mathrm{cl}}^{k-1} \right) + A_{\mathrm{cl}} d^2\left(A_{\mathrm{cl}}^{k-1} \right) \right)}{k!}, \end{aligned}$$

where we used

$$\begin{aligned} d\left(A_{\mathrm{cl}}^{i+1} \right) &= dA_{\mathrm{cl}} A_{\mathrm{cl}}^{i} + A_{\mathrm{cl}} d\left(A_{\mathrm{cl}}^{i} \right), \\ d^2\left(A_{\mathrm{cl}}^{i+1} \right) &= d^2A_{\mathrm{cl}} A_{\mathrm{cl}}^{i} + 2dA_{\mathrm{cl}} d\left(A_{\mathrm{cl}}^{i} \right) + A_{\mathrm{cl}} d^2\left(A_{\mathrm{cl}}^{i} \right), \end{aligned}$$

for $i \geq 1$ and where the infinite series of the matrix exponential can be truncated after a finite number of steps since its convergent for any $A_{\mathrm{cl}}t$ [49, Prop. 2.1].

Deviation input: Analogously to the deviation state, we only require the derivatives of $\tilde{\rho}_{\Delta u}^{(\mathcal{E})}(l,t,z)$ as defined in (6.42), here restated for convenience:

$$\tilde{\rho}_{\Delta u}^{(\mathcal{E})}(l,t,z) = \sqrt{l^T D_{\Delta u}(t,z)^T \, l + \epsilon},$$

where

$$\begin{aligned} D_{\Delta u}(t,z) &= K(t,z) \, D_{\Delta x}(t,z) \, K(t,z), \\ dD_{\Delta u}(t,z) &= K(t,z) \, dD_{\Delta x}(t,z) \, K(t,z) + 2dK(t,z) \, D_{\Delta x}(t,z) \, K(t,z), \\ d^2D_{\Delta u}(t,z) &= K(t,z) \, d^2D_{\Delta x}(t,z) \, K(t,z)^T + 2d^2K(t,z) \, D_{\Delta x}(t,z) \, K(t,z)^T \\ &\quad + 4dK(t,z) \, dD_{\Delta x}(t,z) \, K(t,z)^T + 2dK(t,z) \, D_{\Delta x}(t,z) \, dK(t,z)^T. \end{aligned}$$

Its derivatives are then – analogously to (6.62) and (6.63) – given by

$$\begin{aligned} d\tilde{\rho}_{\Delta u}^{(\mathcal{E})}\left(l,\hat{t},z\right) &= \frac{l^T dD_{\Delta u}\left(\hat{t},z\right) l}{2\tilde{\rho}_{\Delta u}^{(\mathcal{E})}\left(l,\hat{t},z\right)}, \\ d^2\tilde{\rho}_{\Delta u}^{(\mathcal{E})}\left(l,\hat{t},z\right) &= \frac{l^T d^2D_{\Delta u}\left(\hat{t},z\right) l - 2d\tilde{\rho}_{\Delta u}^{(\mathcal{E})}\left(l,\hat{t},z\right) d\tilde{\rho}_{\Delta u}^{(\mathcal{E})}\left(l,\hat{t},z\right)}{2\tilde{\rho}_{\Delta u}^{(\mathcal{E})}\left(l,\hat{t},z\right)}, \end{aligned}$$

so that its Jacobian and Hessian matrix follow identically to (6.64) and (6.65) using Th. 2.4 and Th. 2.5.

6.3.8 Initial Guess

Due to the structure of the iPROC approach (also see Alg. 6), good initial guesses may drastically reduce the number of overall iterations. As a result, we shortly describe possible initial guess computations for both the feedforward controller parameters and the feedback controller parameters.

We initialize the feedforward controller parameters by executing the GSC approach described in Sec. 5.2. If polynomial feedforward controllers are to be used, PGSC can also be employed to compute an initial guess. While iPGSC may also be used, it is not as suitable for the initial guess computation since there is no point in fine tuning P for the given input capacity without considering the input capacity needed for the feedback controller.

For the feedback controller parameters Q and R, one may simply set $Q = I_{n_x}$ and $R = I_{n_u}$. Since the feedback matrices are parameterized as LQR gain matrices, any choice for Q and R results in (locally) asymptotically stable systems for $\Delta x = x - x_{\mathrm{ff}}$, which describes the deviation of the closed-loop, disturbed state from the undisturbed feedforward state.

6.3.9 Computational Complexity

We again use Ass. 5.1 and distinguish between the offline and the online computational complexity in n. For simplicity, let $\kappa \geq 1$ hold for the order of the controller template introduced in Sec. 5.3.2 and let

$$n_z = an_u + n_x + n_u, \tag{6.69}$$

$$O(a) \geq O(n), \tag{6.70}$$

so that $O(n_z) = O(an)$.

We provide an upper bound on the polynomial complexity of the offline computation in Sec. 6.3.9.1 and then briefly describe the online application of the controller in Sec. 6.3.9.2.

6.3.9.1 Offline Complexity

The maximum number of iterations in Alg. 6 is fixed. As a result, it suffices to derive the complexity for a single trust-region iteration. As the numerical experiments in Sec. 6.3.10 will demonstrate, this does not really affect the performance of iPROC as the number of iterations in practice is often small and independent of the state dimension.

Controller Template: The dominating complexity is given by $O\left(an^{1+\kappa}\right)$ according to Sec. 5.3.6.1.

Reachable Sets: The combined complexity of computing the closed-loop reachable set and the parameterized feedforward reachable set can be bounded from above by $O\left(n^5 + n^2a^2 + n^3a\log n\right)$ (see Sec. 5.3.6.1).

Trust-Region Subproblem: Solving (6.50) involves the precomputation of the Jacobian and Hessian matrices with complexity $O\left(a^2 n^4\right)$ (see Sec. 5.3.6.1) of the feedforward reachable set approximation and the computation of the objective and constraint functions during optimization. Since the number of function evaluations $v\left(\epsilon\right)$ can be polynomially bounded in $\frac{1}{\epsilon}$ for the accuracy $\epsilon > 0$ of a first-order critical point if second-order methods are used [18], we focus on the complexity of one evaluation of the objective and constraint function subsequently.

We now analyze the complexity of evaluating the approximations to the state deviation and input deviation support functions (and their corresponding derivatives) from (6.41) and (6.43). For a worst-case complexity analysis, we only consider the computation of the Hessian matrix of these approximations. For their computation, we require the feedback matrix using LQR control with complexity $O\left(n_x^3\right)$ (see Sec. 2.4), its second-order tensor with complexity $O\left(n_x^{2\omega} + n^4 n_z^2\right) = O\left(a^2 n^6\right)$ (see Prop. A.4) using (6.69), (6.70), and $2\omega \leq 6$, and the second-order tensor of the approximated shape matrix $D_{\Delta x}$ which is computed via (6.39). For the computation of $D_{\Delta x}$, we evaluate (6.39) $m_\mathrm{c} m_\mathrm{d}$ times, where one evaluation is dominated by the complexity of computing the second-order tensor of the matrix integral in (6.39). Here, computing the second-order tensor of $e^{A_\mathrm{cl}(t,z)\delta}$ dominates with complexity $O\left(\vartheta n_x^\omega n_z^2\right) \overset{(6.69)}{=} O\left(a^2 n^{\omega+2}\right)$, where $\vartheta \in \mathbb{N}_+$ denotes the finite number of terms from the infinite series of $e^{A_\mathrm{cl}(t,z)\delta}$ to compute the matrix exponential accurately enough; the complexity for each of the ϑ terms is $O\left(n_x^\omega n_z^2\right)$, which follows from the complexity of multiplying two matrices but where each element multiplication is the outer product of two n_z-dimensional vectors (also see Sec. 6.3.7). To derive the computational complexity of solving the integral in (6.39), we solve the ODE corresponding to this integral with complexity $O\left(b\vartheta n_x^\omega n_z^2\right) \overset{(6.69)}{=} O\left(a^2 n^{\omega+2}\right)$, where $b \in \mathbb{N}_+$ collects the fixed number of function evaluations per ODE step and the total number of ODE steps: For instance, the classical Runge-Kutta integration scheme requires four function evaluations per step and the number of ODE steps can be assumed fixed if a solver with a fixed step size is used. Evaluation of (6.39) $m_\mathrm{c} m_\mathrm{d}$ times over $v\left(\epsilon\right)$ optimization iterations and computing the second-order differential tensor for the LQR gain matrix m_c times thus yields the overall complexity

$$O\left(v\left(\epsilon\right) m_\mathrm{c} \left(n_x^{2\omega} + n_x^4 n_z^2 + m_\mathrm{d} b \vartheta n_x^\omega n_z^2\right)\right) = O\left(a^2 n^6\right),$$

for one objective and constraint function evaluation of (6.50).

Overall Complexity: The offline complexity of iPROC can thus be bounded from above by

$$\begin{aligned} O\left(c_\mathrm{iPROC}^\mathrm{(off)}\left(n\right)\right) &= O\left(c_\mathrm{GSC}^\mathrm{(off)}\left(n\right) + a n^{1+\kappa} + n^5 + n^2 a^2 + n^3 a \log n + a^2 n^6\right) \\ &\overset{(5.14)}{=} O\left(c_\mathrm{ref}\left(n\right) + a n^{1+\kappa} + a^2 n^6\right), \end{aligned} \tag{6.71}$$

if the GSC approach is used to compute an initial guess for the feedforward parameters (see Sec. 6.3.8) and where $O\left(c_\mathrm{ref}\left(n\right)\right)$ denotes the complexity of the reference trajectory computation as detailed in Sec. 5.2.1.

6.3.9.2 Online Complexity

The feedforward controller evaluation has the same complexity as the PGSC approach, i.e. $O\left(c_{\mathrm{PGSC}}^{(\mathrm{on})}(n)\right) \overset{(5.34)}{=} O(an)$ since we use the same controller template. The feedback controller evaluation is dominated by the simulation of the feedforward state with complexity $O\left(b\left(n_x e + c_{\mathrm{PGSC}}^{(\mathrm{on})}(n)\right)\right) = O(an)$ using (5.34) and Ass. 5.1, where $b \in \mathbb{N}_+$ collects the total number of function evaluations for a fixed step-size ODE solver: Evaluating the dynamics f requires $O(e) = O(n)$ elementary operations $e \in \mathbb{N}_+$ by Ass. 5.1 for each dimension and we need to evaluate the feedforward controller for all b function evaluations. Thus, the overall online complexity can be bounded from above by

$$O\left(c_{\mathrm{iPROC}}^{(\mathrm{on})}(n)\right) = O\left(c_{\mathrm{PGSC}}^{(\mathrm{on})}(n) + an\right) \overset{(5.34)}{=} O(an).$$

6.3.10 Experiments

In this section, we compare the performance of ROC to our novel iPROC synthesis algorithm. Since derivatives for the optimization of the synthesis problem of ROC cannot be computed analytically (see Sec. 6.3.7), ROC uses parallel computing to numerically approximate the derivative information using finite differences (see [53, Chap. 5] for details on finite differences): For the processor used in this thesis, this means that ROC uses eight processing cores compared to a single core used by iPROC. Put differently, ROC uses around eight times more computational power than iPROC. That said, the experiments will demonstrate that iPROC is still competitive, even with much less computational power. We set $\zeta = 0$ for all benchmarks in this section and use an optimality tolerance $\mu = 10^{-2}$ and controller order $\kappa = 1$ for the iPROC algorithm.

6.3.10.1 Space Rendezvous

We present the space rendezvous benchmark proposed in [38, p. 3.6], [20] for the *approaching* mode. The dynamics of the controlled spacecraft, given in a coordinate system relative to the target as shown in Fig. 6.5, are given by

$$\dot{x}_1 = x_3, \tag{6.72a}$$
$$\dot{x}_2 = x_4, \tag{6.72b}$$
$$\dot{x}_3 = n^2 x_1 + 2n x_4 + \frac{c}{r^2} - \frac{c}{r_c^2}(r + x_1) + \frac{u_1}{m_c}, \tag{6.72c}$$
$$\dot{x}_4 = n^2 x_2 - 2n x_3 - \frac{c}{r_c^3} x_2 + \frac{u_2}{m_c}, \tag{6.72d}$$

where $c = 1.4350 \times 10^{18}\,\mathrm{m^3\,min^{-2}}$, $r = 42\,164 \times 10^3\,\mathrm{m}$, $m_c = 500\,\mathrm{kg}$, $n = \sqrt{\frac{c}{r^3}}$, and $r_c = \sqrt{(r + s_x)^2 + s_y^2}$. Let $x = \begin{bmatrix} s_y, & s_y, & v_x, & v_y \end{bmatrix}$ denote the state with spatial coordinates $\begin{bmatrix} s_x, & s_y \end{bmatrix}$ and velocity $\begin{bmatrix} v_x, & v_y \end{bmatrix}$, and denote with $u = \begin{bmatrix} F_x, & F_y \end{bmatrix}^T$ the input to the system, where F_x and F_y are thrust forces. The parameters of the benchmark are shown

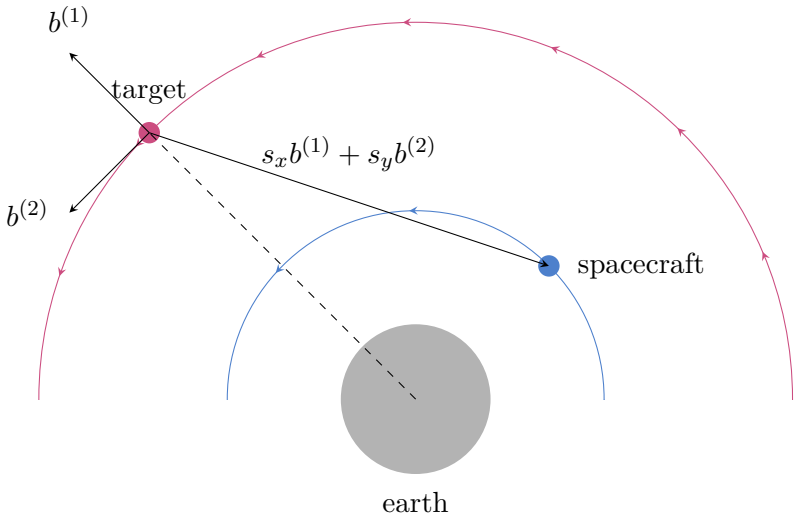


Figure 6.5: Depiction of the relative coordinate system for the spacecraft model. Here, $b^{(1)} \in \mathbb{R}^2$ and $b^{(2)} \in \mathbb{R}^2$ denote the basis vectors of the relative coordinate system in some global coordinate system.

in Tab. 6.1. For ROC, we additionally set $\tilde{\mathcal{U}} = 0.75\,\mathcal{U}$ and $\tilde{\mathcal{X}} = \mathcal{X}$. We note that the original authors of the benchmark in [20] proposed $\mathcal{V}_{\text{orig}} = \left\{ v \in \mathbb{R}^2 \;\middle|\; \sqrt{v_x^2 + v_y^2} \leq 3.3 \right\}$ as state velocity constraints. Generally, an arbitrarily tight inner approximation of $\mathcal{V}_{\text{orig}}$ can be computed by first inner approximating $\mathcal{V}_{\text{orig}}$ using Th. 4.4 and then using [8, Th. 7] for the exact conversion from a zonotope to an H-polytope; however, the resulting H-polytope has potentially many halfspaces. For this benchmark, a crude inner approximation of $\mathcal{V}_{\text{orig}}$ suffices: Looking at Fig. 6.6, the velocity constraints in $\begin{bmatrix} x_3, & x_4 \end{bmatrix}$ are easily satisfied. Therefore, we compute a parallelotope inner approximation of the velocity sphere constraint as $\mathcal{V} = \check{\mathcal{Z}}\left(\left\langle 0, 3.3^2 I_2 \right\rangle_E\right)_1$ using Th. 4.4.

Both ROC and iPROC find a feasible solution in 553 s and 732 s, respectively, where $\text{csize}\left(\mathcal{R}_{\text{ROC}}\left(t_{\text{f}}\right)\right) = 0.19$ and $\text{csize}\left(\mathcal{R}_{\text{iPROC}}\left(t_{\text{f}}\right)\right) = 0.15$. Fig. 6.6 shows the solution for both approaches, with a more detailed comparison of their final reachable sets in Fig. 6.7. Additionally, Fig. 6.8 shows the progress of iPROC over the iterations, which validates the result of Th. 6.1 that the approximated trust-region problem can approximate the formally verified controller cost arbitrarily closely.

Table 6.1: Parameters for the space rendezvous benchmark.

Parameter	Value
m_c	5
$\mathcal{X}^{(0)}$	$\left\langle \begin{bmatrix} -95, & -30, & 0, & 0 \end{bmatrix}^T, \mathrm{diag}\left(\begin{bmatrix} 2, & 2, & 0.1, & 0.1 \end{bmatrix}\right) \right\rangle_Z$
$\mathcal{U}$	$\langle 0, 2 \rangle_Z$
$\mathcal{W}$	$\langle 0, 0.1 I_2 \rangle_Z$
$\mathcal{X}$	$\mathcal{S} \times \mathcal{V}$
$\mathcal{S}$	$\left\langle \begin{bmatrix} -1, & 0 \\ \tan 20°, & -1 \\ \tan 20°, & 1 \end{bmatrix}, \begin{bmatrix} 100 \\ 0 \end{bmatrix} \right\rangle_H$
$\mathcal{V}$	$\check{\mathcal{Z}}\left(\langle 0, 3.3^2 I_2 \rangle_E\right)_1 = \langle 0, G \rangle_Z = \left\langle \begin{bmatrix} G^{-1} \\ -G^{-1} \end{bmatrix}, 1_4 \right\rangle_H$
t_f	$200\,\mathrm{s}$
x_f	$\begin{bmatrix} -1, & 0, & 0, & 0 \end{bmatrix}^T$
κ	1

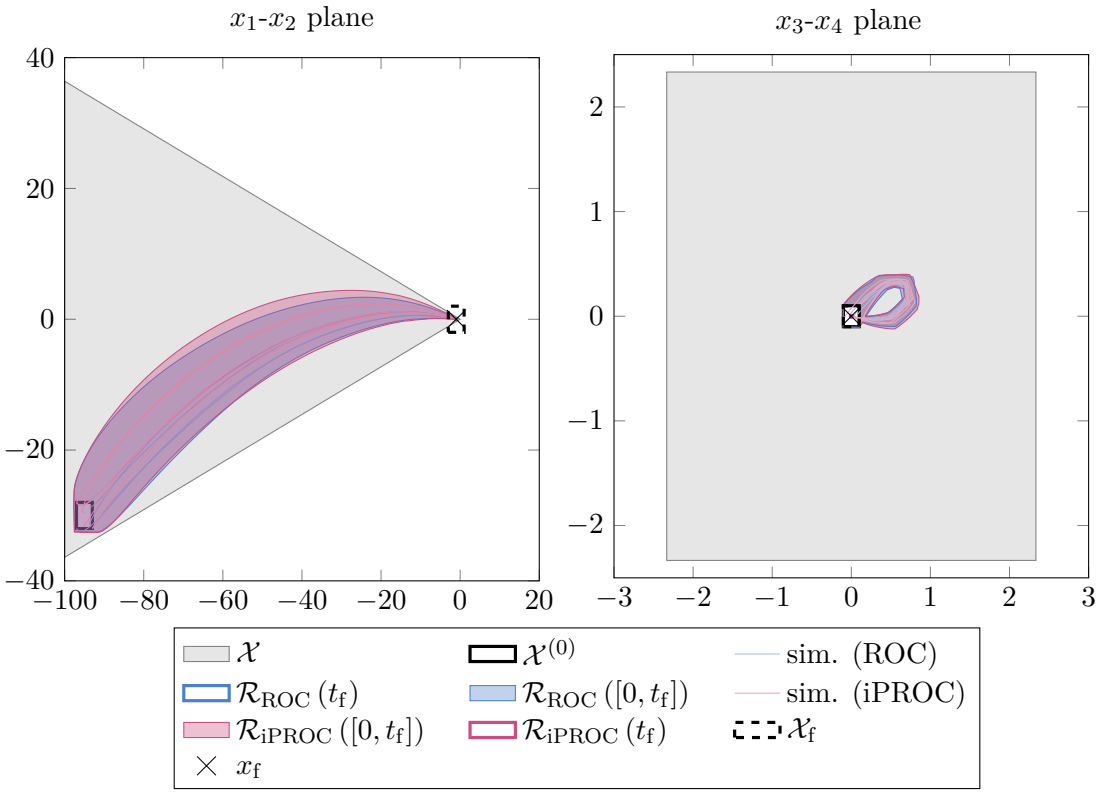


Figure 6.6: Comparison of the ROC approach and the iPROC approach for the space rendezvous benchmark. Fig. 6.7 shows the final, closed-loop reachable sets of both approaches in more detail.

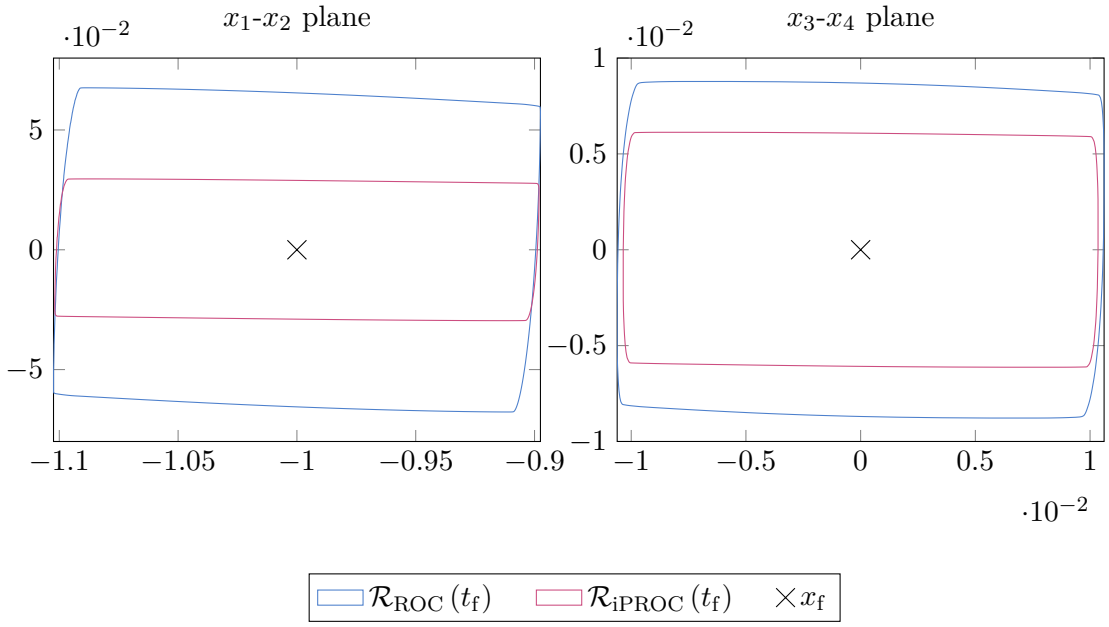


Figure 6.7: Comparison of the final closed-loop reachable set of ROC and iPROC for the space rendezvous benchmark.

6.3.10.2 Kinematic Single-Track Model (Car)

We consider a variant of the kinematic single-track model[1]

$$\dot{x}_1 = u_1 + w_1, \tag{6.73a}$$
$$\dot{x}_2 = u_2 + w_2, \tag{6.73b}$$
$$\dot{x}_3 = x_1 \cos(x_2), \tag{6.73c}$$
$$\dot{x}_4 = x_1 \sin(x_2), \tag{6.73d}$$

where $x = \begin{bmatrix} v, & \Psi, & s_x, & s_y \end{bmatrix}^T$ is the state of the system, v is the longitudinal velocity of the vehicle, $\Psi \in [-\pi, \pi]$ is the vehicle heading, $\begin{bmatrix} s_x, & s_y \end{bmatrix}^T$ denote the spatial coordinates, and $u = \begin{bmatrix} u_1, & u_2 \end{bmatrix}^T = \begin{bmatrix} a, & \phi \end{bmatrix}^T$ is the input to the system, where a is the longitudinal acceleration and ϕ is the steering rate. All benchmark parameters are shown in Tab. 6.2. To arrive at a feasible solution for ROC, we additionally set $\tilde{\mathcal{U}} = \operatorname{diag}\left(\begin{bmatrix} 0.3, & 0.7 \end{bmatrix}\right)\mathcal{U}$, weigh the final reachable set in the objective function of the combined controller in (6.9) with $Q_\mathrm{obj} = \operatorname{diag}\left(\begin{bmatrix} 0.5, & 10, & 1, & 1 \end{bmatrix}\right)$, and weigh the undisturbed, final reachable set in the objective function of the feedforward computation with $Q_\mathrm{obj}^{(\mathrm{ff})} = \operatorname{diag}\left(\begin{bmatrix} 1, & 1000, & 1, & 1 \end{bmatrix}\right)^2$.

[1] Sec. 5, `https://gitlab.lrz.de/tum-cps/commonroad-vehicle-models/-/blob/master/vehicleModels_commonRoad.pdf`

[2] These values are mostly obtained by trial and error.

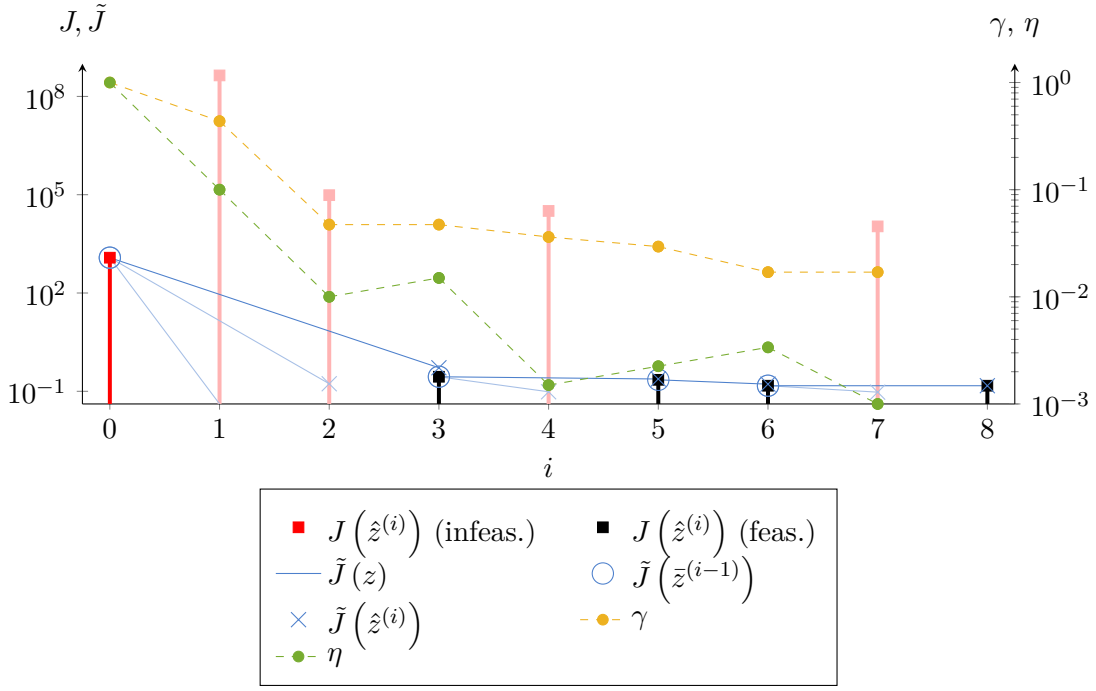


Figure 6.8: Visualization of the solver progress for the iPGSC approach (space rendezvous benchmark). Values between $\tilde{J}\left(\bar{z}^{(i-1)}\right)$ and $\tilde{J}\left(\hat{z}^{(i)}\right)$ for $1 \leq i \leq 8$ – where $\bar{z}^{(i-1)}$ and $\hat{z}^{(i)}$ are the initial and critical points of the trust-region subproblem – are linearly interpolated ($\hat{z}^{(0)} = \bar{z}^{(0)}$). Opaque values at $i \in \{1, 2, 4, 7\}$ visualize rejected iterations.

Table 6.2: Benchmark parameters for the car model.

Parameter	Value
m_c	5
$c_{\mathcal{X}^{(0)}}$	$\begin{bmatrix}20, & 0, & 0, & 0\end{bmatrix}^T$
$\mathcal{X}^{(0)}$	$\left\langle c_{\mathcal{X}^{(0)}}, \operatorname{diag}\left(\begin{bmatrix}0.2, & 0.02, & 0.2, & 0.2\end{bmatrix}\right)\right\rangle_Z$
$\mathcal{U}$	$\left\langle 0, \operatorname{diag}\left(\begin{bmatrix}9.81, & 0.4\end{bmatrix}\right)\right\rangle_Z$
$\mathcal{W}$	$\left\langle 0, \operatorname{diag}\left(\begin{bmatrix}2, & 0.08\end{bmatrix}\right)\right\rangle_Z$
$\mathcal{X}_\mathrm{f}$	$\{-c_{\mathcal{X}^{(0)}} + x_\mathrm{f}\} \oplus \mathcal{X}^{(0)}$
t_f	1 s
x_f	$\begin{bmatrix}20, & 0.2, & 19.87, & 1.99\end{bmatrix}^T$
κ	1

Fig. 6.9 depicts the reachable sets of the ROC and the iPROC approach for the car benchmark, where Fig. 6.10 shows the final reachable sets of both approaches in more detail. The sizes of the final reachable sets using (5.37) are $\operatorname{csize}\left(\mathcal{R}_\mathrm{ROC}\left(t_\mathrm{f}\right)\right) = 0.39$ and $\operatorname{csize}\left(\mathcal{R}_\mathrm{iPROC}\left(t_\mathrm{f}\right)\right) = 0.13$, where computation times for ROC and iPROC where 487 s and 497 s, respectively. While both ROC and iPROC find feasible solutions, ROC requires manual tuning (see, e.g., $\tilde{\mathcal{U}}$ above) while performing worse, i.e., finding a larger final reachable set.

6.3.10.3 Water Tanks

We now compare the scalability of ROC and iPROC for a model of $n \in \mathbb{N}_+$ water tanks [9] as shown in Fig. 6.11. The dynamics are given by

$$\begin{aligned}\dot{x}_1 &= u + w - k\sqrt{2gx_1},\\ \dot{x}_i &= k\left(\sqrt{2gx_{i-1}} - \sqrt{2gx_i}\right),\end{aligned}$$

for $2 \leq i \leq n$, where x_k is the water level of the k-th tank for $1 \leq k \leq n$, u is the inflow into the first tank, w is the uncontrollable inflow into the first tank, and we set $k = 0.015$ and $g = 9.81$. The parameters for this benchmark are listed in Tab. 6.3 and Fig. 6.12 shows the reachable sets for $n = 4$ tanks.

Tab. 6.4 shows the control performance using the metric given in (5.37) and the computation times for both ROC and iPROC for up to eight tanks, where ROC uses parallel computing and thus more computational resources. That said, the iPROC approach seems to scale better with the system dimension in this benchmark than ROC while also performing better overall.

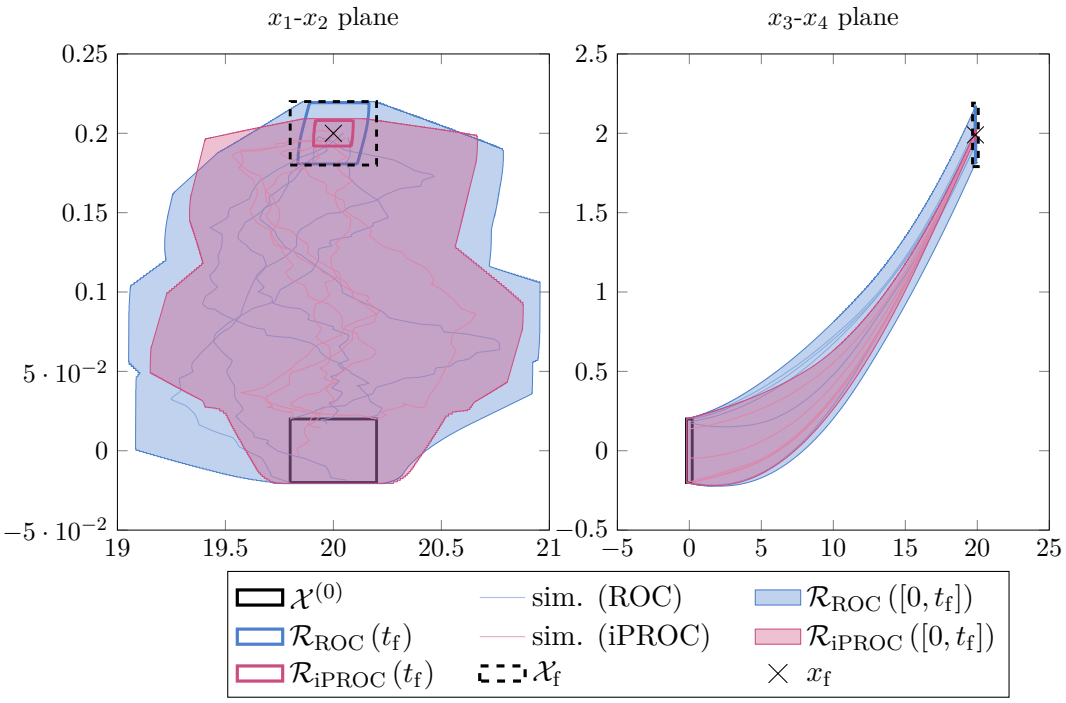


Figure 6.9: Comparison of the ROC approach and the iPROC approach for the kinematic single-track model (car).

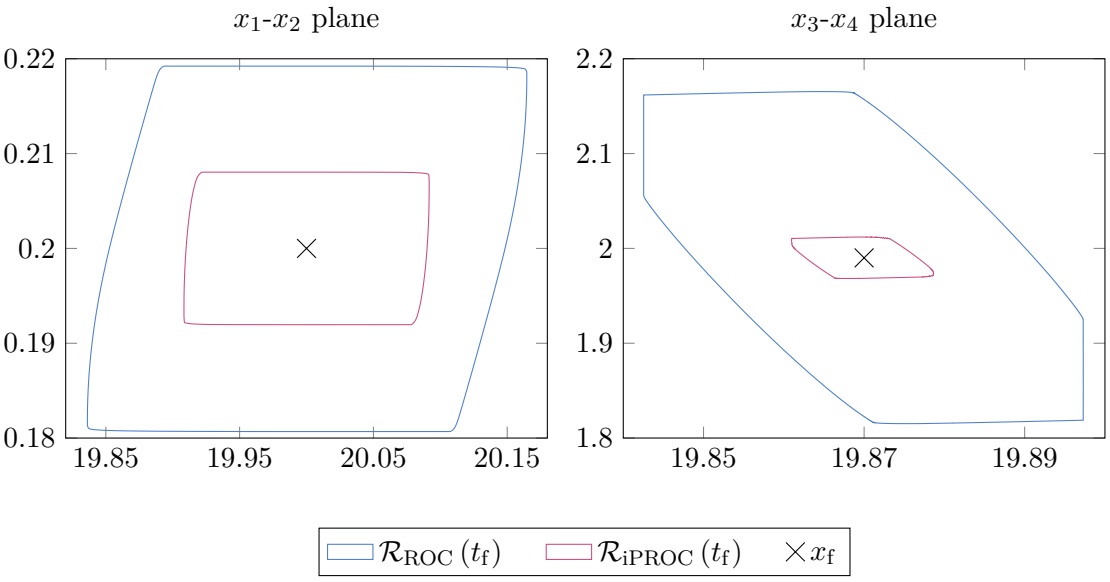


Figure 6.10: Comparison of the final closed-loop reachable set of the ROC approach and the iPROC approach for the car benchmark.

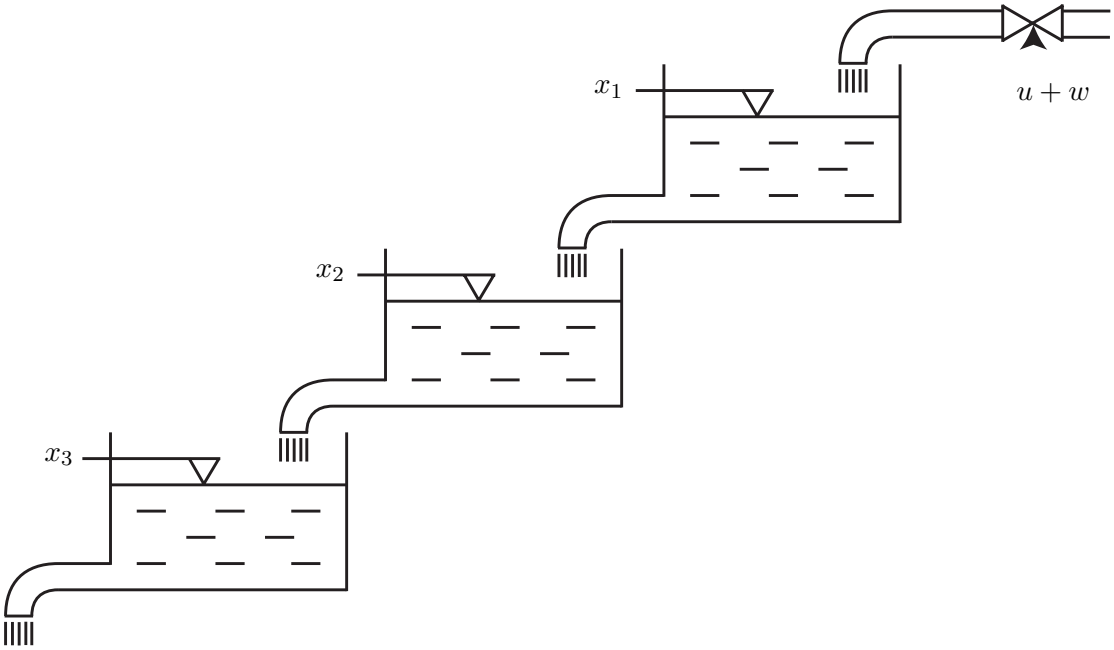


Figure 6.11: Model of the water tank, where x_i denotes the water level of tank i for $1 \leq i \leq 3$ and $u + w$ denotes the inflow rate caused by the input and the disturbance [9, Fig. 5, adapted].

Table 6.3: Benchmark parameters for the n-dimensional water tank model.

Parameter	**Value**
m_c	2
$c_{\mathcal{X}^{(0)}}$	$10 \cdot 1_n$
$\mathcal{X}^{(0)}$	$\langle c_{\mathcal{X}^{(0)}}, I_n \rangle_Z$
$\mathcal{U}$	$\langle 1, 1 \rangle_Z$
$\mathcal{W}$	$\langle 0, 0.02 \rangle_Z$
$\mathcal{X}_\mathrm{f}$	$\{-c_{\mathcal{X}^{(0)}} + x_\mathrm{f}\} \oplus \mathcal{X}^{(0)}$
t_f	120 s
x_f	$10 \cdot 1_n$
κ	1

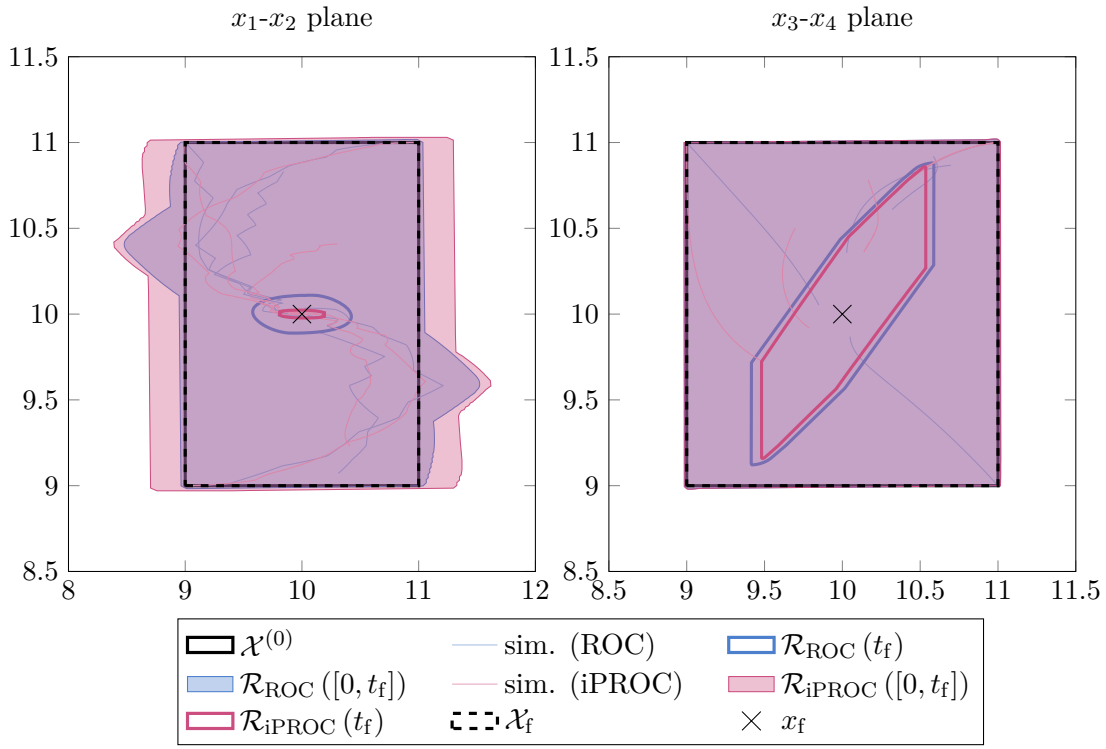


Figure 6.12: Comparison of the ROC approach and the iPROC approach for the $n = 4$ tank benchmark. The final state constraint $\mathcal{X}_\text{f}$ is only shown for reference but not included in the controller synthesis.

Table 6.4: Scalability comparison of iPROC and ROC for the tank benchmark using 2 to 8 tanks with $m_\text{c} = 2$ and no final state constraints.

nr. of tanks	**2**	**4**	**6**	**8**
time [s] **(iPROC)**	116	218	473	1056
time [s] **(ROC)**	30	102	383	1209
csize $(\mathcal{R}(t_\text{f}))$ **(iPROC)**	0.17	1.62	3.66	5.77
csize $(\mathcal{R}(t_\text{f}))$ **(ROC)**	0.48	2.00	4.00	6.06
nr. of iter. (iPROC)	10	8	8	8

6.3.11 Discussion

Compared to ROC, iPROC synthesizes the combined controller jointly and thus does not require the user to provide tightened feedforward input and adapted state constraints or weighting matrices for the feedforward or feedback synthesis. Additionally, the numerical experiments in Sec. 6.3.10 demonstrate that iPROC can synthesize more performant controllers than ROC, albeit at the cost of higher computation times, especially for lower-dimensional systems. Nevertheless, Tab. 6.4 indicates that iPROC scales better than ROC for an increasing number of system states in practice. Overall, iPROC is an almost one-click synthesis approach for the synthesis of a combined controller which requires minimal user input.

That said, the tuning of the trust-region radii can still be improved: While Th. 6.1 provides a general framework for how the radii can be adjusted, choosing the monotonically decreasing function and its parameters adequately so that iPROC works well over a range of systems is not trivial. Additionally, while not discussed here, it might further be possible to choose the m_c controller pairs of feedforward and feedback controller and the feedforward controller order κ automatically. Furthermore, it may be possible to obtain an initial guess for P, Q, and R by solving the synthesis problem for the combined controller for a finite set of initial states using optimal control.

6.4 Summary

In this chapter, we described the reachset optimal control (ROC) approach from [101] and introduced our novel iterative polynomial reachset optimal control (iPROC) algorithm from [35], which – for the first time – synthesizes the feedforward and feedback controller simultaneously and thus avoids the introduction of additional algorithm parameters.

ROC synthesizes a combined controller, consisting of a piecewise constant feedforward controller, which is linear in the initial state, and a continuous state feedback term. Using the GSC approach, the feedforward controller is first synthesized separately. The feedforward controller then is kept fixed and the feedback controller is optimized: To evaluate the objective function and the constraints, an outer approximation of the closed-loop reachable set has to be computed in each iteration. To reduce the number of optimization variables for the feedback controller, LQR control with diagonal weighting matrices for LQR are used. In summary, ROC allows for a straightforward extension of the feedforward synthesis to the combined synthesis of a feedforward controller with continuous state feedback by combining reachability analysis with optimization.

However, even though the feedforward controller has direct influence on the feedback controller, ROC keeps the feedforward controller constant during the feedback synthesis, yielding suboptimal results. Furthermore, this sequential optimization introduces additional algorithm parameters that need to be tuned by the user. Additionally, the feedback synthesis recomputes an outer approximation to the closed-loop reachable set in each optimization iteration, and thus is computationally expensive. As a result, we proposed the novel iPROC approach, which – for the first time – jointly synthesizes the feedforward and feedback controller iteratively: Since the original synthesis prob-

lem is in general hard to solve, we approximate it with a cost function. Because the evaluation of this controller cost is computationally expensive as it requires reachability analysis, we avoid optimizing over it directly and rather propose a smooth optimization problem in the controller parameters, which approximates this cost and can be solved with any off-the-shelf optimization solver. That said, this optimization problem approximates the cost accurately only in a local neighborhood of these parameters. Thus we restrict the domain of the approximated optimization problem to a trust region around the current controller parameters and iteratively update them whenever the cost decreases. We adapt the size of the trust region in each iteration to ensure the accuracy of the approximated parameterized reachable set. Hence, iPROC enables the simultaneous optimization over the combined controller parameters and avoids the introduction of additional algorithm parameters that need to be tuned by the user. The effectiveness of iPROC was demonstrated using numerical benchmarks.

7 Conclusion and Future Directions

7.1 Conclusion

In this thesis, we developed different control synthesis algorithms which combine reachability analysis and optimization theory to obtain non-conservative and formally verified results. To formulate efficient algorithms, we further developed new set conversions between zonotopes and ellipsoids. The presented algorithms all operate on disturbed nonlinear systems, respect input and state constraints, and have polynomial complexity in the state dimension, making them applicable to a wide range of tasks, such as autonomous driving and human-robot interaction tasks.

We first introduced the general synthesis problem and defined the abstracted synthesis problem, which is constructed such that all relevant optimization problems from subsequent chapters can be generalized to this problem. We showed that the abstracted synthesis problem can be reformulated as a smooth non-convex optimization problem. Because many solvers find solutions to an optimization problem by solving the associated Karush-Kuhn-Tucker (KKT) conditions, we furthermore proved that any critical point of said optimization problem is regular. As a result, the first-order KKT conditions are always necessary conditions for all novel optimization problems in subsequent chapters, and thus many off-the-shelf optimization solvers can be used.

For an efficient and numerically stable way to pose and solve these synthesis problems, we further derived novel conversions between sets, such as ellipsoid-zonotope conversions from our work in [36]. These conversions are necessary since not all set operations are closed under all set operations. As a result, the same operation for two different set representations may have different tightness and computational efficiency. Since we synthesize controllers under input and state constraints, we furthermore reviewed existing methods to check the containment of two sets, which are used in subsequent chapters to, e.g., check the state constraint satisfaction of a given reachable set.

Next, we first focused on the synthesis of piecewise constant controllers. Here, we introduced the generator-space control (GSC) approach from previous work in [99] for the synthesis of piecewise constant controllers for disturbed nonlinear systems under constraints. While GSC is computationally efficient since it only requires the solution of a single linear program (LP) for the synthesis of its controller, it only optimizes over the reachable set of the linearized dynamics. Thus, we proposed polynomial generator-space control (PGSC) from our work in [37] as an extension of GSC, which synthesizes piecewise constant controllers that are polynomially dependent on the initial state based on a polynomial abstraction of the nonlinear dynamics. However, since both GSC and PGSC only approximate the reachable set without any formal guarantees, the inclusion of state constraints always requires additional input from the user. To solve this problem, we

introduced the iterative polynomial generator-space control (iPGSC) approach, allowing us to synthesize polynomial, piecewise constant controllers under input and state constraints without the need for additional user input.

However, piecewise constant controllers often cannot counteract disturbances efficiently when disturbances are large or act quickly, resulting in larger reachable sets. Thus, we first described the reachset optimal control (ROC) algorithm from [101], which synthesizes a piecewise constant feedforward controller with continuous state feedback for nonlinear disturbed systems under input and state constraints. However, because this synthesis is executed sequentially, i.e., the feedforward controller is computed first and then fixed in the subsequent optimization of the feedback control term, the resulting controller may be suboptimal. Furthermore, this sequential computation approach produces additional algorithm parameters that require manual tuning by the user. Thus, we proposed the iterative polynomial reachset optimal control (iPROC) approach from our work in [35] which – for the first time – synthesizes a feedforward and a feedback controller simultaneously by solving a single optimization problem. To avoid extensive reachability computations, we approximated the synthesis problem and used trust-region radii to restrict the domain of the controller parameters for this approximated optimization problem to ensure accuracy.

7.2 Future Directions

We presented approaches for the formally verified controller synthesis of disturbed nonlinear systems under input and state constraints. Subsequently, we propose possible directions for future research to extend this work.

In PGSC and iPGSC, we use a piecewise constant controller which is polynomially dependent on the initial state. While this can achieve better control performance than linear controllers as demonstrated in Sec. 5.3.8, it requires the user to choose the appropriate controller order. Formulating the synthesis problem for a single initial state, optimal control may be able to quickly solve the synthesis problem for a set of possible controller orders to find a suitable choice. Additionally, one might be able to directly identify exponent vectors of the controller template which contribute most to good control performance. Such a sparse controller template could avoid an unnecessarily large number of controller parameters.

While our proposed approaches, specifically iPGSC and iPROC, already reduce the number of algorithm parameters that need to be set by the user, the number of piecewise controllers for iPGSC and the number of feedforward and feedback controller pairs in iPROC still needs to be chosen by the user. Here, optimal control for a single initial state could be used to determine the effect of a varying number of controllers on the control performance: By formulating the synthesis problem for a single initial state, optimal control methods can be used to quickly solve the synthesis problem for a given number of controllers. Since these methods are very efficient, evaluating the optimal control problem for a larger number of controllers might be feasible.

Furthermore, we assumed that the time horizon is split uniformly, i.e., each controller is always active for the same amount of time. However, there might be instances where a large number of controllers is only required for a short time interval – e.g., due to quickly changing system dynamics – so that subdividing the time horizon uniformly to achieve the required number of controllers for this short time interval is quite inefficient. Here, ideas similar to numerically integrating differential equations might be able to identify points in time where a higher number of controllers is required: Adaptive step-size solvers for ODEs are able to automatically decrease the time step size when the dynamics require a finer time resolution. A similar approach might be able to identify a non-uniform time discretization for the number of controllers.

Lastly, we approximated the synthesis problem in iPROC to avoid the computation of the reachable set in each optimization iteration. While this approximation is less expensive to compute than the reachable set in practice, it still evaluates the Riccati equation and its derivatives in each optimization iteration to compute the LQR feedback matrix for the feedback controller. To further reduce the computational effort and speed up the optimization, one may approximate the trust-region subproblem using yet another, inner trust-region approximation: We initially compute the Jacobian and Hessian matrices of the outer trust-region subproblem and then form an inner approximation by assuming that these computed Jacobian and Hessian matrices are constant. By again restricting the domain for the controller parameters of this inner approximation to a trust region, we can control the accuracy of this inner approximation. As a result, we avoid the recomputation of the derivative information in each optimization iteration of the trust-region subproblem.

A Appendix

A.1 Abstracted Synthesis Problem

A.1.1 Reformulation Equivalence

Proof of Prop. 3.1. We first prove the second claim by deriving the optimizers $\hat{s}$, $\hat{w}$, $\hat{y}$, $\hat{V}$, $\hat{T}$, and $\hat{p}$ as functions of $\hat{z}$, i.e., in dependence of $\hat{z}$ only. We note that the KKT conditions used in this proof are always necessary for (3.5) due to Prop. 3.2.

Value of $\hat{w}$: We start by analyzing the complementary slackness conditions of (3.5b), given by

$$\hat{\lambda}_i^{(+)} \left(+h_i\left(\hat{z}\right) - \hat{w}_i\right) = 0, \tag{A.1a}$$

$$\hat{\lambda}_i^{(-)} \left(-h_i\left(\hat{z}\right) - \hat{w}_i\right) = 0, \tag{A.1b}$$

where $\hat{\lambda}^{(+)} \in \mathbb{R}_{\geq 0}^{n_w}$ and $\hat{\lambda}^{(-)} \in \mathbb{R}_{\geq 0}^{n_w}$ are the corresponding optimal Lagrange multipliers with $1 \leq i \leq n_w$. Further, the stationary condition with respect to w is given by

$$l - \hat{\lambda}^{(+)} - \hat{\lambda}^{(+)} + M^T \sum_{k=1}^{n_k} \sum_{m=1}^{n_s} {F_{(m)}^{(k)}}^T \hat{\chi}_{(m)}^{(k)} = 0, \tag{A.2}$$

where $\hat{\chi}_{(m)}^{(k)} \in \mathbb{R}_{\geq 0}^{q}$ is the optimal Lagrange multiplier of the (km)-th constraint in (3.5d). Now assume that $\hat{w}_i > \left|h_i\left(\hat{z}\right)\right|$. It then holds that $\hat{\lambda}_i^{(+)} = \hat{\lambda}_i^{(-)} = 0$ due to (A.1). Since (A.2) must hold for general $l \geq 0$ and $M \in \mathbb{R}_{\geq 0}^{n_r \times n_w}$, it must also hold for $l = 1_{n_s}$ and $M = 0$, from which $\hat{\lambda}_i^{(+)} + \hat{\lambda}_i^{(-)} = 1$ follows, and thus by contradiction, it must hold that

$$\hat{w} = \left|h\left(\hat{z}\right)\right|, \tag{A.3}$$

since $\hat{w} \geq \left|h\left(\hat{z}\right)\right|$ (see (3.5b)).

Value of $\hat{y}$: The complementary slackness conditions of (3.5c) and the stationarity condition in y are given by

$$\hat{\underline{\zeta}}_i \left(\underline{R}_{(i,:)} \rho\left(\hat{z}\right) - y_i\right) = 0, \tag{A.4}$$

$$\hat{\bar{\zeta}}_i \left(\bar{R}_{(i,:)} \rho\left(\hat{z}\right) - y_i\right) = 0, \tag{A.5}$$

$$1 - \hat{\underline{\zeta}}_i - \hat{\bar{\zeta}}_i = 0, \tag{A.6}$$

for $1 \leq i \leq n_y$, where $\hat{\underline{\zeta}} \in \mathbb{R}^{n_y}_{\geq 0}$ and $\hat{\bar{\zeta}} \in \mathbb{R}^{n_y}_{\geq 0}$ are the optimal Lagrange multipliers of the respective constraints in (3.5c). It follows from (A.6) that $\hat{\underline{\zeta}}_i + \hat{\bar{\zeta}}_i = 1$. Assuming $\hat{y}_i > \max\left(\underline{R}_{(i,:)}\rho\left(\hat{z}\right), \bar{R}_{(i,:)}\rho\left(\hat{z}\right)\right)$ implies $\hat{\underline{\zeta}}_i = \hat{\bar{\zeta}}_i = 0$ due to (A.4) and (A.5), and thus

$$\hat{y}_i = \max\left(\underline{R}_{(i,:)}\rho\left(\hat{z}\right), \bar{R}_{(i,:)}\rho\left(\hat{z}\right)\right) \tag{A.7}$$

follows by contradiction and due to (3.5c).

Value of $\hat{s}$: The relevant KKT conditions in s are

$$\sigma - \hat{\delta}_m - \hat{\vartheta}_m = 0, \tag{A.8}$$

$$\hat{\delta}_m\left(\hat{g}_m - \hat{s}_m\right) = 0, \tag{A.9}$$

$$-\hat{\vartheta}_m \hat{s}_m = 0, \tag{A.10}$$

for $1 \leq m \leq n_s$, where $\hat{\delta} \in \mathbb{R}^{n_s}_{\geq 0}$ and $\hat{\vartheta} \in \mathbb{R}^{n_s}_{\geq 0}$ are the optimal Lagrange multipliers of the constraints in (3.5g) and (3.5i) and we define $\hat{g} = \hat{p} - b$. Summing (A.9) and (A.10) yields $\hat{\delta}_m \hat{g}_m - \left(\hat{\delta}_m + \hat{\vartheta}_m\right)\hat{s}_m = 0$, where a subsequent substitution of $\hat{\delta}_m + \hat{\vartheta}_m = \sigma$ from (A.8) and solving for $\hat{s}_m$ results in

$$\hat{s}_m = \frac{\hat{\delta}_m \hat{g}_m}{\sigma}. \tag{A.11}$$

Now let $\mathcal{I} = \{i \in \{1, ..., n_s\} \mid \hat{g}_i \leq 0\}$. Then $\forall i \in \mathcal{I} : \hat{\delta}_i \hat{g}_i = 0$ since necessarily $\hat{\delta}_i = 0$ for $\hat{g}_i < 0$ due to $s_i \geq 0$ and (3.5g) (statement trivially holds for $\hat{g}_i = 0$). For $\mathcal{J} = \{1, ..., n_s\} \setminus \mathcal{I}$, we further have $\forall j \in \mathcal{J} : \hat{g}_j > 0 \overset{(3.5g)}{\Longrightarrow} \hat{s}_j > 0 \overset{(A.10)}{\Longrightarrow} \hat{\vartheta}_j = 0 \overset{(A.8)}{\Longrightarrow} \hat{\delta}_j = \sigma$. Thus, (A.11) can be written as

$$\hat{s}_m = \begin{cases} \hat{g}_m, & \hat{g}_m > 0, \\ 0, & \text{otherwise} \end{cases} \tag{A.12}$$

for $1 \leq m \leq n_s$. To express $\hat{s}$ in $\hat{z}$ only, it remains to derive $\hat{g} = \hat{p} - b$ in $\hat{z}$ only. Thus, we assume $\hat{g}_m > 0$ for the remainder of this proof.

Value of $\hat{p}$: The stationarity condition in p and the complementary slackness condition of (3.5f) are given by

$$-1^T_{n_k} \hat{\gamma}_{(m)} + \hat{\delta}_m = 0, \tag{A.13}$$

$$\left[\hat{\gamma}_{(m)}\right]_k \left(\hat{V}_{km} + \hat{T}_{km} - \hat{p}_m\right) = 0, \tag{A.14}$$

for $1 \leq m \leq n_s$ and $1 \leq k \leq n_k$, where $\hat{\gamma}_{(m)} \in \mathbb{R}^{n_k}_{\geq 0}$ is the optimal Lagrange multiplier of the m-th constraint in (3.5f). Now assume that $\hat{p}_m > \max\left(\hat{V}_{(:,m)} + \hat{T}_{(:,m)}\right)$ for $\hat{g}_m > 0$.

It then holds that $\hat{\gamma}_{(m)} = 0$ due to (A.14), and thus $\hat{\delta}_m = 1_{n_k}^T \hat{\gamma}_{(m)} = 0$ due to (A.13); however, we previously proved that $\hat{\delta}_m = \sigma$ for $\hat{g}_m > 0$, and therefore

$$\hat{p}_m = \max\left(\hat{V}_{(:,m)} + \hat{T}_{(:,m)}\right) \text{ for } \hat{g}_m > 0, \tag{A.15}$$

follows by contradiction.

Value of $\hat{V}$: Since $\hat{p}_m = \max\left(\hat{V}_{(:,m)} + \hat{T}_{(:,m)}\right)$ for $\hat{g}_m > 0$, there must be a set of indices $\bar{\mathcal{K}}_{(m)} \subseteq \{1, \ldots, n_k\}$ for which holds that

$$\forall \bar{k} \in \bar{\mathcal{K}}_{(m)} : \hat{V}_{\bar{k}m} + \hat{T}_{\bar{k}m} = \hat{p}_m \text{ for } \hat{g}_m > 0, \tag{A.16a}$$

$$\forall \underline{k} \in \{1, \ldots, n_k\} \setminus \bar{\mathcal{K}}_{(m)} : \hat{V}_{\underline{k}m} + \hat{T}_{\underline{k}m} < \hat{p}_m \text{ for } \hat{g}_m > 0. \tag{A.16b}$$

From (A.13), (A.14), (A.16a) and (A.16b), it then follows that

$$1_{n_k}^T \hat{\gamma}_{(m)} = \sum_{\bar{k} \in \bar{\mathcal{K}}_{(m)}} \left[\hat{\gamma}_{(m)}\right]_{\bar{k}} = \hat{\delta}_m. \tag{A.17}$$

The stationarity conditions in V_{km} for $1 \leq k \leq n_k$ and the complementary slackness condition of (3.5d) are given by

$$-1_q^T \hat{\chi}_{(m)}^{(k)} + \left[\hat{\gamma}_{(m)}\right]_k = 0, \tag{A.18}$$

$$\left[\hat{\chi}_{(m)}^{(k)}\right]_i \left(\left[F_{(m)}^{(k)}\right]_{(i,:)} (Ah(\hat{z}) + M\hat{w}) - \hat{V}_{km}\right) = 0, \tag{A.19}$$

for $1 \leq i \leq q$, where $\hat{\chi}_{(m)}^{(k)} \in \mathbb{R}_{\geq 0}^q$ is the optimal Lagrange multiplier of the (km)-th constraint in (3.5d). Now assume that $\hat{V}_{\bar{k}m} > \max\left(F_{(m)}^{(\bar{k})} (Ah(\hat{z}) + M\hat{w})\right)$. It then follows from (A.19) that $\hat{\chi}_{(m)}^{(\bar{k})} = 0$ and thus $\left[\hat{\gamma}_{(m)}\right]_{\bar{k}} = 0$ due to (A.18). Since then $\sum_{\bar{k} \in \bar{\mathcal{K}}_{(m)}} \left[\hat{\gamma}_{(m)}\right]_{\bar{k}} = 1_{n_k}^T \hat{\gamma}_{(m)} = \hat{\delta} = 0$ (see (A.17)) but we previously showed that $\hat{\delta} = \sigma$ for $\hat{g}_m > 0$ (see derivations for $\hat{s}$), it follows by contradiction that

$$\hat{V}_{\bar{k}m} = \max\left(F_{(m)}^{(\bar{k})} (Ah(\hat{z}) + M\hat{w})\right) \text{ for } \hat{g}_m > 0. \tag{A.20}$$

Value of $\hat{T}$: The stationarity conditions in T_{km} for $1 \leq k \leq n_k$, $1 \leq m \leq n_s$, $1 \leq i \leq q$ and the complementary slackness condition of (3.5e) are given by

$$-1_q^T \hat{\phi}_{(m)}^{(k)} + \left[\hat{\gamma}_{(m)}\right]_k = 0, \tag{A.21}$$

$$\left[\hat{\phi}_{(m)}^{(k)}\right]_i \left(\left[G_{(m)}^{(k)}\right]_{(i,:)} \rho(\hat{z}) - \hat{T}_{km}\right) = 0, \tag{A.22}$$

where $\hat{\phi}_{(m)}^{(k)} \in \mathbb{R}_{\geq 0}^{q}$ is the optimal Lagrange multiplier of the (km)-th constraint in (3.5e). Similarly to $\hat{V}$, assume now that $\hat{T}_{\bar{k}m} > \max\left(G_{(m)}^{(\bar{k})}\rho\left(\hat{z}\right)\right)$ with $\bar{k} \in \bar{\mathcal{K}}_{(m)}$ as defined in (A.16) for $\hat{g}m > 0$. It then follows from (A.22) that $\hat{\phi}_{(m)}^{(\bar{k})} = 0$ and thus $\left[\hat{\gamma}_{(m)}\right]_{\bar{k}} = 0$ due to (A.21). Since then $\sum_{\bar{k}\in\bar{\mathcal{K}}_{(m)}} \left[\hat{\gamma}_{(m)}\right]_{\bar{k}} = 1_{n_k}^{T}\hat{\gamma}_{(m)} = \hat{\delta} = 0$ (see (A.17)) but we previously showed that $\hat{\delta} = \sigma$ for $\hat{g}_m > 0$ (see derivations for $\hat{s}$), it follows by contradiction that

$$\hat{T}_{\bar{k}m} = \max\left(G_{(m)}^{(\bar{k})}\rho\left(\hat{z}\right)\right) \text{ for } \hat{g}_m > 0. \tag{A.23}$$

Collecting all optimizers: We can now write $\hat{s}_m$ for $1 \leq m \leq n_s$ as

$$\begin{aligned}
&\hat{g}_m > 0: \\
&\quad \hat{s}_m \overset{(A.12)}{=} \hat{g}_m = \hat{p}_m - b_m \\
&\qquad \overset{(A.16a)}{=} \hat{V}_{\bar{k}m} + \hat{T}_{\bar{k}m} - b_m \\
&\qquad \overset{(A.3),(A.20),(A.23)}{=} \max\left(F_{(m)}^{(\bar{k})}\left(Ah\left(\hat{z}\right) + M\left|h\left(\hat{z}\right)\right|\right)\right) + \max\left(G_{(m)}^{(\bar{k})}\rho\left(\hat{z}\right)\right) - b_m \\
&\qquad \overset{(A.16a)}{=} \max_{0\leq k\leq n_k}\left\{\max\left(F_{(m)}^{(k)}\left(Ah\left(\hat{z}\right) + M\left|h\left(\hat{z}\right)\right|\right)\right) + \max\left(G_{(m)}^{(k)}\rho\left(\hat{z}\right)\right)\right\} - b_m,
\end{aligned}$$

and thus

$$\hat{s}_m = \begin{cases} g_m\left(\hat{z}\right), & g_m\left(\hat{z}\right) > 0, \\ 0, & \text{otherwise} \end{cases}, \tag{A.24}$$

or equivalently $\hat{s} = \max\left(0, g\left(\hat{z}\right)\right)$, where g is defined in (3.4). Therefore, substitution of $\hat{s}$, $\hat{w}$, and $\hat{y}$ in (3.6) by (A.24), (A.3), and (A.7) yields $J_{\text{smooth}}\left(\hat{z}, \hat{s}, \hat{w}, \hat{y}\right) = \tilde{J}\left(\hat{z}\right)$.

Same minimum: It remains to prove the first claim. Let (z^*, s^*) denote a global optimizer of (3.2). By design, it must hold that $s^* = \max\left(0, g\left(z^*\right)\right)$ since, if $s^* > \max\left(0, g\left(z^*\right)\right)$, choosing $\left(z^*, \max\left(0, g\left(z^*\right)\right)\right)$ achieves a smaller minimum but (z^*, s^*) is assumed to be a global optimizer. Thus, the minimum $\tilde{J}\left(z^*\right) = J\left(z^*, \max\left(0, g\left(z^*\right)\right)\right)$ is attained. Previously, we showed that the minimum objective value at a critical point $\left(\hat{z}, \hat{s}, \hat{w}, \hat{y}, \hat{V}, \hat{T}, \hat{p}\right)$ of (3.5) is $\tilde{J}\left(\hat{z}\right)$. We now prove the equivalence of (3.2) and (3.5) by

contradiction: Assume that $\tilde{J}(z^*) < \tilde{J}(\hat{z})$. Then $\left(\bar{z}, \bar{s}, \bar{w}, \bar{y}, \bar{V}, \bar{T}, \bar{p}\right)$ with

$$\begin{aligned}
\bar{z} &= z^*, \\
\bar{s} &= \max\left(0, g\left(z^*\right)\right), \\
\bar{w} &= \left|h\left(z^*\right)\right|, \\
\bar{y} &= \max\left(\underline{R}\rho\left(z^*\right), \bar{R}\rho\left(z^*\right)\right), \\
\bar{V}_{km} &= \max\left(F_{(m)}^{(k)}\left(Ah\left(z^*\right) + M\bar{w}\right)\right), \\
\bar{T}_{km} &= \max\left(F_{(m)}^{(k)}\rho\left(z^*\right)\right), \\
\bar{p}_m &= \max\left(\bar{V}_{(:,m)} + \bar{T}_{(:,m)}\right),
\end{aligned}$$

for $1 \leq k \leq n_k$ and $1 \leq m \leq n_s$ is a feasible point of (3.5) with

$$J_{\text{smooth}}\left(\bar{z}, \bar{s}, \bar{w}, \bar{y}, \bar{V}, \bar{T}, \bar{p}\right) = \tilde{J}\left(z^*\right) < \tilde{J}\left(\hat{z}\right),$$

but $\left(\hat{z}, \hat{s}, \hat{w}, \hat{r}, \hat{t}\right)$ is a global optimizer by assumption; thus, $\tilde{J}(z^*) \geq \tilde{J}(\hat{z})$ follows by contradiction. Similarly, now assume that $\tilde{J}(z^*) > \tilde{J}(\hat{z})$. Then $(\hat{z}, \max(0, g(\hat{z})))$ is a feasible point of (3.2) with $\tilde{J}(z^*) > \tilde{J}(\hat{z})$, but (z^*, s^*) is a global optimizer by assumption; thus, $\tilde{J}(z^*) = \tilde{J}(\hat{z})$ when combined with $\tilde{J}(z^*) \geq \tilde{J}(\hat{z})$ as proven above, which concludes the proof. □

A.1.2 Necessity of First-Order Optimality Conditions

Proof of Prop. 3.2. Let $\hat{x} = \left[\hat{z}^T, \ \hat{s}^T, \ \hat{w}^T, \ \hat{y}^T, \ \hat{V}_{(:)}^T \ \hat{T}_{(:)}^T, \ \hat{p}^T\right]^T \in \mathbb{R}^n$ denote a collected critical point of (3.5) with $\tilde{g}(\hat{x}) \leq 0$, where

$$\tilde{g}\left(\hat{x}\right) = \begin{bmatrix}
\begin{bmatrix} +h\left(\hat{z}\right) \\ -h\left(\hat{z}\right) \end{bmatrix} - \begin{bmatrix} \hat{w} \\ \hat{w} \end{bmatrix} \\
\begin{bmatrix} \underline{R}\rho\left(\hat{z}\right) \\ \bar{R}\rho\left(\hat{z}\right) \end{bmatrix} - \begin{bmatrix} \hat{y} \\ \hat{y} \end{bmatrix} \\
\vdots \\
F_{(m)}^{(k)}\left(Ah\left(\hat{z}\right) + M\hat{w}\right) - 1_q\hat{V}_{km} \\
\vdots \\
G_{(m)}^{(k)}\rho\left(\hat{z}\right) - 1_q\hat{T}_{km} \\
\vdots \\
\hat{V}_{(:,m)} + \hat{T}_{(:,m)} - 1_{n_k}p_m \\
\vdots \\
\hat{p} - b - \hat{s} \\
-\hat{z} + \underline{z} \\
\hat{z} - \bar{z} \\
-\hat{s}
\end{bmatrix},$$

i.e., $\tilde{g}(\hat{x}) \in \mathbb{R}^m$ collects all constraints of (3.5). Since there are no equality constraints, the Mangasarian-Fromovitz constraint qualification (MFCQ) from Prop. 2.3 requires that there exists a direction

$$d = \left[{d^{(z)}}^T, \quad {d^{(s)}}^T, \quad {d^{(w)}}^T, \quad {d^{(y)}}^T, \quad {D^{(V)}_{(:)}}^T, \quad {D^{(T)}_{(:)}}^T, \quad {d^{(p)}}^T \right]^T,$$

with $d^{(z)} \in \mathbb{R}^{n_z}$, $d^{(s)} \in \mathbb{R}^{n_s}$, $d^{(w)} \in \mathbb{R}^{n_w}$, $D^{(V)} \in \mathbb{R}^{n_k \times n_s}$, $D^{(T)} \in \mathbb{R}^{n_k \times n_s}$, and $d^{(p)} \in \mathbb{R}^{n_s}$, such that $J_{g_\mathcal{A}}(\hat{x})\, d < 0$ for a given $\hat{x}$, where $\mathcal{A} = \{a \in \{1, ..., m\} \mid \tilde{g}_a(\hat{x}) = 0\}$ denotes the active constraint index set and $J_{\tilde{g}_\mathcal{A}}(\hat{x})$ is the Jacobian matrix of all constraints active at $x = \hat{x}$.

In the worst case, all constraints are active. That said, use the fact that – between $z_i \leq \bar{z}_i$ and $z_i \geq \underline{z}_i$ for $1 \leq i \leq n_z$ – only one of the two constraints can be active at any given time since $\underline{z}_i < \bar{z}_i$ by assumption. To make the presentation more compact, we write these two constraints as $\left| z - \frac{1}{2}(\underline{z} + \bar{z}) \right| \leq \frac{1}{2}(\bar{z} - \underline{z})$. The non-differentiability at $\tilde{z} = \frac{1}{2}(\underline{z} + \bar{z})$ is of no consequence as both constraints are inactive at $z = \tilde{z}$, and its derivative is hence given by

$$\frac{d}{dz}\left(|z - \tilde{z}| - \frac{1}{2}(\bar{z} - \underline{z}) \right) = \operatorname{diag}(\operatorname{sign}(z - \tilde{z})).$$

Then we have

$$\begin{aligned}
\max\left(J_{\tilde{g}_\mathcal{A}}(\hat{x})\, d\right) &= \max\left(\begin{bmatrix}
\begin{bmatrix} +J_h(\hat{z}) \\ -J_h(\hat{z}) \end{bmatrix} d^{(z)} - \begin{bmatrix} d^{(w)} \\ d^{(w)} \end{bmatrix} \\
\begin{bmatrix} \bar{R}J_\rho(\hat{z}) \\ \underline{R}J_\rho(\hat{z}) \end{bmatrix} d^{(z)} - \begin{bmatrix} d^{(y)} \\ d^{(y)} \end{bmatrix} \\
\max\limits_{\substack{1 \leq k \leq n_k \\ 1 \leq m \leq n_s}} \left\{ \max\left(F^{(k)}_{(m)} \left(A J_h(\hat{z})\, d^{(z)} + M d^{(w)} \right) \right) - D^{(V)}_{km} \right\} \\
\max\limits_{\substack{1 \leq k \leq n_k \\ 1 \leq m \leq n_s}} \left\{ \max\left(G^{(k)}_{(m)} J_\rho(\hat{z})\, d^{(z)} \right) - D^{(T)}_{km} \right\} \\
\max\limits_{1 \leq m \leq n_s} \left\{ \max\left(D^{(V)}_{(:,m)} - D^{(T)}_{(:,m)} \right) - d^{(p)}_m \right\} \\
d^{(p)} - d^{(s)} \\
\operatorname{diag}(\operatorname{sign}(\hat{z} - \tilde{z}))\, d^{(z)} \\
-d^{(s)}
\end{bmatrix} \right) \\
&= \max\left(\begin{bmatrix}
\bar{h}\left(d^{(z)}\right) - d^{(w)} \\
\bar{r}\left(d^{(z)}\right) - d^{(y)} \\
\bar{F}_{(:)}\left(d^{(r)}\right) - D^{(V)}_{(:)} \\
\bar{G}_{(:)}\left(d^{(z)}\right) - D^{(T)}_{(:)} \\
\bar{d}\left(D^{(V)}, D^{(T)}\right) - d^{(p)} \\
d^{(p)} - d^{(s)} \\
\operatorname{diag}(\operatorname{sign}(\hat{z} - \tilde{z}))\, d^{(z)} \\
-d^{(s)}
\end{bmatrix} \right),
\end{aligned} \tag{A.25}$$

where

$$\bar{h}\left(d^{(z)}\right) = \max\left(+J_h\left(\hat{z}\right)d^{(z)}, -J_h\left(\hat{z}\right)d^{(z)}\right), \tag{A.26a}$$

$$\bar{r}\left(d^{(z)}\right) = \max\left(\underline{R}J_\rho\left(\hat{z}\right)d^{(z)}, \bar{R}J_\rho\left(\hat{z}\right)d^{(z)}\right), \tag{A.26b}$$

$$\bar{F}_{km}\left(d^{(z)}, d^{(w)}\right) = \max\left(F^{(k)}_{(m)}\left(AJ_h\left(\hat{z}\right)d^{(z)} + Md^{(w)}\right)\right), \tag{A.26c}$$

$$\bar{G}_{km}\left(d^{(z)}\right) = \max\left(G^{(k)}_{(m)}J_\rho\left(\hat{z}\right)d^{(z)}\right), \tag{A.26d}$$

$$\bar{d}_m\left(D^{(V)}, D^{(T)}\right) = \max\left(D^{(V)}_{(:,m)} - D^{(T)}_{(:,m)}\right). \tag{A.26e}$$

Since we assume that all functions are sufficiently smooth, their gradients are at least continuous, and since $\hat{z}$ is bounded (see (3.5h)), the expressions in (A.26) are bounded for bounded d since they are also continuous. Choosing

$$\begin{aligned}
\hat{d}^{(z)} &= -\epsilon\,\mathrm{sign}\left(\hat{z} - \tilde{z}\right),\\
\hat{d}^{(w)} &= \bar{h}\left(\hat{d}^{(z)}\right) + \epsilon,\\
\hat{d}^{(y)} &= \bar{r}\left(\hat{d}^{(z)}\right),\\
\hat{D}^{(V)}_{(:)} &= \bar{F}_{(:)}\left(\hat{d}^{(z)}, \hat{d}^{(w)}\right) + \epsilon,\\
\hat{D}^{(T)}_{(:)} &= \bar{G}_{(:)}\left(\hat{d}^{(z)}\right) + \epsilon,\\
\hat{d}^{(p)} &= \bar{d}\left(\hat{D}^{(V)}, \hat{D}^{(T)}\right) + \epsilon,\\
\hat{d}^{(s)} &= \max\left(0, \hat{d}^{(p)}\right) + \epsilon,
\end{aligned}$$

with $\epsilon \in \mathbb{R}_+$ yields

$$\max\left(\begin{bmatrix}
\bar{h}\left(\hat{d}^{(z)}\right) - \hat{d}^{(w)}\\
\bar{r}\left(\hat{d}^{(z)}\right) - \hat{d}^{(y)}\\
\bar{F}_{(:)}\left(\hat{d}^{(z)}, \hat{d}^{(w)}\right) - \hat{D}^{(V)}_{(:)}\\
\bar{G}_{(:)}\left(\hat{d}^{(z)}\right) - \hat{D}^{(T)}_{(:)}\\
\bar{d}\left(\hat{D}^{(V)}, \hat{D}^{(T)}\right) - \hat{d}^{(p)}\\
\hat{d}^{(p)} - \hat{d}^{(s)}\\
\mathrm{diag}\left(\mathrm{sign}\left(\hat{z} - \tilde{z}\right)\right)\hat{d}^{(z)}\\
-d^{(s)}
\end{bmatrix}\right) < 0,$$

and thus there exists a $\hat{d}$ for any $\hat{x}$ such that $J_{\tilde{g}_\mathcal{A}}\left(\hat{x}\right)\hat{d} < 0$, concluding the proof. □

A.2 Complexity Analysis

Proposition A.1 (Parameterized Reachable Set)**.** *Let $f : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \mapsto \mathbb{R}^{n_x}$ denote the system dynamics and let $n = \min\left(n_x, n_u\right)$. Further, we assume that the number of dependent factors and dependent generators of the given initial set grows at most with*

$O(na)$ and $O(n)$, and the number of elementary operations to evaluate f_i for $1 \leq i \leq n_x$ and all its necessary derivatives grows at most with $O(n)$. The complexity of computing the parameterized reachable set using the conservative polynomialization approach (see Sec. 2.7.3) can then be bounded from above by $O\left(n^5 + n^2a^2 + n^3a \log n\right)$.

Proof. We simply follow the proof in [59, Sec. 4.1.4], where we make the necessary changes according to the made assumptions.

We discuss the complexity of all listed operations in [59, Eq. (4.18)] due to the different assumptions separately:

- Linear map: $O\left(n^3\right)$ [59, Prop. 3.1.18]
- Minkowski sum: $O(na)$ [59, Prop. 3.1.19]
- Exact sum: The merge operation has complexity $O\left((na)^2\right)$ [59, Prop. 3.1.5] and the compact operation has complexity $O\left(n^2a \log n\right)$ [59, Prop. 3.1.7], so that the overall complexity can be bounded from above by $O\left(n^2a^2\right)$.
- Cartesian product: $O(1)$ [59, Prop. 3.1.22]
- Linear combination: $O\left(n^2\right) + O\left(n^2\right) = O\left(n^2\right)$ [59, Prop. 3.1.26]
- Quadratic map [59, Prop. 3.1.26]

$$\begin{aligned} &O\left((na)^2\right) + O\left(n^2nn\right) + O(nnnn) + O(nn(n + na\log(nn))) \\ =&O\left(n^2a^2 + n^4 + n^3a\log n\right) \end{aligned}$$

- Reduce: $O\left(n^2\right)$ [59, Tab. 3.5]
- Restructure: Not used to preserve dependencies
- Zonotope enclosure: $O((na)n + nn) = O\left(an^2\right)$ [59, Prop. 3.1.14]
- Interval enclosure: $O\left(n^3\right)$ [59, Tab. 3.3]

Thus, adding all the listed complexities above as in [59, Eq. (4.18)] yields

$$O\left(a^2n^2 + n^4 + an^3 \log n\right).$$

Since we assume a dimension of n and that the number of generators and number of elementary operations to evaluate f and its derivatives grows at most with $O(n)$, the upper bound on the complexity of computing the Lagrange remainder follows directly from [59, Sec. 4.1.4] as $O\left(n^5\right)$. The overall complexity thus follows, concluding the proof. □

Proposition A.2 (Polynomial Zonotope Composition)**.** *Given are the polynomial zonotopes* $\mathcal{A} = \langle \cdot, G, G_{\text{rest}}, E \rangle_{PZ} = \{a(\alpha)\}_\alpha$*, where* $G \in \mathbb{R}^{n \times m}$*,* $G_{\text{rest}} \in \mathbb{R}^{n \times l}$*, and* $E \in \mathbb{N}_{\geq 0}^{n_a \times m}$*, and* $\mathcal{B} = \left\langle \cdot, G^{(1)}, 0, E^{(1)} \right\rangle_{PZ} = \{b(\beta)\}_\beta$*, where* $G^{(1)} \in \mathbb{R}^{n_a \times q}$ *and* $E^{(1)} \in \mathbb{N}^{n_b \times q}$*. Further, let* $d = \max_{1 \leq k \leq m} \max\left(E_{(:,k)}\right)$ *be the maximum monomial exponent. An upper bound on the complexity of computing the polynomial zonotope* $\{a(b(\beta))\}_\beta = \left\langle \cdot, \hat{G}, G_{\text{rest}}, \hat{E} \right\rangle_{PZ}$ *is given by*

$$O\left(\left(d\left(n_a + n_b\right) + mn\right) q^d\right).$$

Proof. We first derive the complexity of computing $\left\{b_i(\beta)^d\right\}_\beta$ for $i \in \{1, ..., n_a\}$ and maximum exponent d of $a(\alpha)$. Then, we extend this complexity result to the computation of $\{a(b(\beta))\}_\beta$. Note that the centers of $\mathcal{A}$ and $\mathcal{B}$ are ignored as they can be included in the generator and exponent matrices.

Let $\left\{b_i(\beta)^k\right\}_\beta = \left\langle \cdot, G_{(i,:)}^{(k)}, 0, E^{(k)} \right\rangle_{PZ}$ for $k \in \{0, ..., d\}$ and any $i \in \{1, ..., n_a\}$ with $G_{(i,:)}^{(k)} \in \mathbb{R}^{1 \times q^k}$ and $E^{(k)} \in \mathbb{N}^{n_b \times q^k}$. The computation of $E^{(k)}$ has complexity $O\left(n_b q^k\right)$ since $E^{(k)} \in \mathbb{N}^{n_b \times q^k}$ (see [61, Prop. 12, Proof]). Similarly, the computation of $G_{(i,:)}^{(k)} \in \mathbb{R}^{1 \times q^k}$ for $i \in \{1, ..., n_a\}$ has complexity $O\left(q^k\right)$. Because $E^{(k)}$ is the same for all i, the complexity of computing $\left\{b_i(\beta)^k\right\}_\beta$ for all $i \in \{1, ..., n_a\}$ and $k \in \{0, ..., d\}$ is $O\left(d\left(n_a + n_b\right) q^d\right)$.

Next, we form the m monomials of $\{a(b(\beta))\}_\beta$. Each monomial consists of at most $n_a - 1$ multiplications of the already computed polynomial zonotopes $\left\{b_i(\beta)^{k_i}\right\}_\beta$ for $i \in \{1, ..., n_a\}$ and corresponding exponent $k_i \in \mathbb{N}$ with $\sum_{k=1}^{n_a} k_i \leq d$. The exponent matrix of the product of n_a polynomial zonotopes has at most dimension $n_b \times q^d$ due to $\sum_{k=1}^{n_a} k_i \leq d$ and thus the complexity of computing any monomial of $\{a(b(\beta))\}_\beta$ is bounded from above by $O\left(n_b q^d\right)$. Finally, we multiply the k-th monomial, now represented as a polynomial zonotope with a generator matrix of dimension at most $1 \times q^d$, with the corresponding column $G_{(:,k)} \in \mathbb{R}^n$ for all $k \in \{1, ..., m\}$ with complexity $O\left(m\left(nq^d\right)\right)$ and bring these m monomials into a single polynomial zonotope $\{a(b(\beta))\}_\beta = \left\langle \cdot, \hat{G}, G_{\text{rest}}, \hat{E} \right\rangle_{PZ}$ by concatenation with complexity $O(m(1))$. Collecting all complexities then yields the desired result. □

A.3 Derivatives of the LQR Feedback Matrix

In order to optimize over the gain matrix $K(z)$, we derive its Jacobian and Hessian matrices in App. A.3.1 and App. A.3.2, respectively.

To that end, let $(A(P), B(P))$ denote a controllable LTI system with $A(P) \in \mathbb{R}^{n_x \times n_x}$, $B(P) \in \mathbb{R}^{n_x \times n_u}$, $P \in [-1, 1]^{m_c n_u \times a}$, $Q \in \mathbb{S}_{++}^{n_x \times n_x}$, and $R \in \mathbb{S}_{++}^{n_u \times n_u}$. Further, let $z =$

$\begin{bmatrix} P_{(:)}^T, & Q_{(:)}^T, & R_{(:)}^T \end{bmatrix}^T \in \mathbb{R}^{n_z}$, and we define $n = \max(n_x, n_u)$. For convenience, we restate that the optimal LQR feedback gain matrix is given as (we omit arguments from now on)

$$K = -R^{-1}B^T X, \tag{A.27}$$

with

$$F = A^T X + XA - XBR^{-1}B^T X + Q = 0. \tag{A.28}$$

For later convenience, we define here the closed-loop system matrix as $A_{\mathrm{cl}} = A + BK$. Further, we introduce the commutation matrix $M_D \in \{0,1\}^{nm \times nm}$ for a matrix $D \in \mathbb{R}^{n \times m}$, which is implicitly defined by

$$M_D D_{(:)} = \left(D^T\right)_{(:)}.$$

According to Sec. 2.2, we can replace differentials with their corresponding differential tensors, and thus we derive the below expressions directly for these tensors. Due to the definition of the sums and products of these differential tensors, one can replace $d\square$ at any point with $\mathrm{d}\square$ to recover the "differential" version. Let the differential tensors $dA \in \mathbb{R}^{1 \times n_z \times n_x \times n_x}$, $dB \in \mathbb{R}^{1 \times n_z \times n_x \times n_u}$, $d^2A \in \mathbb{R}^{n_z \times n_z \times n_x \times n_x}$, $d^2B \in \mathbb{R}^{n_z \times n_z \times n_x \times n_u}$, $dQ \in \mathbb{R}^{1 \times n_z \times n_x \times n_x}$, and $dR \in \mathbb{R}^{1 \times n_z \times n_u \times n_u}$, all with respect to z, be given. Lastly, let $e = c_e n$ denote the number of elementary operations required to evaluate any element of A, B, or any of its differential tensors, where $c_e \in \mathbb{R}_+$. Thus, for a given differential tensor $M \in \mathbb{R}^{n_z \times n_z \times n \times n}$, its evaluation complexity is bounded from above by $O\left(e n_z^2 n^2\right) = O\left(n_z^2 n^3\right)$.

A.3.1 Jacobian of Feedback Matrix

Proposition A.3 (Jacobian Matrix of LQR Feedback Matrix)**.** *Let $(A(P), B(P))$ be a controllable LTI system with $A(P) \in \mathbb{R}^{n_x \times n_x}$ and $B(P) \in \mathbb{R}^{n_x \times n_u}$ for $P \in [-1,1]^{m_c n_u \times a}$, where each element of A and B is at least once differentiable with respect to P. Further, denote with $Q \in \mathbb{S}_{++}^{n_x \times n_x}$ and R given LQR weighting matrices. The vectorized Jacobian matrix $\frac{dK_{(:)}}{dz}$ is given by*

$$\frac{dK_{(:)}}{dz} = -\left(K^T \boxtimes R^{-1}\right) dR_{(:)} - \left(X \boxtimes R^{-1}\right) M_B dB_{(:)} - \left(I_n \boxtimes R^{-1}B^T\right) dX_{(:)}, \tag{A.29}$$

where

$$\begin{aligned} dX_{(:)} = & -\left(A_{\mathrm{cl}}^T \boxplus A_{\mathrm{cl}}^T\right)^{-1} \Big(\left((X \boxtimes I_n) M_A + (I_n \boxtimes X)\right) dA_{(:)} + \left(K^T \boxtimes K^T\right) dR_{(:)} \\ & + \left(\left(X \boxtimes K^T\right) M_B + \left(K^T \boxtimes X\right)\right) dB_{(:)} + dQ_{(:)} \Big), \end{aligned} \tag{A.30}$$

with an upper bound on its computational complexity of $O\left(n_x^{2\omega} + n^4 n_z\right)$, where $\omega \in \mathbb{R}_+$ is the exponent for the current complexity of matrix multiplication.

Proof. Applying the differential operator to (A.28) yields

$$\begin{aligned}
0 &= dA^T X + A^T dX + dXA + XdA + dXBK + XdBK + K^T dRK + K^T dB^T X \\
&\quad + K^T B^T dX + dQ \\
0 &= dX\,(A + BK) + (A + BK)^T dX + dA^T X + XdA + K^T dRK + XdBK \\
&\quad + K^T dB^T X + dQ \\
0 &= \left(\left(A_{\mathrm{cl}}^T \boxtimes I_n\right) + \left(I_n \boxtimes A_{\mathrm{cl}}^T\right)\right) dX_{(:)} + \left(\left(X \boxtimes I_n\right) M_A + \left(I_n \boxtimes X\right)\right) dA_{(:)} \\
&\quad + \left(K^T \boxtimes K^T\right) dR_{(:)} + \left(\left(X \boxtimes K^T\right) M_B + \left(K^T \boxtimes X\right)\right) dB_{(:)} + dQ_{(:)},
\end{aligned}$$

where we used (A.27), and the vectorization follows from (2.1). Solving for $dX_{(:)}$ then yields (A.30). With

$$\begin{aligned}
dK &= d\left(-R^{-1}B^T X\right) \\
&= R^{-1}dRR^{-1}B^T X - R^{-1}dB^T X - R^{-1}B^T dX,
\end{aligned}$$

we finally arrive at (A.29) after vectorization, which concludes the first part of the proof.

Looking at (A.30), the complexity of computing $\left(A_{\mathrm{cl}}^T \boxplus A_{\mathrm{cl}}^T\right)^{-1}$ is at most $O\left(\left(n_x^2\right)^\omega\right)$ which follows from $\left(A_{\mathrm{cl}}^T \boxplus A_{\mathrm{cl}}^T\right) \in \mathbb{R}^{n_x^2 \times n_x^2}$ and the computational complexity of the matrix inverse being equal to that of the matrix multiplication [17, Prop. 16.6]. All remaining operations in (A.29) and (A.30) can then be computed by a fixed number of "matrix-vector" products of a $(n^2 \times n^2)$-dimensional matrix and a $(1 \times n_z \times n^2 \times 1)$-dimensional differential tensor, which has complexity $O\left(n^4 n_z\right)$ since the matrix-vector product of a n^2-by-n^2 matrix and n^2-by-1 vector has complexity $O\left(\left(n^2\right)^2\right)$, but we need to multiple a 1-by-n_z vector with a scalar for each "inner" multiplication. Since forming the Kronecker product of two $n \times n$-dimensional matrices has complexity $O\left(n^4\right)$, the overall complexity of computing the Jacobian of the vectorized gain matrix with respect to z is at most $O\left(n^3 n_z + n^4 + n_x^{2\omega} + n^4 n_z\right) = O\left(n_x^{2\omega} + n^4 n_z\right)$ when including the evaluation complexity $O\left(n^3 n_z\right)$ of the precomputed differential tensors, which concludes the proof. □

A.3.2 Hessian of Feedback Matrix

Proposition A.4 (Hessian Matrix of LQR Feedback Matrix)**.** *Let $(A(P), B(P))$ be a controllable LTI system with $A(P) \in \mathbb{R}^{n_x \times n_x}$ and $B(P) \in \mathbb{R}^{n_x \times n_u}$ for $P \in [-1,1]^{m_c n_u \times a}$, where each element of A and B is at least twice differentiable with respect to P. Further, denote with $Q \in \mathbb{S}_{++}^{n_x \times n_x}$ and R given LQR weighting matrices. The Hessian matrix $H_{K_{ij}}(z)$ of the i-th row and j-th column of $K \in \mathbb{R}^{n_u \times n_x}$ with $1 \leq i \leq n_u$ and $1 \leq j \leq n_x$ is given by*

$$H_{K_{ij}}(z) = \frac{1}{2}\left(d^2 K_{ij} + \left(d^2 K_{ij}\right)^T\right), \tag{A.31}$$

with a computational complexity for all $n_x n_u$ entries of at most $O\left(n_x^{2\omega} + n^4 n_z^2\right)$, where

$$
\begin{aligned}
d^2K^T = & -d^2XBR^{-1} + 2dX\left(-dB + BR^{-1}dR\right)R^{-1} \\
& + X\left(-d^2B + 2\left(dB - BR^{-1}dR\right)R^{-1}dR\right)R^{-1}, && \text{(A.32)} \\
d^2X_{(:)} = & \left(A_{\mathrm{cl}}^T \boxplus A_{\mathrm{cl}}^T\right)^{-1} d^2C_{(:)}, && \text{(A.33)} \\
d^2C = & -\left(d^2M + d^2M^T\right), && \text{(A.34)} \\
d^2M = & + X\left(d^2A + dBR^{-1}\left(-2B^TdX - 2dRK - dB^TX\right) + d^2BK\right) \\
& + dX\left(2dA + 2dBK - 2BR^{-1}dRK - BR^{-1}B^TdX\right) \\
& - K^TdRR^{-1}dRK, && \text{(A.35)}
\end{aligned}
$$

dK and dX are given as in Prop. A.3, and $\omega \in \mathbb{R}_+$ is the exponent for the current complexity of matrix multiplication.

Proof. Applying the differential operator to (A.28) yields (see proof of Prop. A.3)

$$
\begin{aligned}
dF = & \, dA^TX + A^TdX \\
& + dXA + XdA \\
& - \left(dXBR^{-1}B^TX + XdBR^{-1}B^TX - XBR^{-1}dRR^{-1}B^TX + XBR^{-1}dB^TX\right. \\
& \left. + XBR^{-1}B^TdX\right) \\
& + dQ
\end{aligned}
$$

and applying the differential operator again results in

$$
\begin{aligned}
d^2F = &+ d^2A^TX + dA^TdX + dA^TdX + A^Td^2X \\
&+ d^2XA + dXdA + dXdA + Xd^2A \\
&- (-d^2XBK - dXdBK + dXBR^{-1}dRK + dXBR^{-1}dB^TX + dXBR^{-1}B^TdX \\
&- dXdBK - Xd^2BK + XdBR^{-1}dRK + XdBR^{-1}dB^TX + XdBR^{-1}B^TdX \\
&+ dXBR^{-1}dRK + XdBR^{-1}dRK + 2K^TdRR^{-1}dRK + K^TdRR^{-1}dB^TX \\
&+ K^TdRR^{-1}B^TdX \\
&+ dXBR^{-1}dB^TX + XdBR^{-1}dB^TX + K^TdRR^{-1}dB^TX - K^Td^2B^TX \\
&- K^TdB^TdX \\
&+ dXBR^{-1}B^TdX + XdBR^{-1}B^TdX + K^TdRR^{-1}B^TdX - K^TdB^TdX \\
&- K^TB^Td^2X) \\
d^2F = &+ d^2A^TX + 2dA^TdX + A^Td^2X \\
&+ d^2XA + 2dXdA + Xd^2A \\
&+ d^2XBK + 2dXdBK - 2dXBR^{-1}dRK - 2dXBR^{-1}dB^TX - 2dXBR^{-1}B^TdX \\
&+ Xd^2BK - 2XdBR^{-1}dRK - 2XdBR^{-1}dB^TX - 2XdBR^{-1}B^TdX \\
&- 2K^TdRR^{-1}dRK - 2K^TdRR^{-1}dB^TX - 2K^TdRR^{-1}B^TdX \\
&+ K^Td^2B^TX + 2K^TdB^TdX \\
&+ K^TB^Td^2X.
\end{aligned}
$$

Next, we define

$$
d^2F = A^Td^2X + d^2XA + d^2XBK + K^TB^Td^2X - d^2C, \tag{A.36}
$$

where

$$
\begin{aligned}
-d^2C = &+ d^2A^TX + 2dA^TdX \\
&+ 2dXdA + Xd^2A \\
&+ 2dXdBK - 2dXBR^{-1}dRK - 2dXBR^{-1}dB^TX - 2dXBR^{-1}B^TdX \\
&+ Xd^2BK - 2XdBR^{-1}dRK - 2XdBR^{-1}dB^TX - 2XdBR^{-1}B^TdX \\
&- 2K^TdRR^{-1}dRK - 2K^TdRR^{-1}dB^TX - 2K^TdRR^{-1}B^TdX \\
&+ K^Td^2B^TX + 2K^TdB^TdX \\
= &\, d^2M + d^2M^T,
\end{aligned}
$$

collects all terms not containing d^2X, and where

$$\begin{aligned}
d^2M = &+ Xd^2A + 2dXdA + 2dXdBK - 2dXBR^{-1}dRK - 2XdBR^{-1}B^TdX \\
&- dXBR^{-1}B^TdX \\
&+ Xd^2BK - 2XdBR^{-1}dRK - XdBR^{-1}dB^TX - K^TdRR^{-1}dRK \\
= &\, X\left(d^2A - 2dBR^{-1}B^TdX + d^2BK - 2dBR^{-1}dRK - dBR^{-1}dB^TX\right) \\
&+ dX\left(2dA + 2dBK - 2BR^{-1}dRK - BR^{-1}B^TdX\right) \\
&- K^TdRR^{-1}dRK \\
= &+ X\left(d^2A + dBR^{-1}\left(-2B^TdX - 2dRK - dB^TX\right) + d^2BK\right) \\
&+ dX\left(2dA + 2dBK - 2BR^{-1}dRK - BR^{-1}B^TdX\right) \\
&- K^TdRR^{-1}dRK.
\end{aligned}$$

Further, it holds that

$$A^Td^2X + d^2XA + d^2XBK + K^TB^Td^2X = A_{\mathrm{cl}}^Td^2X + d^2XA_{\mathrm{cl}}.$$

After vectorization of $d^2F = 0$ from (A.36), it thus follows that

$$\left(A_{\mathrm{cl}}^T \boxplus A_{\mathrm{cl}}^T\right) d^2X_{(:)} = d^2C_{(:)},$$

and therefore

$$d^2X_{(:)} = \left(A_{\mathrm{cl}}^T \boxplus A_{\mathrm{cl}}^T\right)^{-1} d^2C_{(:)}.$$

Using the definition of K from (A.27) and applying the differential operator twice yields

$$\begin{aligned}
d^2K^T = &\, d\left(-dXBR^{-1} - XdBR^{-1} + XBR^{-1}dRR^{-1}\right) \\
= &- d^2XBR^{-1} - dXdBR^{-1} + dXBR^{-1}dRR^{-1} \\
&- dXdBR^{-1} - Xd^2BR^{-1} + XdBR^{-1}dRR^{-1} \\
&+ dXBR^{-1}dRR^{-1} + XdBR^{-1}dRR^{-1} - 2XBR^{-1}dRR^{-1}dRR^{-1} \\
= &- d^2XBR^{-1} - 2dXdBR^{-1} + 2dXBR^{-1}dRR^{-1} \\
&- Xd^2BR^{-1} + 2XdBR^{-1}dRR^{-1} \\
&- 2XBR^{-1}dRR^{-1}dRR^{-1} \\
= &- d^2XBR^{-1} + 2dX\left(-dB + BR^{-1}dR\right)R^{-1} \\
&+ X\left(-d^2B + 2\left(dB - BR^{-1}dR\right)R^{-1}dR\right)R^{-1}.
\end{aligned}$$

Evaluating these products and sums of tensors, we obtain $d^2K^T \in \mathbb{R}^{n_z \times n_z \times n_x \times n_u}$, which yields d^2K by transposition. Since

$$\mathrm{d}^2K_{ij} = \mathrm{d}z^T d^2K_{ij}\,\mathrm{d}z,\ 1 \le i \le n_u,\ 1 \le j \le n_x,$$

the Hessian tensor $H_K(z) \in \mathbb{R}^{n_z \times n_z \times n_u \times n_x}$ that contains the Hessian matrices of all $n_u n_x$ entries of $K \in \mathbb{R}^{n_u \times n_x}$ is given by

$$[H_K(z)]_{(:,:,i,j)} = H_{K_{ij}}(z) = \frac{1}{2}\left(d^2 K_{ij} + d^2 K_{ij}^T\right), \tag{A.37}$$

for $1 \leq i \leq n_u$ and $1 \leq j \leq n_x$ due to Th. 2.5.

The computation of $\left(A_{\mathrm{cl}}^T \boxplus A_{\mathrm{cl}}^T\right)^{-1}$ in (A.33) has complexity $O\left(n_x^{2\omega}\right)$ (see proof of Prop. A.3). Further, all remaining operations in (A.32) to (A.35) can be computed as a product of an $(n_z \times n_z \times n \times n)$-dimensional differential tensor and a $(n \times n)$-dimensional matrix, the tensor product between two $(1 \times n_z \times n \times n)$-dimensional differential tensors, and a "matrix-vector" tensor product of an $(n_x^2 \times n_x^2)$-dimensional matrix with a $(n_z \times n_z \times n_x^2 \times 1)$-dimensional differential tensor: The complexity of the first operation is $O\left(n^\omega n_z^2\right)$ since normal matrix multiplication has complexity $O\left(n^\omega\right)$ but here each scalar multiplication is replaced by the multiplication of an $(n_z \times n_z)$-dimensional matrix with a scalar. The complexity of the second operation is at most $O\left(n^\omega n_z^2\right)$ since normal matrix multiplication has complexity $O\left(n^\omega\right)$ but here each scalar multiplication is replaced by the outer product of two n_z-dimensional vectors. Finally, the complexity of the third operation is bounded from above by $O\left(\left(n^2\right)^2 n_z^2\right)$ since the matrix-vector product of a $(n^2 \times n^2)$-dimensional matrix and a n^2-dimensional matrix is $O\left(\left(n^2\right)^2\right)$, but here each element is a $(n_z \times n_z)$-dimensional matrix and so each element multiplication of the "outer" matrix-vector product is the multiplication of a scalar with a $(n_z \times n_z)$-dimensional matrix. Thus, the combined complexity is then $O\left(n^3 n_z^2 + n_x^{2\omega} + n^\omega n_z^2 + n^\omega n_z^2 + n^4 n_z^2\right) = O\left(n_x^{2\omega} + n^4 n_z^2\right)$ (computational complexity of dK and dX from Prop. A.3 can be thus be ignored) when including the evaluation complexity of all differential tensors of at most $O\left(n^3 n_z^2\right)$, which concludes the proof. □

Bibliography

[1] A. Alessio and A. Bemporad. "A survey on explicit model predictive control". In: *Nonlinear Model Predictive Control*. Springer, 2009, pp. 345–369.

[2] M. Althoff. "An introduction to CORA 2015". In: *Workshop on Applied Verification for Continuous and Hybrid Systems*. EasyChair, 2015, pp. 120–151.

[3] M. Althoff. "On computing the minkowski difference of zonotopes". In: arXiv, 2015.

[4] M. Althoff. "Reachability analysis and its application to the safety assessment of autonomous cars". PhD thesis. Technische Universität München, 2010.

[5] M. Althoff. "Reachability analysis of nonlinear systems using conservative polynomialization and non-convex sets". In: *Conference on Hybrid Systems: Computation and Control*. ACM, 2013, pp. 173–182.

[6] M. Althoff, G. Frehse, and A. Girard. "Set propagation techniques for reachability analysis". In: *Annual Review of Control, Robotics, and Autonomous Systems* 4.1 (2021), pp. 369–395.

[7] M. Althoff and B.H. Krogh. "Avoiding geometric intersection operations in reachability analysis of hybrid systems". In: *Conference on Hybrid Systems: Computation and Control*. ACM, 2012.

[8] M. Althoff, O. Stursberg, and M. Buss. "Computing reachable sets of hybrid systems using a combination of zonotopes and polytopes". In: *Nonlinear Analysis: Hybrid Systems* 4.2 (2010), pp. 233–249.

[9] M. Althoff, O. Stursberg, and M. Buss. "Reachability analysis of nonlinear systems with uncertain parameters using conservative linearization". In: *Conference on Decision and Control*. IEEE, 2008, pp. 4042–4048.

[10] J.A.E. Andersson et al. "CasADi: a software framework for nonlinear optimization and optimal control". In: *Mathematical Programming Computation* 11.1 (2018), pp. 1–36.

[11] H.E. Bell. "Gershgorin's theorem and the zeros of polynomials". In: *The American Mathematical Monthly* 72.3 (1965).

[12] C. Belta and L. C. G. J. M. Habets. "Controlling a class of nonlinear systems on rectangles". In: *Transactions on Automatic Control* 51.11 (2006), pp. 1749–1759.

[13] 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.

[14] S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge, United Kingdom: Cambridge university press, 2004.

[15] S. Boyd et al. *Linear matrix inequalities in system and control theory*. Philadelphia, USA: SIAM, 1994.

[16] J. van den Brand. "A deterministic linear program solver in current matrix multiplication time". In: *Symposium on Discrete Algorithms*. ACM-SIAM, 2020, pp. 259–278.

[17] P. Bürgisser, M. Clausen, and M.A. Shokrollahi. *Algebraic complexity theory*. Berlin, Germany: Springer, 1997.

[18] C. Cartis, N.I.M. Gould, and P.L. Toint. "On the evaluation complexity of constrained nonlinear least-squares and general constrained nonlinear optimization using second-order methods". In: *Journal on Numerical Analysis* 53.2 (2015), pp. 836–851.

[19] M. Černỳ. "Goffin's algorithm for zonotopes". In: *Kybernetika* 48.5 (2012), pp. 890–906.

[20] N. Chan and S. Mitra. "Verifying safety of an autonomous spacecraft rendezvous mission". In: *Workshop on Applied Verification of Continuous and Hybrid Systems*. EasyChair, 2017, pp. 20–32.

[21] X. Chen and S. Sankaranarayanan. "Decomposed reachability analysis for nonlinear systems". In: *Real-Time Systems Symposium*. IEEE, 2016, pp. 13–24.

[22] M.B. Cohen, Y.T. Lee, and Z. Song. "Solving linear programs in the current matrix multiplication time". In: *Journal of the ACM* 68.1 (2021), pp. 1–39.

[23] P. Collins et al. "Computing the evolution of hybrid systems using rigorous function calculus". In: *IFAC Proceedings Volumes* 45.9 (2012), pp. 284–290.

[24] T.H. Cormen et al. *Introduction to algorithms*. Cambridge, USA: MIT press, 2022.

[25] S. Dasgupta, C.H. Papadimitriou, and U. Vazirani. *Algorithms*. New York, USA: McGraw-Hill Higher Education, 2006.

[26] M. Diehl et al. "Fast direct multiple shooting algorithms for optimal robot control". In: *Lecture Notes in Control and Information Sciences*. Springer, 2006, pp. 65–93.

[27] A.L. Dontchev and R.T. Rockafellar. *Implicit functions and solution mappings*. New York, USA: Springer, 2014.

[28] C. Durieu, É. Walter, and B. Polyak. "Multi-input multi-output ellipsoidal state bounding". In: *Journal of Optimization Theory and Applications* 111.2 (2001), pp. 273–303.

[29] C.H. Edwards. *Advanced calculus of several variables*. Academic Press, New York, USA, 2012.

[30] H. J. Ferreau, H. G. Bock, and M. Diehl. "An online active set strategy to overcome the limitations of explicit MPC". In: *Journal of Robust and Nonlinear Control* 18.8 (2008), pp. 816–830.

[31] J.-A. Ferrez K. Fukuda and T.M. Liebling. "Solving the fixed rank convex quadratic maximization in binary variables by a parallel zonotope construction algorithm". In: *European Journal of Operational Research* 166.1 (2005), pp. 35–50.

[32] K. Fukuda. "From the zonotope construction to the Minkowski addition of convex polytopes". In: *Journal of Symbolic Computation* 38.4 (2004), pp. 1261–1272.

[33] A. A. Gaganov. "Computation complexity of the range of a polynomial in several variables". In: *Cybernetics* 21.4 (1985), pp. 418–421.

[34] V. Gaßmann and M. Althoff. "Implementation of ellipsoidal operations in CORA 2022". In: *Workshop on Applied Verification of Continuous and Hybrid Systems*. Vol. 90. EasyChair, 2022, pp. 1–17.

[35] V. Gaßmann and M. Althoff. "Polynomial controller synthesis of nonlinear systems with continuous state feedback using trust regions". In: *Open Journal of Control Systems* 2 (2023), pp. 310–324.

[36] V. Gaßmann and M. Althoff. "Scalable zonotope-ellipsoid conversions using the Euclidean zonotope norm". In: *American Control Conference*. IEEE, 2020, pp. 4715–4721.

[37] V. Gaßmann and M. Althoff. "Verified polynomial controller synthesis for disturbed nonlinear systems". In: *IFAC-PapersOnLine* 54.5 (2021), pp. 85–90.

[38] L. Geretti et al. "ARCH-COMP22 category report: Continuous and hybrid systems with nonlinear dynamics". In: *Workshop on Applied Verification of Continuous and Hybrid Systems*. Vol. 90. EasyChair, 2022, pp. 86–112.

[39] A. Girard. "Controller synthesis for safety and reachability via approximate bisimulation". In: *Automatica* 48.5 (2012), pp. 947–953.

[40] A. Girard. "Reachability of uncertain linear systems using zonotopes". In: *Hybrid Systems: Computation and Control*. Springer, 2005, pp. 291–305.

[41] A. Girard and C. Le Guernic. "Efficient reachability analysis for linear systems using support functions". In: *IFAC Proceedings Volumes* 41.2 (2008), pp. 8966–8971.

[42] A. Girard, G. Pola, and P. Tabuada. "Approximately bisimilar symbolic models for incrementally stable switched systems". In: *Transactions on Automatic Control* 55.1 (2010), pp. 116–126.

[43] F. De Groote et al. "Evaluation of direct collocation optimal control problem formulations for solving the muscle redundancy problem". In: *Annals of Biomedical Engineering* 44.10 (2016), pp. 2922–2936.

[44] F. Gruber and M. Althoff. "Scalable robust model predictive control for linear sampled-data systems". In: *Conference on Decision and Control*. IEEE, 2019, pp. 438–444.

[45] C. Le Guernic. “Reachability analysis of hybrid systems with linear continuous dynamics”. PhD thesis. Université Joseph-Fourier - Grenoble I, 2010.

[46] L. C. G. J. M. Habets, P. J. Collins, and J. H. Van Schuppen. “Reachability and control synthesis for piecewise-affine hybrid systems on simplices”. In: *Transactions on Automatic Control* 51.6 (2006), pp. 938–948.

[47] A. Halder. “On the parameterized computation of minimum volume outer ellipsoid of Minkowski sum of ellipsoids”. In: *Conference on Decision and Control*. IEEE, 2018.

[48] G.D. Halikias et al. “New bounds on the unconstrained quadratic integer programming problem”. In: *Journal of Global Optimization* 39.4 (2007), pp. 543–554.

[49] B.C. Hall. *Lie groups, lie algebras, and representations*. Cham, Switzerland: Springer, 2015.

[50] S.-P. Han and O.L. Mangasarian. “Exact penalty functions in nonlinear programming”. In: *Mathematical Programming* 17.1 (1979), pp. 251–269.

[51] C. Helmberg. *Semidefinite programming for combinatorial optimization*. Berlin, Germany: ZIB, 2000.

[52] D. Hess, M. Althoff, and T. Sattel. “Formal verification of maneuver automata for parameterized motion primitives”. In: *Conference on Intelligent Robots and Systems*. IEEE, 2014, pp. 1474–1481.

[53] J.D. Hoffman and S. Frankel. *Numerical methods for engineers and scientists*. New York, USA: McGraw-Hill, 2018.

[54] R.A. Horn and C.R. Johnson. *Matrix analysis*. Cambridge, United Kingdom: Cambridge University Press, 1985.

[55] B. Houska, H.J. Ferreau, and M. Diehl. “ACADO Toolkit – an open-source framework for automatic control and dynamic optimization”. In: *Optimal Control Applications and Methods* 32.3 (2011), pp. 298–312.

[56] R.E. Kalman. “Contributions to the theory of optimal control”. In: *Bol. soc. mat. mexicana* 5.2 (1960), pp. 102–119.

[57] D.E. Kirk. *Optimal control theory: An introduction*. Mineola, NY: Dover Publications, 2004.

[58] M. Kloetzer and C. Belta. “A fully automated framework for control of linear systems from temporal logic specifications”. In: *Transactions on Automatic Control* 53.1 (2008), pp. 287–297.

[59] N. Kochdumper. “Extensions of polynomial zonotopes and their application to verification of cyber-physical systems”. PhD thesis. Technische Universität München, 2022.

[60] N. Kochdumper and M. Althoff. “Representation of polytopes as polynomial zonotopes”. In: arXiv, 2019.

[61] N. Kochdumper and M. Althoff. "Sparse polynomial zonotopes: A novel set representation for reachability analysis". In: *Transactions on Automatic Control* 66.9 (2021), pp. 4043–4058.

[62] N. Kochdumper, B. Schürmann, and M. Althoff. "Utilizing dependencies to obtain subsets of reachable sets". In: *International Conference on Hybrid Systems: Computation and Control*. ACM, 2020, pp. 1–10.

[63] N. Kochdumper et al. "AROC: a toolbox for automated reachset optimal controller synthesis". In: *Conference on Hybrid Systems: Computation and Control*. ACM, 2021, pp. 1–6.

[64] A. Kopetzki, B. Schürmann, and M. Althoff. "Methods for order reduction of zonotopes". In: *Conference on Decision and Control*. IEEE, 2017, pp. 5626–5633.

[65] A.B. Kurzhanski and P. Varaiya. "On ellipsoidal techniques for reachability analysis. Part I: External approximations". In: *Optimization Methods and Software* 17.2 (2002), pp. 177–206.

[66] A.A. Kurzhanskiy and P. Varaiya. "Ellipsoidal toolbox (ET)". In: *Conference on Decision and Control*. IEEE, 2006, pp. 1498–1503.

[67] A.A. Kurzhanskiy and P. Varaiya. "Reach set computation and control synthesis for discrete-time dynamical systems with disturbances". In: *Automatica* 47.7 (2011), pp. 1414–1426.

[68] G. Lafferriere, G.J. Pappas, and S. Yovine. "Symbolic reachability computation for families of linear vector fields". In: *Journal of Symbolic Computation* 32.3 (2001), pp. 231–253.

[69] A. Laub. "A schur method for solving algebraic riccati equations". In: *Transactions on Automatic Control* 24.6 (1979), pp. 913–921.

[70] A. J. Laub. *Matrix analysis for scientists and engineers*. Vol. 91. Philadelphia, USA: SIAM, 2005.

[71] P. Leopardi. "Distributing points on the sphere". PhD thesis. University of New South Wales, 2007.

[72] D. Limon et al. "Robust tube-based MPC for tracking of constrained linear systems with additive disturbances". In: *Journal of Process Control* 20.3 (2010), pp. 248–260.

[73] J. Linhart. "Approximation of a ball by zonotopes using uniform distribution on the sphere". In: *Archiv der Mathematik* 53.1 (1989), pp. 82–86.

[74] J.R. Magnus and H. Neudecker. *Matrix differential calculus with applications in statistics and econometrics*. Hoboken, USA: John Wiley & Sons, 2019.

[75] U. Malik et al. "On the gap between the quadratic integer programming problem and its semidefinite relaxation". In: *Mathematical programming* 107.3 (2006), pp. 505–515.

[76] O.L. Mangasarian and S. Fromovitz. "The Fritz John necessary optimality conditions in the presence of equality and inequality constraints". In: *Journal of Mathematical Analysis and Applications* 17.1 (1967), pp. 37–47.

[77] S. Manzinger, M. Leibold, and M. Althoff. "Driving strategy selection for cooperative vehicles using maneuver templates". In: *Intelligent Vehicles Symposium*. IEEE, 2017, pp. 647–654.

[78] D. Q. Mayne et al. "Robust output feedback model predictive control of constrained linear systems". In: *Automatica* 42.7 (2006), pp. 1217–1222.

[79] D. Q. Mayne et al. "Robust output feedback model predictive control of constrained linear systems: Time varying case". In: *Automatica* 45.9 (2009), pp. 2082–2087.

[80] D.Q. Mayne et al. "Tube-based robust nonlinear model predictive control". In: *Journal of Robust and Nonlinear Control* 21.11 (2011), pp. 1341–1353.

[81] F. Messerer and M. Diehl. "An efficient algorithm for tube-based robust nonlinear optimal control with optimal linear feedback". In: *Conference on Decision and Control*. IEEE, 2021, pp. 6714–6721.

[82] M. Morari and J.H. Lee. "Model predictive control: Past, present and future". In: *Computers & Chemical Engineering* 23.4-5 (1999), pp. 667–682.

[83] M.E. Muller. "A note on a method for generating points uniformly on n-dimensional spheres". In: *Communications of the ACM* 2.4 (1959), pp. 19–20.

[84] K.G. Murty and S.N. Kabadi. "Some NP-complete problems in quadratic and nonlinear programming". In: *Mathematical Programming* 39.2 (1987), pp. 117–129.

[85] C.L. Navasca and A.J. Krener. "Solution of Hamilton Jacobi Bellman equations". In: *Conference on Decision and Control*. IEEE, 2000, pp. 4442–4447.

[86] C.H. Papadimitriou. "On the complexity of integer programming". In: *Journal of the ACM* 28.4 (1981), pp. 765–768.

[87] I. Papusha et al. "Automata theory meets approximate dynamic programming: Optimal control with temporal logic constraints". In: *Conference on Decision and Control*. IEEE, 2016, pp. 434–440.

[88] G. Pola, A. Girard, and P. Tabuada. "Approximately bisimilar symbolic models for nonlinear control systems". In: *Automatica* 44.10 (2008), pp. 2508–2516.

[89] G. Pola and P. Tabuada. "Symbolic models for nonlinear control systems: Alternating approximate bisimulations". In: *Journal on Control and Optimization* 48.2 (2009), pp. 719–733.

[90] G. Pola et al. "Symbolic models for nonlinear time-delay systems using approximate bisimulations". In: *Systems & Control Letters* 59.6 (2010), pp. 365–373.

[91] S. V. Rakovic et al. "Parameterized tube model predictive control". In: *Transactions on Automatic Control* 57.11 (2012), pp. 2746–2761.

[92] A.V. Rao. "A survey of numerical methods for optimal control". In: *Advances in the Astronautical Sciences* 135.1 (2009), pp. 497–528.

[93] J. B. Rawlings, D. Q. Mayne, and M. Diehl. *Model predictive control: Theory, computation, and design*. Wisconsin, WI: Nob Hill Publishing, 2009.

[94] G. Reissig. "Computing abstractions of nonlinear systems". In: *Transactions on Automatic Control* 56.11 (2011), pp. 2583–2598.

[95] G. Reissig, A. Weber, and M. Rungger. "Feedback refinement relations for the synthesis of symbolic controllers". In: *Transactions on Automatic Control* 62.4 (2017), pp. 1781–1796.

[96] P. Riedinger, J. Daafouz, and C. Iung. "About solving hybrid optimal control problems". In: *IMACS05* (2005).

[97] S. Sadraddini and R. Tedrake. "Linear encodings for polytope containment problems". In: *Conference on Decision and Control*. IEEE, 2019, pp. 4367–4372.

[98] B. Schürmann. "Using reachability analysis in controller synthesis for safety-critical systems". PhD thesis. Technische Universität München, 2022.

[99] B. Schürmann and M. Althoff. "Guaranteeing constraints of disturbed nonlinear systems using set-based optimal control in generator space". In: *IFAC-PapersOnLine* 50.1 (2017), pp. 11515–11522.

[100] B. Schürmann and M. Althoff. "Optimal control of sets of solutions to formally guarantee constraints of disturbed linear systems". In: *American Control Conference*. IEEE. 2017, pp. 2522–2529.

[101] B. Schürmann and M. Althoff. "Optimizing sets of solutions for controlling constrained nonlinear systems". In: *Transactions on Automatic Control* 66.3 (2021), pp. 981–994.

[102] B. Schürmann, N. Kochdumper, and M. Althoff. "Reachset model predictive control for disturbed nonlinear systems". In: *Conference on Decision and Control*. IEEE, 2018, pp. 3463–3470.

[103] B. Sendov. *Hausdorff approximations*. Dordrecht, The Netherlands: Kluwer Academic Publishers, 1990.

[104] O. von Stryk. "Numerical solution of optimal control problems by direct collocation". In: *Optimal Control*. Birkhäuser, 1993, pp. 129–143.

[105] O. von Stryk and R. Bulirsch. "Direct and indirect methods for trajectory optimization". In: *Annals of Operations Research* 37.1 (1992), pp. 357–373.

[106] P. Sun and R.M. Freund. "Computation of minimum-volume covering ellipsoids". In: *Operations Research* 52.5 (2004), pp. 690–706.

[107] P. Tabuada. *Verification and control of hybrid systems: A symbolic approach*. New York, USA: Springer, 2009.

[108] H.R. Tiwary. "On computing the shadows and slices of polytopes". In: arXiv, 2008.

[109] H.R. Tiwary. "On the hardness of computing intersection, union and Minkowski sum of polytopes". In: *Discrete & Computational Geometry* 40.3 (2008), pp. 469–479.

[110] L. Vandenberghe and S. Boyd. "Semidefinite programming". In: *SIAM Review* 38.1 (1996), pp. 49–95.

[111] A. Wächter and L.T. Biegler. "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming". In: *Mathematical Programming* 106.1 (2005), pp. 25–57.

[112] M. Wetzlinger, N. Kochdumper, and M. Althoff. "Adaptive parameter tuning for reachability analysis of linear systems". In: *Conference on Decision and Control*. IEEE, 2020, pp. 5145–5152.

[113] M. Wetzlinger et al. "Adaptive reachability algorithms for nonlinear systems using abstraction error analysis". In: *Nonlinear Analysis: Hybrid Systems* 46 (2022), p. 101252.

[114] M. Wetzlinger et al. "Fully automated verification of linear systems using inner- and outer-approximations of reachable sets". In: *Transactions on Automatic Control* (2023), pp. 1–16.

[115] M. Wetzlinger et al. "Fully-automated verification of linear systems using reachability analysis with support functions". In: *International Conference on Hybrid Systems: Computation and Control*. ACM, 2023.

[116] E. M. Wolff and R. M. Murray. "Optimal control of nonlinear systems with temporal logic specifications". In: *Robotics Research*. Springer, 2016, pp. 21–37.

[117] A. Yershova and S.M. LaValle. "Deterministic sampling methods for spheres and SO (3)". In: *Conference on Robotics and Automation*. Vol. 4. IEEE, 2004, pp. 3974–3980.

[118] M. Zamani et al. "Symbolic models for nonlinear control systems without stability assumptions". In: *Transactions on Automatic Control* 57.7 (2012), pp. 1804–1809.

[119] M.N. Zeilinger et al. "On real-time robust model predictive control". In: *Automatica* 50.3 (2014), pp. 683–694.

[120] G.M. Ziegler. *Lectures on polytopes*. New York, USA: Springer, 1995.