TECHNISCHE UNIVERSITÄT WIEN

AUTOMATION & CONTROL INSTITUTE
INSTITUT FÜR AUTOMATISIERUNGS- & REGELUNGSTECHNIK

Iterative Linear Quadratic Regulator for Collision-Free Trajectory Optimization and Model Predictive Control of a Timber Crane

DIPLOMA THESIS

Conducted in partial fulfillment of the requirements for the degree of a

Diplom-Ingenieur (Dipl.-Ing.)

supervised by

Univ.-Prof. Dr. techn. A. Kugi
Dr. techn. B. Bischof

submitted at the

TU Wien
Faculty of Electrical Engineering and Information Technology
Automation and Control Institute

by

Marc-Philip Ecker

Wien, October 2022

Complex Dynamical Systems Group
A-1040 Wien, Gußhausstr. 27–29, Internet: https://www.acin.tuwien.ac.at

Preamble

There are so many people that supported me my whole life and I want to thank all of them. However, I am not a person that uses big words and I guess most of them won't read this thesis anyways.

Even though, I want to thank Dr.techn. Bernhard Bischof for his support in writing this thesis. Additionally, I want to thank Dr.techn. Tobias Glück for a lot of interesting discussions and Univ.-Prof. Dr.techn. Andreas Kugi for his motivating lectures and his respectful dealing with students, which motivated me to specialize in the field of robotics and automation.

Last but not least I want to thank by domestic partner for being patient enough living with me for seven years and hopefully many more to come.

Wien, October 2022

Abstract

Collision-free trajectory planning is a major task for any autonomous robotic system. Algorithms that solve the planning task have to be fast enough to operate online, e.g. in order to react to changing environments. Hence, low computation time is a major criteria for algorithms performing online trajectory planning. This thesis deals with a special class of optimal control algorithms with application to collision-free trajectory planning, called iterative linear quadratic regulator (iLQR). The main benefit of these algorithms is the linear time dependency on the considered time horizon length. Two recent algorithms that integrate constraints into the iLQR are compared and applied to offline trajectory planning for a simple 2D kinematic vehicle and, a more realistic example, a timber crane. Additionally, an online model predictive control (MPC) trajectory planner is implemented for the timber crane in order to benchmark the performance of the constrained iLQR algorithms for an online motion planning problem.

Kurzzusammenfassung

Kollisionsfreie Trajektorienplanung ist eine der wichtigsten Augaben für jedes autonome System. Solche Trajektorienplanungen müssen in der Lage sein in Echtzeit zu operieren, um z.B. auf dynamische Umgebungen reagieren zu können. Daher is eine kurze Laufzeit ein wesentliches Kriterium für diese Algorithmen. Diese Arbeit beschäftigt sich mit einer speziellen Klasse von Algorithmen, nämlich Iterative Linear Quadratic Regulator (iLQR), zur Lösung von Optimalsteuerungsproblemen. Der wesentliche Vorteil dieser Algorithmen ist die lineare Abhängingkeit der Laufzeit von der Länge des betrachteten Zeithorizontes. Zwei jüngst veröffentliche Algorithmen zur Integration von Beschränkungen in das iLQR Schema werden als offline Trajektorienplaner für ein einfaches kinematisches Fahrzeugmodell und fÃ¼r einen Holzkran implementiert und verglichen. Zusätzlich wird ein modelprädiktiver Regeler als online Trajektorienplaner für den Holzkran implementiert, um die iLQR Algorithmen mit Beschränkungen in einem online Szenario zu evaluieren.

Nomenclature

Acronyms

NP	Non polynomial
NLP	Nonlinear programming
iLQR	Iterative linear quadratic regulator
DDP	Differential dynamic programming
SQP	Sequential quadratic programming
KKT	Karush-Kuhn-Tucker
AL	Augmented Lagrangian
ALTRO	Augmented Lagrangian trajectory optimizer

General Notation

$a, \gamma, A, \Gamma, \dots$	Scalars
$\mathbf{a}, \boldsymbol{\gamma}, \mathbf{b}, \boldsymbol{\xi}, \dots$	Vectors
$\mathbf{A}, \boldsymbol{\Gamma}, \mathbf{B}, \boldsymbol{\Xi}, \dots$	Matrices
$\mathcal{A}, \mathcal{B}, \dots$	Sets
$(\cdot)^{\mathrm{T}}$	Transpose of a quantity
$\dot{(\cdot)}$	Total time derivative of a quantity
$\mathbb{R}$	Set of real numbers
$\mathbb{N}$	Set of natural numbers
$\mathbb{Z}$	Set of integers
$\mathcal{C}^n(\mathcal{A}, \mathcal{B})$	Set of n-times continuous differentiable functions from $\mathcal{A}$ to $\mathcal{B}$
$\mathbf{f}_{\mathbf{x}}$	Partial derivative of scalar function f w.r.t. $\mathbf{x}$
$\mathbf{F}_{\mathbf{xy}}$	Partial derivative of scalar function f first w.r.t. $\mathbf{x}$, then w.r.t. $\mathbf{y}$
$(\cdot)[i, j]$	Element in row i and column j of a matrix
$\mathrm{diag}(\cdot)$	Diagonal matrix generated from argument vector
$\mathbf{J}_{\mathbf{f},\mathbf{x}}$	Jacobian of a function $\mathbf{f}$ w.r.t. argument $\mathbf{x}$

General Symbols

$\mathbf{0}$	Vector or matrix with all entries being zero
$\mathbf{I}$	Identity matrix
$\mathbf{R}_i^j$	Rotation matrix transforming a point from frame i to frame j
$\mathbf{d}_i^j$	Position of frame j in frame i

$\mathbf{H}_i^j$	Homogeneous transformation from frame i to frame j
$\mathbf{x}$	State of a general system
$\mathbf{u}$	Control input of a general system
$\boldsymbol{\lambda}$	Lagrange multiplier
$\boldsymbol{\zeta}$	Slack variable
$\varphi(\cdot)$	Terminal cost of an optimal control problem
$l(\cdot)$	Step cost of an optimal control problem
$V_k(\cdot)$	Value function for discrete time k
$J(\cdot)$	Cost function of an optimization problem
$\mathbf{A}_k$	State Jacobian of a discrete time system
$\mathbf{B}_k$	Control Jacobian of a discrete time system
$\mathbf{l}_{\mathbf{x},k}, \mathbf{l}_{\mathbf{u},k}$	Partial derivative of step cost at discrete time k
$\mathbf{L}_{\mathbf{xx},k}, \mathbf{L}_{\mathbf{uu},k}, \mathbf{L}_{\mathbf{xu},k}$	Second derivative matrix of step cost at discrete time k
$\mathbf{K}_k, \mathbf{k}_k$	iLQR feedback gains at time k
$\mathbf{s}_k, \mathbf{S}_k$	Quadratic approximation terms for value function at time k
$\Delta\mathbf{u}_k$	Control increment at time k
$\Delta\mathbf{x}_k$	State increment at time k
$\mathbf{f}(\mathbf{x}_k, \mathbf{u}_k)$	Discrete time state equation of general a system
$\mathbf{h}(\mathbf{x}, \mathbf{u})$	Inequality constraint function
$\overline{\mathbf{u}}, \underline{\mathbf{u}}$	Upper and lower bound of control input
$\overline{\mathbf{x}}, \underline{\mathbf{x}}$	Upper and lower bound of state
$\overline{\mathbf{q}}, \underline{\mathbf{q}}$	Upper and lower bound of joint coordinates
$\overline{\dot{\mathbf{q}}}, \underline{\dot{\mathbf{q}}}$	Upper and lower bound of joint velocities
N	Horizon length of optimal control problem
T_s	Sampling time
$\mathbf{x}_{N,d}, \mathbf{q}_{N,d}$	Desired endpoints of an optimal control problem
$d(\mathcal{O}_1, \mathcal{O}_2)$	Distance function between objects $\mathcal{O}_1$ and $\mathcal{O}_2$
$\boldsymbol{\nu}(\mathcal{O})$	Mapping to minimum norm point of object $\mathcal{O}$
$h_{\mathcal{O}}$	Support function of object $\mathcal{O}$
$\mathbf{s}_{\mathcal{O}}$	Support mapping of object $\mathcal{O}$
$\mathcal{S}_k$	Simplex of GJK algorithm at iteration k
$\mathbf{q}$	Generalized coordinates of a rigid body
$\mathbf{q}_A$	Actuated joint coordinates of a rigid body
$\mathbf{q}_U$	Non-actuated joint coordinates of a rigid body
$q_1, \ldots, q_n$	Components of generalized coordinates
$\mathbf{D}(\mathbf{q})$	Mass matrix
$\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$	Coriolis matrix
$\mathbf{g}(\mathbf{q})$	Potential forces
$\boldsymbol{\tau}$	Generalized forces
$\mathbf{F}$	Diagonal matrix with viscous friction coefficients
$\mathcal{L}_i(\mathbf{q})$	Sets representing link collision objects

Contents

1 Introduction

1.1 Motivation

Collision-free trajectory planning is a major task of every autonomous robot. In order to implement a machine that is able to react to its dynamic environment such algorithms must be able to perform the planning tasks in real time. Hence, fast trajectory computation is a major criterion for an online planner. While approaches like grid-based or sampling-based planners benefit from their ability to find globally optimal trajectories, grid-based methods have non-polynomial (NP) complexity and the computational complexity of sampling-based methods is hard to characterize due to their random nature, c.f. [1, 2]. This is why optimization-based planning algorithms, i.e. algorithms that solve an optimal control problem for trajectory generation, are more popular to solve the online (re-)planning task, while combinatorial planners are often used for computing an initial trajectory for optimization-based planners, see, e.g., [3–6]. These optimization-based algorithms can be categorized as follows, c.f. [7].

- **Direct methods:** The original infinite-dimensional optimal control problem is first transformed into a finite-dimensional nonlinear optimization problem (NLP). This can then be solved using standard NLP solvers.
- **Indirect methods:** First, optimality conditions of the original infinite-dimensional optimal control problem are formulated using the calculus of variations, which are then solved in order to get a solution candidate.

Indirect methods require prior knowledge of active parts in path inequality constraints, c.f. [7]. In robotic trajectory planning, a typical path inequality constraint comes from the requirement that the trajectory must be collision free and it is not possible to determine the active set for these constraints in advance. Hence, direct methods are used for collision-free trajectory planning.

Direct optimal control algorithms can be further distinguished as follows, c.f. [7, 8].

- **Multiple shooting methods:** In these methods both (some) states and control inputs are used as optimization variables. The connection between these quantities is considered using the system equation as equality constraint.
- **Single shooting methods:** Here only the control inputs are used as optimization variables and the state is retrieved by forward simulation of the system dynamics.

Since the state in single shooting methods is not handled as optimization variable, the resulting optimization problem has a smaller dimension, which leads to a lower number of linear equations and thus less computational complexity per iteration. However, as a

consequence, the initial state trajectory cannot be chosen arbitrarily because it has to be consistent with the initial control trajectory. This is not the case in multiple shooting methods.

This thesis deals with optimization-based trajectory planning for collision avoidance applications. More specifically, a special class of algorithms for solving discrete time optimal control problems is studied, called *iterative linear quadratic regulator* (iLQR), c.f. [9]. In their original formulation, these algorithms belong to the class of direct single shooting methods and, compared to other common NLP solvers, the main benefit lies in the linear time complexity in the number of discrete trajectory sampling points, which is why these algorithms are especially well suited for fast trajectory planning.

The iLQR algorithms are evaluated for two applications, namely a 2D kinematic vehicle and a timber crane. The former has a lower state-space dimension and simpler collision constraints, which is why it is used as a first test example. The latter one deals with more realistic collision constraints and a higher state space and aims at evaluating the algorithms for a more realistic application.

Finally, an online planner based on *model predictive control* (MPC) is implemented for the timber crane using the constrained iLQR algorithms presented in this thesis.

1.2 Related Work

Unconstrained *differential dynamic programming* (DDP) was originally proposed in the 1960's [10]. The main idea is to use a second-order approximation of both, the nonlinear system dynamics and the nonlinear cost function and solve the resulting problem using *Bellman's principle of optimality*. Hence, the main benefit of DDP compared to other common optimization techniques, such as *sequential quadratic programming* (SQP), is that instead of optimizing all control inputs of size m in the prediction horizon of size N, i.e. Nm variables in one turn, m controls are optimized by formulating N separate optimization problems. Considering that second-order optimization methods are used, this leads to N equations of size m that have to be solved instead of solving one system of size Nm. Hence, the computational costs of DDP increase only linearly with N and lower computation times can be achieved. Additionally, the gain matrix retrieved in the last DDP iteration can be used as optimal state feedback gain without any additional computation required. Conditions in order to guarantee quadratic convergence behavior of DDP are dicussed in [11].

However, the original DDP formulation does not allow to initialize the system with infeasible trajectories. The work [8] extends the idea of DDP by a multiple shooting approach. This allows to initialize the DDP algorithm with state and control trajectories that do not satisfy the system equations. While this is an interesting approach for systems where it is not easy to compute control inputs for a given state trajectory, the control inputs of the crane considered in this thesis are the second-order time derivatives of the joint coordinates. Additionally, no special initialization technique is considered for both applications studied in this thesis, which is why this approach is not used. However, note that another interesting advantage of this multiple shooting formulation is that it enables a parallel implementation of the forward calculation, see, e.g.,Â´ [12].

Iterative linear-quadratic regulator (iLQR) algorithms, originally proposed in [9], are a subclass of DDP. The difference is that iLQR uses a linear approximation of the system dynamics instead of the quadratic approximation used in DDP, which can be quite expensive to compute. The authors achieved an increase in efficiency by a factor of 10 using iLQR instead of DDP. This idea is extended to stochastic systems subject to control constraints in [13] .

The first work that integrates control and terminal constraints in DDP is [14], which presents a variational form of DDP using the *Hamilton-Jacobi-Bellman equations*. In [15], the initial works on DDP are extended to handle terminal constraints by solving the *Karush-Kuhn-Tucker* (KKT) first-order optimality conditions. An early application of constrained DDP is presented in [16], where general nonlinear inequality constraints are incorporated using the *augmented Lagrangian* (AL) method in order to solve a hydroelectric generation scheduling problem.

A combined AL and KKT based approach is proposed in [17, 18], where soft constraints are considered using an AL method and hard constraints are incorporated using KKT conditions.

In [19], box constraints for control inputs are incorporated into iLQR using projected Newton quadratic programming combined with an active set method. The resulting algorithm is compared to simpler heuristic methods, called clamping and penalizing. Clamping, which simply crops box control constraints in the forward calculation, has the drawback that the search direction might not be a descent direction anymore. On the other hand, squashing, where the control signal is parameterized using a nonlinear function that ensures the constraints, e.g. the sigmoid function for box constraints, suffers from the cost function becoming a highly nonlinear function of the control inputs, which is reflected in bad convergence properties. However, the proposed projected Newton method can only be applied to box control constraints and not to arbitrary nonlinear control and state constraints. A projected Newton method able to handle such nonlinear state and control equality constraints is presented in [20], where the control inputs are projected onto the nullspace of the linearized constraints. However, it does not enable the integration of nonlinear state inequality constraints such as collision avoidance constraints.

One approach that is able to handle general nonlinear state and control constraints is provided in [21]. The method solves the KKT conditions using an active set approach. Lagrange multiplier based sensitivity analysis is used in order to determine constraints that have to be removed from the active set. In the forward calculation, a quadratic program is solved in order to guarantee that the new trajectory satisfies the constraints. In their experiments, they additionally show that penalty methods are more sensitive to local minima compared to their method. However, the number of possible active sets is exponential in the number of constraints, which is why this approach is not chosen in this thesis.

The *augmented Lagrangian trajectory optimizer* (ALTRO) provided in [22] is an open source iLQR framework based on the augmented Lagrangian method. Additionally, a square root formulation of the iLQR equations is used in order to increase numerical stability. While in [17, 18] the control variables and the Lagrange multipliers are updated simultaneously in the forward calculation, [22] uses the standard *Powell-Hestenes-Rockafellar* (PHR) approach [23–26]. Finally, a projection method combined with an

active set approach is used to fine-tune the final results.

A similar augmented Lagrangian approach is presented in [27]. Additionally, a sample-based gradient and Hessian approximation method is provided that is motivated by the unscented Kalman filter.

A solution that combines benefits of AL and KKT is presented in [28]. An initial solution is generated by a standard AL method with a modified penalty term that is two times differentiable in order to prevent numerical instabilities when using second-order methods. The solution is then fed to an algorithm that solves the KKT conditions using the iLQR scheme with slack variables for inequality constraints. This aims at combining the benefit of AL based methods, which is the robustness with respect to initialization, and simultaneously overcome the slow convergence rate near the optimal solutions, as pointed out in [29].

A primal-dual interior-point based DDP algorithm is proposed in [30], where the primal-dual system is optimized using barrier functions for the inequality constraints. Additionally, a formulation based on slack variables is provided, which allows violations of the original inequality constraints in the initialization trajectories.

1.3 Contribution and structure of the thesis

This thesis deals with online and offline optimization-based trajectory planning using constrained iLQR as the optimization framework. As the main application, offline trajectory planning and online model predictive control (MPC) based planning for a timber crane with collision avoidance constraints is considered. The thesis is structured as follows:

- The main theory behind the constrained iLQR algorithms used in this thesis is summarized in Chapter 2. As the main goal is to achieve low computational complexity, this thesis focuses on augmented Lagrangian and primal-dual interior point methods, since they do not require to solve the combinatorial active set problem.
- In Chapter 3, the algorithms discussed in Chapter 2 are benchmarked for a simple 2D kinematic vehicle model with very simple collision objects.
- Chapter 4 covers offline trajectory planning for a timber crane with more complex collision constraints and system dynamics compared to the kinematic vehicle application in Chapter 3. The mathematical model of the crane is presented and the optimal control problem as well as the mathematical treatment of the collision constraints is formulated. Additionally, a MPC for the timber crane based on the constrained iLQR algorithms as presented in Chapter 2 is implemented and evaluated.

2 Iterative Linear Quadratic Regulator

Unconstrained DDP is originally proposed in [10]. The main idea is to use a second-order approximation of both, the nonlinear system dynamics and the nonlinear cost function and solve the optimal control problem using *Bellman's principle*. In this work, as proposed in [9], iLQR is used instead of DDP. The difference is that only a first-order approximation of the system dynamics is used in iLQR. This results in less computation time, since the second-order terms of the system dynamics can be quite complex. The main benefit of these algorithms compared to other nonlinear optimization techniques for solving discrete time optimal control problems is the linear time complexity in the length of the time horizon.

This chapter summarizes the main ideas of the algorithms from [9, 19, 22, 30] used in this thesis and explains the implementation details. Additionally, a simple kinematic vehicle model is used as a benchmark example in order discuss some properties of the different methods.

2.1 Unconstrained iLQR

Consider a discrete time optimal control problem given by

$$\min_{\mathbf{u}_0,\dots,\mathbf{u}_{N-1}\in\mathbb{R}^m} \quad J(\mathbf{u}_0,\dots,\mathbf{u}_{N-1}) \tag{2.1a}$$

$$\text{s.t.} \quad \mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k,\mathbf{u}_k),\ \mathbf{x}_0 = \boldsymbol{\xi}_0\ , \tag{2.1b}$$

with a fixed $\boldsymbol{\xi}_0 \in \mathbb{R}^n$ and the cost function

$$J(\mathbf{u}_0,\dots,\mathbf{u}_{N-1}) = \varphi(\mathbf{x}_N) + \sum_{j=0}^{N-1} l(\mathbf{x}_j,\mathbf{u}_j)\ , \tag{2.2}$$

where the cost $l \in \mathcal{C}^2(\mathbb{R}^n \times \mathbb{R}^m, \mathbb{R})$ and the terminal cost $\varphi \in \mathcal{C}^2(\mathbb{R}^n, \mathbb{R})$ are two times continuously differentiable functions. The discrete time system state at time $t_k = kT_s$, with the sampling time T_s, is denoted by $\mathbf{x}_k \in \mathbb{R}^n$, $k = 1,\dots,N$ and the control input is denoted by $\mathbf{u}_k \in \mathbb{R}^m$, $k = 1,\dots,N-1$. For the discrete time system dynamics $\mathbf{f} \in \mathcal{C}^1(\mathbb{R}^n \times \mathbb{R}^m, \mathbb{R}^n)$ only continuous differentiability is required.

Due to *Bellman's principle of optimality*, the solution of the optimal control problem (2.1) is equivalent to the solution $V_0^*(\mathbf{x}_k)$, which is recursively defined by

$$V_k^*(\mathbf{x}_k) = \min_{\mathbf{u}_k\in\mathbb{R}^m} \Big\{ \underbrace{l(\mathbf{x}_k,\mathbf{u}_k) + V_{k+1}^*(\mathbf{f}(\mathbf{x}_k,\mathbf{u}_k))}_{=:V_k(\mathbf{x}_k,\mathbf{u}_k)} \Big\} \quad \text{with} \quad V_N^*(\mathbf{x}_N) = \varphi(\mathbf{x}_N) \tag{2.3}$$

for $k = 0,\dots,N-1$, c.f. [10].

In iLQR, the nonlinear optimization problem (2.3) is solved iteratively using a quadratic approximation of $V_k(\mathbf{x}_k, \mathbf{u}_k)$ and a linear approximation of the discrete time system dynamics (2.1b). The quadratic approximation of $V_k(\mathbf{x}_k, \mathbf{u}_k)$ is derived using small perturbations of the state $\mathbf{x}_k + \tilde{\mathbf{x}}_k$, $\tilde{\mathbf{x}}_k \in \mathbb{R}^n$, and the control input $\mathbf{u}_k + \tilde{\mathbf{u}}_k$, $\tilde{\mathbf{u}}_k \in \mathbb{R}^m$, around the actual values $\mathbf{x}_k$ and $\mathbf{u}_k$

$$V_k(\mathbf{x}_k + \tilde{\mathbf{x}}_k, \mathbf{u}_k + \tilde{\mathbf{u}}_k) \approx V_k(\mathbf{x}_k, \mathbf{u}_k) + \tilde{V}_k(\tilde{\mathbf{x}}_k, \tilde{\mathbf{u}}_k) \ , \tag{2.4}$$

with the quadratic cost perturbation

$$\tilde{V}_k(\tilde{\mathbf{x}}_k, \tilde{\mathbf{u}}_k) := \tilde{\mathbf{x}}_k^{\mathrm{T}} \mathbf{v}_{\mathbf{x},k} + \tilde{\mathbf{u}}_k^{\mathrm{T}} \mathbf{v}_{\mathbf{u},k} + \frac{1}{2} \tilde{\mathbf{x}}_k^{\mathrm{T}} \mathbf{V}_{\mathbf{xx},k} \tilde{\mathbf{x}}_k + \frac{1}{2} \tilde{\mathbf{u}}_k^{\mathrm{T}} \mathbf{V}_{\mathbf{uu},k} \tilde{\mathbf{u}}_k + \tilde{\mathbf{x}}_k^{\mathrm{T}} \mathbf{V}_{\mathbf{xu},k} \tilde{\mathbf{u}}_k \ . \tag{2.5}$$

Using the definition of $V_k(\mathbf{x}_k, \mathbf{u}_k)$ in (2.3) with a linear approximation of (2.1b), i.e.

$$\tilde{\mathbf{x}}_{k+1} = \mathbf{A}_k \tilde{\mathbf{x}}_k + \mathbf{B}_k \tilde{\mathbf{u}}_k \tag{2.6}$$

with $\mathbf{A}_k \in \mathbb{R}^{n \times n}$ and $\mathbf{B}_k \in \mathbb{R}^{n \times m}$ being the Jacobian of $\mathbf{f}(\mathbf{x}, \mathbf{u})$ w.r.t. to $\mathbf{x}$ and $\mathbf{u}$ evaluated at $(\mathbf{x}_k, \mathbf{u}_k)$, respectively, yields

$$\mathbf{v}_{\mathbf{x},k} = \mathbf{l}_{\mathbf{x},k} + \mathbf{A}_k^{\mathrm{T}} \mathbf{s}_{k+1} \tag{2.7a}$$
$$\mathbf{v}_{\mathbf{u},k} = \mathbf{l}_{\mathbf{u},k} + \mathbf{B}_k^{\mathrm{T}} \mathbf{s}_{k+1} \tag{2.7b}$$
$$\mathbf{V}_{\mathbf{xx},k} = \mathbf{L}_{\mathbf{xx},k} + \mathbf{A}_k^{\mathrm{T}} \mathbf{S}_{k+1} \mathbf{A}_k \tag{2.7c}$$
$$\mathbf{V}_{\mathbf{uu},k} = \mathbf{L}_{\mathbf{uu},k} + \mathbf{B}_k^{\mathrm{T}} \mathbf{S}_{k+1} \mathbf{B}_k \tag{2.7d}$$
$$\mathbf{V}_{\mathbf{xu},k} = \mathbf{L}_{\mathbf{xu},k} + \mathbf{A}_k^{\mathrm{T}} \mathbf{S}_{k+1} \mathbf{B}_k \ , \tag{2.7e}$$

with

$$\mathbf{L}_{\mathbf{xx},k} = \frac{\partial^2 l(\mathbf{x}, \mathbf{u})}{\partial \mathbf{x}^2}\bigg|_{\substack{\mathbf{x}=\mathbf{x}_k \\ \mathbf{u}=\mathbf{u}_k}} \qquad \mathbf{l}_{\mathbf{x},k} = \left(\frac{\partial l(\mathbf{x}, \mathbf{u})}{\partial \mathbf{x}}\bigg|_{\substack{\mathbf{x}=\mathbf{x}_k \\ \mathbf{u}=\mathbf{u}_k}} \right)^{\mathrm{T}} \tag{2.8a}$$

$$\mathbf{L}_{\mathbf{uu},k} = \frac{\partial^2 l(\mathbf{x}, \mathbf{u})}{\partial \mathbf{u}^2}\bigg|_{\substack{\mathbf{x}=\mathbf{x}_k \\ \mathbf{u}=\mathbf{u}_k}} \qquad \mathbf{l}_{\mathbf{u},k} = \left(\frac{\partial l(\mathbf{x}, \mathbf{u})}{\partial \mathbf{u}}\bigg|_{\substack{\mathbf{x}=\mathbf{x}_k \\ \mathbf{u}=\mathbf{u}_k}} \right)^{\mathrm{T}} \tag{2.8b}$$

$$\mathbf{L}_{\mathbf{xu},k} = \frac{\partial}{\partial \mathbf{u}} \left(\frac{\partial l(\mathbf{x}, \mathbf{u})}{\partial \mathbf{x}} \right)^{\mathrm{T}}\bigg|_{\substack{\mathbf{x}=\mathbf{x}_k \\ \mathbf{u}=\mathbf{u}_k}} \ . \tag{2.8c}$$

Hence, (2.3) can be numerically minimized by iteratively minimizing (2.4), and equivalently (2.5), w.r.t. $\tilde{\mathbf{u}}_k$. If $\mathbf{V}_{\mathbf{uu},k} \in \mathbb{R}^{n \times n}$ in (2.5) is positive definite, the function (2.5) is stricly convex and the unique minima can be computed by setting the derivative w.r.t. $\tilde{\mathbf{u}}_k$ to zero. This yields the optimal control perturbation $\tilde{\mathbf{u}}_k^*$ as a function of the state perturbation $\tilde{\mathbf{x}}_k$ with

$$\tilde{\mathbf{u}}_k^* = \mathbf{K}_k \tilde{\mathbf{x}}_k + \mathbf{d}_k \ , \qquad \mathbf{K}_k := -\mathbf{V}_{\mathbf{uu},k}^{-1} \mathbf{V}_{\mathbf{xu},k}^{\mathrm{T}} \qquad \text{and} \qquad \mathbf{d}_k := -\mathbf{V}_{\mathbf{uu},k}^{-1} \mathbf{v}_{\mathbf{u},k} \ . \tag{2.9}$$

The updated control inputs $\bar{\mathbf{u}}_k \in \mathbb{R}^m$, $k = 1, \ldots, N-1$ are then computed using

$$\bar{\mathbf{u}}_k = \mathbf{u}_k + \tilde{\mathbf{u}}_k^* \tag{2.10}$$

and the updated states $\bar{\mathbf{x}}_k \in \mathbb{R}^n$, $k = 1, \ldots, N$ are retrieved by forward simulation of the discrete time system dynamics (2.1b). The state perturbation $\tilde{\mathbf{x}}_k$, $k = 1, \ldots, N$ is computed using the difference $\tilde{\mathbf{x}}_k = \bar{\mathbf{x}}_k - \mathbf{x}_k$.

The vector $\mathbf{s}_{k+1} \in \mathbb{R}^n$ and the matrix $\mathbf{S}_{k+1} \in \mathbb{R}^{n \times n}$, $k = 0, \ldots, N-1$, are gradient and Hessian approximation of $V^*_{k+1}(\mathbf{f}(\mathbf{x}_k, \mathbf{u}_k))$, respectively. They can be computed recursively in a backward iteration as follows:
For $k = N$ we get $V^*_N(\mathbf{x}_N) = \varphi(\mathbf{x}_N)$, which yields

$$\mathbf{s}_N = \left(\frac{\partial \varphi}{\partial \mathbf{x}}\Big|_{\mathbf{x}=\mathbf{x}_N} \right)^{\mathrm{T}} \qquad \text{and} \qquad \mathbf{S}_N = \left(\frac{\partial^2 \varphi}{\partial \mathbf{x}^2}\Big|_{\mathbf{x}=\mathbf{x}_N} \right)^{\mathrm{T}} . \tag{2.11}$$

Assuming that $\mathbf{s}_{k+1}$, $\mathbf{S}_{k+1}$ is known, $\mathbf{s}_k$ and $\mathbf{S}_k$ can be computed by plugging the optimal control perturbation $\tilde{\mathbf{u}}^*_k$ from (2.9) into (2.5) which yields a quadratic function $\tilde{V}^*_k(\tilde{\mathbf{x}}_k)$ in the state increment $\tilde{\mathbf{x}}_k$. Equating coefficients of the linear and quadratic parts yields

$$\mathbf{s}_k = \mathbf{v}_{\mathbf{x}_k} + \mathbf{K}_k^{\mathrm{T}} \mathbf{V}_{\mathbf{uu},k} \mathbf{d}_k + \mathbf{K}_k^{\mathrm{T}} \mathbf{v}_{\mathbf{u},k} + \mathbf{V}_{\mathbf{xu},k} \mathbf{d}_k \tag{2.12a}$$

$$\mathbf{S}_k = \mathbf{V}_{\mathbf{xx},k} + \mathbf{K}_k^{\mathrm{T}} \mathbf{V}_{\mathbf{uu},k} \mathbf{K}_k + \mathbf{K}_k^{\mathrm{T}} \mathbf{V}_{\mathbf{xu},k}^{\mathrm{T}} + \mathbf{V}_{\mathbf{xu},k} \mathbf{K}_k \ . \tag{2.12b}$$

Hence, given an initial control sequence $(\mathbf{u}_k)_{k=0}^{N-1}$ and a corresponding feasible state sequence $(\mathbf{x}_k)_{k=0}^{N}$, the algorithm first computes $\mathbf{s}_k$, $\mathbf{S}_k$ in (2.12) and $\mathbf{K}_k$, $\mathbf{d}_k$ in (2.9) backwards for $k = N-1, \ldots, 0$. Then $\tilde{\mathbf{x}}_k$ and $\tilde{\mathbf{u}}^*_k$ can be computed in a forward iteration using (2.1b) and (2.9). Algorithm 1 summarizes the whole iLQR optimization procedure.

For the concrete implementation, the change in cost is considered as termination criteria, i.e. as soon as the change in cost falls below a certain threshold, the algorithm terminates. Additionally, a line search parameter $\alpha > 0$ with

$$\tilde{\mathbf{u}}^*_k = \mathbf{K}_k \tilde{\mathbf{x}}_k + \alpha \mathbf{d}_k \tag{2.13}$$

is used in order to improve the convergence behavior. Let $\bar{J} = J(\bar{\mathbf{u}}_0, \ldots, \bar{\mathbf{u}}_{N-1})$ be the cost (2.2) of the update state and control sequences and $J = J(\mathbf{u}_0, \ldots, \mathbf{u}_{N-1})$ the one of the original state and control sequences. The actual cost change $\tilde{J}$ is computed as

$$\tilde{J} = \bar{J} - J \ . \tag{2.14}$$

As described in [31], the expected cost decrease $\hat{\tilde{J}}$ can be written

$$\hat{\tilde{J}}(\alpha) = \frac{\alpha^2}{2} \sum_{k=0}^{N-1} \mathbf{d}_k^{\mathrm{T}} \mathbf{V}_{\mathbf{uu},k} \mathbf{d}_k + \alpha \sum_{k=0}^{N-1} \mathbf{d}_k^{\mathrm{T}} \mathbf{v}_{\mathbf{u},k} \ . \tag{2.15}$$

Starting from $\alpha = 1$, this parameter is decreased until $\frac{\tilde{J}}{\hat{\tilde{J}}} \in [10^{-4}, 10]$.

Regularization is implemented by additionally penalizing large control perturbations $\tilde{\mathbf{u}}_k$ in the cost function (2.5), which yields

$$\begin{aligned} \tilde{V}^*_{k+1}(\tilde{\mathbf{x}}_{k+1}) = \min_{\tilde{\mathbf{u}}_k \in \mathbb{R}^m} \Big\{ & \mathbf{v}_{\mathbf{x},k}^{\mathrm{T}} \tilde{\mathbf{x}}_k + \mathbf{v}_{\mathbf{u},k}^{\mathrm{T}} \tilde{\mathbf{u}}_k + \frac{1}{2} \tilde{\mathbf{x}}_k^{\mathrm{T}} \mathbf{V}_{\mathbf{xx},k} \tilde{\mathbf{x}}_k \\ & + \frac{1}{2} \tilde{\mathbf{u}}_k^{\mathrm{T}} \Big(\mathbf{V}_{\mathbf{uu},k} + \nu \mathbf{I} \Big) \tilde{\mathbf{u}}_k + \tilde{\mathbf{x}}_k^{\mathrm{T}} \mathbf{V}_{\mathbf{xu},k} \tilde{\mathbf{u}}_k \Big\} \ , \end{aligned} \tag{2.16}$$

Algorithm 1 Unconstrained iLQR

Require: An initial control sequence $(\mathbf{u}_k)_{k=0}^{N-1} \in \mathbb{R}^{m \times N-1}$ and the corresponding feasible state sequence $(\mathbf{x}_k)_{k=0}^{N} \in \mathbb{R}^{n \times N}$.

while $\tilde{J} > \varepsilon$ **do**

$\mathbf{s}_N = \left(\frac{\partial \varphi}{\partial \mathbf{x}} \Big|_{\mathbf{x}=\mathbf{x}_N} \right)^{\mathrm{T}}$

$\mathbf{S}_N = \left(\frac{\partial^2 \varphi}{\partial \mathbf{x}^2} \Big|_{\mathbf{x}=\mathbf{x}_N} \right)^{\mathrm{T}}$

for $k = N-1 \dots 0$ **do** ▷ Backward-Pass

$\mathbf{v}_{\mathbf{x},k} \leftarrow \mathbf{l}_{\mathbf{x},k} + \mathbf{A}_k^{\mathrm{T}} \mathbf{s}_{k+1}$

$\mathbf{v}_{\mathbf{u},k} \leftarrow \mathbf{l}_{\mathbf{u},k} + \mathbf{B}_k^{\mathrm{T}} \mathbf{s}_{k+1}$

$\mathbf{V}_{\mathbf{xx},k} \leftarrow \mathbf{L}_{\mathbf{xx},k} + \mathbf{A}_k^{\mathrm{T}} \mathbf{S}_{k+1} \mathbf{A}_k$

$\mathbf{V}_{\mathbf{uu},k} \leftarrow \mathbf{L}_{\mathbf{uu},k} + \mathbf{B}_k^{\mathrm{T}} \mathbf{S}_{k+1} \mathbf{B}_k$

$\mathbf{V}_{\mathbf{ux},k} \leftarrow \mathbf{L}_{\mathbf{ux},k} + \mathbf{B}_k^{\mathrm{T}} \mathbf{S}_{k+1} \mathbf{A}_k$

$\mathbf{K}_k \leftarrow -\mathbf{V}_{\mathbf{uu},k}^{-1} \mathbf{V}_{\mathbf{xu},k}^{\mathrm{T}}$

$\mathbf{d}_k \leftarrow -\mathbf{V}_{\mathbf{uu},k}^{-1} \mathbf{v}_{\mathbf{u},k}$

$\mathbf{s}_k \leftarrow \mathbf{v}_{\mathbf{x}_k} + \mathbf{K}_k^{\mathrm{T}} \mathbf{V}_{\mathbf{uu},k} \mathbf{d}_k + \mathbf{K}_k^{\mathrm{T}} \mathbf{v}_{\mathbf{u},k} + \mathbf{V}_{\mathbf{xu},k} \mathbf{d}_k$

$\mathbf{S}_k \leftarrow \mathbf{V}_{\mathbf{xx},k} + \mathbf{K}_k^{\mathrm{T}} \mathbf{V}_{\mathbf{uu},k} \mathbf{K}_k + \mathbf{K}_k^{\mathrm{T}} \mathbf{V}_{\mathbf{xu},k}^{\mathrm{T}} + \mathbf{V}_{\mathbf{xu},k} \mathbf{K}_k$

end for

$\bar{\mathbf{x}}_0 \leftarrow \mathbf{x}_0$

for $k = 0 \dots N-1$ **do** ▷ Forward-Pass

$\tilde{\mathbf{x}}_k \leftarrow \bar{\mathbf{x}}_k - \mathbf{x}_k$

$\tilde{\mathbf{u}}_k \leftarrow \mathbf{K}_k \tilde{\mathbf{x}}_k + \mathbf{d}_k$

$\bar{\mathbf{u}}_k \leftarrow \mathbf{u}_k + \tilde{\mathbf{u}}_k$

$\bar{\mathbf{x}}_{k+1} \leftarrow \mathbf{f}(\bar{\mathbf{x}}_k, \bar{\mathbf{u}}_k)$

end for

$(\mathbf{x}_k)_{k=0}^{N} \leftarrow (\bar{\mathbf{x}}_k)_{k=0}^{N}$

$(\mathbf{u}_k)_{k=0}^{N-1} \leftarrow (\bar{\mathbf{u}}_k)_{k=0}^{N-1}$

end while

with the identity matrix $\mathbf{I} \in \mathbb{R}^{m \times m}$.

The regularization parameter $\nu > 0$ is decreased in every iteration. The linear system (2.9) is solved using the *Cholesky factorization*. If this factorization fails due to missing positive definiteness of $\mathbf{V}_{\mathbf{uu},k}$ the penalty weights ν are increased again. Additionally, if the step size α becomes too small ν is increased as well.

2.2 Constrained iLQR

This section deals with the constrained optimal control problem given by

$$\min_{\mathbf{u}_0,\dots,\mathbf{u}_{N-1} \in \mathbb{R}^m} \quad J(\mathbf{u}_0, \dots, \mathbf{u}_{N-1}) \tag{2.17a}$$

$$\text{s.t.} \quad \mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k) \tag{2.17b}$$

$$\mathbf{h}(\mathbf{x}_k, \mathbf{u}_k) \leq \mathbf{0} \tag{2.17c}$$

with J being defined as in (2.2). Again, $l \in \mathcal{C}^2(\mathbb{R}^n \times \mathbb{R}^m, \mathbb{R})$, $\varphi \in \mathcal{C}^2(\mathbb{R}^n, \mathbb{R})$ are two times continuously differentiable and $\mathbf{f} \in \mathcal{C}^1(\mathbb{R}^n \times \mathbb{R}^m, \mathbb{R}^n)$ is continuously differentiable. Additionally, for $\mathbf{h} \in \mathcal{C}^1(\mathbb{R}^n \times \mathbb{R}^m, \mathbb{R}^q)$ continuous differentiability is required.

2.2.1 Projected Newton method for control limits

A method for incorporating box control constraints using the projected Newton method is provided in [19]. As seen in the derivation of the unconstrained iLQR, in each iteration we solve a quadratic program in order to get an expression for the optimal control perturbation $\tilde{\mathbf{u}}_k \in \mathbb{R}^m$. If we take into account the control constraints we get, compare (2.5)

$$\min_{\tilde{\mathbf{u}} \in \mathbb{R}^m} \quad \left\{ \frac{1}{2} \tilde{\mathbf{u}}^{\mathrm{T}} \mathbf{V}_{\mathbf{uu}} \tilde{\mathbf{u}} + \tilde{\mathbf{u}}^{\mathrm{T}} (\mathbf{V}_{\mathbf{xu}}^{\mathrm{T}} \tilde{\mathbf{x}} + \mathbf{v}_{\mathbf{u}}) \right\} \tag{2.18a}$$

$$\text{s.t.} \quad \underline{\mathbf{u}} \leq \mathbf{u} + \tilde{\mathbf{u}} \leq \overline{\mathbf{u}} \,, \tag{2.18b}$$

where $\underline{\mathbf{u}}, \overline{\mathbf{u}} \in \mathbb{R}^m$ are lower and upper bounds of the control input. The time index is omitted for simplicity.

During the backward computation $\tilde{\mathbf{x}}$ is unknown. However, the assumption $\tilde{\mathbf{x}} \approx \mathbf{0}$ can be used, which allows to solve

$$\min_{\tilde{\mathbf{u}} \in \mathbb{R}^m} \quad \left\{ \frac{1}{2} \tilde{\mathbf{u}}^{\mathrm{T}} \mathbf{V}_{\mathbf{uu}} \tilde{\mathbf{u}} + \tilde{\mathbf{u}}^{\mathrm{T}} \mathbf{v}_{\mathbf{u}} \right\} \tag{2.19a}$$

$$\text{s.t.} \quad \underline{\mathbf{u}} \leq \mathbf{u} + \tilde{\mathbf{u}} \leq \overline{\mathbf{u}} \tag{2.19b}$$

during the backward pass in order to estimate the active inequality constraints as shown in the following.

Let $\mathbf{g} := \mathbf{V}_{\mathbf{uu}} \tilde{\mathbf{u}} + \mathbf{v}_{\mathbf{u}}$ be the gradient of the objective function. If $u_j = \underline{u}_j$ and $g_j > 0$, where $\cdot_j$ indicates the j-th component of the corresponding vector, we know that the steepest descent direction points towards smaller values of u_j, i.e. it allows to assume that the inequality constraint is active for u_j. A similar argument can be made for $u_j = \overline{u}_j$ and $g_j < 0$. The active set of controls can be estimated using

$$\mathcal{A} = \left\{ j \in \{1, \dots, m\} : (u_j = \underline{u}_j \wedge g_j > 0) \vee (u_j = \overline{u}_j \wedge g_j < 0 \right\} . \tag{2.20}$$

This allows to transform the original quadratic program with inequality constraints into one with equality constraints, that is

$$\min_{\tilde{\mathbf{u}}\in\mathbb{R}^m} \quad \left\{\frac{1}{2}\tilde{\mathbf{u}}^{\mathrm{T}}\mathbf{V}_{\mathbf{uu}}\tilde{\mathbf{u}} + \tilde{\mathbf{u}}^{\mathrm{T}}\mathbf{v}_{\mathbf{u}}\right\} \tag{2.21a}$$

$$\text{s.t.} \quad \tilde{u}_j = 0,\ j \in \mathcal{A}\ . \tag{2.21b}$$

Since the equality constraints trivially fulfill the *linear independence constraint qualification (LICQ)* condition, the search space is reduced to the manifold $\mathcal{U} = \{\tilde{\mathbf{u}} \in \mathbb{R}^m : \tilde{u}_j = 0, j \in \mathcal{A}\}$ with dimension $d = m - |\mathcal{A}|$.

The general idea of projected Newton methods is to project the gradient and Hessian of the objective function onto the tangent space of $\mathcal{U}$ defined by the equality constraints and then use the projected quantities to perform a Newton step. With general nonlinear equality constraints, an additional projection from the tangent space to the manifold is required, but since they are equal for linear constraints, this is not required for the purpose of the considered limited control problem.

Given a matrix $\mathbf{T} \in \mathbb{R}^{m\times d}$, where the columns form a basis of $\mathcal{U}$, every $\tilde{\mathbf{u}} \in \mathcal{U}$ can be written as $\mathbf{T}\tilde{\mathbf{z}}$ with $\tilde{\mathbf{z}} \in \mathbb{R}^d$. This allows to finally reduce the problem to an unconstrained quadratic program given by

$$\min_{\tilde{\mathbf{z}}\in\mathbb{R}^d} \quad \left\{\frac{1}{2}\tilde{\mathbf{z}}^{\mathrm{T}}\mathbf{T}^{\mathrm{T}}\mathbf{V}_{\mathbf{uu}}\mathbf{T}\tilde{\mathbf{z}} + \tilde{\mathbf{z}}^{\mathrm{T}}\mathbf{T}^{\mathrm{T}}(\mathbf{V}_{\mathbf{xu}}^{\mathrm{T}}\tilde{\mathbf{x}} + \mathbf{v}_{\mathbf{u}})\right\}\ . \tag{2.22}$$

Note that for this special problem, we can choose $\mathbf{T} = (\mathbf{e}_j)_{j\in\{1,\dots,m\}\setminus\mathcal{A}}$ where $\mathbf{e}_j$ has a 1 in its j-th component and zeros else. From $\tilde{\mathbf{u}} = \mathbf{T}\tilde{\mathbf{z}}$, the update rule from (2.9) changes to

$$\tilde{\mathbf{u}} = \mathbf{K}\tilde{\mathbf{x}} + \alpha\mathbf{d}\ , \tag{2.23a}$$

with

$$\mathbf{K} = \mathbf{T}(\mathbf{T}^{\mathrm{T}}\mathbf{V}_{\mathbf{uu}}\mathbf{T})^{-1}\mathbf{T}^{\mathrm{T}}\mathbf{V}_{\mathbf{xu}} \qquad \text{and} \qquad \mathbf{d} = \mathbf{T}(\mathbf{T}^{\mathrm{T}}\mathbf{V}_{\mathbf{uu}}\mathbf{T})^{-1}\mathbf{T}^{\mathrm{T}}\mathbf{v}_{\mathbf{u}} \tag{2.23b}$$

and the step size $\alpha > 0$. This control update is then used to perform the projected update

$$\mathbf{u} \longleftarrow \max(\min(\overline{\mathbf{u}}, \mathbf{u} + \tilde{\mathbf{u}}), \underline{\mathbf{u}})\ . \tag{2.24}$$

2.2.2 Augmented Lagrangian method

The *Augmented Lagrangian* method is similar to the penalty method, but additionally tries to include an estimate of the Lagrange multipliers $\boldsymbol{\lambda}_k \in \mathbb{R}^q$, $k = 0, \dots, N-1$. As discussed in [32], this avoids the ill-conditioning of the pure penalty method, which is that convergence is only guaranteed if the penalties $\mu \in \mathbb{R}_{\geq 0}$ go to infinity. Among others, it is used in [22, 28] for solving constrained iLQR in different variants.

The main idea of augmented Lagrangian methods is to minimize the constrained optimiziation problem (2.17) by minimizing the so called *augmented Lagrangian*

$$L_A(\mathbf{u}_0, \dots, \mathbf{u}_{N-1}, \boldsymbol{\lambda}_0, \dots, \boldsymbol{\lambda}_{N-1}, \mu) = J(\mathbf{u}_0, \dots, \mathbf{u}_{N-1}) + \sum_{k=0}^{N-1}\sum_{i=1}^{q} p(h_i(\mathbf{x}_k, \mathbf{u}_k), \lambda_{i,k}, \mu) \tag{2.25}$$

where $\boldsymbol{\lambda}_k = [\lambda_{1,k}, \dots, \lambda_{q,k}]^\mathrm{T} \in \mathbb{R}^q$, $k = 0, \dots, N-1$ are the Lagrange multipliers, $\mu \in \mathbb{R}_{\geq 0}$ is the penalty weight and $p : \mathbb{R} \times \mathbb{R}_{\geq 0} \times \mathbb{R}_{\geq 0} \to \mathbb{R}$ is a suitable *penalty-Lagrangian function*, c.f. [29].

For fixed Lagrange multipliers $\boldsymbol{\lambda}_k \in \mathbb{R}^q$, $k = 0, \dots, N-1$ and a fixed penalty weight $\mu \in \mathbb{R}_{\geq 0}$, the resulting unconstrained optimal control problem (2.25) is first minimized w.r.t. $\mathbf{u}_0, \dots, \mathbf{u}_{N-1} \in \mathbb{R}^m$, which yields a solution $\mathbf{u}_0^*, \dots, \mathbf{u}_{N-1}^* \in \mathbb{R}^m$ and the corresponding state sequence $\mathbf{x}_0^*, \dots, \mathbf{x}_N^* \in \mathbb{R}^n$. Then the Lagrange multipliers and the penalty weight are adapted according to

$$\lambda_{i,k} \leftarrow \frac{\partial p(h_i(\mathbf{x}_k^*, \mathbf{u}_k^*), \lambda_{i,k}, \mu)}{\partial h_i} \tag{2.26a}$$

$$\mu \leftarrow \phi\mu \tag{2.26b}$$

where $\phi > 1$ is a constant scaling parameter. This process is repeated until all constraints are satisfied. In [32], the author strongly suggests to select ϕ not much larger than one. Otherwise the penalty weights increase too fast and ill-conditioning, similar to pure penalty methods, might occur. This is why the algorithm is implemented with $\phi = 1.5$ for this thesis. The Lagrange multipliers are initialized with $\boldsymbol{\lambda}_k = \mathbf{0}$, $k = 0, \dots, N-1$, since constraints would be treated as being active otherwise.

In order to solve the unconstrained optimal control problem (2.25) using iLQR, (2.2) is plugged into (2.25), which yields

$$L_A = \varphi(\mathbf{x}_N) + \sum_{k=0}^{N-1} \underbrace{\left(l(\mathbf{x}_k, \mathbf{u}_k) + \sum_{i=1}^{q} p(h_i(\mathbf{x}_k, \mathbf{u}_k), \boldsymbol{\lambda}_k, \mu) \right)}_{\bar{l}(\mathbf{x}_k, \mathbf{u}_k)} . \tag{2.27}$$

A quadratic approximation (2.5) of (2.27) is computed using the linear approximation of the inequality constraints

$$\mathbf{h}(\mathbf{x}_k + \tilde{\mathbf{x}}_k, \mathbf{u}_k + \tilde{\mathbf{u}}_k) \approx \mathbf{h}(\mathbf{x}_k, \mathbf{u}_k) + \mathbf{H}_{\mathbf{x},k}^\mathrm{T} \tilde{\mathbf{x}}_k + \mathbf{H}_{\mathbf{u},k}^\mathrm{T} \tilde{\mathbf{u}}_k \tag{2.28a}$$

with

$$\mathbf{H}_{\mathbf{x},k} := \left(\frac{\partial \mathbf{h}(\mathbf{x}, \mathbf{u})}{\partial \mathbf{x}} \Big|_{\substack{\mathbf{x}=\mathbf{x}_k \\ \mathbf{u}=\mathbf{u}_k}} \right)^\mathrm{T} \quad \text{and} \quad \mathbf{H}_{\mathbf{u},k} := \left(\frac{\partial \mathbf{h}(\mathbf{x}, \mathbf{u})}{\partial \mathbf{u}} \Big|_{\substack{\mathbf{x}=\mathbf{x}_k \\ \mathbf{u}=\mathbf{u}_k}} \right)^\mathrm{T} . \tag{2.28b}$$

Hence, instead of using (2.7) to compute the quadratic approximation (2.5), the following quantities are used

$$\bar{\mathbf{v}}_{\mathbf{x},k} = \mathbf{v}_{\mathbf{x},k} + \mathbf{H}_{\mathbf{x},k} \mathbf{p}_{\mathbf{h},k} \tag{2.29a}$$

$$\bar{\mathbf{v}}_{\mathbf{u},k} = \mathbf{v}_{\mathbf{u},k} + \mathbf{H}_{\mathbf{u},k} \mathbf{p}_{\mathbf{h},k} \tag{2.29b}$$

$$\bar{\mathbf{V}}_{\mathbf{xx},k} = \mathbf{V}_{\mathbf{xx},k} + \mathbf{H}_{\mathbf{x},k} \mathbf{P}_{\mathbf{hh},k} \mathbf{H}_{\mathbf{x},k}^\mathrm{T} \tag{2.29c}$$

$$\bar{\mathbf{V}}_{\mathbf{uu},k} = \mathbf{V}_{\mathbf{uu},k} + \mathbf{H}_{\mathbf{u},k} \mathbf{P}_{\mathbf{hh},k} \mathbf{H}_{\mathbf{u},k}^\mathrm{T} \tag{2.29d}$$

$$\bar{\mathbf{V}}_{\mathbf{ux},k} = \mathbf{V}_{\mathbf{ux},k} + \mathbf{H}_{\mathbf{u},k} \mathbf{P}_{\mathbf{hh},k} \mathbf{H}_{\mathbf{x},k}^\mathrm{T} , \tag{2.29e}$$

where $\mathbf{v}_{\mathbf{x},k}$, $\mathbf{v}_{\mathbf{u},k}$, $\mathbf{V}_{\mathbf{xx},k}$, $\mathbf{V}_{\mathbf{uu},k}$, $\mathbf{V}_{\mathbf{ux},\mathbf{k}}$ are computed using (2.7) and $\mathbf{p}_{\mathbf{h},k} = \left[\frac{\partial p}{\partial h_1}, \dots, \frac{\partial p}{\partial h_q}\right]^{\mathrm{T}}$ as well as $\mathbf{P}_{\mathbf{hh},k} = \operatorname{diag}\left(\frac{\partial^2 p}{\partial h_1^2}, \dots, \frac{\partial^2 p}{\partial h_q^2}\right)$.

For the penalty-Lagrangian function p, the following methods are discussed in this thesis.

1. The first method developed to solve constrained optimization problems is the well known *Powell-Hestenes-Rockafellar* (PHR) approach, which is originally proposed in [24, 25]. The penalty-Lagrangian function is chosen as

$$p(h, \lambda, \mu) = \frac{1}{2\mu}\Big(\max(0, \lambda + \mu h)^2 - \lambda^2\Big) = \begin{cases} \lambda h + \frac{1}{2}\mu h^2 \ , & \lambda + \mu h > 0 \\ -\frac{\lambda^2}{2\mu} \ , & \text{otherwise} \ . \end{cases} \tag{2.30}$$

   The corresponding first and second derivatives are

$$\frac{\partial p(h, \lambda, \mu)}{\partial h} = \begin{cases} \lambda + \mu h \ , & \lambda + \mu h > 0 \\ 0 \ , & \text{otherwise} \end{cases} \tag{2.31}$$

   and

$$\frac{\partial^2 p(h, \lambda, \mu)}{\partial h^2} = \begin{cases} \mu \ , & \lambda + \mu h > 0 \\ 0 \ , & \text{otherwise} \ . \end{cases} \tag{2.32}$$

   Hence, the algorithm treats the constraint $h_i(\mathbf{x}_k, \mathbf{u}_k)$ as active and thus gets penalized, if $\lambda_{i,k} + \mu h_i(\mathbf{x}_k, \mathbf{u}_k) > 0$. Otherwise it is set inactive and is not penalized.

2. One drawback of the standard PHR approach is that the second derivative (2.32) is not continuous, i.e. p is not twice continuously differentiable. This may yield to a bad performance of second-order methods, c.f. [28, 29]. A smooth approximation of the PHR (S-PHR) method is proposed in [29] and used in combination with iLQR in [28]. The penalty-Lagrangian function is given by

$$p(h, \lambda, \mu) = \begin{cases} \lambda h + \frac{1}{2}\mu h^2 \ , & h \geq -\frac{1}{2}\frac{\lambda}{\mu} \\ -\frac{\lambda^2}{\mu}\left(\frac{1}{4}\log\left(-2\frac{\mu}{\lambda}h\right) + \frac{3}{8}\right) \ , & \text{otherwise} \ . \end{cases} \tag{2.33}$$

   The corresponding first and second derivatives are

$$\frac{\partial p(h, \lambda, \mu)}{\partial h} = \begin{cases} \lambda + \mu h \ , & h \geq -\frac{1}{2}\frac{\lambda}{\mu} \\ -\frac{1}{4}\frac{\lambda^2}{\mu}\frac{1}{h} & \text{otherwise} \end{cases} \tag{2.34}$$

   and

$$\frac{\partial^2 p(h, \lambda, \mu)}{\partial h^2} = \begin{cases} \mu & h \geq -\frac{1}{2}\frac{\lambda}{\mu} \\ \frac{1}{4}\frac{\lambda^2}{\mu}\frac{1}{h^2} & \text{otherwise} \ . \end{cases} \tag{2.35}$$

   Plugging $h = -\frac{1}{2}\frac{\lambda}{\mu}$ into (2.35) shows that p is twice continuously differentiable in h. However, note that p in (2.33) is not defined for $\lambda = 0$. Hence, as in [29], the Lagrange parameter is limited to $\lambda \geq \underline{\lambda}$ where $\underline{\lambda} = 10^{-6}$ is chosen.

3. The approach used in ALTRO [22] is a slight adaption of the standard PHR approach. The penalty-Lagrangian function is chosen as

$$p(h, \lambda, \mu) = \begin{cases} \lambda h + \frac{1}{2}\mu^2 h \ , & h \geq 0 \vee \lambda > 0 \\ 0 \ , & \text{otherwise} \ . \end{cases} \tag{2.36}$$

The corresponding first and second derivatives are

$$\frac{\partial p(h, \lambda, \mu)}{\partial h} = \begin{cases} \lambda + \mu h \ , & h \geq 0 \vee \lambda > 0 \\ 0 & \text{otherwise} \end{cases} \tag{2.37}$$

and

$$\frac{\partial^2 p(h, \lambda, \mu)}{\partial h^2} = \begin{cases} \mu \ , & h \geq 0 \vee \lambda > 0 \\ 0 \ , & \text{otherwise} \ . \end{cases} \tag{2.38}$$

Additionally, ALTRO uses a slightly modified update rule for the Lagrange multipliers than (2.26), i.e.

$$\boldsymbol{\lambda}_k = \max\left(\mathbf{0}, \lambda_k + \mu \mathbf{h}(\mathbf{x}_k^*, \mathbf{u}_k^*)\right) \ , \qquad k = 1, \ldots, N-1 \ . \tag{2.39}$$

Hence, a constraint $h_i(\mathbf{x}_k, \mathbf{u}_k)$ is treated as active and thus gets penalized, if $h_i(\mathbf{x}_k, \mathbf{u}_k) \geq 0$ or $\lambda_{i,k} > 0$. Otherwise it is inactive and not penalized.

The augmented Lagrangian based iLQR algorithm is summarized in Algorithm 2.

Algorithm 2 AL-iLQR

Require: Initial control sequence $(\mathbf{u}_k)_{k\in\{0,\ldots,N-1\}}$, corresponding state trajectory $(\mathbf{x}_k)_{k\in\{0,\ldots,N\}}$, initial penalty factor μ, scaling parameter ϕ and initial Lagrange multipliers $(\boldsymbol{\lambda}_k)_{k\in\{0,\ldots,N-1\}}$.

while $\neg$ converged **do**
  Solve unconstrained iLQR using (2.29)
  Update $\boldsymbol{\lambda}_k$, $k = 0, \ldots, N-1$ and μ with the appropriate update rule
end while

2.2.3 Primal-Dual Interior Point

In this section, the *Primal-Dual Interior Point* based iLQR method from [30] is presented. Two variants of this algorithm are given in [30]. The first directly solves the dual problem of the constrained optimal control problem (2.17). However, this approach suffers from the requirement of feasible control and state trajectories that fulfill the inequality constraints during the whole optimization process. Hence, it is not possible to initialize the algorithm with infeasible trajectories, which might be desired in e.g. MPC applications, where the solution of the previous time horizon could be reused for the next time horizon, but, e.g., due to a model-plant mismatch it might not be feasible. Hence, another formulation provided in [30] uses slack variables in order to allow for initialization with infeasible trajectories.

Feasible Primal-Dual Interior Point

Consider the constrained optimal control problem (2.17). It can be shown, that a local solution to (2.17) is equal to a local solution of the *dual problem*

$$\min_{\mathbf{u}_0,\dots,\mathbf{u}_{N-1}\in\mathbb{R}^m} \max_{\boldsymbol{\lambda}_0,\dots,\boldsymbol{\lambda}_{N-1}\in\mathbb{R}^q} \quad \Big\{\varphi(\mathbf{x}_N) + \sum_{k=0}^{N-1} \bar{l}(\mathbf{x}_k,\mathbf{u}_k,\boldsymbol{\lambda}_k)\Big\} \tag{2.40a}$$

$$\text{s.t.} \quad \mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k,\mathbf{u}_k) \tag{2.40b}$$

$$\boldsymbol{\lambda}_k \geq 0, \qquad k = 0,\dots,N-1\ , \tag{2.40c}$$

with

$$\bar{l}(\mathbf{x}_k,\mathbf{u}_k,\boldsymbol{\lambda}_k) = l(\mathbf{x}_k,\mathbf{u}_k) + \boldsymbol{\lambda}_k^{\mathrm{T}}\mathbf{h}(\mathbf{x}_k,\mathbf{u}_k)\ , \tag{2.41}$$

provided some convexity assumptions and the LICQ condition hold, c.f. [26].

As in the derivation of the unconstrained iLQR in Section 2.1, *Bellman's principle of optimality* yields

$$\bar{V}_k^*(\mathbf{x}_k) = \min_{\mathbf{u}_k\in\mathbb{R}^m} \max_{\boldsymbol{\lambda}_k\in\mathbb{R}^q} \Big\{\bar{l}(\mathbf{x}_k,\mathbf{u}_k,\boldsymbol{\lambda}_k) + \bar{V}_{k+1}^*(\mathbf{f}(\mathbf{x}_k,\mathbf{u}_k))\Big\} \quad \text{with} \quad \bar{V}_N(\mathbf{x}_N) = \varphi(\mathbf{x}_N) \tag{2.42}$$

for $k = 1,\dots,N-1$. Analogous to Section 2.1, the perturbations $\tilde{\mathbf{x}}_k \in \mathbb{R}^n$, $\tilde{\mathbf{u}}_k \in \mathbb{R}^m$ and $\tilde{\boldsymbol{\lambda}}_k \in \mathbb{R}^q$ around $\mathbf{x}_k$, $\mathbf{u}_k$ and $\boldsymbol{\lambda}_k$, $k = 1,\dots,N-1$ allow for the approximation

$$\bar{V}_k(\mathbf{x}_k+\tilde{\mathbf{x}}_k,\mathbf{u}_k+\tilde{\mathbf{u}}_k,\boldsymbol{\lambda}_k+\tilde{\boldsymbol{\lambda}}_k) \approx \bar{V}_k(\mathbf{x}_k,\mathbf{u}_k,\lambda_\mathbf{k}) + \tilde{\bar{V}}_k(\tilde{\mathbf{x}}_k,\tilde{\mathbf{u}}_k,\tilde{\boldsymbol{\lambda}}_k)\ , \tag{2.43}$$

with

$$\begin{aligned}\tilde{\bar{V}}_k(\tilde{\mathbf{x}}_k,\tilde{\mathbf{u}}_k,\tilde{\boldsymbol{\lambda}}_k) := {}& \tilde{\mathbf{x}}_k^{\mathrm{T}}\bar{\mathbf{v}}_{\mathbf{x},k} + \tilde{\mathbf{u}}_k^{\mathrm{T}}\bar{\mathbf{v}}_{\mathbf{u},k} + \tilde{\boldsymbol{\lambda}}_k^{\mathrm{T}}\bar{\mathbf{v}}_{\boldsymbol{\lambda},k} + \frac{1}{2}\tilde{\mathbf{x}}_k^{\mathrm{T}}\bar{\mathbf{V}}_{\mathbf{xx},k}\tilde{\mathbf{x}}_k + \frac{1}{2}\tilde{\mathbf{u}}_k^{\mathrm{T}}\bar{\mathbf{V}}_{\mathbf{uu},k}\tilde{\mathbf{u}}_k \\ & + \frac{1}{2}\tilde{\boldsymbol{\lambda}}_k^{\mathrm{T}}\bar{\mathbf{V}}_{\boldsymbol{\lambda\lambda},k}\tilde{\boldsymbol{\lambda}}_k + \tilde{\mathbf{x}}_k^{\mathrm{T}}\bar{\mathbf{V}}_{\mathbf{xu},k}\tilde{\mathbf{u}}_k + \tilde{\mathbf{x}}_k^{\mathrm{T}}\bar{\mathbf{V}}_{\mathbf{x\lambda},k}\tilde{\boldsymbol{\lambda}}_k + \tilde{\mathbf{u}}_k^{\mathrm{T}}\bar{\mathbf{V}}_{\mathbf{u\lambda},k}\tilde{\boldsymbol{\lambda}}_k\ .\end{aligned} \tag{2.44}$$

Using the linear approximations (2.6) and (2.28) yields

$$\bar{\mathbf{v}}_{\mathbf{x},k} = \mathbf{v}_{\mathbf{x},k} + \mathbf{H}_{\mathbf{x},k}\boldsymbol{\lambda}_k \qquad \bar{\mathbf{V}}_{\mathbf{xx},k} = \mathbf{V}_{\mathbf{xx},k} \tag{2.45a}$$

$$\bar{\mathbf{v}}_{\mathbf{u},k} = \mathbf{v}_{\mathbf{u},k} + \mathbf{H}_{\mathbf{u},k}\boldsymbol{\lambda}_k \qquad \bar{\mathbf{V}}_{\mathbf{uu},k} = \mathbf{V}_{\mathbf{uu},k} \tag{2.45b}$$

$$\bar{\mathbf{v}}_{\boldsymbol{\lambda},k} = \mathbf{h}(\tilde{\mathbf{x}}_k,\tilde{\mathbf{u}}_k) \qquad \bar{\mathbf{V}}_{\boldsymbol{\lambda\lambda},k} = \mathbf{0} \tag{2.45c}$$

$$\bar{\mathbf{V}}_{\mathbf{x\lambda},k} = \mathbf{H}_{\mathbf{x},k} \qquad \bar{\mathbf{V}}_{\mathbf{u\lambda},k} = \mathbf{H}_{\mathbf{u},k} \tag{2.45d}$$

$$\bar{\mathbf{V}}_{\mathbf{ux},k} = \mathbf{V}_{\mathbf{ux},k}\ , \tag{2.45e}$$

where $\mathbf{v}_{\mathbf{x},k}$, $\mathbf{v}_{\mathbf{u},k}$, $\mathbf{V}_{\mathbf{xx},k}$, $\mathbf{V}_{\mathbf{uu},k}$ and $\mathbf{V}_{\mathbf{xu},k}$ are given in (2.7). Minimizing (2.44) w.r.t. $\tilde{\mathbf{u}}_k \in \mathbb{R}^m$ results in one first-order necessary condition

$$\bar{\mathbf{V}}_{\mathbf{uu},k}\tilde{\mathbf{u}}_k^* + \bar{\mathbf{V}}_{\mathbf{xu},k}^{\mathrm{T}}\tilde{\mathbf{x}}_k + \bar{\mathbf{V}}_{\mathbf{u\lambda},k}\tilde{\boldsymbol{\lambda}}_k + \bar{\mathbf{v}}_{\mathbf{u},k} \stackrel{!}{=} \mathbf{0}\ . \tag{2.46}$$

Additionally, (2.44) has to be maximized w.r.t. $\tilde{\boldsymbol{\lambda}}_k \in \mathbb{R}^q$ s.t. $\boldsymbol{\lambda}_k + \tilde{\boldsymbol{\lambda}}_k \geq 0$. This yields another first-order necessary condition, c.f. [30]

$$\operatorname{diag}(\boldsymbol{\lambda}_k + \tilde{\boldsymbol{\lambda}}_k^*)\left(\bar{\mathbf{v}}_{\boldsymbol{\lambda},k} + \bar{\mathbf{V}}_{\mathbf{x}\boldsymbol{\lambda},k}^{\mathrm{T}}\tilde{\mathbf{x}}_k + \bar{\mathbf{V}}_{\mathbf{u}\boldsymbol{\lambda},k}^{\mathrm{T}}\tilde{\mathbf{u}}_k\right) \stackrel{!}{=} \mathbf{0} \ . \tag{2.47}$$

Combining (2.46) and (2.47) and adding the perturbation vector $\boldsymbol{\eta} = [\eta, \dots, \eta]^{\mathrm{T}} \in \mathbb{R}^q$ with $\eta \in \mathbb{R}_{>0}$ to the left-hand side of (2.47) brings about

$$\begin{bmatrix} \bar{\mathbf{V}}_{\mathbf{uu},k} & \mathbf{H}_{\mathbf{u},k} \\ \boldsymbol{\Lambda}_k\mathbf{H}_{\mathbf{u},k}^{\mathrm{T}} & \mathbf{H}_k \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{u}}_k \\ \tilde{\boldsymbol{\lambda}}_k \end{bmatrix} = -\begin{bmatrix} \bar{\mathbf{v}}_{\mathbf{u},k} \\ \mathbf{r}_k \end{bmatrix} - \begin{bmatrix} \bar{\mathbf{V}}_{\mathbf{xu},k}^{\mathrm{T}} \\ \boldsymbol{\Lambda}_k\mathbf{H}_{\mathbf{x},k}^{\mathrm{T}} \end{bmatrix} \tilde{\mathbf{x}}_k \ , \tag{2.48}$$

where $\boldsymbol{\Lambda}_k := \operatorname{diag}(\boldsymbol{\lambda}_k)$, $\mathbf{H}_k := \operatorname{diag}(\mathbf{h}(\mathbf{x}_k, \mathbf{u}_k))$ and $\mathbf{r}_k = \boldsymbol{\Lambda}_k\mathbf{h}(\mathbf{x}_k, \mathbf{u}_k) + \boldsymbol{\eta}$. The second order terms in (2.47) are neglected. The perturbation vector $\boldsymbol{\eta}$ aims at penalizing points near the inequality constraints. It can be interpreted as weighting of a barrier function applied to the inequalities, which is why it is also called *barrier parameter*, c.f. [26]. Hence, $\boldsymbol{\eta}$ together with a proper step size selection strategy asserts that all inequalities are fulfilled during optimization, which justifies the name *interior point* method.

Solving for $\tilde{\boldsymbol{\lambda}}_k \in \mathbb{R}^q$ in the second row of (2.48) yields

$$\tilde{\boldsymbol{\lambda}}_k = -\mathbf{H}_k^{-1}\left(\boldsymbol{\Lambda}_k\mathbf{H}_{\mathbf{u},k}^{\mathrm{T}}\tilde{\mathbf{u}}_k + \boldsymbol{\Lambda}_k\mathbf{H}_{\mathbf{x},k}^{\mathrm{T}}\tilde{\mathbf{x}}_k + \mathbf{r}_k\right) \ . \tag{2.49}$$

Plugging the result (2.49) for $\tilde{\boldsymbol{\lambda}}_k$ in the first row of (2.48) yields the update rule

$$\tilde{\mathbf{u}}_k = \mathbf{K}_k\tilde{\mathbf{x}}_k + \mathbf{d}_k \ , \tag{2.50}$$

with

$$\mathbf{K}_k = -\left(\bar{\mathbf{V}}_{\mathbf{uu},k} - \mathbf{H}_{\mathbf{u},k}\mathbf{H}_k^{-1}\boldsymbol{\Lambda}_k\mathbf{H}_{\mathbf{u},k}^{\mathrm{T}}\right)^{-1}\left(\bar{\mathbf{V}}_{\mathbf{xu},k}^{\mathrm{T}} - \mathbf{H}_{\mathbf{u},k}\mathbf{H}_k^{-1}\boldsymbol{\Lambda}_k\mathbf{H}_{\mathbf{x},k}^{\mathrm{T}}\right) \tag{2.51a}$$

$$\mathbf{d}_k = -\left(\bar{\mathbf{V}}_{\mathbf{uu},k} - \mathbf{H}_{\mathbf{u},k}\mathbf{H}_k^{-1}\boldsymbol{\Lambda}_k\mathbf{H}_{\mathbf{u},k}^{\mathrm{T}}\right)^{-1}\left(\bar{\mathbf{v}}_{\mathbf{u},k} - \mathbf{H}_{\mathbf{u},k}\mathbf{H}_k^{-1}\mathbf{r}_k\right) \ . \tag{2.51b}$$

Similarly, substituting $\tilde{\mathbf{u}}_k$ in (2.49) with (2.50) yields

$$\tilde{\boldsymbol{\lambda}}_k = \boldsymbol{\Psi}_k\tilde{\mathbf{x}}_k + \boldsymbol{\psi}_k \tag{2.52}$$

with

$$\boldsymbol{\Psi}_k = -\mathbf{H}_k^{-1}\boldsymbol{\Lambda}_k\left(\mathbf{H}_{\mathbf{x},k}^{\mathrm{T}} + \mathbf{H}_{\mathbf{u},k}^{\mathrm{T}}\mathbf{K}_k\right) \tag{2.53a}$$

$$\boldsymbol{\psi}_k = -\mathbf{H}_k^{-1}\left(\mathbf{r}_k + \boldsymbol{\Lambda}_k\mathbf{H}_{\mathbf{u},k}^{\mathrm{T}}\mathbf{d}_k\right) \ . \tag{2.53b}$$

Back-substitution of (2.50) and (2.52) into (2.44) yields

$$\mathbf{s}_k = \hat{\mathbf{v}}_{\mathbf{x}_k} + \mathbf{K}_k^{\mathrm{T}}\hat{\mathbf{V}}_{\mathbf{uu},k}\mathbf{d}_k + \mathbf{K}_k^{\mathrm{T}}\hat{\mathbf{v}}_{\mathbf{u},k} + \hat{\mathbf{V}}_{\mathbf{xu},k}\mathbf{d}_k \tag{2.54a}$$

$$\mathbf{S}_k = \hat{\mathbf{V}}_{\mathbf{xx},k} + \mathbf{K}_k^{\mathrm{T}}\hat{\mathbf{V}}_{\mathbf{uu},k}\mathbf{K}_k + \mathbf{K}_k^{\mathrm{T}}\hat{\mathbf{V}}_{\mathbf{xu},k}^{\mathrm{T}} + \hat{\mathbf{V}}_{\mathbf{xu},k}\mathbf{K}_k \ , \tag{2.54b}$$

where

$$\hat{\mathbf{v}}_{\mathbf{x},k} = \bar{\mathbf{v}}_{\mathbf{x},k} - \mathbf{H}_{\mathbf{x},k}\mathbf{H}_k^{-1}\mathbf{r}_k \qquad \hat{\mathbf{V}}_{\mathbf{xx},k} = \bar{\mathbf{V}}_{\mathbf{xx},k} - \mathbf{H}_{\mathbf{x},k}\mathbf{H}_k^{-1}\mathbf{\Lambda}_k\mathbf{H}_{\mathbf{x},k}^{\mathrm{T}} \tag{2.55a}$$

$$\hat{\mathbf{v}}_{\mathbf{u},k} = \bar{\mathbf{v}}_{\mathbf{x},k} - \mathbf{H}_{\mathbf{u},k}\mathbf{H}_k^{-1}\mathbf{r}_k \qquad \hat{\mathbf{V}}_{\mathbf{uu},k} = \bar{\mathbf{V}}_{\mathbf{uu},k} - \mathbf{H}_{\mathbf{u},k}\mathbf{H}_k^{-1}\mathbf{\Lambda}_k\mathbf{H}_{\mathbf{u},k}^{\mathrm{T}} \tag{2.55b}$$

$$\hat{\mathbf{V}}_{\mathbf{xu},k} = \bar{\mathbf{V}}_{\mathbf{xu},k} - \mathbf{H}_{\mathbf{x},k}\mathbf{H}_k^{-1}\mathbf{\Lambda}_k\mathbf{H}_{\mathbf{u},k}^{\mathrm{T}} \ . \tag{2.55c}$$

Following [30], the perturbation η is initialized using $\eta = \frac{J(\mathbf{u}_0,\ldots,\mathbf{u}_{N-1})}{Nq}$. The step size α is selected s.t. the inequalities $\mathbf{h}(\mathbf{x}_k, \mathbf{u}_k) \leq 0$ and $\boldsymbol{\lambda}_k \geq 0$, $k = 0, \ldots, N-1$ are fulfilled in all iterations.

Infeasible Primal-Dual Interior Point

In order to allow initialization of the algorithm with infeasible trajectories, [30] reformulated (2.17) by slack variables $\boldsymbol{\zeta}_k \in \mathbb{R}^q$, $k = 1, \ldots, N-1$, i.e.

$$\min_{\substack{\mathbf{u}_0,\ldots,\mathbf{u}_{N-1}\in\mathbb{R}^m \\ \boldsymbol{\zeta}_0,\ldots,\boldsymbol{\zeta}_{N-1}\in\mathbb{R}^q}} \quad J(\mathbf{u}_0, \ldots, \mathbf{u}_{N-1}) \tag{2.56a}$$

$$\text{s.t.} \quad \mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k) \tag{2.56b}$$

$$\mathbf{g}(\mathbf{x}_k, \mathbf{u}_k, \boldsymbol{\zeta}_k) := \mathbf{h}(\mathbf{x}_k, \mathbf{u}_k) + \boldsymbol{\zeta}_k = \mathbf{0} \tag{2.56c}$$

$$\boldsymbol{\zeta}_k \geq \mathbf{0} \ . \tag{2.56d}$$

Similar to the derivation of the feasible Primal-Dual Interior Point algorithm, the dual problem can be used to derive the system of linear equations

$$\begin{bmatrix} \bar{\mathbf{V}}_{\mathbf{uu},k} & \mathbf{H}_{\mathbf{u},k} & \mathbf{0} \\ \mathbf{H}_{\mathbf{u},k}^{\mathrm{T}} & \mathbf{0} & \mathbf{I} \\ \mathbf{0} & \mathbf{Z}_k & \mathbf{\Lambda}_k \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{u}}_k \\ \tilde{\boldsymbol{\lambda}}_k \\ \tilde{\boldsymbol{\zeta}}_k \end{bmatrix} = - \begin{bmatrix} \bar{\mathbf{v}}_{\mathbf{u},k} \\ \mathbf{g}(\mathbf{x}_k, \mathbf{u}_k, \boldsymbol{\zeta}_k) \\ \mathbf{r}_k \end{bmatrix} - \begin{bmatrix} \bar{\mathbf{V}}_{\mathbf{xu},k}^{\mathrm{T}} \\ \mathbf{H}_{\mathbf{x},k}^{\mathrm{T}} \\ \mathbf{0} \end{bmatrix} \tilde{\mathbf{x}}_k \ , \tag{2.57}$$

with $\mathbf{r}_k := \mathbf{\Lambda}_k\boldsymbol{\zeta}_k + \boldsymbol{\eta}$, $\mathbf{Z}_k := \mathrm{diag}(\boldsymbol{\zeta}_k)$, $\mathbf{\Lambda}_k := \mathrm{diag}(\boldsymbol{\lambda}_k)$ and $\mathbf{I}$ being the identity matrix, c.f. [30]. Again, $\boldsymbol{\eta} \in \mathbb{R}^q$ is the perturbation vector and $\tilde{\boldsymbol{\lambda}}_k, \tilde{\boldsymbol{\zeta}}_k \in \mathbb{R}^q$, $\tilde{\mathbf{x}}_k \in \mathbb{R}^n$ as well as $\tilde{\mathbf{u}}_k \in \mathbb{R}^m$ are perturbations around the actual values $\boldsymbol{\lambda}_k, \boldsymbol{\zeta}_k \in \mathbb{R}^q$, $\mathbf{x}_k \in \mathbb{R}^n$ and $\mathbf{u}_k \in \mathbb{R}^m$. This brings about

$$\tilde{\mathbf{u}}_k = \mathbf{K}_k\tilde{\mathbf{x}}_k + \mathbf{d}_k \tag{2.58a}$$

$$\tilde{\boldsymbol{\lambda}}_k = \mathbf{\Psi}_k\tilde{\mathbf{x}}_k + \boldsymbol{\psi}_k \tag{2.58b}$$

$$\tilde{\boldsymbol{\zeta}}_k = \mathbf{\Phi}_k\tilde{\mathbf{x}}_k + \boldsymbol{\phi}_k \ , \tag{2.58c}$$

where

$$\mathbf{K}_k = \left(\bar{\mathbf{V}}_{\mathbf{uu},k} + \mathbf{H}_{\mathbf{u},k}\mathbf{Z}_k^{-1}\mathbf{\Lambda}_k\mathbf{H}_{\mathbf{u},k}^{\mathrm{T}}\right)^{-1}\left(\bar{\mathbf{v}}_{\mathbf{u},k} + \mathbf{H}_{\mathbf{u},k}\mathbf{Z}_k^{-1}\mathbf{r}_k\right) \tag{2.59a}$$

$$\mathbf{d}_k = \left(\bar{\mathbf{V}}_{\mathbf{uu},k} + \mathbf{H}_{\mathbf{u},k}\mathbf{Z}_k^{-1}\mathbf{\Lambda}_k\mathbf{H}_{\mathbf{u},k}^{\mathrm{T}}\right)^{-1}\left(\bar{\mathbf{V}}_{\mathbf{xu},k}^{\mathrm{T}} + \mathbf{H}_{\mathbf{u},k}\mathbf{Z}_k^{-1}\mathbf{\Lambda}\mathbf{H}_{\mathbf{x},k}^{\mathrm{T}}\right) \tag{2.59b}$$

$$\mathbf{\Psi}_k = \mathbf{Z}_k^{-1}\mathbf{\Lambda}_k\left(\mathbf{H}_{\mathbf{x},k}^{\mathrm{T}} + \mathbf{H}_{\mathbf{u},k}^{\mathrm{T}}\mathbf{K}_k\right) \tag{2.59c}$$

$$\boldsymbol{\psi}_k = \mathbf{Z}_k^{-1}\left(\mathbf{r}_k + \mathbf{\Lambda}_k\mathbf{H}_{\mathbf{u},k}^{\mathrm{T}}\mathbf{d}_k\right) \tag{2.59d}$$

$$\mathbf{\Phi}_k = -\mathbf{H}_{\mathbf{x},k}^{\mathrm{T}} - \mathbf{H}_{\mathbf{u},k}^{\mathrm{T}}\mathbf{K}_k \tag{2.59e}$$

$$\boldsymbol{\phi}_k = -\mathbf{g}(\mathbf{x}_k, \mathbf{u}_k, \boldsymbol{\zeta}_k) - \mathbf{H}_{\mathbf{u},k}^{\mathrm{T}}\mathbf{d}_k \ . \tag{2.59f}$$

Back-substitution of (2.58) into (2.59) yields

$$\mathbf{s}_k = \hat{\mathbf{v}}_{\mathbf{x}_k} + \mathbf{K}_k^{\mathrm{T}}\hat{\mathbf{V}}_{\mathbf{uu},k}\mathbf{d}_k + \mathbf{K}_k^{\mathrm{T}}\hat{\mathbf{v}}_{\mathbf{u},k} + \hat{\mathbf{V}}_{\mathbf{xu},k}\mathbf{d}_k \tag{2.60a}$$

$$\mathbf{S}_k = \hat{\mathbf{V}}_{\mathbf{xx},k} + \mathbf{K}_k^{\mathrm{T}}\hat{\mathbf{V}}_{\mathbf{uu},k}\mathbf{K}_k + \mathbf{K}_k^{\mathrm{T}}\hat{\mathbf{V}}_{\mathbf{xu},k}^{\mathrm{T}} + \hat{\mathbf{V}}_{\mathbf{xu},k}\mathbf{K}_k \ , \tag{2.60b}$$

where

$$\hat{\mathbf{v}}_{\mathbf{x},k} = \bar{\mathbf{v}}_{\mathbf{x},k} - \mathbf{H}_{\mathbf{x},k}\mathbf{Z}_k^{-1}\mathbf{r}_k \qquad \hat{\mathbf{V}}_{\mathbf{xx},k} = \bar{\mathbf{V}}_{\mathbf{xx},k} - \mathbf{H}_{\mathbf{x},k}\mathbf{Z}_k^{-1}\mathbf{\Lambda}_k\mathbf{H}_{\mathbf{x},k}^{\mathrm{T}} \tag{2.61a}$$

$$\hat{\mathbf{v}}_{\mathbf{u},k} = \bar{\mathbf{v}}_{\mathbf{x},k} - \mathbf{H}_{\mathbf{u},k}\mathbf{Z}_k^{-1}\mathbf{r}_k \qquad \hat{\mathbf{V}}_{\mathbf{uu},k} = \bar{\mathbf{V}}_{\mathbf{uu},k} - \mathbf{H}_{\mathbf{u},k}\mathbf{Z}_k^{-1}\mathbf{\Lambda}_k\mathbf{H}_{\mathbf{u},k}^{\mathrm{T}} \tag{2.61b}$$

$$\hat{\mathbf{V}}_{\mathbf{xu},k} = \bar{\mathbf{V}}_{\mathbf{xu},k} - \mathbf{H}_{\mathbf{x},k}\mathbf{Z}_k^{-1}\mathbf{\Lambda}_k\mathbf{H}_{\mathbf{u},k}^{\mathrm{T}} \ . \tag{2.61c}$$

Similar to the feasible interior point method, the step size α is selected s.t. the constraints $\boldsymbol{\zeta}_k \geq 0$ and $\boldsymbol{\lambda}_k \geq 0$ are fulfilled in all iterations. Note that this strategy does not require to evaluate the inequality constraints for the step size selection procedure. Considering that the constraints can be elaborate functions, this can lead to advantages regarding runtime.

3 Application: Kinematic Vehicle Model

This chapter studies iLQR-based trajectory planning for a simple 2D kinematic vehicle model. The mathematical model is presented in Section 3.1 while in Section 3.2 the results generated by the different constrained iLQR algorithms from Chapter 2 are given. All algorithms are implemented in MATLAB/SIMULINK R2021a and all measurements are performed on an Intel Core i7-10850H CPU @ 2.7 GHz × 12.

3.1 Model

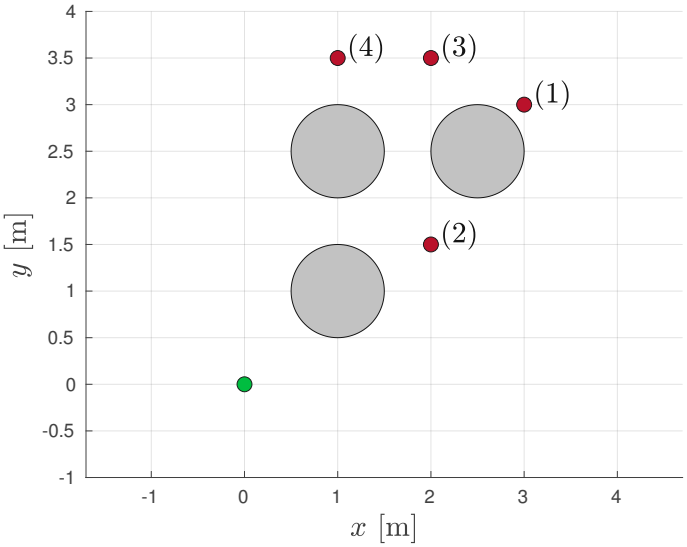


Figure 3.1: Experiment setup for the 2D vehicle benchmark. The green point represents the initial state $\mathbf{x}_0$ and the red ones show the desired end points $\mathbf{x}_{N,di}$, $i = 1, 2, 3, 4$. The gray circles represent the collision objects

Figure 3.1 shows the experimental setup for the 2D kinematic vehicle application. The goal is to find a feasible, collision-free trajectory from the initial state $\mathbf{x}_0$, which is visualized as a green point in Figure 3.1, to the desired endpoints $\mathbf{x}_{N,di}$, $i = 1, 2, 3, 4$, visualized as red points in Figure 3.1. The collision objects are represented by the gray circles in Figure 3.1.

3.1.1 Vehicle Kinematics

The dynamic equations of the kinematic vehicle model are given by, c.f. [28]

$$\frac{\mathrm{d}}{\mathrm{d}t}\underbrace{\begin{bmatrix} x \\ y \\ v \\ \theta \end{bmatrix}}_{\mathbf{x}} = \underbrace{\begin{bmatrix} v\sin(\theta) \\ v\cos(\theta) \\ a \\ \omega v \end{bmatrix}}_{\mathbf{f}(\mathbf{x},\mathbf{u})} , \tag{3.1}$$

where the control input $\mathbf{u} := [\omega, a]^{\mathrm{T}} \in \mathbb{R}^2$ comprises the steering command ω in $\frac{\mathrm{rad}}{\mathrm{m}}$ and the acceleration a in $\frac{\mathrm{m}}{\mathrm{s}^2}$ of the vehicle. The state $\mathbf{x} := [x, y, v, \theta]^{\mathrm{T}} \in \mathbb{R}^4$ consists of the position represented by the 2D Euclidean coordinates x and y given in m, the velocity v in $\frac{\mathrm{m}}{\mathrm{s}}$ and the heading θ in rad. The discrete time system equations are retrieved using Euler forward integration

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k)T_s \ , \tag{3.2}$$

where $T_s = 100$ ms is the sampling time. The initial state is chosen as

$$\mathbf{x}_0 = \begin{bmatrix} 0 \text{ m} \\ 0 \text{ m} \\ 0 \ \frac{\mathrm{m}}{\mathrm{s}} \\ 0 \text{ rad} \end{bmatrix} . \tag{3.3}$$

3.1.2 Constraints

Three circles with radius $r_i = 0.5$ m, $i = 1, 2, 3$ and centers $\mathbf{c}_1 = [1 \text{ m}, 1 \text{ m}]^{\mathrm{T}}$, $\mathbf{c}_2 = [1 \text{ m}, 2.5 \text{ m}]^{\mathrm{T}}$ and $\mathbf{c}_3 = [2.5 \text{ m}, 2.5 \text{ m}]^{\mathrm{T}}$ are used as collision objects, see Figure 3.1. The collision constraints are formulated using the distance to the center of these circles, i.e.

$$\left\| \mathbf{c}_i - \begin{bmatrix} x & y \end{bmatrix}^{\mathrm{T}} \right\|_2^2 \geq r_i, \qquad i = 1, 2, 3 \ . \tag{3.4}$$

Unless otherwise stated, the control constraints are chosen as

$$\omega \in [-\pi/4, \pi/4] \text{ rad/m} \qquad \text{and} \qquad a \in [-0.6, 0.6] \text{ m/s}^2 \ . \tag{3.5}$$

The vehicle forward velocity is limited by

$$-8.3 \ \frac{\mathrm{m}}{\mathrm{s}} \leq v \leq 8.3 \ \frac{\mathrm{m}}{\mathrm{s}} \qquad k = 1, \ldots, N-1 \ . \tag{3.6}$$

3.1.3 Cost Function

In order generate a trajectory that controls the system towards a desired final configuration $\mathbf{x}_{N,d} \in \mathbb{R}^4$, a quadratic cost function of the form

$$J(\mathbf{u}_0, \ldots, \mathbf{u}_{N-1}) := (\mathbf{x}_N - \mathbf{x}_{N,d})^{\mathrm{T}} \mathbf{L}_{\mathbf{xx},N} (\mathbf{x}_N - \mathbf{x}_{N,d}) + \sum_{k=0}^{N-1} \mathbf{u}_k^{\mathrm{T}} \mathbf{L}_{\mathbf{uu}} \mathbf{u}_k \ , \tag{3.7}$$

with the matrix $\mathbf{L}_{\mathbf{xx},N} := \text{diag}(500, 500, 100, 500)$ which penalizes the deviation from the desired endpoints $\mathbf{x}_{N,d}$ and the matrix $\mathbf{L}_{\mathbf{uu}} := \text{diag}(1, 30)$ which constitutes a regularization term. The algorithms are evaluated with four target configurations

$$\mathbf{x}_{N,d1} := [3 \text{ m}, 3 \text{ m}, 0 \text{ m/s}, \pi/2 \text{ rad}]^{\mathrm{T}} \tag{3.8a}$$
$$\mathbf{x}_{N,d2} := [2 \text{ m}, 1.5 \text{ m}, 0 \text{ m/s}, \pi/2 \text{ rad}]^{\mathrm{T}} \tag{3.8b}$$
$$\mathbf{x}_{N,d3} := [2 \text{ m}, 3.5 \text{ m}, 0 \text{ m/s}, \pi/2 \text{ rad}]^{\mathrm{T}} \tag{3.8c}$$
$$\mathbf{x}_{N,d4} := [1 \text{ m}, 3.5 \text{ m}, 0 \text{ m/s}, \pi/2 \text{ rad}]^{\mathrm{T}} \ , \tag{3.8d}$$

which are shown in Figure 3.1 by the red points (1)-(4).

3.2 Offline Planning

This section shows the results of the algorithms presented in Chapter 2 applied to offline trajectory planning for the 2D kinematic vehicle model introduced in Section 3.1. The overall discrete time optimal control problem

$$\min_{\mathbf{u}_1,\dots,\mathbf{u}_N \in \mathbb{R}^m} \quad J(\mathbf{u}_0, \dots, \mathbf{u}_{N-1}) \tag{3.9a}$$
$$\text{s.t.} \quad \mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k) T_s \qquad \text{(System equations)} \tag{3.9b}$$
$$- \|\mathbf{c}_i - [x_k, y_k]^{\mathrm{T}}\|_2^2 \leq -0.5 \text{ m} \ , \ i = 0, \dots, N \qquad \text{(Collision constraints)} \tag{3.9c}$$
$$-\frac{\pi}{4}\frac{\text{rad}}{\text{m}} \leq \omega_k \leq \frac{\pi}{4}\frac{\text{rad}}{\text{m}} \qquad \text{(Control limit)} \tag{3.9d}$$
$$-0.6\,\frac{\text{m}}{\text{s}^2} \leq a_k \leq 0.6\,\frac{\text{m}}{\text{s}^2} \qquad \text{(Control limit)} \tag{3.9e}$$
$$-8.3\,\frac{\text{m}}{\text{s}} \leq v_k \leq 8.3\,\frac{\text{m}}{\text{s}} \qquad \text{(State limit)} \tag{3.9f}$$

is solved using the constrained iLQR algorithms from Chapter 2.

3.2.1 Augmented Lagrangian

The resulting trajectories generated by the augmented Lagrangian method presented in Section 2.2.2 are depicted in Figure 3.2, and Figure 3.3 shows the corresponding control sequence. Since there is no qualitative difference in the results of the algorithms using the variants of the augmented Lagrangian approach (2.30), (2.33) and (2.36), only the ones computed with (2.36) are visualized. The algorithm is able to generate trajectories that move the kinematic car model to the vicinity of the desired endpoints and fulfill both, the collision and control constraints.

The evolution of the costs (3.7) over the optimization iterations are depicted in Figure 3.4. Hence, the algorithm is able to find a feasible solution for the optimal control problem (3.9) in a finite number of iterations. Figure 3.4 shows that, in all three cases, the overall costs decrease at the beginning and then slightly increase again in later iterations. The algorithm first computes a solution with fixed penalties and fixed Lagrange multipliers,

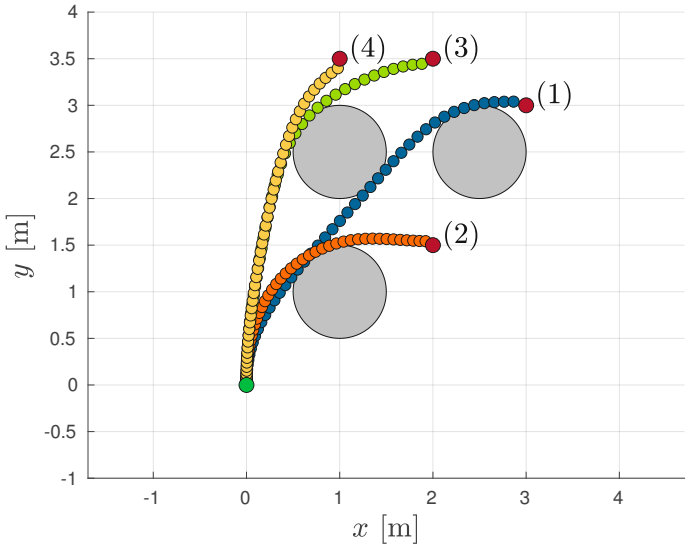


Figure 3.2: Resulting state trajectories of the augmented Lagrangian method.

which in general does not fulfill the constraints. The penalties and the Lagrange multipliers are increased after convergence which leads to a larger influence of the constraint violations than in the previous iteration with smaller penalties and Lagrange multipliers. Hence, the actual cost function starts to increase again, since a solution of the constrained problem in general yields higher costs than the unconstrained one.

The number of iterations required to converge are approximately equal for all three penalty-Lagrangian functions (ALTRO, PHR and S-PHR), which confirms the results from [29] where PHR and S-PHR is considered as well. This suggests that, similar to [29], the lack of a two times continuously differentiable penalty-Lagrangian in the standard PHR method does not have a huge influence on the algorithm's convergence. However, the chosen step sizes, which are depicted in Figure 3.5, are smaller on average for the PHR penalty-Lagrangian than for the other ones.

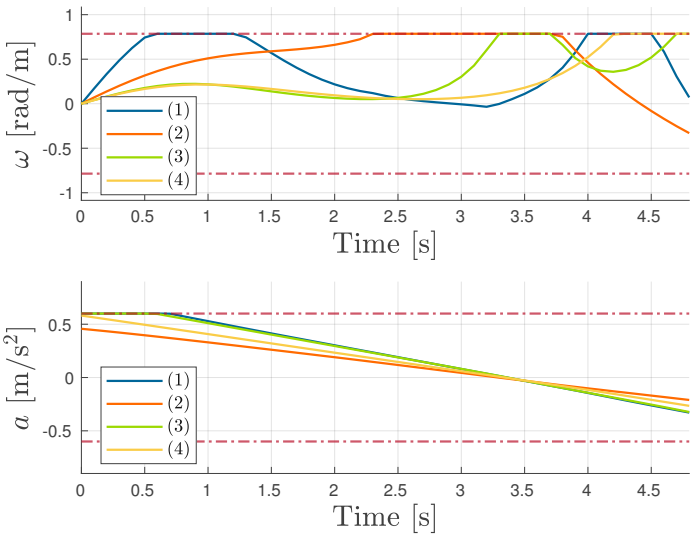
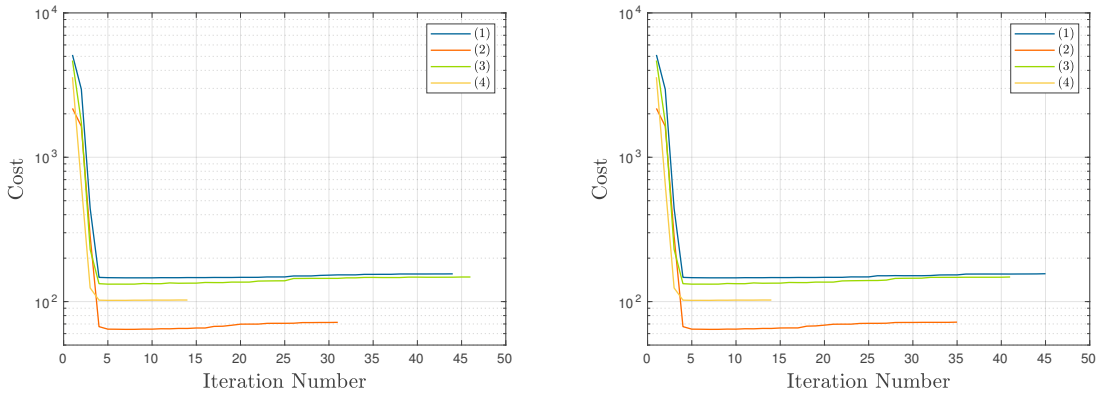
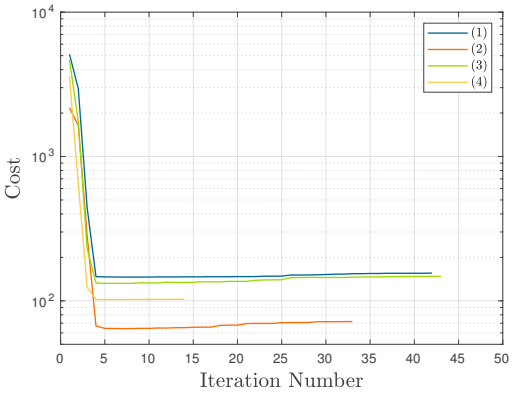


Figure 3.3: Resulting control trajectories of the augmented Lagrangian method.



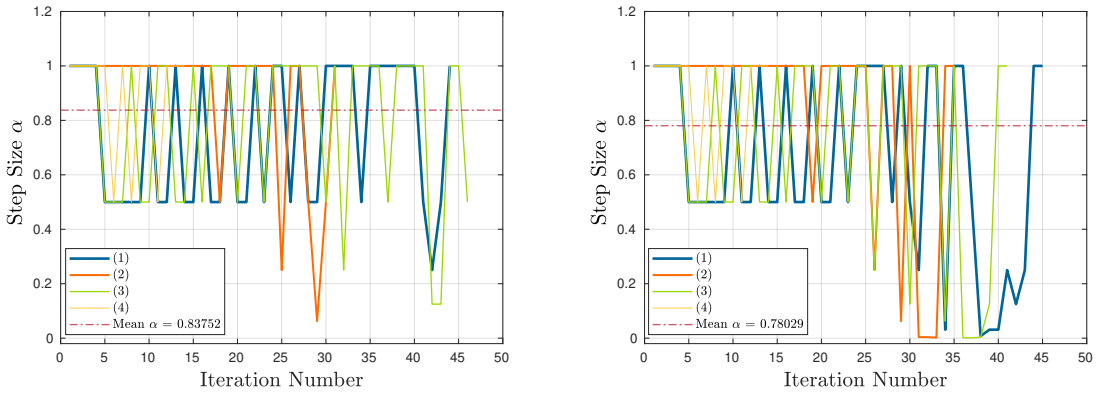
(a) ALTRO

(b) PHR

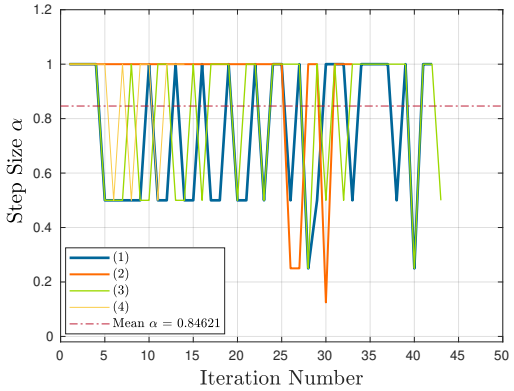


(c) S-PHR

Figure 3.4: Cost evolution of the augmented Lagrangian method.



(a) ALTRO (b) PHR

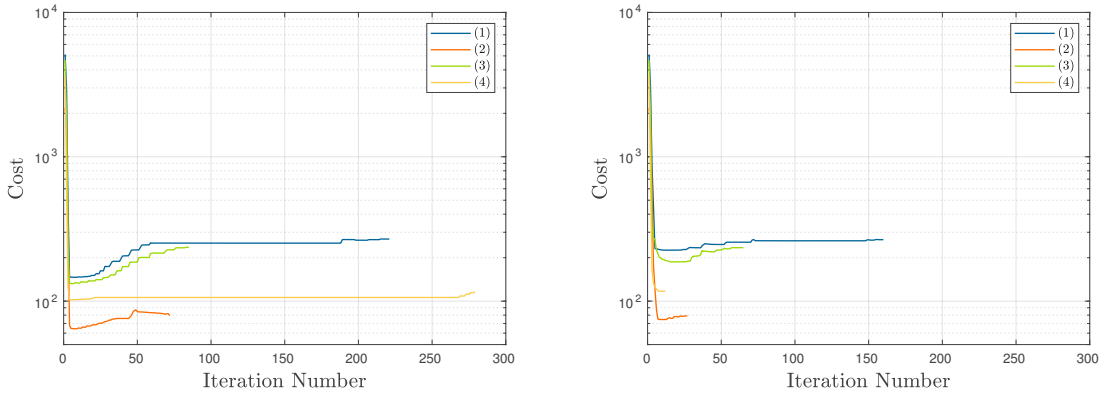


(c) S-PHR

Figure 3.5: Step size evolution of the augmented Lagrangian method.

3.2.2 Augmented Lagrangian combined with Projected Newton

A combined augmented Lagrangian (AL) approach applied to the 2D kinematic vehicle example is discussed in this section. While the control limits are handled using the projected Newton method from Section 2.2.1, collision and state constraints are incorporated using the augmented Lagrangian method from Section 2.2.2.



(a) Augmented Lagrangian

(b) Augmented Lagrangian with Projected Newton

Figure 3.6: Cost evolution of the pure and the combined AL approach with the projected Newton method for more restrictive control constraints.

One drawback of the pure augmented Lagrangian is the degradation of the performance when multiple conflicting constraints shall be fulfilled, e.g. the trajectory has to be collision free but the control limit is very small. This leads to convergence issues, since the algorithm jumps from one constraint, e.g. a collision constraint, to another one, e.g. a control limit. Figure 3.6a shows the cost evolution of the augmented Lagrangian approach using more restrictive control constraints $\omega \in [-\pi/5, \pi/5]$ rad/s and $a \in [-0.35, 0.35]$ $\frac{\mathrm{m}}{\mathrm{s}^2}$. On the other hand, Figure 3.6b depicts the cost evolution of the combined augmented Lagrangian with the projected Newton method. Hence, handling the control constraints with the projected Newton method frees the augmented Lagrangian method from the burden to find a solution that fulfills conflicting constraints and improves the convergence behavior. Especially for end configuration (4) the combined approach requires a much lower number of iterations, c.f. Figure 3.6b, since the algorithm mainly has to handle the control limits and only little effort has to be taken in order to fulfill the collision constraints.

The following experiments discuss the results of the combined augmented Lagrangian with the projected Newton method using the original control constraints (3.9d) and (3.9e).

Figure 3.7 shows the resulting state trajectories, Figure 3.8 the resulting control trajectories and Figure 3.9 the cost evolution. Since the resulting state and control trajectories of the algorithms using the different penalty-Lagrangian functions (2.30), (2.33) and (2.36) are qualitatively the same, only the results of the approach that uses (2.36) is depicted in Figures 3.7 and 3.8.

Comparing Figure 3.9 with Figure 3.4 shows that the cost decrease at the beginning is

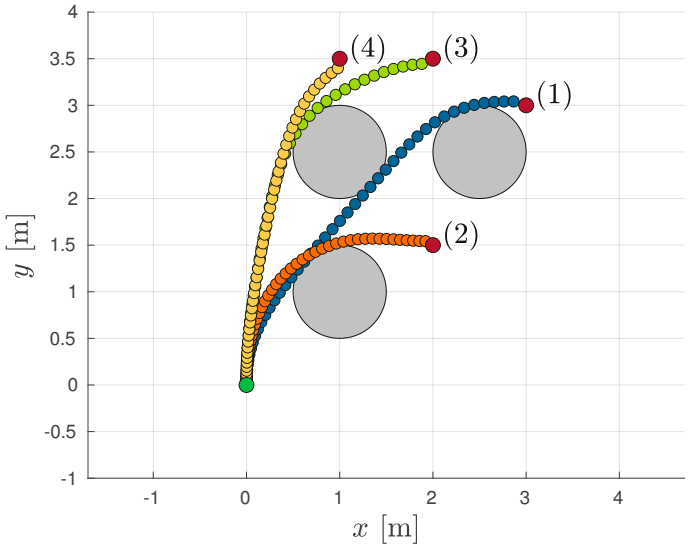


Figure 3.7: Resulting state trajectories of the combined AL with the projected Newton method.

steeper for the pure augmented Lagrangian approach than for the combined one. This comes from the update rule (2.24). Since the active set is determined by looking at the gradient at $\tilde{\mathbf{x}} = 0$, all control inputs that are not labeled active, but violate the constraints due to the influence of $\tilde{\mathbf{x}} \neq \mathbf{0}$, are simply clamped. This might result in a non-descending search direction.

Similar to the pure augmented Lagrangian approach, the choice of the actual penalty-Lagrangian does not seem to be highly influential on the convergence behavior.

Figure 3.10 visualizes the selected step sizes in each optimization iteration. The average step size is slightly larger in all cases compared to the ones of the pure augmented Lagrangian approach in Figure 3.5. Note that the clamping operation (2.24) already yields smaller control updates, which is why larger step sizes can be accepted. Since the line search strategy requires additional evaluations of the system dynamics and the constraints, a larger average step size may be beneficial, especially for applications where the evaluation of these functions is computationally expensive.

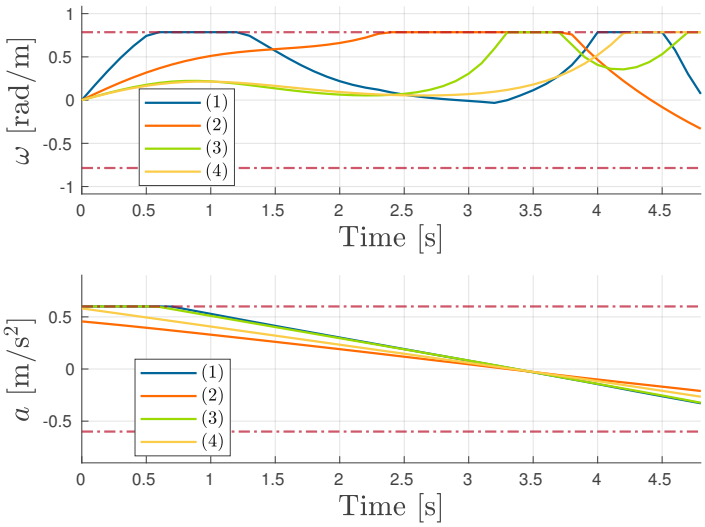
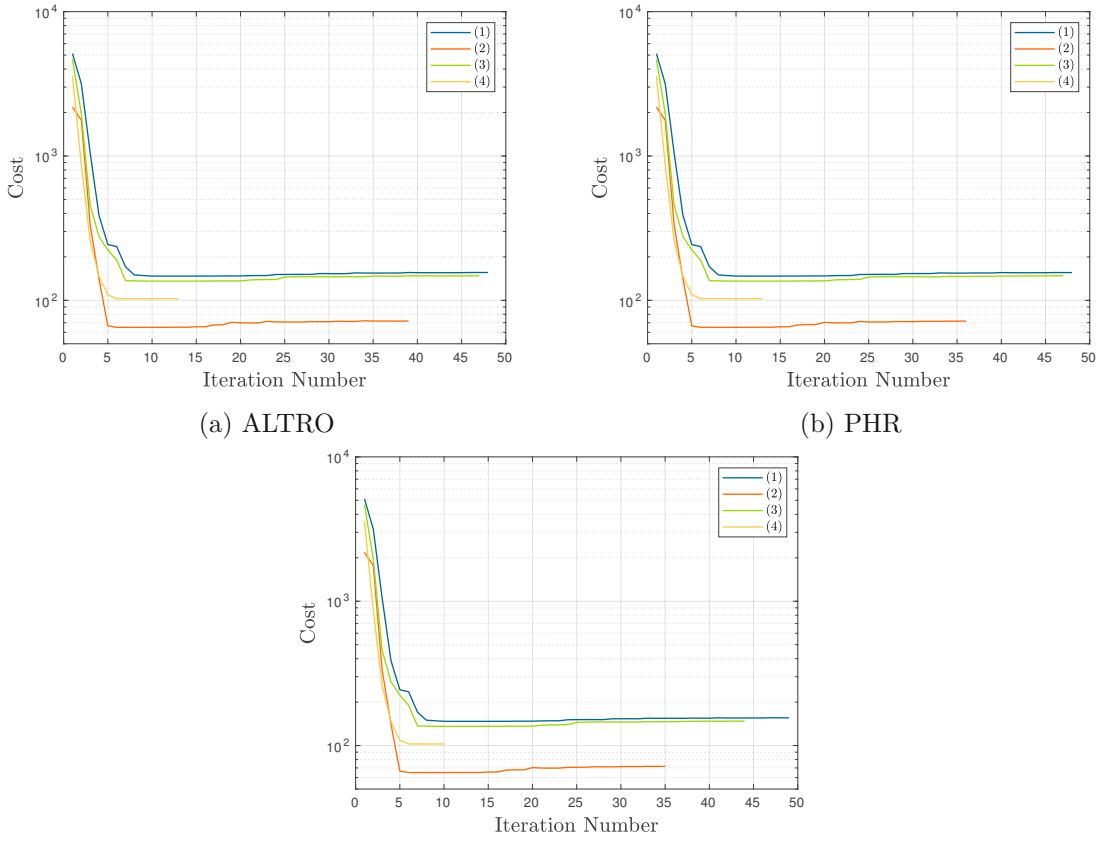


Figure 3.8: Resulting control trajectories of the combined AL with the projected Newton method.

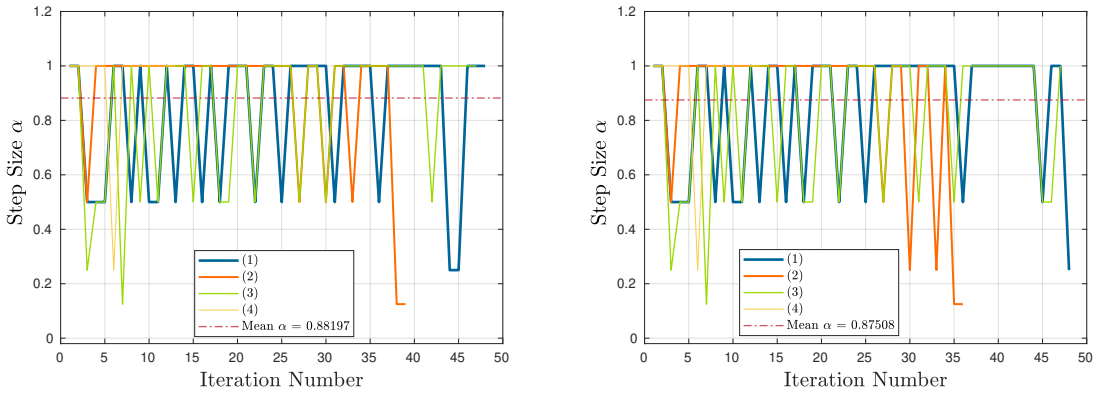


(a) ALTRO

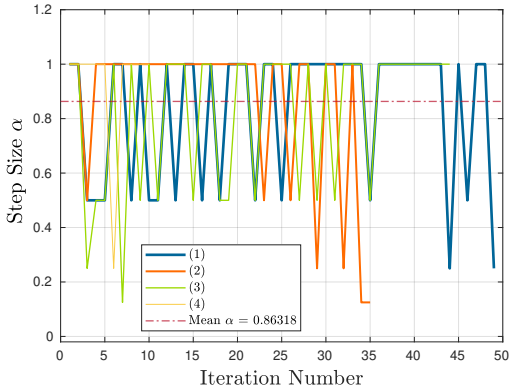
(b) PHR

(c) S-PHR

Figure 3.9: Cost evolution of the combined AL with the projected Newton method.



(a) ALTRO (b) PHR



(c) S-PHR

Figure 3.10: Step size evolution of the combined AL with the projected Newton method.

3.2.3 Primal-Dual Interior Point

Figure 3.11 shows the resulting trajectories of the feasible primal-dual interior point method from Section 2.2.3 and the corresponding control trajectories are depicted in Figure 3.13. Again, since the qualitative results of the feasible and the infeasible primal-dual interior point method are equal, only the solutions of the former one are visualized in Figure 3.11 and Figure 3.13. Other than the augmented Lagrangian method, the inequality constraints are fulfilled during the whole optimization procedure with the interior point methods. The consequence is that once the trajectory starts passing an obstacle at one side there is nearly no chance to change this decision. Trajectory (2) in Figure 3.11 exactly reveals this behavior.

The cost evolution for both variants of the primal-dual interior point algorithm is depicted in Figure 3.12. Closeness to obstacles is penalized more at the beginning and then becomes less as the barrier parameter increases, which is exactly the opposite strategy compared to the augmented Lagrangian method. It turns out that due to the barrier characteristics, the algorithm proceeds very slowly at the beginning, especially for more complex environments where free space gets narrower. This can be observed by comparing Figure 3.12 with Figure 3.4. Due to the slow progress, the decrease of the cost, especially in the earlier iterations, is much steeper for the augmented Lagrangian method in Figure 3.4 than for the interior point method in Figure 3.12.

Figure 3.12 shows that the feasible interior point method shows slightly faster convergence compared to the infeasible variant. Additionally, the step size history depicted in Figure 3.14 suggests that the infeasible interior point method requires smaller step sizes than the feasible one. However, as already stated in Section 2.2.3, the requirement of initial control and state trajectories that observe all collision constraints is a huge limitation of the feasible primal-dual interior point method. This is why the infeasible interior point method might still be preferable in applications.

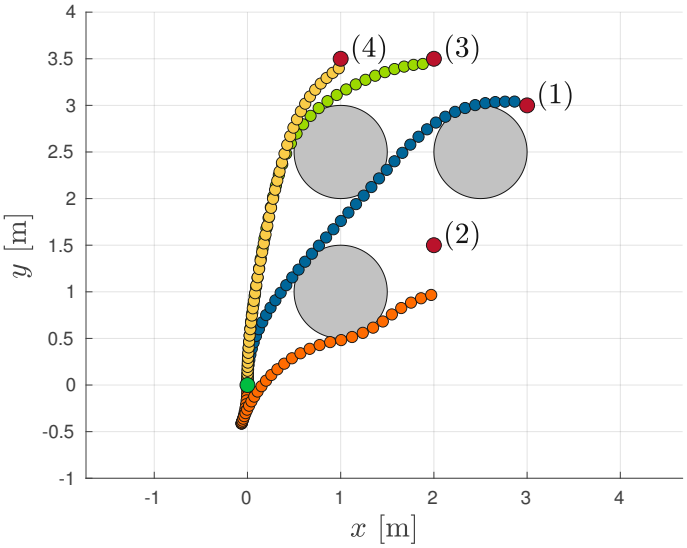
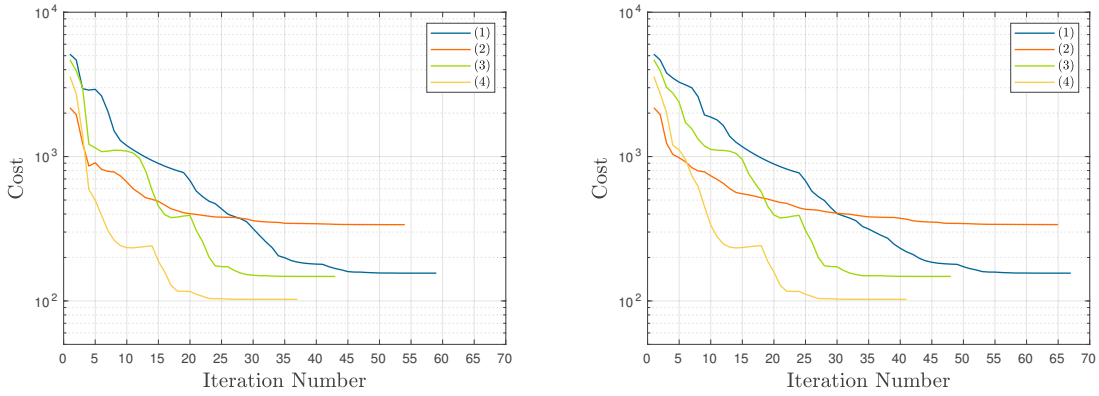


Figure 3.11: Resulting state trajectories of the primal-dual interior point method.



(a) Feasible Primal-Dual Interior Point (b) Infeasible Primal-Dual Interior Point

Figure 3.12: Cost evolution of the primal-dual interior point method.

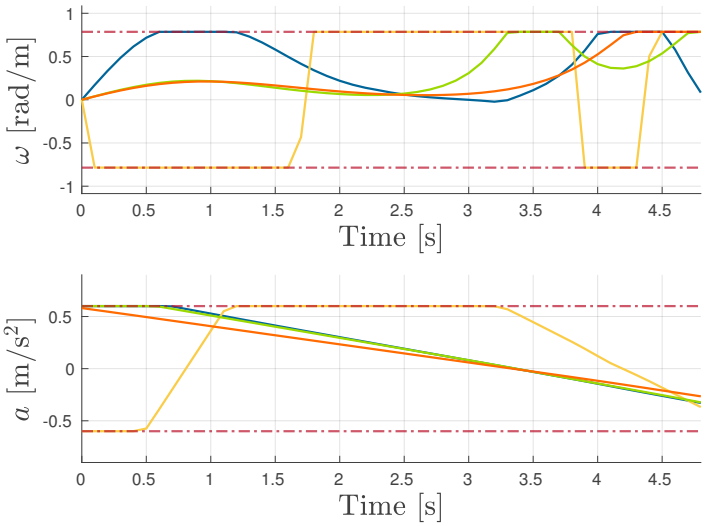
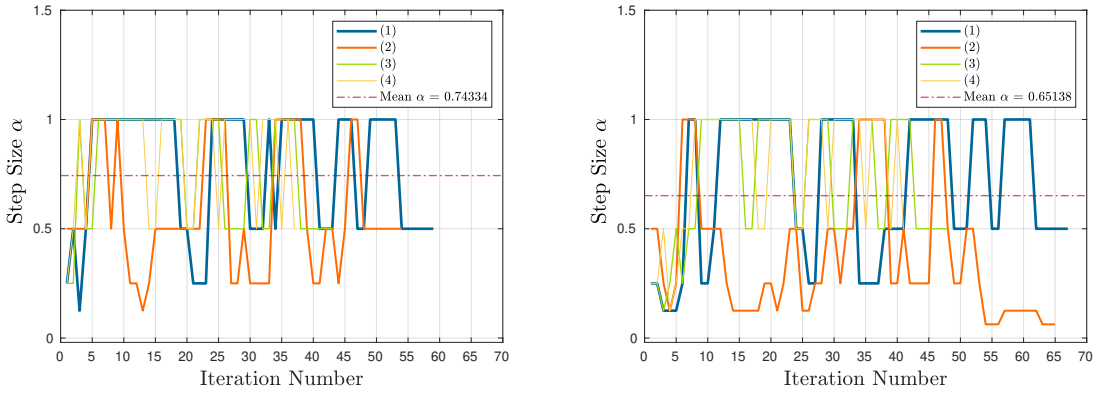


Figure 3.13: Resulting control trajectories of the primal-dual interior point method.



(a) Feasible Primal-Dual Interior Point (b) Infeasible Primal-Dual Interior Point

Figure 3.14: Step size evolution of the primal-dual interior point method.

3.2.4 Conclusions

The numerical experiments show different properties of the presented algorithms. The augmented Lagrangian method exhibits good convergence, even if the algorithm is initialized with the control inputs and velocities being zero. However, increasing the penalties of the constraint violations may lead to bad convergence for competing and restrictive constraints. It is shown that the augmented Lagrangian can be relieved from handling control limits by combining it with the projected Newton method. The performance of the projected Newton method depends on the quality of the active set estimation during the backward calculation. The primal-dual interior point method converges quite robustly, even for very restrictive constraints. However, for collision avoidance applications with narrow free space the algorithm requires more iterations to converge, at least if the initial trajectory equals the starting point with zero acceleration and zero velocity.

4 Application: Timber Crane

This chapter presents iLQR-based trajectory planning for a timber crane. First, the crane model is presented in Section 4.1 while in Section 4.2 the optimization problem is formulated and the results generated by the different constrained iLQR algorithms introduced in Chapter 2 are presented. Finally, Section 4.3 describes an online trajectory planner based on model predictive control (MPC). All algorithms are implemented in MATLAB/SIMULINK R2021a and all experiments are performed on an Intel Core i7-10850H CPU @ 2.7 GHz × 12.

4.1 Model

The kinematic chain of the timber crane is visualized in Figure 4.1. It has eight degrees of

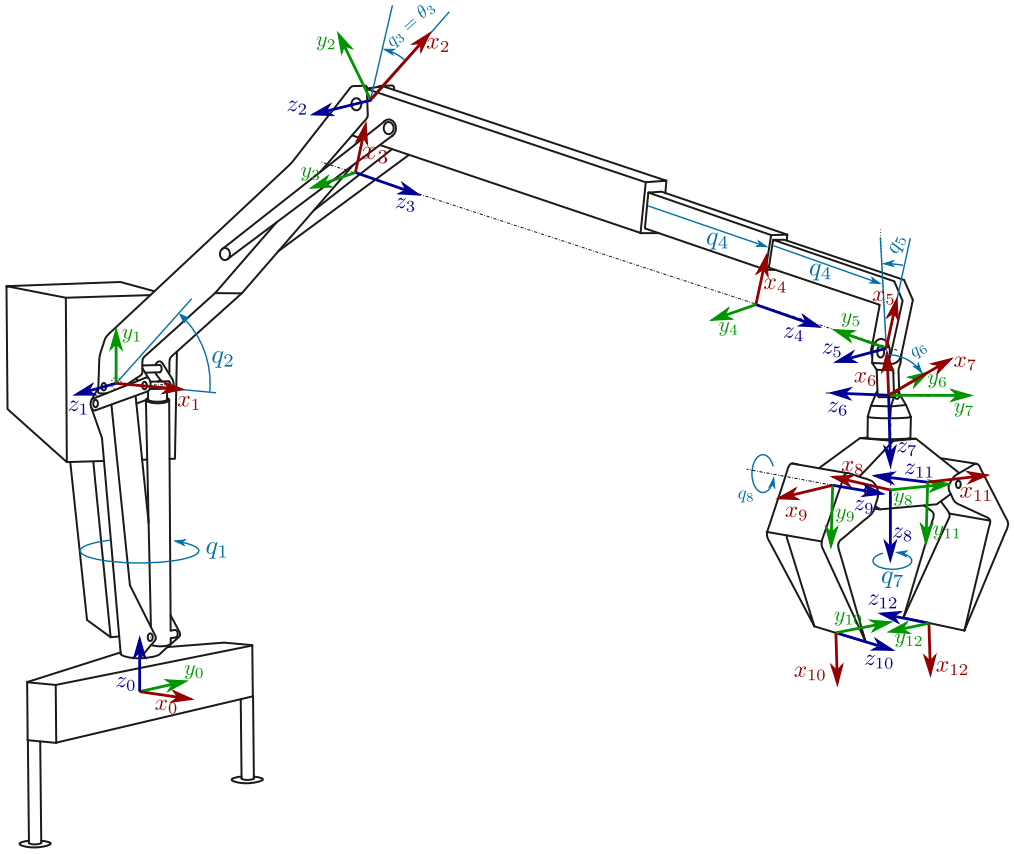


Figure 4.1: Kinematic chain of the timber crane.

freedom (DoFs), but in order to reduce the complexity of the model used for planning the jaw angle q_8 is assumed to be a constant parameter. This allows to reduce the complete grab to a single rigid body with only one mass, center of gravity and moment of inertia. Hence, the kinematic chain considered for planning has only seven degrees of freedom, consisting of five actuated $\mathbf{q}_A := [q_1, q_2, q_3, q_4, q_7]^{\mathrm{T}} \in \mathbb{R}^{n_A}$ and two non-actuated ones $\mathbf{q}_U := [q_5, q_6]^{\mathrm{T}} \in \mathbb{R}^{n_U}$. Thereby, q_4 is a linear DoF, whereas all others are rotational DoFs.

4.1.1 Crane Kinematics

The mathematical model of the crane's kinematic chain is described by transformations from a coordinate Frame $\mathcal{F}_i$ attached to joint i to a coordinate frame $\mathcal{F}_{i-1}$ attached to joint $i-1$

$$\mathbf{H}_{i-1}^{i} = \begin{bmatrix} \mathbf{R}_{i-1}^{i} & \mathbf{d}_{i-1}^{i} \\ \mathbf{0}^{\mathrm{T}} & 1 \end{bmatrix} \in \mathcal{SE}(3) \tag{4.1}$$

where $\mathbf{R}_{i-1}^{i} \in \mathcal{SO}(3)$ is a three dimensional rotation matrix and $\mathbf{d}_{i-1}^{i} \in \mathbb{R}^3$ a three dimensional translation vector. The coordinate frames depicted in Figure 4.1 are selected according to the *Denavit-Hartenberg convention*, see, e.g., [1, 33]. However, note that frame $\mathcal{F}_{11}$ is not defined by Denavit-Hartenberg parameters w.r.t. frame $\mathcal{F}_{10}$, but rather from frame $\mathcal{F}_8$. Hence, every transformation $\mathbf{H}_{i-1}^{i}$, $i = 1, \ldots, 10, 12$ and $\mathbf{H}_{8}^{11}$ can be described using four *Denavit-Hartenberg parameters* θ_i, d_i, a_i and α_i as

$$\mathbf{H}_{i-1}^{i} = \mathbf{H}_{Rz}(\theta_i)\mathbf{H}_{Tz}(d_i)\mathbf{H}_{Tx}(a_i)\mathbf{H}_{Rx}(\alpha_i) \ , \tag{4.2}$$

where $\mathbf{H}_{Ri}$ is a pure rotation around the i-axis and $\mathbf{H}_{Ti}$ is a pure translation in direction of the i-axis. The transformation from $\mathcal{F}_j$ to $\mathcal{F}_i$, $0 \leq i < j$ can be computed using

$$\mathbf{H}_{i}^{j} = \begin{cases} \prod_{l=i+1}^{j} \mathbf{H}_{l-1}^{l} & , \text{ for } j \leq 10 \\ \left(\prod_{l=i+1}^{8} \mathbf{H}_{l-1}^{l} \right) \mathbf{H}_{8}^{11} \mathbf{H}_{11}^{j} & , \text{ for } 11 \leq j \leq 12 \end{cases} , \tag{4.3}$$

where $\prod_{l=i+1}^{j} \mathbf{H}_{l-1}^{l}$ being the identity for $j \leq i$. The Denavit-Hartenberg parameters for the crane are given in Table 4.1.

i	θ_i [rad]	d_i [m]	a_i [m]	α_i [rad]
1	q_1	2.425	0.1800	$\frac{\pi}{2}$
2	q_2	0	3.4931	0
3	q_3	0	-0.3925	$\frac{\pi}{2}$
4	0	q_4	0	0
5	0	q_4	0	$-\frac{\pi}{2}$
6	q_5	0	-0.2130	$-\frac{\pi}{2}$
7	q_6	0	0	$-\frac{\pi}{2}$
8	q_7	0.578	0	0
9	$-\frac{\pi}{2}$	0	0.3402	$\frac{\pi}{2}$
10	1.5708	0	0.8566	0
11	$\frac{\pi}{2}$	0	0.3248	$\frac{\pi}{2}$
12	1.5708	0	0.8566	0

Table 4.1: Denavit-Hartenberg parameters of the timber crane.

4.1.2 Crane Dynamics

For a general rigid-body system the equations of motion can be derived using the *Lagrange formalism*, c.f. [1, 33]. This yields

$$\mathbf{D}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}) + \mathbf{F}\dot{\mathbf{q}} = \boldsymbol{\tau} \ , \tag{4.4}$$

where $\mathbf{q} \in \mathbb{R}^n$ are the joint coodinates, $\mathbf{D}(\mathbf{q}) \in \mathbb{R}^{n\times n}$ is the positive definite mass matrix, $\mathbf{C}(\mathbf{q},\dot{\mathbf{q}}) \in \mathbb{R}^{n\times n}$ is the Coriolis matrix, $\mathbf{g}(\mathbf{q}) \in \mathbb{R}^n$ are the potential forces and $\boldsymbol{\tau} \in \mathbb{R}^n$ are the generalized joint forces. The diagonal matrix $\mathbf{F} = \mathrm{diag}(\boldsymbol{\tau}_v) \in \mathbb{R}^{n\times n}$ contains the viscous friction coefficients $\boldsymbol{\tau}_v \in \mathbb{R}^n$. The joint coordinates comprise the coordinates $\mathbf{q}_A \in \mathbb{R}^{n_A}$ of the actuated joints and the coordinates $\mathbf{q}_U \in \mathbb{R}^{n_U}$ of the non-actuated ones, i.e.

$$\mathbf{q} := \begin{bmatrix} \mathbf{q}_A \\ \mathbf{q}_U \end{bmatrix} \ . \tag{4.5}$$

Since no external forces can be applied to non-actuated joints, (4.4) can be partitioned into

$$\underbrace{\begin{bmatrix} \mathbf{D}_{AA}(\mathbf{q}) & \mathbf{D}_{AU}(\mathbf{q}) \\ \mathbf{D}_{UA}(\mathbf{q}) & \mathbf{D}_{UU}(\mathbf{q}) \end{bmatrix}}_{\mathbf{D}(\mathbf{q})} \begin{bmatrix} \ddot{\mathbf{q}}_A \\ \ddot{\mathbf{q}}_U \end{bmatrix} + \left(\underbrace{\begin{bmatrix} \mathbf{C}_A(\mathbf{q},\dot{\mathbf{q}}) \\ \mathbf{C}_U(\mathbf{q},\dot{\mathbf{q}}) \end{bmatrix}}_{\mathbf{C}(\mathbf{q},\dot{\mathbf{q}})} + \underbrace{\begin{bmatrix} \mathbf{F}_A \\ \mathbf{F}_U \end{bmatrix}}_{\mathbf{F}} \right) \dot{\mathbf{q}} + \underbrace{\begin{bmatrix} \mathbf{g}_A(\mathbf{q}) \\ \mathbf{g}_U(\mathbf{q}) \end{bmatrix}}_{\mathbf{g}(\mathbf{q})} = \underbrace{\begin{bmatrix} \boldsymbol{\tau}_A \\ \boldsymbol{\tau}_U \end{bmatrix}}_{\boldsymbol{\tau}} \tag{4.6}$$

where

$$\boldsymbol{\tau}_U = \mathbf{D}_{UA}(\mathbf{q})\ddot{\mathbf{q}}_A + \mathbf{D}_{UU}(\mathbf{q})\ddot{\mathbf{q}}_U + \Big(\mathbf{C}_U(\mathbf{q},\dot{\mathbf{q}}) + \mathbf{F}_U\Big)\dot{\mathbf{q}} + \mathbf{g}_U(\mathbf{q}) = \mathbf{0} \tag{4.7}$$

for the non-actuated joints.

For iLQR, a discrete time state-space description of the system is required. Hence, the state is defined as

$$\mathbf{x} := \begin{bmatrix} \mathbf{q} \\ \dot{\mathbf{q}} \end{bmatrix} = \begin{bmatrix} \mathbf{q}_A \\ \mathbf{q}_U \\ \dot{\mathbf{q}}_A \\ \dot{\mathbf{q}}_U \end{bmatrix} \tag{4.8}$$

and the second derivatives of the actuated joint coordinates serve as control inputs, i.e.

$$\mathbf{u} := \ddot{\mathbf{q}}_A \ . \tag{4.9}$$

Assuming that the control $\mathbf{u}(t)$ is constant for $t \in [t_k, t_{k+1})$ yields the discrete time state equation of the actuated joints

$$\mathbf{q}_{A,k+1} = \mathbf{q}_{A,k} + \dot{\mathbf{q}}_{A,k} T_s + \frac{1}{2} \mathbf{u}_k T_s^2 \tag{4.10a}$$

$$\dot{\mathbf{q}}_{A,k+1} = \dot{\mathbf{q}}_{A,k} + \mathbf{u}_k T_s \ , \tag{4.10b}$$

where $T_s > 0$ is the sampling time and $\mathbf{q}_{A,k} = \mathbf{q}_A(kT_s)$, $\dot{\mathbf{q}}_{A,k} = \dot{\mathbf{q}}_A(kT_s)$ and $\mathbf{u}_k = \mathbf{u}(kT_s)$. From (4.7) an expression for the second derivative of the non-actuated joint coordinates can be derived as

$$\ddot{\mathbf{q}}_U(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}_A) = -\mathbf{D}_{UU}(\mathbf{q})^{-1} \Big(\mathbf{D}_{UA}(\mathbf{q}) \ddot{\mathbf{q}}_A + \Big(\mathbf{C}_U(\mathbf{q}, \dot{\mathbf{q}}) + \mathbf{F}_U \Big) \dot{\mathbf{q}} + \mathbf{g}_U(\mathbf{q}) \Big) \ . \tag{4.11}$$

Assuming that $\ddot{\mathbf{q}}_U(t)$ is constant for $t \in [t_k, t_{k+1})$ yields

$$\mathbf{q}_{U,k+1} \approx \mathbf{q}_{U,k} + \dot{\mathbf{q}}_{U,k} T_s + \frac{1}{2} \ddot{\mathbf{q}}_{U,k}(\mathbf{q}_k, \dot{\mathbf{q}}_k, \ddot{\mathbf{q}}_{A,k}) T_s^2 \tag{4.12a}$$

$$\dot{\mathbf{q}}_{U,k+1} \approx \dot{\mathbf{q}}_{U,k} + \ddot{\mathbf{q}}_U(\mathbf{q}_k, \dot{\mathbf{q}}_k, \ddot{\mathbf{q}}_{A,k}) T_s \ , \tag{4.12b}$$

where $\mathbf{q}_{A,k} = \mathbf{q}_U(kT_s)$, $\dot{\mathbf{q}}_{U,k} = \dot{\mathbf{q}}_U(kT_s)$ and $\ddot{\mathbf{q}}_{U,k} = \ddot{\mathbf{q}}_U(kT_s)$. However, note that, other than for the actuated joints, the assumption that $\ddot{\mathbf{q}}_U$ is constant is just an approximation here.

Due to the nonlinear dependence of the non-actuated joint coordinate's second derivative on the joint coordinates, their first derivatives and the second derivatives of the actuated joint coordinates, see (4.11), the discrete time system equations (4.10) and (4.12) are nonlinear. However, iLQR requires a linearization of this nonlinear dynamics and so the derivatives $\frac{\partial \ddot{\mathbf{q}}_U}{\partial \mathbf{q}}$, $\frac{\partial \ddot{\mathbf{q}}_U}{\partial \dot{\mathbf{q}}}$ and $\frac{\partial \ddot{\mathbf{q}}_U}{\partial \ddot{\mathbf{q}}_A}$ are required. The expression in (4.7) gives the following derivatives

$$\frac{\partial \ddot{\mathbf{q}}_U}{\partial \mathbf{q}} = -\Big(\frac{\partial \boldsymbol{\tau}_U}{\partial \ddot{\mathbf{q}}_U} \Big)^{-1} \frac{\partial \boldsymbol{\tau}_U}{\partial \mathbf{q}} = -\mathbf{D}_{UU}(\mathbf{q})^{-1} \frac{\partial \boldsymbol{\tau}_U}{\partial \mathbf{q}} \tag{4.13a}$$

$$\frac{\partial \ddot{\mathbf{q}}_U}{\partial \dot{\mathbf{q}}} = -\Big(\frac{\partial \boldsymbol{\tau}_U}{\partial \ddot{\mathbf{q}}_U} \Big)^{-1} \frac{\partial \boldsymbol{\tau}_U}{\partial \dot{\mathbf{q}}} = -\mathbf{D}_{UU}(\mathbf{q})^{-1} \frac{\partial \boldsymbol{\tau}_U}{\partial \dot{\mathbf{q}}} \tag{4.13b}$$

$$\frac{\partial \ddot{\mathbf{q}}_U}{\partial \ddot{\mathbf{q}}_A} = -\Big(\frac{\partial \boldsymbol{\tau}_U}{\partial \ddot{\mathbf{q}}_U} \Big)^{-1} \frac{\partial \boldsymbol{\tau}_U}{\partial \ddot{\mathbf{q}}_A} = -\mathbf{D}_{UU}(\mathbf{q})^{-1} \mathbf{D}_{UA}(\mathbf{q}) \ . \tag{4.13c}$$

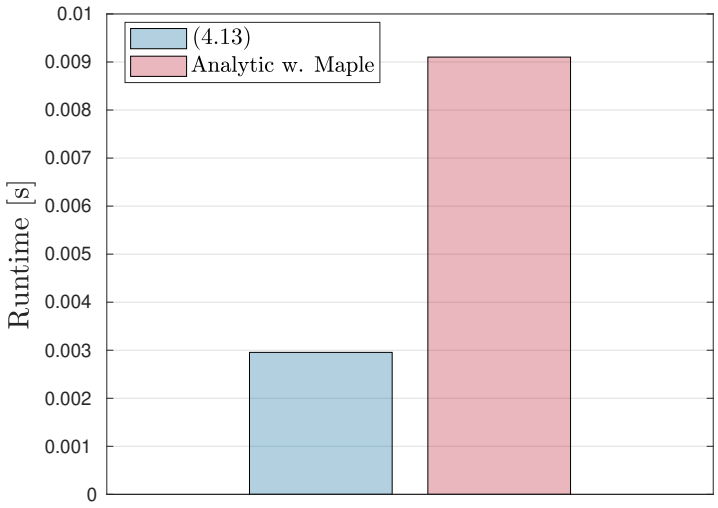


Figure 4.2: Mean runtime for derivative computation using 100 repetitions.

This method to compute the derivatives aims at avoiding symbolic differentiation using MAPLE, which can be quite time consuming considering that $\mathbf{D}_{UU}(\mathbf{q})$ can be a very complex expression that has to be inverted before the derivative is computed. Figure 4.2 shows the mean measured runtime for both variants of derivative computation using 100 repetitions. It can be seen that using the formulation based on (4.13) instead of the symbolic maple computation yields a speedup of 3.

4.1.3 Collision Constraints

In order to incorporate the collision constraints, a simplified geometrical model of the crane is used according to in Figure 4.3. The relevant links $\mathcal{L}_i(\mathbf{q})$, $i = 1, 2, 3$ are modeled as line segments from which a certain minimum distance is required, which geometrically leads to the red capsules that can be seen in Figure 4.3. Note that the space occupied by these capsules depends on the actual joint coordinates $\mathbf{q} \in \mathbb{R}^n$. The collision objects $\mathcal{O}_i$, $i = 1, \ldots, 8$, i.e. the truck in these experiments, are modeled using static oriented bounding boxes (OBB), visualized by the blue transparent cuboids in Figure 4.3. The minimal distances between the crane and the collision objects are computed using the GJK algorithm described in Appendix A. Every object is defined by its support mapping, which is

$$\mathbf{s}_{\mathcal{L}_i(\mathbf{q})}(\mathbf{d}) = \begin{cases} \mathbf{p}_{i1} & , \text{ if } \mathbf{d}^{\mathrm{T}}(\mathbf{p}_{i1} - \mathbf{p}_{i2}) > 0 \\ \mathbf{p}_{i2} & , \text{ otherwise} \end{cases} \qquad i = 1, 2, 3 \tag{4.14}$$

for the line segments, where $\mathbf{p}_{i1}, \mathbf{p}_{i2} \in \mathbb{R}^3$, $i = 1, 2, 3$ denote the endpoints of the corresponding line. Note that if the line is parallel to a surface of a bounding box the support mapping is in general not unique. However, in this case the point $\mathbf{p}_{i2}$ is returned in this case. OBBs could be modeled by projecting each endpoint onto $\mathbf{d} = [d_x, d_y, d_z]^{\mathrm{T}} \in \mathbb{R}^3$.

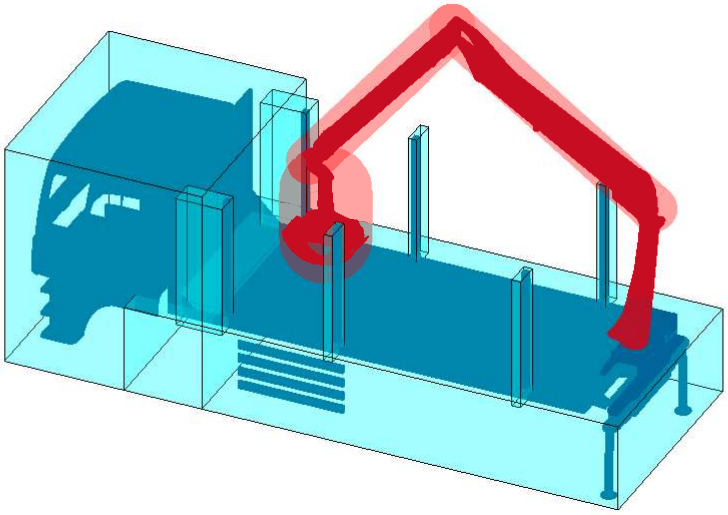


Figure 4.3: Collision model for the timber crane.

However, there is a computationally cheaper method. Let $\mathbf{c}_j \in \mathbb{R}^3$, $j = 1, \ldots, 8$ be the origins of the OBBs and let $\mathbf{R}_j \in \mathcal{SO}(3)$, $j = 1, \ldots, 8$ be the orientations. Further let $\mathbf{l}_j = [l_{xj}, l_{yj}, l_{zj}]^{\mathrm{T}}$ be the length of the edges in x-, y- and z-direction. Then the support mapping can be computed as

$$\mathbf{s}_{\mathcal{O}_j}(\mathbf{d}) = \mathbf{c}_j + \mathbf{R}_j\Big(\mathrm{diag}\big(\mathrm{sign}(\mathbf{d})\big)\frac{\mathbf{l}_j}{2}\Big) \qquad j = 1, \ldots, 8\ , \tag{4.15}$$

where

$$\mathrm{sign}(\mathbf{d}) = \begin{bmatrix} \mathrm{sign}(d_x) \\ \mathrm{sign}(d_y) \\ \mathrm{dign}(d_z) \end{bmatrix} \qquad \text{with} \qquad \mathrm{sign}(d) = \begin{cases} 1 & ,\ d \geq 0 \\ 0 & ,\ d < 0 \end{cases}. \tag{4.16}$$

Self collision is avoided by the limits of the joint angles. Hence, for every link in the crane covered by a red capsule in Figure 4.3 $\mathcal{L}_i(\mathbf{q})$, $i = 1, 2, 3$ and every object representing the truck $\mathcal{O}_i$, $i = 1, \ldots, 8$, there is one collision constraint of the form

$$h_{8(i-1)+j}(\mathbf{q}) = -\Big(d(\mathcal{L}_i(\mathbf{q}), \mathcal{O}_j) - r_i\Big) \leq 0 \text{ m}, \qquad i = 1, 2, 3 \text{ and } j = 1 \ldots 8 \tag{4.17}$$

where d is the distance between $\mathcal{L}_i(\mathbf{q})$ and $\mathcal{O}_j$ computed by the GJK algorithm as discussed in Appendix A and r_i is the radius of the capsule around the link. Hence, for $n_{\mathcal{L}} = 3$ links and $n_{\mathcal{O}} = 8$ objects there are $n_{\mathcal{L}} \cdot n_{\mathcal{O}}$ collision constraints in total which are stacked into the vector inequality

$$\begin{bmatrix} h_1(\mathbf{q}) \\ \vdots \\ h_{n_{\mathcal{L}} \cdot n_{\mathcal{O}}}(\mathbf{q}) \end{bmatrix} =: \mathbf{h}(\mathbf{q}) \leq \mathbf{0}\ , \tag{4.18}$$

where $\mathbf{h} : \mathbb{R}^n \to \mathbb{R}^{n_\mathcal{L} \cdot n_\mathcal{O}}$ is assumed to be sufficiently smooth such that the required derivatives do exist. However, note that differentiability of the distance function is not guaranteed in every case. In order to use the constrained iLQR techniques presented in Chapter 2 a linear approximation of the constraints and hence the Jacobian

$$\mathbf{J}_{\mathbf{h},\mathbf{q}}(\mathbf{q}) := \frac{\partial \mathbf{h}(\mathbf{q})}{\partial \mathbf{q}} \tag{4.19}$$

is required. The method described in [34] is used to approximate the Jacobian (4.19). Let $\mathbf{p}_i(\mathbf{q}) \in \mathcal{L}_i(\mathbf{q})$ be the point on the link and $\mathbf{p}_j \in \mathcal{O}_j$ be the point on the object such that they are the closest points to each other. Then the distance between $\mathcal{L}_i(\mathbf{q})$ and $\mathcal{O}_j$ can be written as

$$d(\mathcal{L}_i(\mathbf{q}), \mathcal{O}_j) = \mathbf{n}^\mathrm{T}(\mathbf{p}_i(\mathbf{q}) - \mathbf{p}_j) \qquad \text{with} \qquad \mathbf{n} = \frac{\mathbf{p}_i(\mathbf{q}) - \mathbf{p}_j}{\|\mathbf{p}_i(\mathbf{q}) - \mathbf{p}_j\|_2} \,. \tag{4.20}$$

Assuming that $\mathbf{p}_i(\mathbf{q})$ and $\mathbf{p}_j$ are fixed in their local frames and that $\mathbf{n}$ does not change, we get

$$\frac{\partial d(\mathcal{L}_i(\mathbf{q}), \mathcal{O}_j)}{\partial \mathbf{q}} \approx \mathbf{n}^\mathrm{T} \mathbf{J}_{\mathbf{p}_i,\mathbf{q}}(\mathbf{q}) \qquad \text{with} \qquad \mathbf{J}_{\mathbf{p}_i,\mathbf{q}} = \frac{\partial \mathbf{p}_i(\mathbf{q})}{\partial \mathbf{q}} \,. \tag{4.21}$$

4.2 Offline Planning

In summary, the overall optimization problem to solve the trajectory planning task reads as

$$\min_{\mathbf{u}_0,\ldots,\mathbf{u}_{N-1} \in \mathbb{R}^m} \quad J(\mathbf{u}_0, \ldots, \mathbf{u}_{N-1}) \tag{4.22a}$$

s.t.

$$\mathbf{q}_{A,k+1} = \mathbf{q}_{A,k} + \dot{\mathbf{q}}_{A,k} T_s + \frac{1}{2} \mathbf{u}_k T_s^2 \qquad \text{(Sys. dyn.)} \tag{4.22b}$$

$$\mathbf{q}_{U,k+1} = \mathbf{q}_{U,k} + \dot{\mathbf{q}}_{U,k} T_s + \frac{1}{2} \ddot{\mathbf{q}}_{U,k}(\mathbf{q}_k, \dot{\mathbf{q}}_k, \ddot{\mathbf{q}}_{A,k}) T_s^2 \tag{4.22c}$$

$$\dot{\mathbf{q}}_{A,k+1} = \dot{\mathbf{q}}_{A,k} + \mathbf{u} T_s \tag{4.22d}$$

$$\dot{\mathbf{q}}_{U,k+1} = \dot{\mathbf{q}}_{U,k} + \ddot{\mathbf{q}}_U(\mathbf{q}_k, \dot{\mathbf{q}}_k, \ddot{\mathbf{q}}_{A,k}) T_s \tag{4.22e}$$

$$\tag{4.22f}$$

$$\mathbf{h}(\mathbf{q}) \leq \mathbf{0} \text{ m} \qquad \text{(Collisions)}$$

$$\tag{4.22g}$$

$$\underline{\mathbf{u}} \leq \mathbf{u}_k \leq \overline{\mathbf{u}} \qquad \text{(Ctrl. limits)}$$

$$\tag{4.22h}$$

$$\underline{\mathbf{q}} \leq \mathbf{q}_k \leq \overline{\mathbf{q}} \qquad \text{(State limits)}$$

with

$$J(\mathbf{u}_0, \ldots, \mathbf{u}_{N-1}) = \varphi(\mathbf{q}_{A,N}, \dot{\mathbf{q}}_N) + \sum_{k=0}^{N-1} \Big(l_1(\dot{\mathbf{q}}_k) + l_2(\mathbf{u}_k) \Big) T_s \tag{4.23a}$$

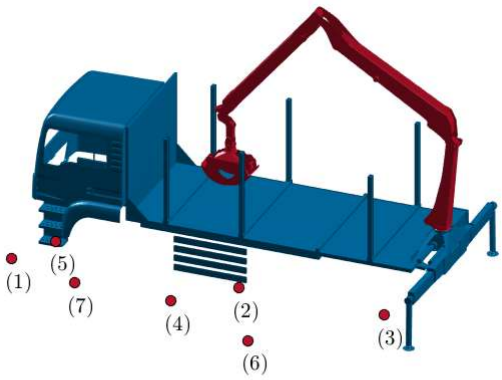


Figure 4.4: Experimental setup for the crane trajectory planning problem.

where

$$\varphi(\mathbf{q}_A, \dot{\mathbf{q}}) = (\mathbf{q}_A - \mathbf{q}_{A,d})^{\mathrm{T}}\mathbf{P}(\mathbf{q}_A - \mathbf{q}_{A,d}) + \dot{\mathbf{q}}^{\mathrm{T}}\mathbf{R}\dot{\mathbf{q}} \tag{4.23b}$$

$$l_1(\dot{\mathbf{q}}) = \dot{\mathbf{q}}\mathbf{V}\dot{\mathbf{q}} \tag{4.23c}$$

$$l_2(\mathbf{u}) = \mathbf{u}^{\mathrm{T}}\mathbf{Q}\mathbf{u}\ . \tag{4.23d}$$

The desired end configuration is denoted by $\mathbf{q}_{A,d} \in \mathbb{R}^{n_A}$ and $\mathbf{P}, \mathbf{R}, \mathbf{V}$ as well as $\mathbf{R}$ are positive (semi-) definite diagonal weighting matrices. The sampling time is chosen as $T_s = 100$ ms and the horizon length is $N = 50$. The matrix $\mathbf{P} \in \mathbb{R}^{n_A \times n_A}$ penalizes deviations from the desired end configuration and with the diagonal elements being 500. Note that only the actuated joints can be freely chosen, which is why the end configuration of non-actuated joints are not considered in (4.23b). Remaining velocities at the last trajectory sample are penalized using the matrix $\mathbf{R} \in \mathbb{R}^{n \times n}$. The diagonal elements here are chosen as 500 as well. For all other time instances of the trajectory there are two terms serving as regularization. The first one comes with the weighting matrix $\mathbf{V} \in \mathbb{R}^{n \times n}$ and aims at penalizing large velocities. This is especially important in order to avoid abrupt motions of the non-actuated joints, which is why velocities of the non-actuated joints are penalized more than the actuated ones. To be more specific, the diagonal elements are chosen as 1 for the non-actuated degrees of freedom and 0.1 for the actuated ones. The second term penalizes large control inputs and thus accelerations. The matrix $\mathbf{Q} \in \mathbb{R}^{m \times m}$ is chosen as diagonal matrix with 1 as a diagonal elements.

The experimental setup is visualized in Figure 4.4. The initial state is visualized by the actual configuration of the red crane in Figure 4.4. The goal is to compute collision-free trajectories that additionally fulfill the control and state limits to reach seven different desired end configurations, visualized by the red dots (1)-(7) in Figure 4.4. However, note that the end configurations are given by the generalized coordinates $\mathbf{q}_{A,d} \in \mathbb{R}^{n_A}$, but for clarity the red points in Figure 4.4 visualize the corresponding Euclidean positions of the

frame $\mathcal{F}_5$. The control limits are chosen as

$$\overline{\mathbf{u}} = \begin{bmatrix} 0.3 \text{ rad/s}^2 & 0.3 \text{ rad/s}^2 & 0.3 \text{ rad/s}^2 & 0.3 \text{ rad/s}^2 & 0.3 \text{ rad/s}^2 \end{bmatrix}^{\mathrm{T}} \tag{4.24a}$$

$$\underline{\mathbf{u}} = \begin{bmatrix} -0.3 \text{ rad/s}^2 & -0.3 \text{ rad/s}^2 & -0.3 \text{ rad/s}^2 & -0.3 \text{ rad/s}^2 & -0.3 \text{ rad/s}^2 \end{bmatrix}^{\mathrm{T}} \tag{4.24b}$$

and for the state limits

$$\overline{\mathbf{q}} = \begin{bmatrix} 212.57° & 89.5° & 263.56° & 2.24 \text{ m} & 89.95° & 180° & 360° \end{bmatrix}^{\mathrm{T}} \tag{4.25a}$$

$$\underline{\mathbf{q}} = \begin{bmatrix} -212.57° & -63.03° & -52.14° & 0 \text{ m} & -89.95° & 0° & -360° \end{bmatrix}^{\mathrm{T}} \tag{4.25b}$$

is used.

In the following experiments, the iLQR algorithms from Chapter 2 are evaluated and compared regarding their performance in solving the optimal control problem (4.22).

4.2.1 Augmented Lagrangian

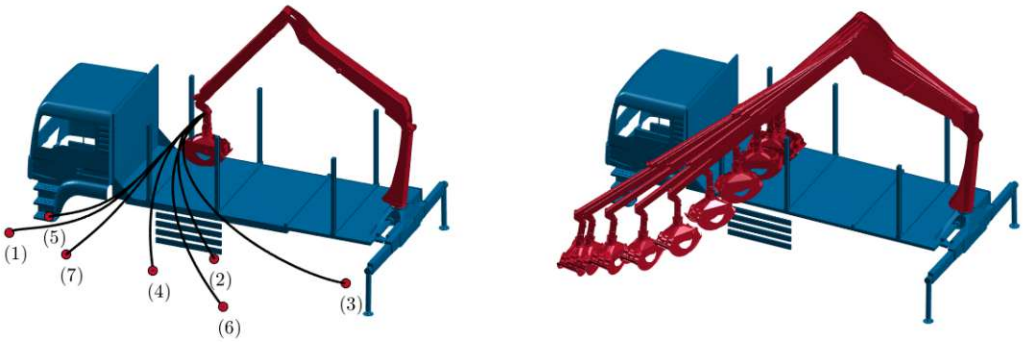


Figure 4.5: Resulting state trajectories of the augmented Lagrangian method.

Figure 4.5 shows the resulting trajectories generated by the augmented Lagrangian method, while being initialized with a zero velocity and acceleration. The red points in the left image depict the locations of the coordinate frame $\mathcal{F}_5$ from Figure 4.1, which result from the desired end points $\mathbf{q}_{A,d}$ and the black lines show the corresponding computed trajectories. Additionally, the joint coordinates of every fifth point of trajectory (1) are visualized on the right-hand side of Figure 4.5 and Figure 4.6 shows the corresponding control trajectories with their limits. Note that these are just illustrative examples in order to avoid overloading of the figures. The qualitative results for the other trajectories are equivalent to those shown in Figure 4.5 and Figure 4.6. There is no qualitative difference in the results generated by the methods using the different penalty-Lagrangian functions (2.30), (2.33) and (2.36), which is why only the results generated by the one used in

the ALTRO approach are shown in Figure 4.5 and Figure 4.6. Hence, the algorithm is able to find collision-free trajectories that additionally fulfill all control and state limits. Furthermore, abrupt movements of the non-actuated degrees of freedom is avoided.

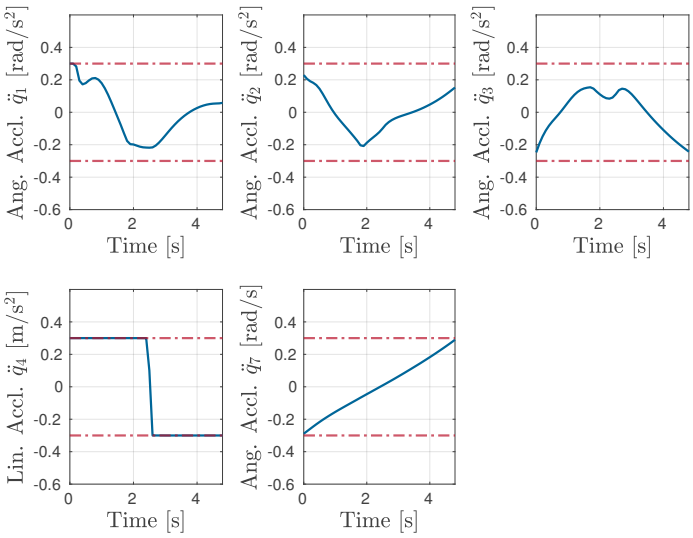
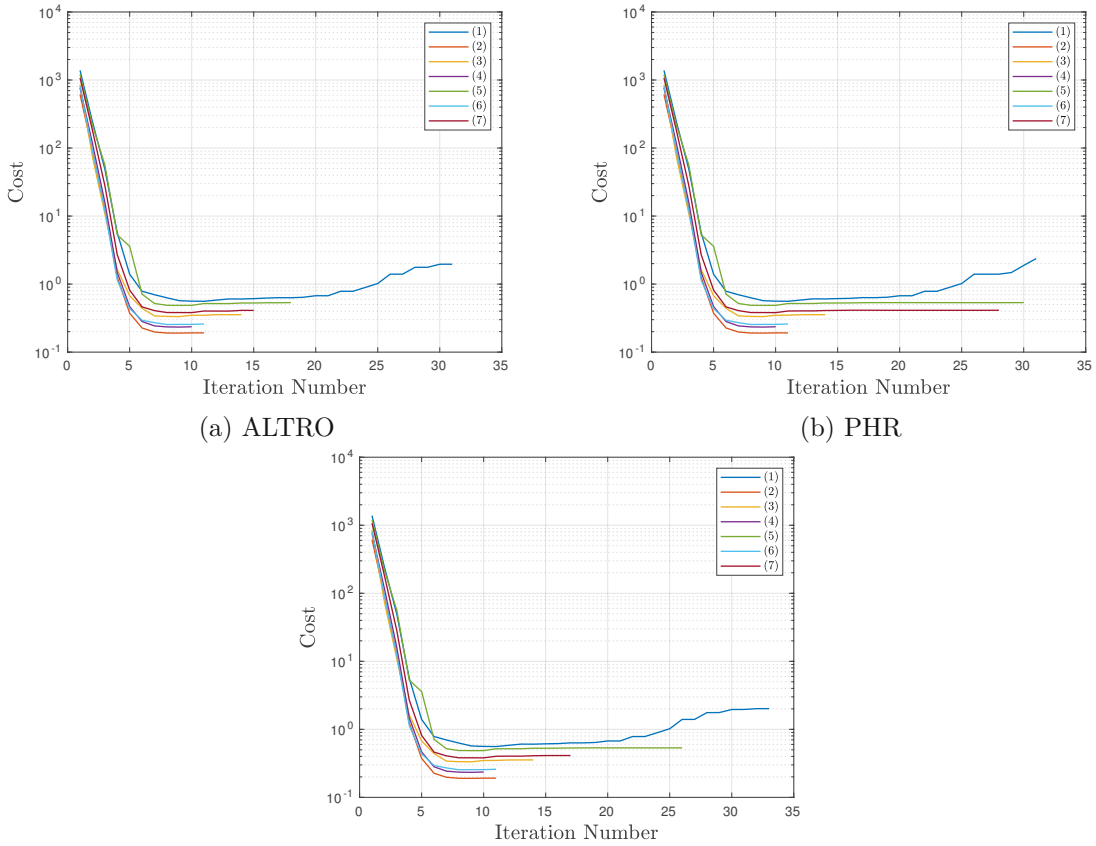


Figure 4.6: Resulting control trajectories for target configuration (1) using the augmented Lagrangian method.

Figure 4.7 shows the cost evolution of the augmented Lagrangian method with the penalty-Lagrangian functions (2.30), (2.33) and (2.36), respectively. As already discussed in Chapter 3, the cost increases in iterations where the penalty factor is increased. It seems that for this application the ALTRO approach performs better than the PHR and S-PHR approaches. However, the influence of the penalty-Lagrangian function on the convergence behavior is limited.



(a) ALTRO

(b) PHR

(c) S-PHR

Figure 4.7: Cost evolution of the augmented Lagrangian method.

4.2.2 Augmented Lagrangian combined with Projected Newton

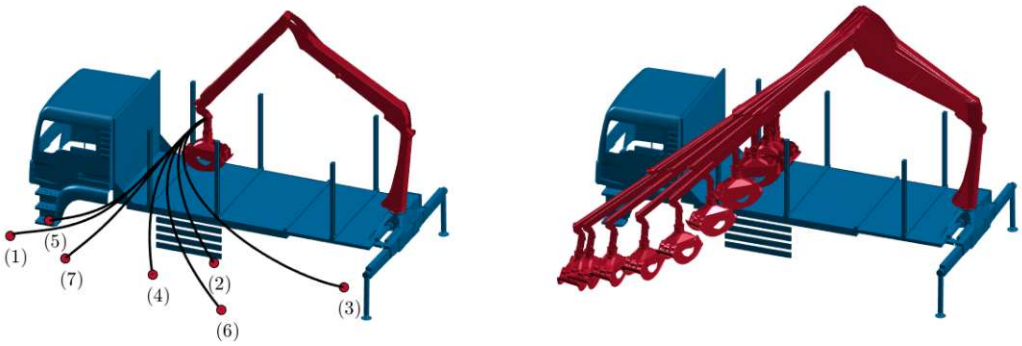


Figure 4.8: Resulting state trajectories of the combined AL with projected Newton method.

Figure 4.8 shows the resulting trajectories generated by the combined augmented Lagrangian with the projected Newton approach, while being initialized with zero velocity and accelerations. Similar to the pure augmented Lagrangian approach, this algorithm is able to generate collision-free trajectories in all seven cases, while the control and state limits are observed. Additionally, on the right-hand side of Figure 4.8 trajectory (1) generated by the combined augmented Lagrangian with the projected Newton method is visualized for every fifth trajectory sample in joint coordinates. Figure 4.9 shows the corresponding control inputs. Again, note that this trajectory is used as one illustrative example and the results of the other ones are qualitatively equal. Additionally, there is no qualitative difference in the results of the algorithms using the different penalty-Lagrangian functions (2.30), (2.33) and (2.36), which is why only the results of the algorithm with the ALTRO penalty-Lagrangian are depicted in Figure 4.8 and Figure 4.9. Hence, the algorithm is able to find trajectories that fulfill all constraints and avoid abrupt motions of the non-actuated joints.

Figure 4.10 shows the cost evolution of the trajectories generated by the combined augmented Lagrangian and the projected Newton method. Note that while the cost function (4.23) is depicted in Figure 4.10, the algorithm minimizes the augmented Lagrangian (2.25). Hence, the step size selection strategy forces the augmented Lagrangian (2.25) to decrease and not the cost (4.23), which is why the costs in Figure 4.10 can increase even without an increase in the penalties. The reason for depicting the costs (4.23) and not the augmented Lagrangian (2.25) is to be able to compare the evolution of the different algorithms (the interior point method does not minimize the augmented Lagrangian and the pure and combined approaches differ in that the latter ones do not penalize control violations).

As already discussed in Chapter 3, the projected Newton method might yield bad search directions if the active set estimate is inaccurate. In this application, the cost decrease in earlier iterations suffers even more than in the 2D car example from Chapter 3. Again, the

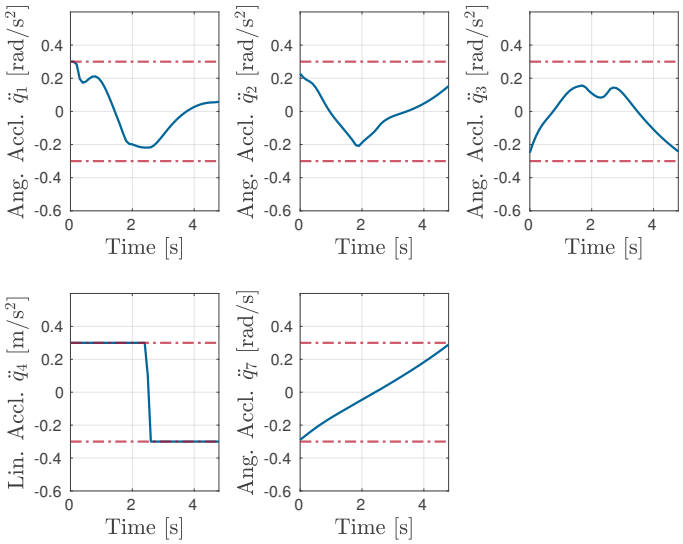
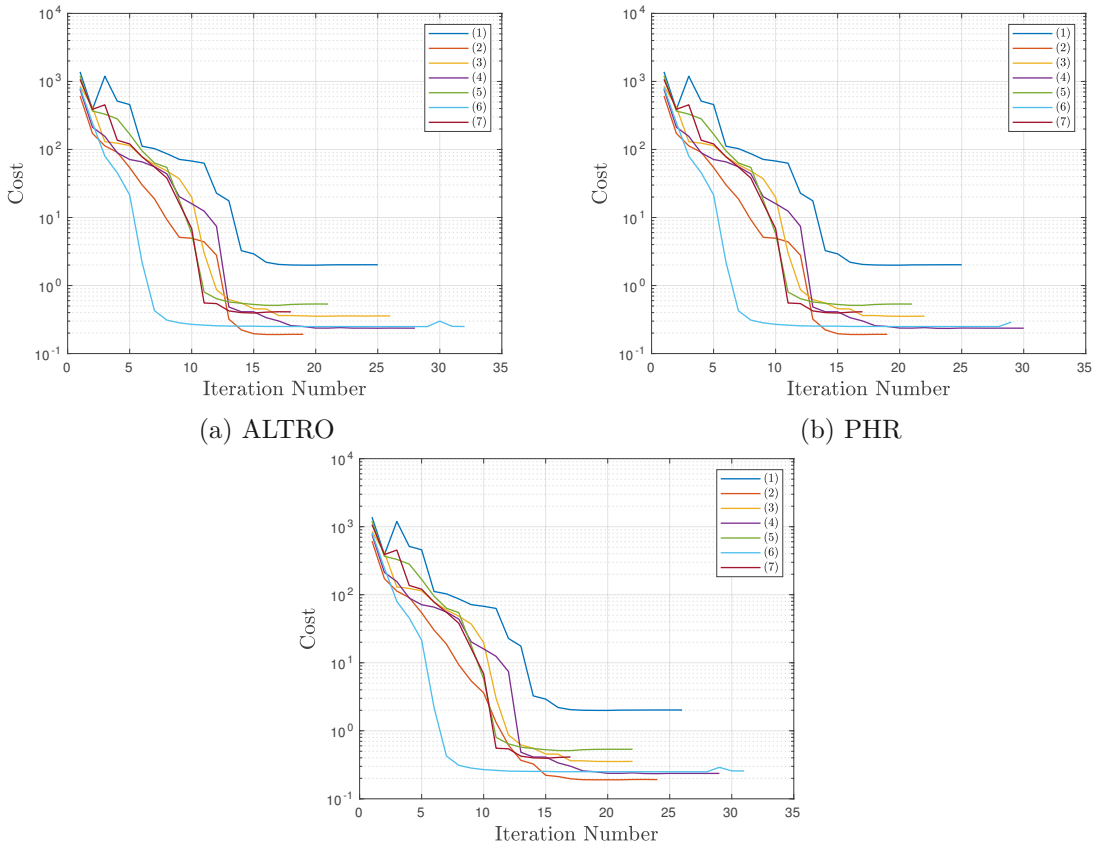


Figure 4.9: Resulting control trajectories using the combined AL with projected Newton method.

specific choice in penalty-Lagrangian function has only little influence on the convergence behavior.



(a) ALTRO

(b) PHR

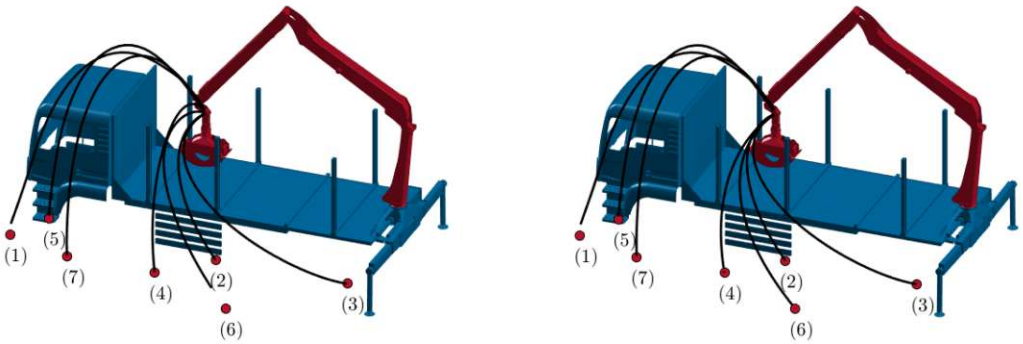
(c) S-PHR

Figure 4.10: Cost evolution of the combined AL with projected Newton method.

4.2.3 Primal-Dual Interior Point

Figure 4.11 shows trajectories generated by the primal-dual interior point based iLQR method, while being initialized with zero velocity and accelerations. While the trajectories of the feasible interior point method are depicted in Figure 4.11a, Figure 4.11b shows the trajectories computed by the infeasible interior point algorithm. From Figure 4.11a it can be seen that the feasible interior point algorithm is not able to generate trajectories that reach the desired endpoints for (1) and (6). Due to the barrier approach to handle the inequality constraints, the solution stays feasible during the whole optimization procedure and the algorithm tends to avoid narrow areas. Hence the algorithm prefers a longer path to the desired end configuration and the control limits are too small to actually reach these configurations. Since the infeasible interior point algorithm allows constraint violations (but keeps them as small as possible), it is actually able to reach end configuration (6), as can be seen in Figure 4.11b. Hence, the infeasible interior point algorithm has some advantages in narrow environments compared to the feasible one.

Figure 4.12 shows the control trajectory that belongs to trajectory (1) in Figure 4.11. Since there is no qualitative difference in the time evolution of the feasible and the



(a) Feasible Interior Point (b) Infeasible Interior Point

Figure 4.11: Resulting state trajectories using the primal-dual interior point method.

infeasible interior point algorithm, only the results of the former one are visualized in Figure 4.12.

Figure 4.13 shows the cost evolution of the interior point based constrained iLQR. It can be seen that far more iterations are required compared to the augmented Lagrangian method in Figure 4.7 and the combined approach in Figure 4.10. Similarly, the cost reduction is much slower compared to the augmented Lagrangian method. As discussed in Chapter 3, the cost decreases when the barrier parameters are reduced, which is in contrast to the augmented Lagrangian method. Again, note that the cost function (4.23) is not directly optimized by the interior point method, but rather (2.40) and (2.56), respectively.

All in all, the results suggest that the augmented Lagrangian approach is better suited to incorporate collision constraints compared to feasible and infeasible primal-dual interior point methods.

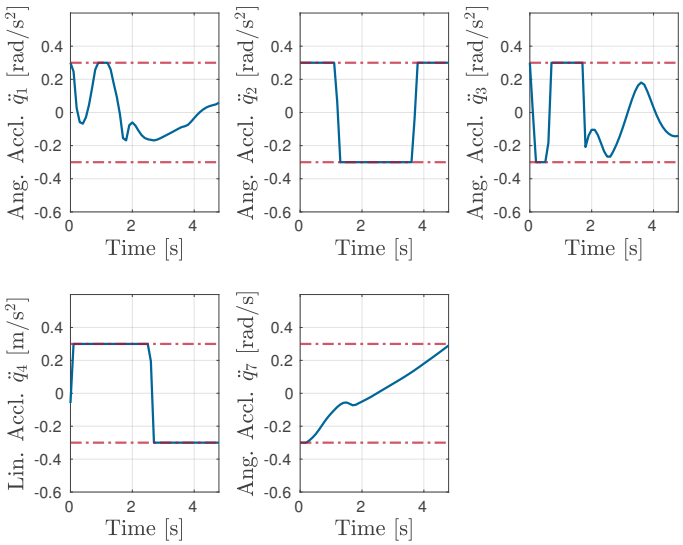
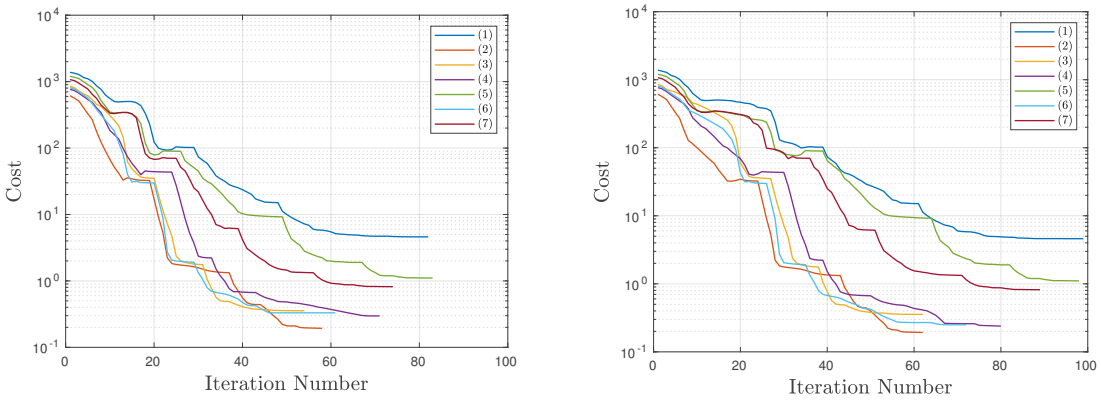


Figure 4.12: Resulting control trajectories using the primal-dual interior point method.



(a) Feasible Interior Point (b) Infeasible Interior Point

Figure 4.13: Cost evolution of the primal-dual interior point method.

4.2.4 Runtime Comparison

This section deals with the runtimes required by the different constrained iLQR methods. The algorithms are implemented in MATLAB and converted to MEX using MATLAB `codegen`. For all seven end configurations, the runtime is measured 100 times for each algorithm and the resulting distributions are depicted in Figure 4.14. Hence, the pure augmented Lagrangian approach is clearly the fastest for this application.

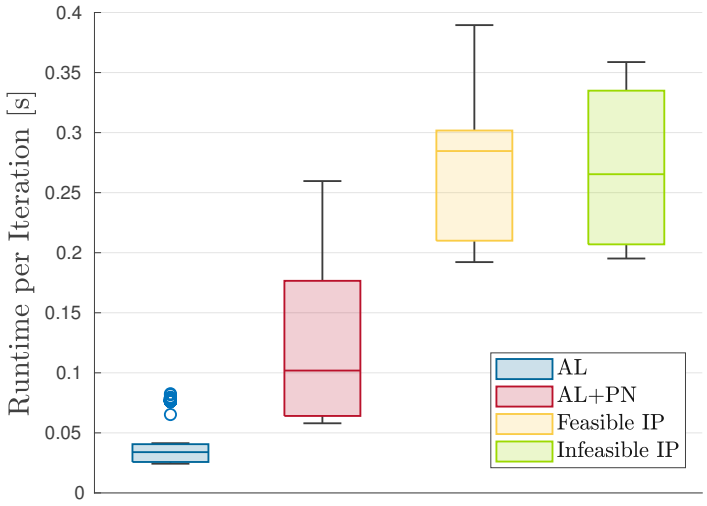


Figure 4.14: Runtimes of different trajectory optimization techniques (**blue:** augmented Lagrangian, **red:** augmented Lagrangian with projected Newton, **yellow:** feasible primal-dual interior point, **green:** infeasible primal-dual interior point).

The primal-dual interior point algorithms require higher computation times compared to the other iLQR based methods. Figure 4.15 depicts the average runtime per iteration for the different algorithms. It shows that, even though the overall runtime of the interior point based method is worse than the augmented Lagrangian based ones, the computation time required per iteration is still comparable to the pure augmented Lagrangian approach. Hence, the main reason for the slower performance of the interior point method is the large number of iterations required, which, however, originates from the barrier nature of the algorithm. Especially at the beginning, the algorithm requires a large number of iterations until the trajectory starts passing the objects. Figure 4.16 shows the step size distribution of the algorithms. The pure augmented Lagrangian approach clearly uses larger step sizes than both interior point methods, which is the source of the remaining difference in runtime shown in Figure 4.15. Additionally, it can be observed that the infeasible interior point algorithm achieves lower runtimes per iteration than the feasible counterpart, even though the step size distributions in Figure 4.16 are comparable. The infeasible interior point algorithm does not require additional evaluation of the inequality constraints for the step size selection procedure, which is why one iteration can be faster

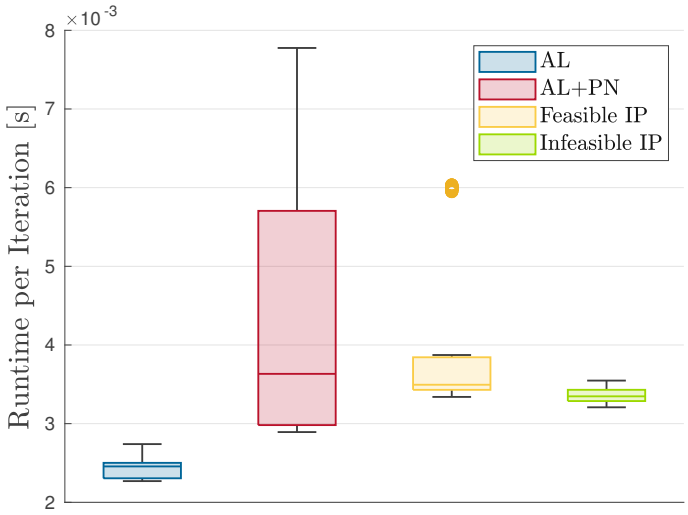


Figure 4.15: Mean runtimes per iteration of different trajectory optimization techniques (**blue:** augmented Lagrangian, **red:** augmented Lagrangian with projected Newton, **yellow:** feasible primal-dual interior point, **green:** infeasible primal-dual interior point).

than for the feasible interior point method.

The relation of the runtimes in Figure 4.14 and the runtimes per iteration in Figure 4.15 of the pure augmented Lagrangian approach and the one combined with the projected Newton method is quite similar. Figure 4.16 shows that one source of runtime difference is the requirement for smaller step sizes of the combined approach. However, while the mean runtime of the combined approach is smaller than the one of the interior point methods, Figure 4.15 shows that the mean runtime per iteration becomes larger, even tough the average step size in Figure 4.16 is still larger. Hence, the larger runtime per iteration originates from the underlying constrained quadratic programming algorithm.

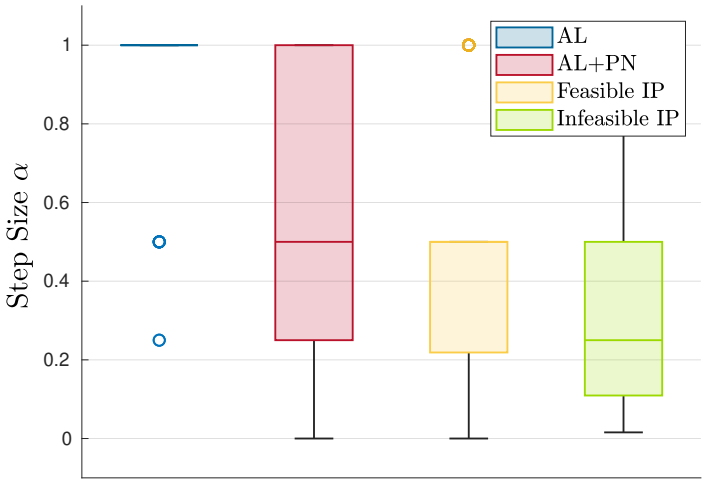


Figure 4.16: Step size distribution of the different trajectory optimization techniques (**blue:** augmented Lagrangian, **red:** augmented Lagrangian with projected Newton, **yellow:** feasible primal-dual interior point, **green:** infeasible primal-dual interior point).

4.3 Model Predictive Control

This section deals with the implementation of an MPC based online trajectory planner using the iLQR algorithms from Chapter 2. The requirements for the MPC are as follows:

R1) The MPC has to be able to bring the crane to a desired end configuration without any collisions and without any other constraint violations.

R2) The MPC has to be able to stabilize the non-actuated degrees of freedom on arrival at the desired end configuration, i.e. the velocity of the non-actuated joints has to be zero and no oscillations shall occur.

R3) The MPC has to be able to compute the control command fast enough, i.e. an upper limit is given by the sampling time.

The following experiments focus on the ability of the MPC algorithm to fulfill R1)-R3). Note that, in general, there are many more requirements, e.g. robustness to parameter uncertainties and so on. However, since the main focus of this thesis is the evaluation of iLQR algorithms for collision-free motion planning problems, no further studies regarding robustness are performed. Additionally, the actuated joints of the crane are assumed to be velocity controlled and the design of these lower level controllers is out of scope of this thesis. Due to these reasons, inaccuracies due to tracking errors of the velocity controllers are not studied here.

The MPC computes acceleration commands $\mathbf{u}_d(\tilde{k}T_s) = \mathbf{u}^*_{\tilde{k}} \in \mathbb{R}^m$ at each discrete time $\tilde{k} \in \mathbb{N}$ as a solution of the optimal control problem

$$\min_{\mathbf{u}_{\tilde{k}},\dots,\mathbf{u}_{\tilde{k}+N-1}\in\mathbb{R}^m} \quad J(\mathbf{u}_{\tilde{k}},\dots,\mathbf{u}_{\tilde{k}+N-1}) \tag{4.26a}$$

$$\text{s.t.} \quad \mathbf{q}_{A,k+1} = \mathbf{q}_{A,k} + \dot{\mathbf{q}}_{A,k}T_s + \frac{1}{2}\mathbf{u}_k T_s^2 \quad \text{(Sys. dyn.)} \tag{4.26b}$$

$$\mathbf{q}_{U,k+1} = \mathbf{q}_{U,k} + \dot{\mathbf{q}}_{U,k}T_s + \frac{1}{2}\ddot{\mathbf{q}}_{U,k}(\mathbf{q}_k, \dot{\mathbf{q}}_k, \ddot{\mathbf{q}}_{A,k})T_s^2 \tag{4.26c}$$

$$\dot{\mathbf{q}}_{A,k+1} = \dot{\mathbf{q}}_{A,k} + \mathbf{u}T_s \tag{4.26d}$$

$$\dot{\mathbf{q}}_{U,k+1} = \dot{\mathbf{q}}_{U,k} + \ddot{\mathbf{q}}_U(\mathbf{q}_k, \dot{\mathbf{q}}_k, \ddot{\mathbf{q}}_{A,k})T_s \tag{4.26e}$$

$$\tag{4.26f}$$

$$\mathbf{h}(\mathbf{q}) \leq \mathbf{0} \quad \text{(Collisions)} \tag{4.26g}$$

$$\underline{\mathbf{u}} \leq \mathbf{u}_k \leq \overline{\mathbf{u}} \quad \text{(Ctrl. limits)} \tag{4.26h}$$

$$\underline{\mathbf{q}} \leq \mathbf{q}_k \leq \overline{\mathbf{q}} \quad \text{(State limits)} \; .$$

The desired velocities $\dot{\mathbf{q}}_{A,d_{vc}}$ passed to the lower level velocity controllers are computed using

$$\dot{\mathbf{q}}_{A,d_{vc}}(t) = \dot{\mathbf{q}}_{A,k} + (t - kT_s)\mathbf{u}_k \; , \quad kT_s \leq t < (k+1)T_s \; , \tag{4.27}$$

where $\dot{\mathbf{q}}_{A,k} = \dot{\mathbf{q}}_A(kT_s)$ is the measured or estimated velocity of the actual system at time kT_s. Since exact velocity tracking is assumed here, the desired and true velocities and accelerations of the actuated joints are assumed to be equal.

Motivated by [35], the MPC is formulated as endpoint stabilization with the cost for the optimal control problem (4.26) being

$$J(\mathbf{u}_{\tilde{k}},\dots,\mathbf{u}_{\tilde{k}+N-1}) = \varphi(\mathbf{q}_{A,\tilde{k}+N}, \dot{\mathbf{q}}_{\tilde{k}+N}) + \sum_{k=\tilde{k}}^{\tilde{k}+N-1} l_3(\mathbf{q}_{A,k}, \dot{\mathbf{q}}_k, \mathbf{u}_k)T_s \tag{4.28}$$

with

$$\varphi(\mathbf{q}_A, \dot{\mathbf{q}}) = (\mathbf{q}_A - \mathbf{q}_{A,d})^{\mathrm{T}}\mathbf{P}(\mathbf{q}_A - \mathbf{q}_{A,d}) + \dot{\mathbf{q}}^{\mathrm{T}}\mathbf{R}\dot{\mathbf{q}} \tag{4.29a}$$

$$l_3(\mathbf{q}_A, \dot{\mathbf{q}}, \mathbf{u}) = (\mathbf{q}_A - \mathbf{q}_{A,d})^{\mathrm{T}}\mathbf{W}(\mathbf{q}_A - \mathbf{q}_{A,d}) + \dot{\mathbf{q}}\mathbf{V}\dot{\mathbf{q}} + \mathbf{u}^{\mathrm{T}}\mathbf{Q}\mathbf{u} \; , \tag{4.29b}$$

where $\mathbf{q}_{A,d} \in \mathbb{R}^{n_A}$ is the desired end configuration of the crane. The positive definite matrices $\mathbf{P} \in \mathbb{R}^{n_A \times n_A}$ and $\mathbf{R} \in \mathbb{R}^{n \times n}$ are chosen as diagonal matrices with entries 100. Similarly, $\mathbf{W} \in \mathbb{R}^{n_A \times n_A}$ is a diagonal, positive definite matrix with diagonal elements 1. For the diagonal, positive definite matrix $\mathbf{V} \in \mathbb{R}^{n \times n}$ the diagonal elements are 0.1 for those elements which penalize actuated velocities and 1 for those ones penalizing non-actuated velocities. Finally, $\mathbf{Q} \in \mathbb{R}^{m \times m}$ is a positive definite diagonal matrix with diagonal elements 1. The controller is operated with $T_s = 100$ ms and $N = 50$.

The runtime required in the first iteration is not that critical because it can be computed offline in advance. In order to speed up the optimization in the following iterations, the control trajectories gained in the previous iteration are used for initialization.

In the following, the performance of the different constrained iLQR algorithms from Chapter 2 is evaluated as underlying solver for the MPC. However, since the specific choice of the penalty-Lagrangian function (2.30), (2.33) and (2.36) has little impact on the result, (2.36) is used for the following experiments. Additionally, since the feasible interior point algorithm from Section 2.2.3 in general does no allow to initialize the algorithm with the resulting control trajectories of the previous iteration, only the infeasible counterpart is considered. The experimental setup is equal to the one used in Section 4.2 and is depicted in Figure 4.4. Point (1) visualizes the position of frame $\mathcal{F}_5$ the MPC aims to reach. However, note that in the optimization problem the desired end configuration is given by the generalized coordinates $\mathbf{q}_{A,d}$ of the actuated joints.

4.3.1 Augmented Lagrangian

Figure 4.17: Resulting state trajectory of the simulated, augmented Lagrangian based MPC-controlled crane.

The resulting trajectory of the simulated MPC-controlled crane that uses augmented Lagrangian-based iLQR as underlying solver, with the desired end configuration (1) from Section 4.2, is depicted in Figure 4.17 and Figure 4.21 shows the corresponding control inputs with their limits. The trajectory of the generalized coordinates are visualized in Figure 4.19. The MPC is able to bring the crane into the desired end configuration without any collision and any violation of the control limits. Additionally, no abrupt movements of the non-actuated joints occur. Figure 4.20 depicts the generalized velocities of the non-actuated joints. Hence, the controller is able to stabilize the non-actuated degrees of freedom in the final end configuration.

The required runtime of the MPC at each sampling time is depicted in Figure 4.18. Hence, the MPC is able to compute the control inputs with runtimes lower than $T_s = 100$ ms (even in the first iteration).

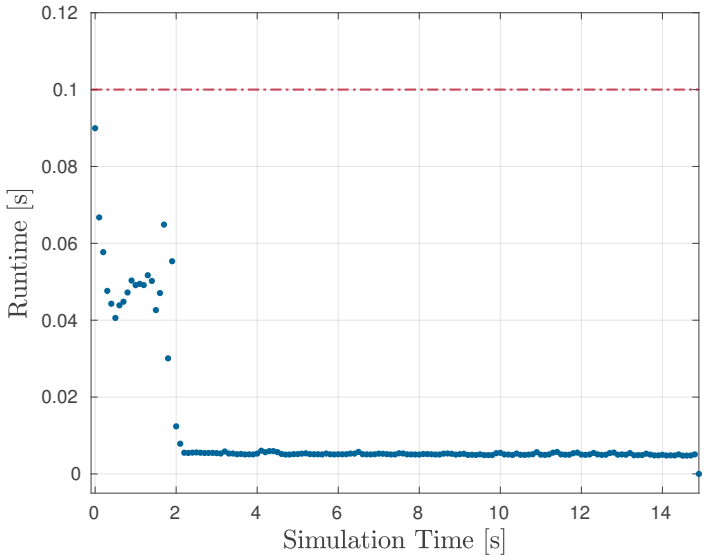


Figure 4.18: Runtime required by the MPC based on augmented Lagrangian iLQR.

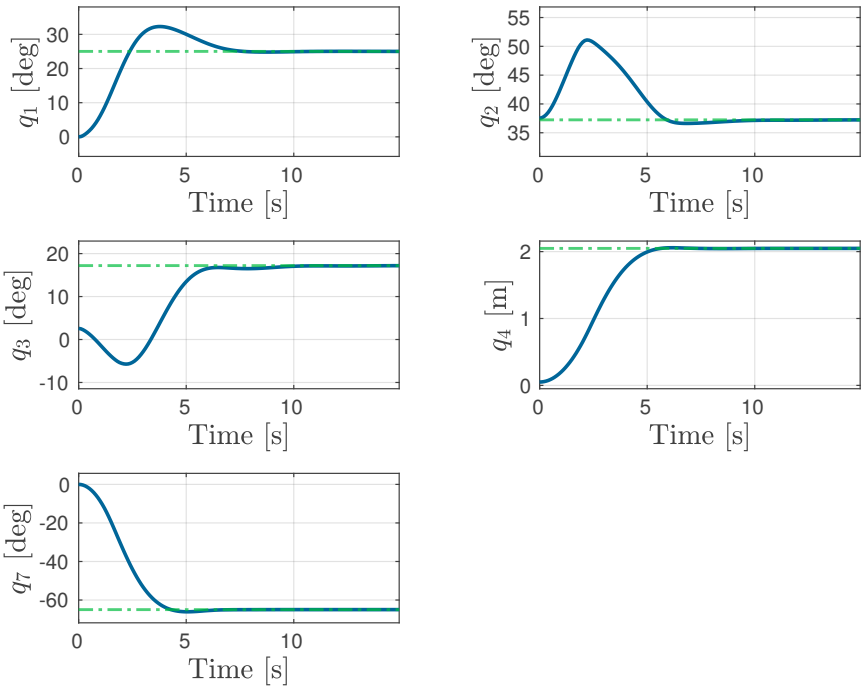


Figure 4.19: Resulting generalized actuated coordinates $\mathbf{q}_A$ of the simulated, augmented Lagrangian based MPC-controlled crane.

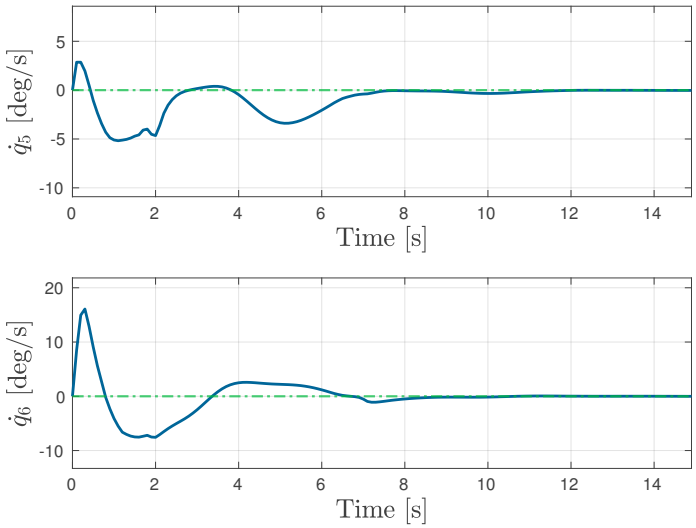


Figure 4.20: Resulting generalized non-actuated velocities $\dot{\mathbf{q}}_U$ of the simulated, augmented Lagrangian based MPC-controlled crane.

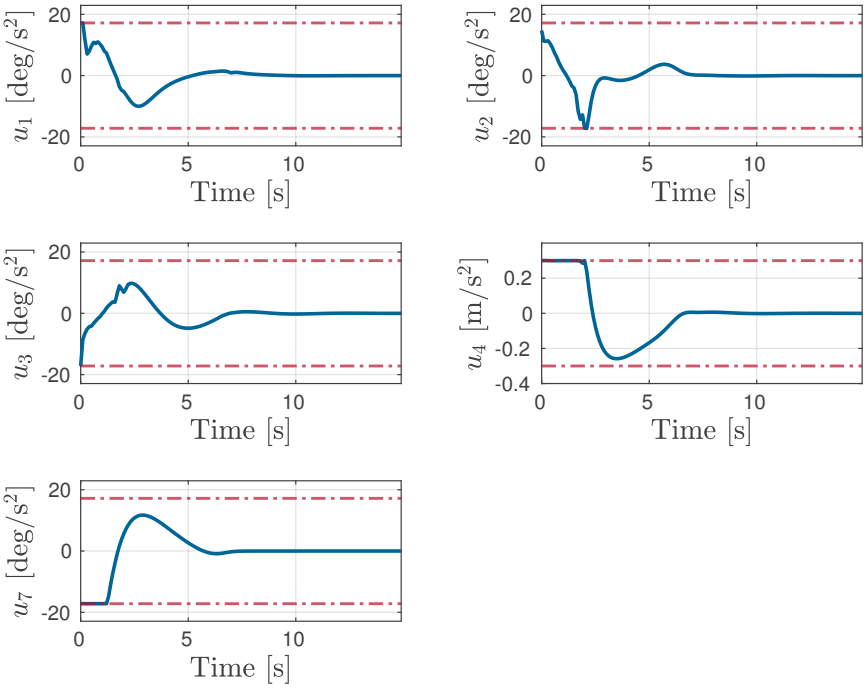


Figure 4.21: Resulting control inputs $\mathbf{u}$ of the simulated, augmented Lagrangian MPC-controlled crane.

4.3.2 Augmented Lagrangian with Projected Newton

The resulting trajectories of the combined augmented Lagrangian with the projected Newton method is qualitatively equal to the ones of the pure augmented Lagrangian approach. Hence, in order to save space, they are not explicitly outlined here.

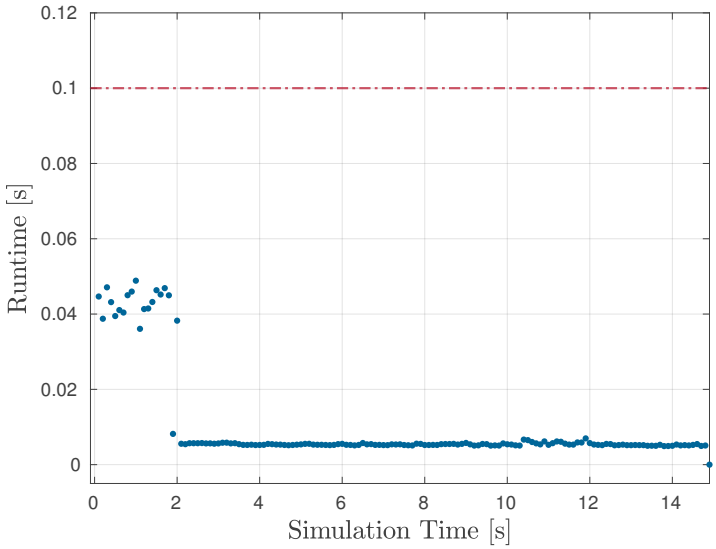


Figure 4.22: Runtime required by the MPC based on augmented Lagrangian with the projected Newton iLQR.

Figure 4.22 shows the runtimes of the combined augmented Lagrangian with the projected Newton approach. The first iteration, which requires approximately 0.58 s, is not depicted for clarity. Hence, the runtime required in order to compute the initial trajectory is by far larger than for the pure augmented Lagrangian approach. As discussed in Section 4.2 and Section 3.2, this comes from the bad initial active set guesses for the control inputs. However, note that all other runtimes of the MPC based on the combined approach in Figure 4.22 are comparable to the ones of the MPC based on the pure augmented Lagrangian approach in Figure 4.18 and at the beginning (approximately ≤ 2 s) they are on average even slightly smaller. Since the resulting control trajectories from the previous optimization are used as initialization, the active set estimation of the projected Newton method is better than in the initial iteration. In summary, all time-critical optimizations converge fast enough, i.e. in less time than $T_s = 100$ ms.

4.3.3 Primal-Dual Interior Point

Figure 4.23 visualizes the resulting trajectory of the simulated MPC-controlled crane that uses primal-dual interior point based iLQR. Other than the results from the algorithms based on the augmented Lagrangian method to incorporate collision constraints, c.f. Figure 4.17, the algorithm avoids the narrow space between the stanchions. As discussed in Section 4.2 and Section 3.2, this behavior comes from the barrier characteristics of

the algorithm. However, the algorithm is able to bring the crane into the desired end configuration without any collision, c.f. Figure 4.23 and Figure 4.25, and without any violations of the control limits, c.f. Figure 4.27. Additionally, abrupt movement of the non-actuated joints are avoided.

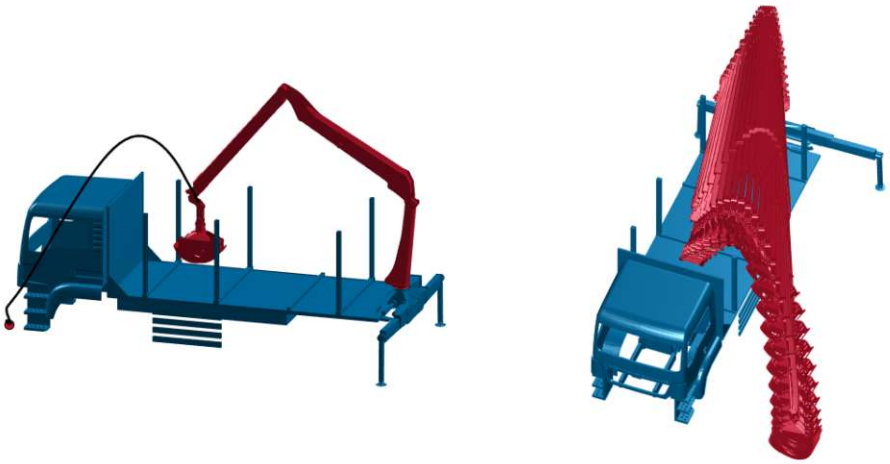


Figure 4.23: Resulting state trajectory of the simulated, primal-dual interior point based MPC-controlled crane.

Figure 4.26 shows the generalized velocities of the non-actuated joints. Hence, the MPC is able to stabilize the non-actuated joints at the end configuration.

The runtime of the primal-dual interior point algorithm is depicted in Figure 4.24. The first iteration, which requires approximately 0.18 s, is not depicted for clarity. The later runtimes are comparable to the ones of the augmented Lagrangian based approaches and even faster in some cases. However, note that the algorithm avoids the narrow space between the stanchions. Passing them introduces two conflicting constraints the algorithm has to handle, which is why it is difficult to directly compare these runtimes.

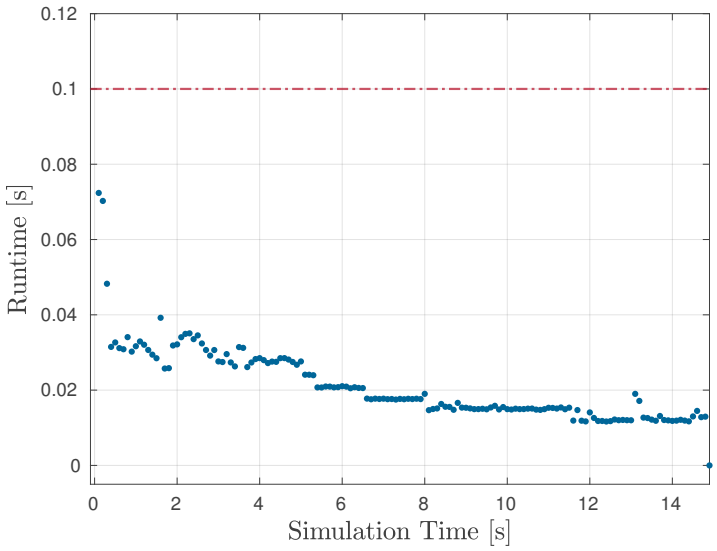


Figure 4.24: Runtime required by the MPC based on primal-dual interior point iLQR.

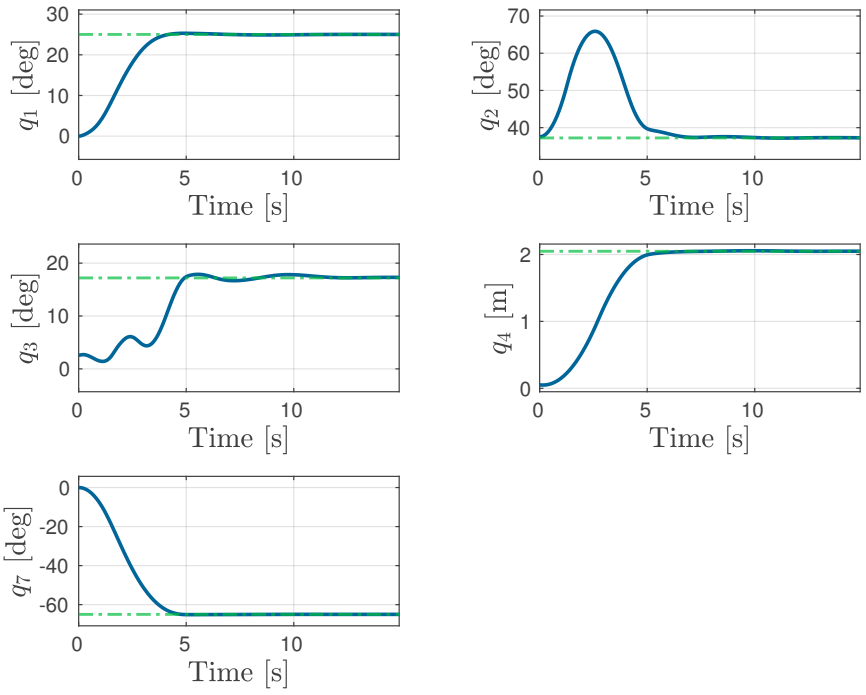


Figure 4.25: Resulting generalized actuated coordinates $\mathbf{q}_A$ of the simulated, primal-dual interior point based MPC-controlled crane.

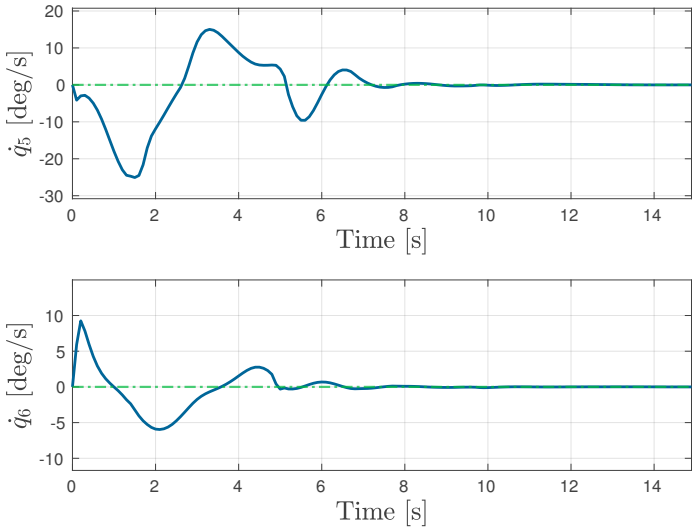


Figure 4.26: Resulting generalized non-actuated velocities $\dot{\mathbf{q}}_U$ of the simulated, primal-dual interior point based MPC-controlled crane.

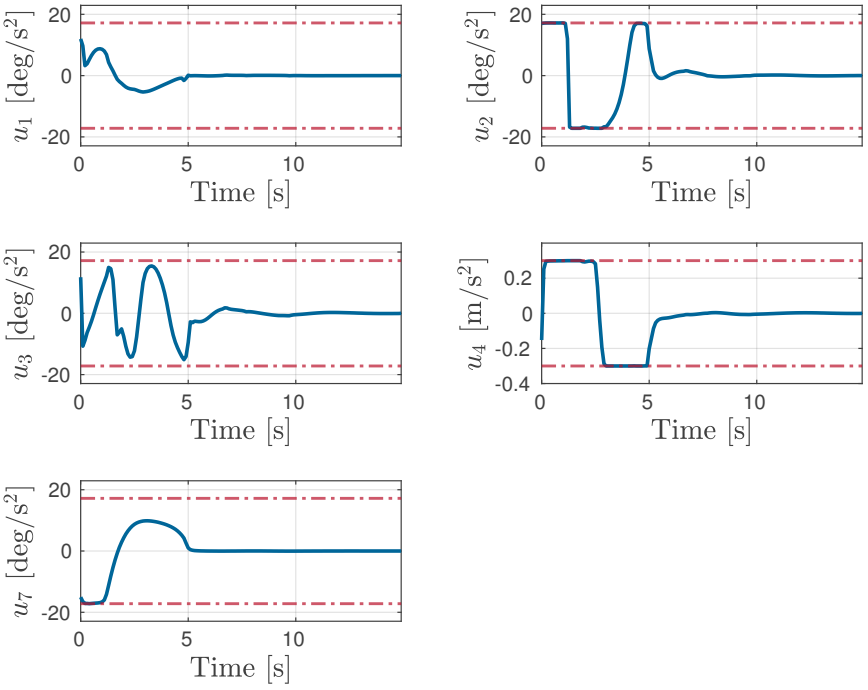


Figure 4.27: Resulting control inputs $\mathbf{u}$ of the simulated, primal-dual interior point based MPC-controlled crane.

5 Conclusions and Outlook

This thesis studied different constrained iLQR methods with application to collision-free trajectory planning of a timber crane. Two recent methods based on augmented Lagrangian, as presented in [22], and primal-dual interior point, as presented in [30], are investigated in order to integrate the collision constraints as well as state and control limits into the iLQR approach. Additionally, a third algorithm that uses the method from [22] for collision and state limits combined with the projected Newton method, presented in [19], to incorporate control box constraints is implemented and compared to the other concept. Numerical experiments show that the augmented Lagrangian method is very well suited for collision avoidance applications and the combined augmented Lagrangian with the projected Newton method leads to improved performance for conflicting state and control constraints. However, the combined approach might result in a degraded convergence if the clamping operations impair the quality of the search direction too much. While there are for sure constraints where the barrier nature of the interior point method has nice properties, like e.g. it avoids too fast progress into the direction of state and control limits, it turns out to be not that suited for collision constraints. Especially when free space becomes narrower, the algorithm requires a large number of iterations until the trajectory starts passing an object. On the one hand, this increases the runtimes compared to the augmented Lagrangian method and on the other hand, it may result in suboptimal trajectories because the barrier function forces the trajectory to keep larger distances to objects, in particular at the beginning of the optimization.

Based on these results, an MPC based trajectory planner is implemented using the constrained iLQR algorithms presented in Chapter 2. The results show that all three algorithms are able to navigate the crane to the desired end configuration without any collisions. Additionally, all algorithms are able to perform the optimization in less than $T_s = 100$ ms, which renders all of them fast enough. However, similar to the offline trajectory planning, the primal-dual interior point methods avoid narrow spaces.

In this thesis, collision objects are assumed to be known and static, This does not necessarily hold true for autonomous systems operating in unknown and dynamic environments. Hence, the system must be able to handle static and dynamic components of the environment. This leads to additional challenges for future research:

- **Static Map Representation:** There are many different map representations, most of them developed for small-scale systems, that can be represented by a single, or a few, point locations like cars or drones. However, the question arises which of these representations is suited for motion planning of large-scale structures like the timber crane regarding both, runtime and memory requirements. Additionally, sensors used to generate those maps are noisy and parts of the environment are occluded from certain sensor locations. Hence, a representation not only has to be well suited for

motion planning, but for the mapping task as well. Additionally, it is clear that the accuracy of the generated map influences the quality of the planned trajectory, but it is possible to consider certain aspects regarding the mapping task during planning as well, e.g. to plan a trajectory that in some sense optimizes occluded areas or viewpoint locations.

- **Dynamic Object Representation:** In real-world applications there are not only static collision objects. Similar to the static case, the question arises which representation is best to be able to estimate location and other properties of moving objects and to incorporate the acquired information into an online motion planning algorithm. One of the main challenging requirements for all algorithms used for this task is the necessity of low computation times in order to be able to perform real-time online planning.

In order to handle these requirements, this work suggests iLQR solvers based on the augmented Lagrangian method might be for such tasks. However, in more complex environments proper trajectory initialization methods must be incorporated in order to prevent the planner to get stuck at local minima not reaching the desired end configuration.

A GJK Algorithm for Collision Detection

Collision detection is a problem with applications in robotics, computer-aided design and computer graphics. The so called *Gilbert-Johnson-Keerthi (GJK) Algorithm* [36], named after its authors, is specifically designed for optimal path planning and allows to determine the distance between two convex objects. However, the initial algorithm was limited to objects represented as polytopes and an extension to general convex object was presented in [37]. Instead of modeling objects as convex hull of a finite set of points as done in [36], [37] uses the corresponding support mappings to model convex objects.

For the application of collision avoidance in robotics it is sufficient to consider objects in the Euclidean space $\mathbb{R}^2$ or $\mathbb{R}^3$. In order to tackle both cases, this section gives a more general formulation in the Euclidean space $\mathbb{R}^n$. An object is defined as the set of points that is occupied by the volume of the object, i.e. an object in $\mathbb{R}^n$ is a compact and convex set $\mathcal{O} \subset \mathbb{R}^n$. The set of all objects, i.e. the set of all convex and compact subsets of $\mathbb{R}^n$, is denoted by $\mathfrak{O} \subseteq 2^{\mathbb{R}^n}$.

The distance between two objects $\mathcal{O}_1, \mathcal{O}_2 \in \mathfrak{O}$ is defined by the mapping

$$d : \mathfrak{O} \times \mathfrak{O} \to \mathbb{R},\ (\mathcal{O}_1, \mathcal{O}_2) \mapsto \min\{\|\mathbf{p}_1 - \mathbf{p}_2\|_2 : \mathbf{p}_1 \in \mathcal{O}_1, \mathbf{p}_2 \in \mathcal{O}_2\} \ . \tag{A.1}$$

Note that the minimum is well defined because each object is a compact set.

Another important mapping is the one that maps an object $\mathcal{O} \in \mathfrak{O}$ to a point $\mathbf{p} \in \mathcal{O}$ that is closest to the origin, i.e.

$$\boldsymbol{\nu}(\mathcal{O}) := \arg\min\{\|\mathbf{p}\|_2 : \mathbf{p} \in \mathcal{O}\} \ . \tag{A.2}$$

Again, since $\mathcal{O}$ is a compact set the minimum always exists. However, $\boldsymbol{\nu}(\mathcal{O})$ might not be unique. It is easy to verify that for $\mathcal{O}_1, \mathcal{O}_2 \in \mathfrak{O}$ the relation

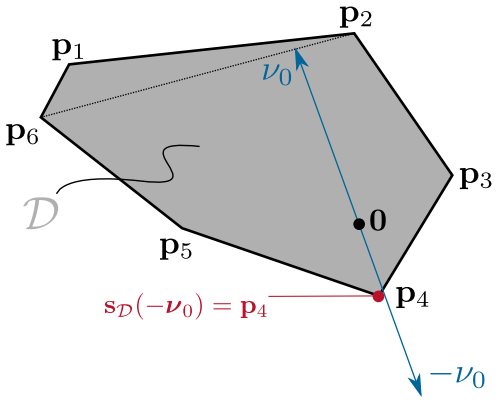
$$d(\mathcal{O}_1, \mathcal{O}_2) = \min\{\|\mathbf{p}\|_2 : \mathbf{p} \in \mathcal{D}\} = \|\boldsymbol{\nu}(\mathcal{D})\|_2 \qquad \mathcal{D} := \mathcal{O}_1 - \mathcal{O}_2 \tag{A.3}$$

holds. Hence, two object collide, if and only if $\mathbf{0} \in \mathcal{D}$. While the *Minkowski-difference* $\mathcal{D} := \{\mathbf{p}_1 - \mathbf{p}_2 : \mathbf{p}_1 \in \mathcal{O}_1, \mathbf{p}_2 \in \mathcal{O}_2\}$ of two convex sets $\mathcal{O}_1, \mathcal{O}_2$ is convex and compact, i.e. $\mathcal{D} \in \mathfrak{O}$, it is generally more complex than both, $\mathcal{O}_1$ and $\mathcal{O}_2$. However, the GJK algorithm allows to avoid the explicit computation of this difference.

The *support function* of an object $\mathcal{O} \in \mathfrak{O}$ maps any direction $\mathbf{d} \in \mathbb{R}^n$ to the maximum distance of a point $\mathbf{p} \in \mathcal{O}$ in the direction of $\mathbf{d}$, i.e. it is defined by

$$h_{\mathcal{O}} : \mathbb{R}^n \to \mathbb{R},\ \mathbf{d} \mapsto \max\{\mathbf{d} \cdot \mathbf{p} : \mathbf{p} \in \mathcal{O}\} \ . \tag{A.4}$$

The point that maximizes the distance in direction of $\mathbf{d}$ can be retrieved by the corresponding *support mapping* $\mathbf{s}_{\mathcal{O}}(\mathbf{d})$.



$$\mathcal{S}_0 = \{\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_6\} \Longrightarrow \bar{\mathcal{S}}_0 = \{\mathbf{p}_2, \mathbf{p}_6\} \Longrightarrow \mathcal{S}_1 = \{\mathbf{p}_2, \mathbf{p}_4, \mathbf{p}_6\}$$

Figure A.1: Visualization of the GJK algorithm in two dimensions.

The GJK algorithm determines $d(\mathcal{O}_1, \mathcal{O}_2)$ using repeated evaluations of $h_\mathcal{D}$ and $\mathbf{s}_\mathcal{D}$, $\mathcal{D} := \mathcal{O}_1 - \mathcal{O}_2$. However, as already mentioned, it is not required to explicitly compute $\mathcal{D}$, because

$$h_\mathcal{D}(\mathbf{d}) = h_{\mathcal{O}_1}(\mathbf{d}) - h_{\mathcal{O}_2}(-\mathbf{d}) \qquad \text{and} \qquad \mathbf{s}_\mathcal{D}(\mathbf{d}) = \mathbf{s}_{\mathcal{O}_1}(\mathbf{d}) - \mathbf{s}_{\mathcal{O}_2}(-\mathbf{d}) \tag{A.5}$$

for any two objects $\mathcal{O}_1, \mathcal{O}_2 \in \mathfrak{O}$ and $\mathbf{d} \in \mathbb{R}^n$.

The main idea of the GJK algorithm is to generate a sequence $(\mathcal{S}_k)_{k\in\mathbb{N}}$ with $\mathcal{S}_k \subseteq \mathcal{D}$ s.t. $\boldsymbol{\nu}(\text{conv}(\mathcal{S}_k)) \to \boldsymbol{\nu}(\mathcal{D})$ as $k \to \infty$, where conv$(\cdot)$ denotes the convex hull. The

Algorithm 3 GJK Algorithm

Require: Objects $\mathcal{O}_1, \mathcal{O}_2 \in \mathfrak{O}$, initial points $\mathbf{p}_1, \ldots, \mathbf{p}_m \in \mathcal{D} := \mathcal{O}_1 - \mathcal{O}_2$, $1 \le m \le n+1$
  $k \leftarrow 0$
  Initialize the simplex $\mathcal{S}_0 \leftarrow \{\mathbf{p}_1, \ldots, \mathbf{p}_m\}$, i.e. $\mathcal{S}_0 \subseteq \mathcal{D}$ is a simplex in $\mathcal{D}$.
  while true **do**
    Compute the minimum norm point $\boldsymbol{\nu}_k \leftarrow \boldsymbol{\nu}(\text{conv}(\mathcal{S}_k))$
    if $g_\mathcal{D}(\boldsymbol{\nu}_k) = 0$ **then**
      return $\boldsymbol{\nu}_k$
    else
      Compute the smallest subset $\bar{\mathcal{S}}_k \subseteq \mathcal{S}_k$ s.t. $\boldsymbol{\nu}_k \in \text{conv}(\bar{\mathcal{S}}_k)$ and $|\bar{\mathcal{S}}_k| \le n$.
      Update $\mathcal{S}_{k+1} \leftarrow \bar{\mathcal{S}}_k \cup \{\mathbf{s}_\mathcal{D}(-\boldsymbol{\nu}_k)\}$ using $\mathbf{s}_\mathcal{D}(-\boldsymbol{\nu}_k) = \mathbf{s}_{\mathcal{O}_1}(-\boldsymbol{\nu}_k) - \mathbf{s}_{\mathcal{O}_2}(\boldsymbol{\nu}_k)$.
      $k \leftarrow k + 1$
    end if
  end while

GJK algorithm is summarized in Algorithm 3 and Figure A.1 visualizes one iteration of the algorithm for $n = 2$. Given an initial simplex $\mathcal{S}_0 := \{\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_6\}$, the algorithm first determines the point $\boldsymbol{\nu}_0$ that has minimum norm visualized by the blue vector in Figure A.1. The set $\bar{\mathcal{S}}_0 = \{\mathbf{p}_2, \mathbf{p}_6\}$ is then determined choosing at most $n = 2$ points in $\mathcal{S}_0$

for which $\boldsymbol{\nu}_0 \in \text{conv}(\bar{\mathcal{S}}_0)$ is satisfied. In Figure A.1 $\boldsymbol{\nu}_0$ lies on the line that connects $\mathbf{p}_2$ and $\mathbf{p}_6$, which is why $\bar{\mathcal{S}}_0 := \{\mathbf{p}_2, \mathbf{p}_6\}$. Finally, the point $\mathbf{s}_\mathcal{D}(-\boldsymbol{\nu}_0)$ that maximizes the projection onto $\boldsymbol{\nu}_0$ is used to form $\mathcal{S}_1 = \bar{\mathcal{S}}_0 \cup \{\mathbf{s}_\mathcal{D}(-\boldsymbol{\nu}_0)\}$. Since $\mathbf{s}_\mathcal{D} = \mathbf{s}_{\text{conv}(\{\mathbf{p}_1,\ldots,\mathbf{p}_6\})} = \mathbf{s}_{\{\mathbf{p}_1,\ldots,\mathbf{p}_6\}}$ it holds that $\mathbf{s}_\mathcal{D}(-\boldsymbol{\nu}_0) = \mathbf{p}_4$.

The function g used as termination criterion is defined as

$$g_\mathcal{D} : \mathbb{R}^n \to \mathbb{R}_{\geq 0},\ \|\boldsymbol{\nu}_k\|_2^2 + h_\mathcal{D}(-\boldsymbol{\nu}_k)\ . \tag{A.6}$$

There are two cases:

- **Case** $g_\mathcal{D}(\boldsymbol{\nu}_k) = 0$**:** This case occurs if and only if $h_\mathcal{D}(-\boldsymbol{\nu}_k) = -\|\boldsymbol{\nu}_k\|_2^2 = -\boldsymbol{\nu}_k^\text{T}\boldsymbol{\nu}_k$. By the definition of $h_\mathcal{D}$ is follows that for every $\mathbf{p} \in \mathcal{D}$ it holds that $-\boldsymbol{\nu}_k^\text{T}\mathbf{p} \leq -\|\boldsymbol{\nu}_k\|_2^2$ and thus

$$\|\mathbf{p}\|_2 = \frac{\mathbf{p}^\text{T}\mathbf{p}}{\|\mathbf{p}\|_2} \geq \frac{\boldsymbol{\nu}_k^\text{T}\mathbf{p}}{\|\boldsymbol{\nu}_k\|} \geq \|\boldsymbol{\nu}\|_2\ . \tag{A.7}$$

  Hence, there is no $\mathbf{p} \in \mathcal{D}$ with $\|\mathbf{p}\|_2 < \|\boldsymbol{\nu}_k\|$, which is why the property

$$g_\mathcal{D}(\boldsymbol{\nu}_k) = 0 \iff \boldsymbol{\nu}_k = \boldsymbol{\nu}(\mathcal{D}) \tag{A.8}$$

  holds. Hence, we know that $\boldsymbol{\nu}_k$ is the minimum norm point in $\mathcal{D}$, i.e. it is the minimum distance vector between $\mathcal{O}_1$ and $\mathcal{O}_2$.

- **Case** $g_\mathcal{D}(\boldsymbol{\nu}_k) > 0$**:** In this case it can be shown that the property

$$\exists \boldsymbol{\nu}_{k+1} \in \text{conv}(\{\boldsymbol{\nu}_k, \mathbf{s}_\mathcal{D}(-\boldsymbol{\nu}_k)\}) : \|\boldsymbol{\nu}_{k+1}\| < \|\boldsymbol{\nu}_k\| \tag{A.9}$$

  holds. Since $\boldsymbol{\nu}_k \in \text{conv}(\mathcal{S}_k)$, $\mathcal{S}_k \subseteq \mathbb{R}^n$ and $|\mathcal{S}_k| \leq n+1$ it is possible to represent $\boldsymbol{\nu}_k$ by a convex combination (which is a linear combination too) of at most n elements of $\mathcal{S}_k$. Hence, the existence of $\bar{\mathcal{S}}_{k+1}$ with $|\bar{\mathcal{S}}_k| \leq n$ and $\boldsymbol{\nu}_k \in \bar{\mathcal{S}}_k$ is guaranteed. Obviously $\text{conv}(\{\boldsymbol{\nu}_k, \mathbf{s}_\mathcal{D}(-\boldsymbol{\nu}_k)\}) \subseteq \bar{\mathcal{S}}_k \cup \{\mathbf{s}_\mathcal{D}(-\boldsymbol{\nu}_k)\}$ from what follows that $\boldsymbol{\nu}_{k+1} = \boldsymbol{\nu}(\mathcal{S}_k)$ with $\|\boldsymbol{\nu}_{k+1}\| < \|\boldsymbol{\nu}_k\|$ exists.

A convergence proof for polytopes generated by a finite set of points is given in [36]. More specifically, if an object is represented by a convex polytope generated by a finite number of points, the algorithm terminates in a finite number of iterations. An extension to general convex objects not necessarily generated by the convex hull of a finite number of points is presented in [37]. In this case, an object is fully characterized by its support mapping $\mathbf{s}_\mathcal{O}$. However, there is no proof that the algorithm terminates after a finite number of iterations, but asymptotic convergence can be shown. Additionally, a proper termination constraint has to be chosen. In order to compute the minimum distance from the origin, the original work [36] uses the so called *Johnson's algorithm*, originally provided in [38].

Bibliography

[1] K. Lynch and F. Park, *Modern Robotics: Mechanics, Planning, and Control*. Cambridge: Cambridge Univeristy Press, 2017.

[2] S. M. LaValle, *Planning Algorithms*. Cambridge: Cambridge University Press, 2006.

[3] C. Park, F. Rabe, S. Sharma, C. Scheurer, U. E. Zimmermann, and D. Manocha, "Parallel cartesian planning in dynamic environments using constrained trajectory planning," in *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, Seoul, Nov. 2015, pp. 983–990.

[4] L. Li, X. Long, and M. A. Gennert, "Birrtopt: A combined sampling and optimizing motion planner for humanoid robots," in *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, Cancun, Nov. 2016, pp. 469–476.

[5] C. Rösmann, F. Hoffmann, and T. Bertram, "Kinodynamic trajectory optimization and control for car-like robots," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Vancouver, Sep. 2017, pp. 5681–5686.

[6] J. Leu, G. Zhang, L. Sun, and M. Tomizuka, "Efficient robot motion planning via sampling and optimization," in *2021 American Control Conference (ACC)*, New Orleans, May 2021, pp. 4196–4202.

[7] J. T. Betts, *Practical methods for optimal control and estimation using nonlinear programming*. Philadelphia: Advances in Design and Control, SIAM, 2010.

[8] M. Giftthaler, M. Neunert, M. Stäuble, J. Buchli, and M. Diehl, "A family of iterative gauss-newton shooting methods for nonlinear optimal control," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Madrid, Oct. 2018, pp. 1–9.

[9] W. Li. and E. Todorov, "Iterative linear quadratic regulator design for nonlinear biological movement systems," in *2004 First International Conference on Informatics in Control, Automation and Robotics*, Setúbal, Aug. 2004, pp. 222–229.

[10] D. MAYNE, "A second-order gradient method for determining optimal trajectories of non-linear discrete-time systems," *International Journal of Control*, vol. 3, no. 1, pp. 85–95, 1966.

[11] L.-Z. Liao and C. Shoemaker, "Convergence in unconstrained discrete-time differential dynamic programming," *IEEE Transactions on Automatic Control*, vol. 36, no. 6, pp. 692–706, 1991.

[12] B. Plancher and S. Kuindersma, "A performance analysis of parallel differential dynamic programming on a gpu," in *Algorithmic Foundations of Robotics XIII*, M. Morales, L. Tapia, G. Sánchez-Ante, and S. Hutchinson, Eds., Mérida: Springer, Dec. 2020, pp. 656–672.

[13] E. Todorov and W. Li, "A generalized iterative lqg method for locally-optimal feedback control of constrained nonlinear stochastic systems," in *2005 American Control Conference*, Portland, Jun. 2005, pp. 300–306.

[14] D. H. Jacobson, "New second-order and first-order algorithms for determining optimal control: A differential dynamic programming approach," *Journal of Optimization Theory and Applications*, vol. 2, pp. 411–440, 1968.

[15] S. B. Gerschwin and D. H. Jacobcon, "A discrete-time differential dynamic programming algorithm with application to optimal orbit transfer," *AIAA Journal*, vol. 8, no. 9, pp. 1616–1626, 1970.

[16] S.-C. Chang, C.-H. Chen, I.-K. Fong, and P. Luh, "Hydroelectric generation scheduling with an effective differential dynamic programming algorithm," *IEEE Transactions on Power Systems*, vol. 5, no. 3, pp. 737–743, 1990.

[17] G. Lantoine and R. P. Russell, "A hybrid differential dynamic programming algorithm for constrained optimal control problems. part 1: Theory," *Journal of Optimization Theory and Applications*, vol. 154, pp. 382–417, 2012.

[18] ——, "A hybrid differential dynamic programming algorithm for constrained optimal control problems. part 2: Application," *Journal of Optimization Theory and Applications*, vol. 154, pp. 418–442, 2012.

[19] Y. Tassa, N. Mansard, and E. Todorov, "Control-limited differential dynamic programming," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, Hong Kong, May 2014, pp. 1168–1175.

[20] M. Giftthaler and J. Buchli, "A projection approach to equality constrained iterative linear quadratic optimal control," in *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*, Birmingham, Nov. 2017, pp. 61–66.

[21] Z. Xie, C. K. Liu, and K. Hauser, "Differential dynamic programming with nonlinear constraints," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, Singapore, May 2017, pp. 695–702.

[22] T. A. Howell, B. E. Jackson, and Z. Manchester, "Altro: A fast solver for constrained trajectory optimization," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Macau, Nov. 2019, pp. 7674–7679.

[23] M. R. Hestenes, "Multiplier and gradient methods," *Journal of Optimization Theory and Applications*, vol. 4, no. 5, pp. 303–320, 1969.

[24] R. T. Rockafellar, "A dual approach to solving nonlinear programming problems by unconstrained optimization," *Mathematical Programming*, vol. 5, no. 1, pp. 354–373, 1973.

[25] R. T. Rockafellar, "The multiplier method of hestenes and powell applied to convex programming," *Journal of Optimization Theory and Applications*, vol. 12, no. 6, pp. 555–562, 1973.

[26] J. Nocedal and S. Wright, *Numerical optimization*. New York: Springer, 2006.

[27] B. Plancher, Z. Manchester, and S. Kuindersma, "Constrained unscented dynamic programming," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Vancouver, Sep. 2017, pp. 5674–5680.

[28] Y. Aoyama, G. Boutselis, A. Patel, and E. A. Theodorou, "Constrained differential dynamic programming revisited," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, Xi'an, May 2021, pp. 9738–9744.

[29] R. A. Castillo and J. M. Martinez, "Numerical comparison of augmented lagrangian algorithms for nonconvex problems," *Computational Optimization and Applications*, vol. 31, no. 1, pp. 31–55, 2005.

[30] A. Pavlov, I. Shames, and C. Manzie, "Interior point differential dynamic programming," *IEEE Transactions on Control Systems Technology*, vol. 29, no. 6, pp. 2720–2727, 2021.

[31] Y. Tassa, T. Erez, and E. Todorov, "Synthesis and stabilization of complex behaviors through online trajectory optimization," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Vilamoura-Algarve, Oct. 2012, pp. 4906–4913.

[32] D. P. Bertsekas, "Multiplier methods: A survey," *Automatica*, vol. 12, no. 2, pp. 133–145, 1976.

[33] M. W. Spong, S. Hutchinson, and M. Vidyasagar, *Robot modeling and control*. New York: Wiley, 2006.

[34] J. Schulman, Y. Duan, J. Ho, A. Lee, I. Awwal, H. Bradlow, J. Pan, S. Patil, K. Goldberg, and P. Abbeel, "Motion planning with sequential convex optimization and convex collision checking," *The International Journal of Robotics Research*, vol. 33, no. 9, pp. 1251–1270, 2014.

[35] K. Graichen, M. Egretzberger, and A. Kugi, "Suboptimal model predictive control of a laboratory crane," *IFAC Proceedings Volumes*, vol. 43, no. 14, pp. 397–402, 2010.

[36] E. Gilbert, D. Johnson, and S. Keerthi, "A fast procedure for computing the distance between complex objects in three-dimensional space," *IEEE Journal on Robotics and Automation*, vol. 4, no. 2, pp. 193–203, 1988.

[37] E. Gilbert and C.-P. Foo, "Computing the distance between general convex objects in three-dimensional space," *IEEE Transactions on Robotics and Automation*, vol. 6, no. 1, pp. 53–61, 1990.

[38] D. W. Johnson, "The optimization of robot motion in the presence of obstacles," PhD thesis, Michigan, 1987.

Eidesstattliche Erklärung

Hiermit erkläre ich, dass die vorliegende Arbeit gemäß dem Code of Conduct – Regeln zur Sicherung guter wissenschaftlicher Praxis (in der aktuellen Fassung des jeweiligen Mitteilungsblattes der TU Wien), insbesondere ohne unzulässige Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel, angefertigt wurde. Die aus anderen Quellen direkt oder indirekt übernommenen Daten und Konzepte sind unter Angabe der Quelle gekennzeichnet. Die Arbeit wurde bisher weder im In– noch im Ausland in gleicher oder in ähnlicher Form in anderen Prüfungsverfahren vorgelegt.

<Vienna, im Monat Jahr>

Marc-Philip Ecker