Fast Model Predictive Control of Robotic Systems with Rigid Contacts

Sotaro Katayama
Department of Systems Science, Graduate School of Informatics
Kyoto University

A thesis submitted for the degree of
Doctor of Philosophy in Informatics

2022

Supervisor:

Prof. Toshiyuki Ohtsuka

Examination committee:

Prof. Toshiyuki Ohtsuka

Prof. Shin Ishii

Prof. Jun Morimoto

Abstract

Robotic systems such as legged robots are expected to work in a wider variety of places. Planning and control are key technologies to give them a certain intelligence for autonomous decision-making.

This thesis pursues this goal by establishing a real-time algorithm for model predictive control (MPC) of robotics systems, which can be a unified approach for versatile, efficient, and dynamic planning and control. However, robotic systems have fast, large-scale, and nonlinear dynamics and involve switches of dynamics and state jumps, which makes online optimization of MPC difficult. In this thesis, we tackle these computational challenges with the following algorithmic developments.

First, we propose an inverse dynamics-based solution method for the MPC problem of rigid body systems. It can reduce the computational time of each Newton-type iteration by leveraging efficient rigid body inverse dynamics algorithms. Second, we propose a lifted Newton-type method for MPC problems of rigid body systems with rigid contacts. This method can reduce the required number of iterations by relaxing high nonlinearity without additional computational demands and therefore can reduce the total computational time.

We also propose efficient MPC algorithms for switched systems, which formally model the robotic systems with rigid contacts under a given contact sequence. First, we present an efficient Riccati recursion for MPC problems with pure-state equality constraints. We propose a constraint-transformation method to efficiently treat the pure-state equality constraints with the Riccati recursion algorithm. Second, we propose a structure-exploiting Newton-type method to optimize switching times (i.e., contact timings in robotic problems), state, and control input simultaneously. The proposed algorithm enables significantly fast computational time and also improves numerical robustness.

Finally, we implement an MPC of robotic systems built on top of these algorithm developments. We propose a whole-body MPC that can optimize the switching times, state, and control input in milliseconds range. We conduct a simulation comparison of the proposed MPC and the conventional MPC with fixed switching times and demonstrate that the proposed approach expands the ability of MPC of robotic systems. We further conduct hardware experiments on a real quadrupedal robot and show that the proposed method achieves dynamic motions on the real robot.

Acknowledgements

First of all, I would like to take the opportunity to express my deepest gratitude to my supervisor Professor Toshiyuki Ohtsuka. He has consistently given me valuable guidance and great freedom to pursue my research interests. His deep knowledge of control, system science, and optimization also always helps my Ph.D. studies. Thank you for inspiring, motivating, and supporting my Ph.D. research with boundless enthusiasm. I also would like to thank examination committee members Prof. Toshiyuki Ohtsuka, Prof. Shin Ishii, and Prof. Jun Morimoto for their helpful advice and suggestions to prepare this thesis.

I would like to thank Associate Professor Kazunori Sakurama and Assistant Professor Kenta Hoshino for valuable discussions and suggestions in the research seminars in Ohtsuka lab. Their comments have always opened my mind to the wider research world. I am also grateful to Assos. Prof. Yasuyuki Satoh at Tokyo University of Science and Dr. Masahiro Doi at Toyota Motor Corporation for supporting my Master's degree and my first journal paper gently. I also would like to thank Assoc. Prof. Yuichi Tazaki at Kobe University and Dr. Masaki Murooka at the National Institute of Advanced Industrial Science and Technology (AIST) for valuable discussion on humanoid robotics research.

I also thank the current and past Ohtuka lab members and staff. Assis. Prof. Iori at Osaka University had provided me with various interesting mathematical topics. Dr. Deng indicates to me a direction of practical MPC algorithm research. I would like to thank all members to let me spend a great time with delightful conversations at the Ohtsuka lab.

I would also like to thank the JST SPRING and JSPS KAKENHI for the financial support of my Ph.D. study.

Finally, I would like to thank my parents who always believe in me and support my challenges.

Contents

Notation **1**

1 Introduction **3**

1.1 Background . . . 3

1.2 Overview of Planning and Control of Robotic Systems with Contacts 4

1.2.1 Contact and motion planning . . . 4

1.2.1.1 Sampling-based motion planning . . . 5

1.2.1.2 Optimization-based motion planning . . . 6

1.2.1.3 Hierarchical contact and motion planning . . . 7

1.2.1.4 Simultaneous contact and motion planning . . . 7

1.2.2 Control . . . 8

1.2.2.1 Reduced-order models for legged robot control . . . 9

1.2.2.2 Whole-body control . . . 10

1.2.2.3 State estimation for legged robots . . . 11

1.2.3 Online motion planning as control: Model predictive control (MPC) . . . 11

1.3 Challenges in Real-Time MPC of Robotic Systems . . . 13

1.3.1 Overview of MPC algorithms . . . 13

1.3.2 MPC algorithms for fast and large-scale robotic systems . . . 14

1.3.3 MPC algorithms for switched systems . . . 15

1.4 Outline and Contributions . . . 16

2 Preliminaries **21**

2.1 Rigid Body Systems . . . 21

2.1.1 Kinematics . . . 22

2.1.1.1 Forward and differential kinematics . . . 22

2.1.1.2 Contact kinematics . . . 22

2.1.2 Dynamics . . . 24

2.1.2.1 Inverse dynamics 24

2.1.2.2 Forward dynamics 24

2.1.2.3 Contact-consistent forward dynamics 25

2.1.2.4 Impulse dynamics 25

2.1.2.5 State space representation 25

2.1.3 Software 26

2.2 Model Predictive Control 26

2.2.1 Overview 26

2.2.2 Numerical optimal control techniques for fast MPC of large-scale systems 27

2.2.2.1 Direct multiple shooting method 27

2.2.2.2 Primal-dual interior point method 28

2.2.2.3 Gauss-Newton Hessian approximation 31

2.2.2.4 Riccati recursion 32

2.2.2.5 Summary of Newton-type method for MPC 33

3 Inverse Dynamics-Based Solution Method of Optimal Control of Rigid Body Systems 35

3.1 Introduction 35

3.2 Optimal Control Problem Based on Inverse Dynamics 37

3.2.1 Rigid-body systems 37

3.2.2 Optimal control problem 38

3.2.3 KKT conditions 39

3.3 Solution Method of Optimal Control Problem 40

3.3.1 Linearization for Newton's method 40

3.3.2 Condensing inverse dynamics 42

3.3.3 Algorithm 43

3.4 Numerical Experiments 45

3.4.1 Experimental settings 45

3.4.2 Computational time 45

3.4.3 Numerical robustness 46

3.4.4 MPC for floating base systems 48

3.5 Summary 48

4 Lifted Contact Dynamics for Efficient Optimal Control of Rigid Body Systems with Contacts 50
4.1 Introduction 50
4.2 Overview of Optimal Control Problems of Rigid-Body Systems with Contacts 53
4.2.1 Contact dynamics 53
4.2.2 Impulse dynamics 55
4.2.3 Conventional formulation of optimal control 56
4.3 Lifted Contact Dynamics in Optimal Control 56
4.3.1 Lifted contact dynamics 56
4.3.2 Lifted impulse dynamics 59
4.3.3 Riccati recursion for LQR subproblem 60
4.3.4 Primal-dual interior-point method for inequality constraints . 60
4.3.5 Algorithm 61
4.3.6 Comparison with existing methods 62
4.3.6.1 Comparison with non-lifted formulations 62
4.3.6.2 Comparison with inverse dynamics-based algorithm . 62
4.4 Numerical Experiments: Whole-Body Optimal Control of Quadrupedal Gaits 62
4.4.1 Experimental settings 62
4.4.2 Results and discussion 65
4.5 Summary 66

5 Efficient Riccati Recursion for Optimal Control Problems with Pure-State Equality Constraints 70
5.1 Introduction 70
5.2 Transformation of Optimal Control Problem with Pure-State Equality Constraints 73
5.2.1 Original optimal control problem 73
5.2.2 Transformation of optimal control problem 74
5.2.3 Optimality conditions 75
5.3 Riccati Recursion 76
5.3.1 Linearization for Newton's method 76
5.3.1.1 Terminal stage 76
5.3.1.2 Intermediate stages without equality constraint . . . 76
5.3.1.3 Intermediate stage with an equality constraint 77

5.3.1.4 Initial stage 77
5.3.2 Derivation of Riccati recursion 77
5.3.2.1 Terminal stage 78
5.3.2.2 Intermediate stages without an equality constraint . 78
5.3.2.3 Intermediate stage with an equality constraint 78
5.3.3 Algorithm, convergence, and computational analysis 79
5.4 Theoretical Properties of Optimal Control Problem Transformation 81
5.5 Numerical Experiments on Whole-Body Quadrupedal Gaits Optimization 83
5.5.1 Experimental settings 83
5.5.2 Trotting gait for different numbers of steps 84
5.5.3 Trotting, jumping, and running gait problems 85
5.6 Summary 87

6 Structure-Exploiting Newton-Type Method for Optimal Control of Switched Systems 89
6.1 Introduction 89
6.2 Problem Formulation 93
6.3 KKT System for Newton-Type Method 96
6.4 Riccati Recursion to Solve KKT Systems 98
6.4.1 Backward recursion 99
6.4.1.1 Terminal stage 99
6.4.1.2 Intermediate stages 99
6.4.1.3 Phase transition stages 102
6.4.2 Forward recursion 103
6.4.3 Properties of proposed Riccati recursion 103
6.4.4 Reduced Hessian modification via Riccati recursion 105
6.4.5 Algorithm 107
6.5 State Jumps and Switching Conditions 107
6.5.1 Switching conditions 109
6.6 Numerical Experiments 112
6.6.1 Comparison with off-the-shelf solvers 112
6.6.1.1 Problem settings 112
6.6.1.2 Results 113
6.6.2 Whole-body optimal control of quadrupedal gaits 114

6.6.2.1 Problem settings 114
6.6.2.2 Results 117
6.7 Summary 118

7 Whole-Body Model Predictive Control with Rigid Contacts via Online Switching Time Optimization 121
7.1 Introduction 121
7.2 Optimal Control Problem Formulation 124
7.2.1 Rigid body systems with rigid contacts 124
7.2.2 Inequality constraints 127
7.2.3 Optimal control problem of switched systems 128
7.3 Model Predictive Control with Online Switching Time Optimization . 129
7.3.1 Direct multiple shooting method with mesh-refinement 129
7.3.2 Riccati recursion to compute Newton step 130
7.3.3 Heuristic regularization to improve convergence property . . . 131
7.3.4 Minimum dwell-time constraints 131
7.3.5 Software implementation 132
7.4 Simulation Study: Comparison to Conventional MPC with Fixed Contact Timings 132
7.4.1 Experimental settings 132
7.4.2 Results 133
7.5 Hardware Experiments on Quadrupedal Robot Unitree A1 135
7.5.1 Experimental settings 135
7.5.2 Results 136
7.6 Summary 137

8 Conclusions 140
8.1 Summary of Contributions 140
8.2 Discussion and Future Work 141
8.2.1 Application to a wider variety of robotic problems 141
8.2.2 Improving robustness 142

A `robotoc`: Open-Source Software for Whole-Body Model Predictive Control 145
A.1 Introduction 145
A.2 Interface Overview 147
A.2.1 `robotoc::Robot` 147

A.2.2 `robotoc::CostFunction` 148
A.2.3 `robotoc::Constraints` 148
A.2.4 `robotoc::ContactSequence` 149
A.2.5 `robotoc::STOCostFunction` and
`robotoc::STOConstraints` 150
A.2.6 `robotoc::OCP` and `robotoc::OCPSolver` 151
A.3 Implementation Details 152
A.4 Application Examples 153
A.4.1 Whole-body MPC of a humanoid robot walking 153
A.4.1.1 Problem settings and MPC design 153
A.4.1.2 Results 154
A.4.2 Whole-body MPC of quadruped robot gaits 155
A.4.2.1 Problem settings and MPC design 155
A.4.2.2 Results 155
A.5 Summary 157

Bibliography **159**

List of Publications **173**

Notation

For vectors $x \in \mathbb{R}^n$ and $y \in \mathbb{R}^m$, a matrix $A \in \mathbb{R}^{n\times n}$, a differentiable scalar function $f(x, y) : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$, and a differentiable vector-valued function $g(x) : \mathbb{R}^n \to \mathbb{R}^p$, we have the following notation:

Notation	Meaning
$x^{(i)}$	i-th component of x
$\|x\|_2$	ℓ_2 norm of x
$A \succ 0$	A is positive-definite
$A \succeq 0$	A is positive-semidefinite
f_x, $\nabla_x f$	Gradient vector of a function $f(x, y)$ with respect to x ($f_x, \nabla_x f \in \mathbb{R}^n$)
f_{xx}, $\nabla_{xx} f$	Second-order derivative of a function $f(x, y)$ ($f_{xx}, \nabla^2_{xx} f \in \mathbb{R}^{n\times n}$)
f_{xy}, $\nabla_{xy} f$	Second-order derivative of a function $f(x, y)$ ($f_{xy}, \nabla^2_{xy} f \in \mathbb{R}^{n\times m}$)
g_x, $\nabla_x g$	Jacobian matrix of a function $g(x)$ with respect to x ($g_x, \nabla_x g \in \mathbb{R}^{p\times n}$)
$\mathrm{diag}(x)$	Diagonal matrix whose diagonal is the vector x.
I, I_n	Identity matrix, identity matrix with n rows and columns
0, O_n	Zero or zero matrix, zero matrix with n rows and columns
$\mathbb{1}$	Vector all of whose elements are 1

Abbreviations

Abbreviation	Meaning
ABA	Articulated body algorithm
AL	Augmented Lagrangian
CBF	Control barrier function
CLF	Control Lyapunov function
CITO	Contact-implicit trajectory optimization
COM	Center of mass
CPU	Central processing unit
CRBA	Composite rigid-body algorithm
DDP	Differential dynamic programming
DMS	Direct multiple shooting (method)
DOF	Degrees of freedom
EKF	Extended Kalman filter
HZD	Hybrid zero dynamics
FONC	First-order necessary conditions
iLQR	Iterative linear quadratic regulator
KKT	Karush–Kuhn–Tucker (conditions)
LICQ	Linear independence constraint qualification
LQR	Linear quadratic regulator
MPC	Model predictive control
NLP	Nonlinear program/programming
OCP	Optimal control problem
PD	Proportional-derivative (control)
PDIP	Primal-dual interiror point (method)
QP	Quadratic program/programming
RNEA	Recursive Newton-Euler Algorithm
SOSC	Second-order sufficient conditions
SQP	Sequential quadratic programming
SRBD	Single rigid body dynamics
STO	Switching time optimization
TO	Trajectory optimization
WBC	Whole-body control
ZMP	Zero-moment point

Chapter 1

Introduction

1.1 Background

Robots are essential for our lives. Industrial robot manipulators do boring and iterative tasks in place of people, typically faster and more accurately than humans. They can also work in dangerous places such as chemical and pharmaceutical plants.

Nowadays, robots are expected to work in more variety of places. For example, they are expected to move around us and help our tasks in our daily lives. On the other hand, they are also expected to enter harsh environments where humans cannot stay and conduct tasks. A significant requirement is that they have to move to various places from households to disaster sites. Legged robots such as quadrupedal robots are promising platforms for this purpose. Another requirement is that they have to conduct various tasks. Versatility is therefore required as well as industrial robotic manipulators. Furthermore, these activities are desired to be achieved autonomously.

Toward this goal, in the past decade, there has been drastic progress on hardware platforms, particularly in legged robots, which can move around in various places. For example, series elastic actuators have been made for compliant and robust locomotion (Hutter et al. (2017); Kamikawa et al. (2021)). Also, powerful low-gear ratio motors have been realized for compliant motions and accurate torque control (Katz, Di Carlo, and Kim (2019); Unitree Robotics (n.d.)). Based on these actuators, quadrupedal robots toward dynamic and agile locomotions have been developed, such as ANYmal (Hutter et al. (2017)), MIT Mini Cheetah (Katz et al. (2019)), Solo (Grimminger et al. (2020)), Stanford Doggo (Kau, Schultz, Ferrante, and Slade (2019)), Spot (Boston Dynamics (n.d.-b)), A1 (Unitree Robotics (n.d.)), Tachyon (Kamikawa et al. (2021)), etc. Humanoid robot hardware also has been evolved, such as iCub (Natale, Bartolozzi, Pucci, Wykowska, and Metta (2017)), TALOS (Stasse et al. (2017)), HRP-5P (Kaneko et al. (2019)), and Cassie (Agility Robotics (n.d.)), whose performance are siginificantly improved from the past humanoid robot platforms. A symbolical example of the hardware evolution is Boston Dynamics Atlas (Boston Dynamics (n.d.-a)) which has surprised the world with its human-level acrobatic motions.

Perception and recognition technologies have also been developed drastically, which give a certain intelligence to robots. In particular, deep neural networks (DNN) have been playing a central role in these developments. For example, DNN made breakthroughs in object detection and reconstruction (Han, Laga, and Bennamoun (2019); Z.-Q. Zhao, Zheng, Xu, and Wu (2019)). These methods have been successfully applied for robotic manipulation problems (Florence, Manuelli, and Tedrake (2018); Schwarz, Milan, Periyasamy, and Behnke (2018)). Simultaneous localization and mapping (SLAM), a fundamental technology for autonomy including legged locomotion, has also evolved (Mur-Artal, Montiel, and Tardos (2015); Saputra, Markham, and Trigoni (2018); Zhang and Singh (2014)). Furthermore, terrain mapping, which is a unique perception problem for legged locomotion on rough terrain, has been developed (Fankhauser, Bloesch, Gehring, Hutter, and Siegwart (2014); Fankhauser, Bloesch, and Hutter (2018)).

Another key technology for intelligence is decision-making. The problem is, how to actuate the joints to achieve desired tasks for given perception and recognition information. This decision-making problem is formally categorized as planning and control. In this thesis, we tackle this decision-making problem for robotic systems with contacts, particularly for legged robots.

1.2 Overview of Planning and Control of Robotic Systems with Contacts

Next, we provide an overview of the planning and control methods for robotic systems with contacts. We particularly focus on legged locomotion problems, in which the dynamical system behaves as a hybrid system and can involve underactuated dynamics.

1.2.1 Contact and motion planning

First, we introduce planning methods of contacts and motions of robotic systems. In 1.2.1.1 and 1.2.1.2, we give overviews of sampling-based and optimization-based motion planning methods, which have been widely utilized, e.g., from traditional manipulation planning to complicated motion generation of humanoid robots. In general, the sampling-based methods are suitable for long-horizon and non-convex problems such as complicated collision avoidance because they do not suffer from the undesired local minima or numerical ill-conditioning, which often appear in optimization-based methods. However, the sampling-based methods can only generate kinematically-feasible motions, i.e., kinodynamic motions (LaValle and Kuffner Jr (2001)). On the other hand, optimization-based methods can generate dynamically-consistent trajectories and incorporate various performance indices such as energy efficiency. In practice, the sampling-based method is first utilized to find the feasible configuration path under obstacles. Subsequently, the optimization-based methods seek a dynamically-consistent and reasonable trajectory to track the paths computed by the sampling-based methods.

Contact planning has a different characteristic than the above motion planning problems: the decision-making on contacts, e.g., making/breaking and sticking/sliding contacts, are discrete (binary) decisions while motion planning is formulated as problems to determine continuous configuration variables. Therefore, in principle, planning of contact and motion involves decisions on both continuous and discrete (or binary) variables. To avoid such complicated problems, in practice, we typically take hierarchical approaches: first determine the contact sequence and subsequently plan the motions by using methods introduced in 1.2.1.1 and 1.2.1.2. In 1.2.1.3, we introduce such hierarchical approaches with promising contact planning methods under perceptive information. In contrast to such hierarchical approaches, in 1.2.1.4, we describe challenging approaches to planning the contact and motion simultaneously, which involve several issues in practice.

1.2.1.1 Sampling-based motion planning

The sampling-based motion planning methods seek the trajectory of the configuration that satisfies pre-defined constraints from start to goal. Typical sampling-based methods are probabilistic road-map (PRM) method (Kavraki, Svestka, Latombe, and Overmars (1996)) and rapidly-exploring random trees (RRT) (LaValle and Kuffner Jr (2001); LaValle et al. (1998)). The PRM method is a multiple-query planning method that grows road maps from start to goal via successive random samplings. After constructing a road map, we can extract a path from the map, e.g., by using A*. We can also reuse the road map for the same or similar planning situations. The RRT method is a single-query method. That is, it focuses on a particular planning problem and can reduce the number of samples. These methods have been traditionally utilized for manipulation planning, e.g., of the joint trajectory under obstacles (Kavraki et al. (1996); Kuffner and LaValle (2000); LaValle et al. (1998)). Furthermore, they can be applied to robotic problems involving contacts such as statically-stable motion planning of humanoid robots (Kuffner, Kagami, Nishiwaki, Inaba, and Inoue (2002)).

1.2.1.2 Optimization-based motion planning

In optimization-based methods, we plan the trajectory by solving constrained optimization problems, that is, by minimizing a cost function under constraints. This class of approaches is also known as trajectory optimization (TO) and its characteristics differ depending on the constraints and cost function. The optimization-based approaches are versatile because they can consider various physical characteristics and task specifications as the cost or constraints in a unified manner.

Zucker et al. (2013), Kalakrishnan, Chitta, Theodorou, Pastor, and Schaal (2011), and Schulman et al. (2014) proposed optimization-based methods for kinematic trajectory optimization problems, which are the same or similar planning problems as the aforementioned sampling-based planning methods. Zucker et al. (2013) proposed a planning method that solves the optimization problem by combining gradient information and sampling-based methods. Kalakrishnan et al. (2011) proposed a sampling-based approach without any gradient information to solve the TO problems that possibly involve non-smooth costs. Schulman et al. (2014) applied sequential convex programming for motion planning problems under non-convex constraints such as complex collision avoidance. These approaches can be successfully applied to various problems from manipulation in a cluttered environment to the kinematic motion planning of legged robots with obstacles.

If we incorporate the dynamics of the system into the optimization problem, the planning problems, i.e., the TO problems, are formulated as optimal control problems (OCPs) (Betts (2010); Bryson and Ho (1975)). Because the robotic problems typically involve nonlinear dynamics, non-convex cost, and constraints, the OCPs are approximately solved by numerical optimization techniques (Betts (2010); Rawlings, Mayne, and Diehl (2017)). Dai, Valenzuela, and Tedrake (2014) realized highly complicated motions of a humanoid robot under a given contact sequence by solving the OCP for the whole-body kinematics and centroidal dynamics model (Orin, Goswami, and Lee (2013)) with constraints such as collision avoidance and friction cones. Schultz and Mombaur (2010), Lengagne, Vaillant, Yoshida, and Kheddar (2013), and Posa, Kuindersma, and Tedrake (2016) successfully generated human-like dynamic motions of humanoid robots by solving OCPs based on whole-body dynamics of the robots under a predefined contact sequence. More detailed descriptions of these approaches, particularly of algorithmic perspective, are described later in 1.2.3.

1.2.1.3 Hierarchical contact and motion planning

Next, we present hierarchical contact and motion planning approaches. The overall planning starts with common devices to perceive the external environment such as depth cameras and LIDAR. First, from perceptive data obtained by these devices, the perception and recognition algorithms reconstruct environment models such as 2D grid map (Thrun (2002)), 2.5D grid-height-map (Fankhauser et al. (2014)), or 3D environment (Wurm, Hornung, Bennewitz, Stachniss, and Burgard (2010)). Subsequently, contact planning methods plan a contact sequence, the sequence of contact patterns and placements. After determining the contact sequence, the motion planners introduced in 1.2.1.2 and 1.2.1.1 are typically utilized to determine the robot trajectory. Practical applications of these hierarchical approaches to complicated multi-contact planning problems are found in Jenelten, Miki, Vijayan, Bjelonic, and Hutter (2020); Kuindersma et al. (2016); Mastalli, Havoutis, Focchi, Caldwell, and Semini (2020).

There are various strategies for contact planning for legged locomotion problems. An early work (Bretl (2006)) proposed a two-stage sampling planning method, in which configurations at contacts are first determined and then transitions between them are planned. As the work only treated simple 2D problem settings, researchers have tried to solve more complicated and practical planning situations. A successful approach is to first plan the COM trajectory with reachability analysis on limbs and then determine the contact sequence (Fernbach, Tonneau, and Taïx (2018); Jenelten et al. (2020); Mastalli et al. (2020); Tonneau et al. (2018)). The sampling-based methods such as RRT and PRM are typically used. For further difficult situations such as foot step-planning on a cluttered terrain, optimization-based methods such as mixed-integer programming are also effective while it can require a large amount of computational time (Deits and Tedrake (2014)).

As well as legged locomotion, contact planning is also important in challenging manipulation planning such as contact-rich manipulation and non-prehensile manipulation. The hierarchical frameworks and sampling-based contact planning such as PRM and RRT are also utilized in these manipulation problems (Cheng, Huang, Hou, and Mason (2021); Saut, Sahbani, El-Khoury, and Perdereau (2007)).

1.2.1.4 Simultaneous contact and motion planning

Finally, we introduce an approach to plan the contact sequence and motion simultaneously. A well-known approach is *contact-implicit trajectory optimization* (CITO), which utilizes *time-stepping scheme* (Stewart and Trinkle (1996)) to simulate the system dynamics in numerical optimal control. The time-stepping scheme models the contacts by a set of complementarity constraints. The CITO is then formulated into *mathematical programs with complementarity constraints* (MPCC) or bilevel optimization (BO) problems. Pioneer works of the CITO are Yunt and Glocker (2006) and Yunt (2011), which formulated a BO problem via the augmented Lagrangian method for the CITO. In Posa, Cantu, and Tedrake (2014), more complicated CITO problems were formulated as MPCC. Successively, further practical applications of these methods have been investigated (Carius, Ranftl, Koltun, and Hutter (2018, 2019); Sleiman, Carius, Grandia, Wermelinger, and Hutter (2019)). However, both MPCC and BO involve theoretical and practical issues. MPCC inherently lacks constraint qualifications (Scheel and Scholtes (2000)) such as linear independence constraint qualification (LICQ), which causes numerical ill-conditionings. Moreover, numerical alleviations for such problems (Hoheisel, Kanzow, and Schwartz (2013)) often lead to undesirable stationary points (Nurkanović, Albrecht, and Diehl (2020)). BO problems are also difficult to solve in general. For example, even a linear bi-level optimization problem is NP-hard (Hansen, Jaumard, and Savard (1992)).

More computationally cheap but inaccurate contact models were considered in Neunert et al. (2018), Todorov (2014), and Chatzinikolaidis, You, and Li (2020). Neunert et al. (2018) modeled the contacts via a mass-spring-damper system, which is simple but leads to stiff and ill-conditioned optimization problems, and inaccurate solutions. Todorov (2014) proposed an analytically invertible and differentiable contact model. Chatzinikolaidis et al. (2020) investigated its applicability for the TO. However, the model lacks physical accuracy, e.g., allowing penetration between rigid objects, as we can see in a contact model comparison of Acosta, Yang, and Posa (2022).

1.2.2 Control

Next, we provide overviews of control methods of robotic systems with contacts, particularly of legged robots. For classical tracking control of robot manipulators, simple methods such as joint-space or task-space proportional-derivative (PD) controllers with gravity compensation work well (Lynch and Park (2017)). However, such methods cannot work for locomotion or non-trivial nonprehensile manipulation due to the following reasons. First, the robots (and the manipulated objects in nonprehensile manipulation) behave as a hybrid system due to contacts. Second, they can be underactuated due to a floating base of the legged robot or lacking the force-closure in non-prehensile manipulation. In the following, we introduce methods to tackle such problems, particularly for legged locomotion. Note that we herein exclude model predictive control (MPC)-based approaches; instead, we summarize them later in 1.2.3.

1.2.2.1 Reduced-order models for legged robot control

In the past robot hardware up to the early 2010s, we typically could not control the joint torques accurately due to low-power and high-gear ratio motors. Therefore, at the early stage, the legged robot control methods have been developed based on the position-controlled hardware. Moreover, the performance of CPUs was poor, and therefore we could not implement complicated algorithms. For these reasons, researchers have tried to capture simple laws of locomotion from physical perspectives and construct simple stabilization controllers.

In humanoid robotics, a promising approach has been to model the robot dynamics by its center of mass (COM) and contact interactions by a contact wrench from the zero-moment point (ZMP) (Vukobratović and Stepanenko (1972)). Based on the idea, Sugihara, Nakamura, and Inoue (2002) proposed a stabilizing walking controller for humanoid robots based on a linear inverted pendulum model (LIPM), which computes reference ZMP for given velocity commands of COM. Kajita et al. (2003) proposed a cart-table model for humanoid robots, which computes reference COM trajectory for a given ZMP trajectory. Although such COM-ZMP-based methods can successfully generate walking motions in real-time, they are restricted to walking on flat ground. That is, these methods cannot be applied to a robot walking on uneven terrains or having multi-contacts with the environment other than the ground.

To achieve stable control even in such a difficult situation, extensions of the COM-ZMP model have been studied. A successful extension is the divergent component of motion (DCM) (Takenaka, Matsumoto, and Yoshiike (2009)), which decomposes the LIPM dynamics into a stable part and an unstable part. DCM enables further analysis while it also simplifies the problem formulation. Englsberger, Ott, and Albu-Schäffer (2015) realized humanoid walking on uneven terrain by a DCM-based controller. Pratt, Carff, Drakunov, and Goswami (2006) achieved push-recovery control via the DCM (the DCM is referred to as the capture point in that paper). In contrast to these approximation model of the COM dynamics, Orin et al. (2013) proposed *centroidal momentum matrix*, which represents the (exact) translational and rotational dynamics of COM under various contact wrenches, which are essential in multi-contact situations.

Note that the aforementioned methods compute certain references such as acceleration of COM and trajectory of ZMP online but not the joint actuation. Therefore, a low-level joint controller is required to actuate the full body of the robot. In the past robot hardware, the joint position commands are first computed by inverse kinematics, e.g., to follow the reference position of COM, and then position-controlled motors track them. As the torque-controllable high-power motors become realistic, force-based controllers have been developed, such as torque allocation at each limb (Ott, Roa, and Hirzinger (2011)) and the following whole-body controllers.

1.2.2.2 Whole-body control

In contrast to the above methods with the reduced-order models, *whole-body control* (WBC) is based on the whole-body (or full-body) dynamics of the robotic systems. The WBC has gained attention extensively in recent years along with considerable developments in torque-controllable robot hardware, CPUs, and efficient quadratic programming (QP) solvers. The WBC was initially recognized as an extension of the well-known inverse dynamics control of robot manipulators to a robot with a floating base and contacts (Nakanishi, Mistry, and Schaal (2007)). Nowadays, in most studies such as Abe, Da Silva, and Popović (2007) and Saab et al. (2013), WBC is formulated as a QP; the equation of motion and contact constraints are regarded as the equality constraints, and additionally inequality constraints such as joint limitations and friction cones are considered. The QP-based WBC is versatile because it can incorporate various physical constraints, tasks, and other conditions in optimization problems as cost or constraints. There are a large number of variants of QP-based WBC. Ames and Powell (2013) considered the time derivative of the control Lyapunov function (CLF) directly as inequality constraints of the QP. Nguyen, Hereid, Grizzle, Ames, and Sreenath (2016) further introduced the control barrier function (CBF) into the QP in the same manner as the CLF to guarantee certain safety in the WBC. Ramuzat, Boria, and Stasse (2022) guaranteed the passivity of the closed-loop system by introducing a kind of storage function as the inequality constraint of the QP.

While the above WBC methods are promising for robotic systems with contacts, they are not proven to work well for systems with harsh impacts, which often arise in dynamic motions. Hybrid zero dynamics (HZD) is a promising virtual-constraint-based approach that can achieve periodic stability for systems with a floating base and impacts (Ames, Galloway, Sreenath, and Grizzle (2014); Westervelt, Grizzle, and Koditschek (2003)). Reher and Ames (2021) successfully realizes QP-based WBC with HZD on a bipedal robot. However, HZD lacks versatility; it can only treat periodic motions such as walking on a flat floor. To achieve more versatile and dynamic motions with WBC, Posa et al. (2016) first solved the TO to compute the optimal trajectory and a constrained linear quadratic regulator (LQR) to compute the optimal cost-to-go function offline. They then implemented QP-based WBC to track the optimized trajectory with the optimal cost-to-go function online.

1.2.2.3 State estimation for legged robots

The QP-based WBC method requires the full state of the robot including the floating base. Because we cannot measure the position, orientation, and linear velocity of the floating base directly, estimation of them is necessary to implement the WBC methods. Legged robots typically equip the joint encoders that measure the joint positions and an inertia measurement unit (IMU) that can measure the angular velocity and linear acceleration of the floating base in the body local coordinate. Bloesch et al. (2013) proposed an extended Kalman filter (EKF) that fuses the IMU measurements and kinematics information. Hartley, Ghaffari, Eustice, and Grizzle (2020) improved the convergence property of the EKF-based estimation by using invariant EKF (Barrau and Bonnabel (2016)). Camurri et al. (2017) proposed contact estimation without contact sensors for these EKF-based approaches. Camurri, Ramezani, Nobili, and Fallon (2020) incorporated additional observations in the sensor fusion such as the landmarks observed by camera and LIDAR.

1.2.3 Online motion planning as control: Model predictive control (MPC)

MPC is an advanced control method in which the OCP is solved at every sampling time for each measured or estimated state (Rawlings et al. (2017)). Since MPC solves the OCP, it shares the same versatility as the optimization-based motion planning methods presented in 1.2.1.2; it can consider physical characteristics and task specifications as the cost or constraints in a unified manner. Meanwhile, a drawback of MPC is that we have to solve an optimization problem in real-time.

Due to the computational limitation, in the early days, MPC could be implemented only for simple linear quadratic OCPs with low-control frequency, such as linear MPC based on cart-table-ZMP for reference COM trajectory generation for humanoid walking (Kajita et al. (2003)). However, along with the recent drastic developments of CPUs and numerical optimization algorithms (Diehl, Ferreau, and Haverbeke (2009); Kouzoupis, Frison, Zanelli, and Diehl (2018)), MPC has become to be able to treat more and more complicated problem settings with high control frequency.

Di Carlo, Wensing, Katz, Bledt, and Kim (2018) achieved fast and highly dynamic motion of a quadrupedal robot via a linear MPC. In the method, the robot dynamics are approximated by a linearized single rigid-body dynamics (SRBD) model. The contact wrenches, i.e., ground reaction forces (GRFs), of the four feet are then treated as the control input to the system under a given contact sequence. After determining the GRFs by solving the linear MPC problem, a low-level controller (typically a QP-based whole-body controller) determines the joint torques to track the GRFs. Such linearized SRBD-based methods are particularly effective when the weights of limbs are sufficiently small compared with the fully-body weight and pitch and roll rotations of the floating base are small. Ding, Pandala, and Park (2019) further improved the linearization of the rotational motion of SRBD model and achieved acrobatic motion on a quadrupedal robot.

Bledt and Kim (2019); Bledt, Wensing, and Kim (2017) included the contact positions as the optimization variables in the linearized SRBD-based MPC, which yields nonlinear MPC. Rathod et al. (2021) formulated nonlinear MPC based on the nonlinear SRBD model to capture the rotational dynamics of the SRBD model and realized robust locomotion that traverses a large object. Farshidian, Jelavic, Satapathy, Giftthaler, and Buchli (2017) further incorporated the full-kinematics in the nonlinear SRBD-based nonlinear MPC. The joint velocity and GRFs are treated as the control input in the method. Sleiman, Farshidian, Minniti, and Hutter (2021) introduced the centroidal momentum matrix (CMM) (Orin et al. (2013)) instead of the nonlinear SRBD model to consider exact COM dynamics in the nonlinear MPC with full-kinematics. These full-kinematics (and CMM) models have realized versatile tasks such as loco-manipulation (Orin et al. (2013)) and wheeled-legged control (Bjelonic et al. (2020)).

In recent years, MPC with whole-body (full-body) dynamics, which is called *whole-body MPC* (Dantec et al. (2021); Dantec, Taix, and Mansard (2022); Ishihara, Itoh, and Morimoto (2019); Koenemann et al. (2015); Mastalli et al. (2022); Neunert et al. (2018)), becomes possible thanks to recently developed efficient algorithms for rigid body dynamics and their derivatives (Carpentier and Mansard (2018); Giftthaler et al. (2017)). Since the whole-body MPC is based on accurate full-body robot dynamics, it can achieve energy-efficient, reasonable, and highly dynamic motion for a variety of problems. In particular, the whole-body MPC does not involve limitations on hardware platforms, working environment, and desired tasks, which often arise in MPC based on reduced-order models, i.e., the SRBD and centroidal models, due to the problem approximations. Furthermore, whole-body MPC can determine the control action with taking the future impact events into account, which cannot be achieved in the typical QP-based WBC methods. Therefore, the whole-body MPC is expected to be a unified and versatile approach that can achieve highly dynamic motions in various hardware platforms.

1.3 Challenges in Real-Time MPC of Robotic Systems

MPC, particularly whole-body MPC, has the potential to achieve highly dynamic motions in a unified manner. However, it is still challenging due to computational limitations. This section introduces these difficulties, which are tackled in this thesis.

1.3.1 Overview of MPC algorithms

Before presenting the challenges in real-time MPC of robotic systems, we herein review MPC algorithms. In particular, we focus on numerical optimization methods, which are useful in large-scale and complicated robotic applications. Other methods such as indirect methods and explicit MPC (see Chapters 8 and 7 of Rawlings et al. (2017)) are difficult to use for nonlinear and high-dimensional robotic problems and therefore omitted.

If the dynamical system and constraints are linear and the cost function is quadratic, the problem is classified as *linear MPC* problem. The MPC optimization problems are then formulated as QPs. To solve this class of problems efficiently, dedicated QP solvers have been developed, such as active-set methods (Ferreau, Kirches, Potschka, Bock, and Diehl (2014); Frasch, Sager, and Diehl (2015)), interior-point methods (Frison and Diehl (2020); Pandala, Ding, and Park (2019); Wang and Boyd (2010)), and alternating direction method of multipliers (ADMM)-based method (Stellato, Banjac, Goulart, Bemporad, and Boyd (2020)). Their efficiency typically depends on problem settings, e.g., numbers of optimization variables and inequality constraints. A historical review of QP solvers for MPC is found in Kouzoupis et al. (2018).

If the dynamical systems and constraints are nonlinear and the cost function is non-convex, the MPC optimization problem is classified as *nonlinear MPC* problem.

In this case, we typically need to solve nonlinear programs (NLPs) to find an optimal solution. Ohtsuka and Fujii (1997), Diehl et al. (2002), Ohtsuka (2004), and Diehl, Bock, and Schlöder (2005) are pioneer works of nonlinear MPC. These two works utilize tangential predictors to track the solution manifold of the NLP and perform only one (inexact) Newton iteration to update the solution at each time. Such ideas are further utilized in Zavala and Biegler (2009) and Zanelli, Quirynen, Jerez, and Diehl (2017).

Significant efforts also have been made to speed up Newton-type methods for solving NLPs. In particular, sequential quadratic programming (SQP) methods and (nonlinear) interior point methods have been developed. In SQP methods, Newton-type iteration corresponds to solving a QP subproblem which is typically solved by the aforementioned QP solvers for linear MPC problems. SQP methods, particularly with active-set QP solvers, can leverage warm-start but can lack efficiency for large-scale problems. In contrast, interior-point methods can treat large-scale problems efficiently while they cannot leverage warm-start. Further, interior-point methods have another computational advantage: in interior-point methods, the QP subproblem of the Newton-type iteration takes the same structure as the unconstrained time-varying linear-quadratic OCP. Efficient algorithms that leverage the structure to compute the Newton steps have been proposed in Rao, Wright, and Rawlings (1998), Wang and Boyd (2010), Zanelli, Domahidi, Jerez, and Morari (2020), Frison and Diehl (2020), and Deng and Ohtsuka (2019).

1.3.2 MPC algorithms for fast and large-scale robotic systems

Robotic systems are fast and large-scale systems. That is, whole-body MPC of them typically requires millisecond-range control frequency while it involves a high-dimensional state. Therefore, fast Newton-type algorithms are required to implement MPC. A promising approach is *Riccati recursion* approach (Frison (2016); Rao et al. (1998)), which can perform the Newton-type iteration for unconstrained OCPs in linear time complexity with respect to the length of the horizon. As it can solve only the unconstrained OCPs, we have to consider the inequality constraints by methods other than the active-set methods, e.g., interior point methods.

The Riccati approaches have been implemented in robotic applications. In robotics community, Riccati recursion-based Newton's methods are often interpreted as *differential dynamic programming* (DDP) (Jacobson and Mayne (1970)). Its Gauss-Newton counterpart is called as *iterative linear quadratic regulator* (iLQR) (Todorov and Li (2005)). They are almost identical to the Riccati recursion-based Newton (or Gauss-Newton) method for the direct single-shooting method in deterministic MPC problems (see Chapter 8 of Rawlings et al. (2017)). A variant of interior point methods (Grandia, Farshidian, Ranftl, and Hutter (2019); Hauser and Saccon (2006)) or augmented Lagrangian method (Howell, Jackson, and Manchester (2019)) are often used to consider constraints with these methods. In Koenemann et al. (2015), Neunert et al. (2018), Ishihara et al. (2019), Dantec et al. (2021), and Mastalli et al. (2022), Riccati-based whole-body MPC based on these methods has been successfully implemented in real-hardware, which shows the efficiency of the Riccati recursion algorithms.

Even though the whole-body MPC is implemented in actual robots, recent literature still indicates the necessity of further speeding up numerical optimization. In Dantec et al. (2021), whole-body MPC was implemented for a humanoid robot having 28-degrees of freedom (DOF). It reports that we typically need a large number of iterations until convergence, particularly with constraints, which implies that the whole-body MPC problems are highly nonlinear. Moreover, per Newton iteration took a large computational time due to high DOF, which results in a slow sampling interval from 10 ms to 100 ms. In Mastalli et al. (2022), whole-body MPC successfully achieved the dynamic and agile motions of a quadrupedal robot. However, the MPC still required a 10 ms sampling interval for the 18 DOF-robot. In summary, it is still required to reduce both the number of iterations until convergence and computational time per iteration.

1.3.3 MPC algorithms for switched systems

There are other difficulties in MPC of robotic systems with contacts. Robotic systems with contacts involve switches of dynamics and state jumps, which are formally modeled as *hybrid systems* (Goebel, Sanfelice, and Teel (2012)). A straightforward way to implement MPC for such systems is to formulate mixed-integer programs (MIPs), which is unrealistic due to exponentially growing computational time (Belotti et al. (2013)). Another way is to implement the CITO online. While the whole-body MPC based on the (approximated) CITO approaches have been implemented (Ishihara et al. (2019); Koenemann et al. (2015); Neunert et al. (2018)), they have limitations as discuss in 1.2.1.4. Koenemann et al. (2015) and Ishihara et al. (2019) implemented the whole-body MPC with smooth approximation model of contacts (Todorov (2014)). However, it lacks physical accuracy as it allows larger penetrations than other standard rigid-contact models (Acosta et al. (2022)). Neunert et al. (2018) utilized a simple spring-damper contact approximation model, which also lack accuracy. Moreover, it involves stiff optimization problems, which require very small time-steps of the numerical integration, i.e., a very large number of optimization variables in MPC.

Therefore, in practice, we have to take a hierarchical strategy composed of a high-level contact planner and a low-level MPC. The high level contact planner determines a contact sequence as in 1.2.1.3 and give it to low-level MPC. The low-level MPC then considers the (bi-lateral) contact constraints explicitly based on the contact sequence. By considering the contact constraints explicitly, there are no difficulties unique to the CITO approaches. The resultant MPC optimization problem for the given contact sequence is classified as OCP of *switched systems* (Xu and Antsaklis (2004)).

A characteristic of the OCP of the switched systems is that we have to optimize the *switching times* as well as the trajectory (Patterson and Rao (2014); Xu and Antsaklis (2004)). In robotic applications, this corresponds to optimizing contact timings for a given contact sequence (Farshidian, Kamgarpour, Pardo, and Buchli (2017); Li and Wensing (2020)). However, existing optimization methods (Farshidian, Kamgarpour, et al. (2017); Li and Wensing (2020); Patterson and Rao (2014); Xu and Antsaklis (2004)) for this problem are inefficient and therefore cannot achieve real-time MPC. Another issue is that robotic systems involve pure-state equality constraints when making contacts. Riccati recursion algorithms, a class of promising efficient MPC algorithms for large-scale systems, cannot treat such constraints efficiently. Therefore, existing robotic applications of the Riccati-based methods including Farshidian, Kamgarpour, et al. (2017) and Li and Wensing (2020) approximate such constraints by penalty function or augmented Lagrangian methods, which can lack accuracy or computational efficiency, and involve troublesome parameter tunings.

1.4 Outline and Contributions

This thesis presents algorithmic developments toward fast whole-body MPC of robotic systems with rigid contacts. The aforementioned challenges and our approaches are summarized in Fig. 1.1. Chapter 2 introduces the preliminaries of this thesis: rigid-body systems and the basics of MPC. Chapter 3 presents a fast optimization algorithm for the MPC problem (OCP) of rigid body systems using inverse dynamics. Chapter 4 presents a fast optimization algorithm for the MPC problem (OCP) of rigid body systems with rigid contacts via lifting the optimization problems. Chapter 5 presents a fast Riccati recursion algorithm for MPC problem (OCP) involving pure-state constraint arising in robotic MPC problems. Chapter 6 presents a fast Riccati recursion

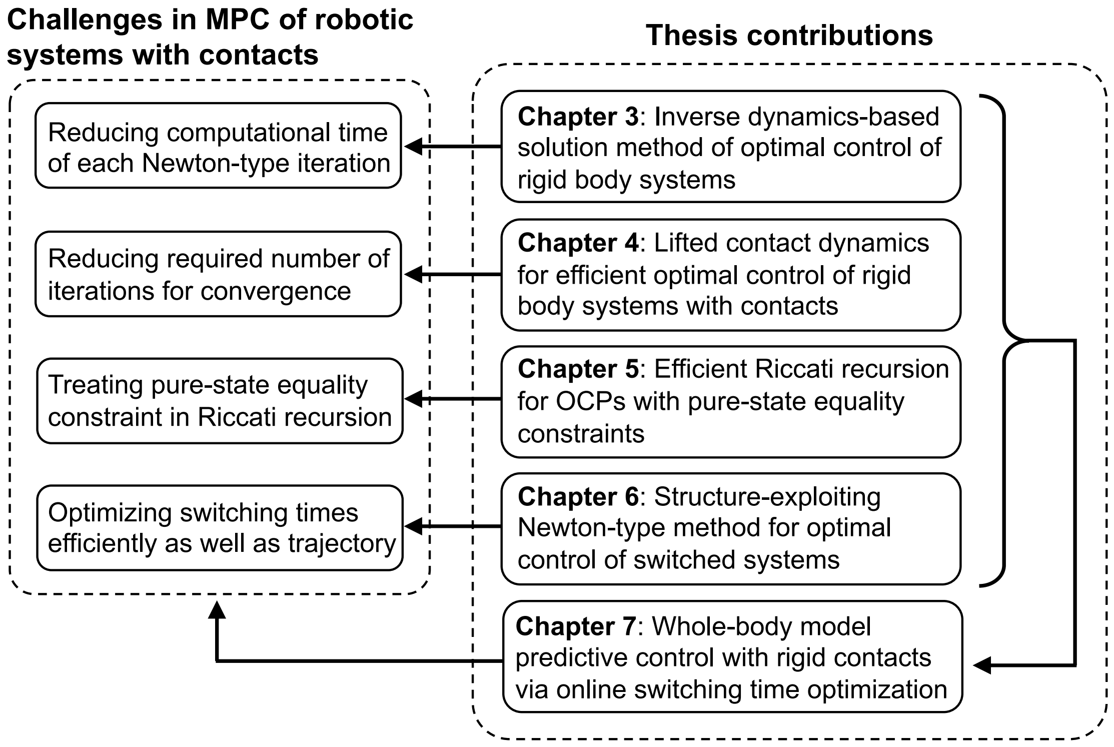

Challenges in MPC of robotic systems with contacts
Thesis contributions
Reducing computational time of each Newton-type iteration
Reducing required number of iterations for convergence
Treating pure-state equality constraint in Riccati recursion
Optimizing switching times efficiently as well as trajectory
Chapter 3: Inverse dynamics-based solution method of optimal control of rigid body systems
Chapter 4: Lifted contact dynamics for efficient optimal control of rigid body systems with contacts
Chapter 5: Efficient Riccati recursion for OCPs with pure-state equality constraints
Chapter 6: Structure-exploiting Newton-type method for optimal control of switched systems
Chapter 7: Whole-body model predictive control with rigid contacts via online switching time optimization

Figure 1.1: Thesis approaches for MPC of robotic systems with rigid contacts.

algorithm for MPC problem (OCP) involving switching time optimization problems. Chapter 7 presents a whole-body MPC of a quadrupedal robot using the STO algorithm. We conclude the thesis and present an outlook in Chapter 8. We shortly discuss the main contributions of each chapter as follows.

Chapter 2 – Preliminaries. This chapter introduces two basics of MPC of robotic systems which are used throughout this thesis. The first section of this chapter introduces rigid body systems, a modeling framework of real-world robotic systems such as robot manipulators and legged robots. In particular, we provide an overview of the mathematical formulation and numerical algorithms of the kinematics and dynamics of the rigid body systems. The second section of this chapter introduces off-the-shelf formulations and algorithms for MPC problems. We particularly focus on efficient numerical methods for large-scale systems such as rigid body systems. Specifically, we utilize the direct multiple shooting method to discretize the continuous-time problem into non-linear program (NLP) problems, the primal-dual interior point method to treat inequality constraints, and the Gauss-Newton Hessian approximation to compute the Hessian matrix. We further introduce the Riccati recursion algorithm that can perform Newton-type iteration in linear-time complexity with respect to the length of the horizon.

Chapter 3 – Inverse dynamics-based solution method of optimal control of rigid body systems. This chapter presents an solution method of OCPs for rigid-body systems on the basis of inverse dynamics, which are particulary suitable for systems with only few contacts, e.g., robot manipulators. In this method, we treat all variables, including the state, acceleration, and control input torques, as optimization variables and treat the inverse dynamics as an equality constraint. We eliminate the update of the control input torques from the linear equation of Newton's method by applying condensing for inverse dynamics. The size of the resultant linear equation is the same as that of the multiple-shooting method based on forward dynamics except for the variables related to the passive joints and contacts. Compared with the conventional methods based on forward dynamics, the proposed method reduces the computational cost of the dynamics and their sensitivities by utilizing the recursive Newton-Euler algorithm (RNEA) and its partial derivatives. Numerical experiments show that the proposed method outperforms state-of-the-art implementations of differential dynamic programming based on forward dynamics in terms of computational time and numerical robustness.

Chapter 4 – Lifted contact dynamics for efficient optimal control of rigid body systems with contacts. This chapter presents an efficient lifting approach for the OCPs of rigid-body systems with contacts to improve the convergence properties of Newton-type methods. To relax the high nonlinearity, we consider all variables, including the state, acceleration, contact forces, and control input torques, as optimization variables and the inverse dynamics and acceleration-level contact constraints as equality constraints. We eliminate the update of the acceleration, contact forces, and their dual variables from the linear equation to be solved in each Newton-type iteration in an efficient manner. As a result, the computational cost per Newton-type iteration is almost identical to that of the conventional non-lifted Newton-type iteration that embeds contact dynamics in the state equation. We conducted numerical experiments on the whole-body optimal control of various quadrupedal gaits subject to the friction cone constraints considered in interior-point methods and demonstrated that the proposed method can significantly increase the convergence speed to more than twice that of the conventional non-lifted approach.

Chapter 5 – Efficient Riccati recursion for optimal control problems with pure-state equality constraints. This chapter presents a novel approach to efficiently treat pure-state equality constraints in OCPs using a Riccati recursion algorithm. The proposed method transforms a pure-state equality constraint into a mixed state-control constraint such that the constraint is expressed by variables at a certain previous time stage. A Riccati recursion algorithm is derived to solve the OCP using the transformed constraints with linear time complexity in the grid number of the horizon, in contrast to a previous approach that scales cubically with respect to the total dimension of the pure-state equality constraints. Numerical experiments on the whole-body optimal control of quadrupedal gaits that involve pure-state equality constraints owing to contact switches demonstrate the effectiveness of the proposed method over existing approaches.

Chapter 6 – Structure-exploiting Newton-type method for optimal control of switched systems. This chapter presents an efficient Newton-type method for the optimal control of switched systems under a given mode sequence. A mesh-refinement-based approach is utilized to discretize continuous-time OCP and formulate a NLP, which guarantees the local convergence of a Newton-type method. A dedicated structure-exploiting algorithm (Riccati recursion) is proposed to perform a Newton-type method for the NLP efficiently because its sparsity structure is different from a standard OCP. The proposed method computes each Newton step with linear time-complexity for the total number of discretization grids as the standard Riccati recursion algorithm. Additionally, the computation is always successful if the solution is sufficiently close to a local minimum. Conversely, general quadratic programming (QP) solvers cannot accomplish this because the Hessian matrix is inherently indefinite. Furthermore, a modification on the reduced Hessian matrix is proposed using the nature of the Riccati recursion algorithm as the dynamic programming for a QP subproblem to enhance the convergence. A numerical comparison is conducted with off-the-shelf NLP solvers, which demonstrates that the proposed method is up to two orders of magnitude faster. Whole-body optimal control of quadrupedal gaits is also demonstrated and shows that the proposed method can achieve the whole-body MPC of robotic systems with rigid contacts.

Chapter 7 – Whole-body model predictive control with rigid contacts via online switching time optimization. This chapter presents a whole-body MPC of robotic systems built top on the algorithmic developments of Chapters 3–6. We treat robot dynamics with rigid contacts as a switched system and formulate an optimal control problem of switched systems to implement the MPC. We utilize the Newton-type solution algorithm proposed in Chapter 6 for the MPC problem that optimizes the switching times and trajectory simultaneously. Further, we treat the switching constraints by the constraint-transformation proposed in Chapter 5 and lifted algorithm proposed in Chapter in 4. The present efficient algorithm, unlike inefficient existing methods, enables online optimization as well as switching times.

The proposed MPC with online STO is compared over the conventional MPC with fixed switching times, through numerical simulations of dynamic jumping motions of a quadruped robot. In the simulation comparison, the proposed MPC successfully controls the dynamic jumping motions in twice as many cases as the conventional MPC, which indicates that the proposed method extends the ability of the whole-body MPC. We further conduct hardware experiments on the quadrupedal robot Unitree A1 and prove that the proposed method achieves dynamic motions on the real robot.

Chapter 2

Preliminaries

This chapter introduces the preliminaries of model predictive control (MPC) of robotic systems. The first section introduces *rigid body systems*, a fundamental modeling framework for various real-world robotic systems such as robot manipulators, quadruped robots, and humanoid robots. The second section introduces the basics of MPC, in particular, off-the-shelf numerical algorithms of MPC for nonlinear and large-scale systems, which are utilized throughout of this thesis.

2.1 Rigid Body Systems

The rigid body systems are dynamical systems composed of multiple rigid bodies and represent various real-world robotic systems. The kinematics and dynamics of the rigid body systems are fundamentals of planning and control methods including MPC. In particular, numerical algorithms to compute the kinematics and dynamics of large-scale rigid body systems have been extensively studied in the past literature. This section introduces these fundamentals, particularly focusing on useful ones in our MPC developments. Specifically, we assume a given contact sequence and model the impact explicitly by means of Newton's law to formulate tractable OCPs. This is in contrast to the time-stepping scheme (Stewart and Trinkle (1996)) that models the unilateral contacts by a set of complementarity constraints.

In the following of this section, we consider the rigid body system whose degree of freedom (DOF) is n. In general, the configuration, i.e., the generalized coordinate, of the rigid body system lies in a differentiable manifold, which is expressed as Q. For example, a legged robot has a floating base that lies in $SE(3)$, that is, the configuration space of the legged robot contains $SE(3)$. As such, generalized velocity is not just the time derivative of the configuration: it is defined at the tangent space of the configuration manifold. Because the tangent space and the Euclidean space are equivalent for typical manifolds such as $SE(3)$ (see Solà, Deray, and Atchuthan (2020) for detailed explanations on Lie groups), we denote the generalized velocity as $v \in \mathbb{R}^n$.

2.1.1 Kinematics

2.1.1.1 Forward and differential kinematics

A position and rotation of a frame of the rigid body system, which we denote as $p(q) \in \mathbb{R}^3$ and $R \in \mathbb{R}^{3\times3}$, are expressed as a differentiable function of the configuration q. This is called as *forward kinematics* and we write it as

$$\mathrm{FK}(q) := \begin{bmatrix} R(q) & p(q) \\ 0 & 1 \end{bmatrix} : Q \to SE(3). \tag{2.1}$$

We can compute the forward kinematics (2.1) efficiently by utilizing the tree structure of the system.

The differential kinematics considers the time-derivative of the forward kinematics. It mainly describes the relation between the velocity of a frame of interest and the generalized velocity. The stack of the linear and angular velocity of the frame $\mathcal{V} \in \mathcal{R}^6$ is expressed as

$$\mathcal{V}(q, v) = J_{\mathrm{FK}}(q)v \tag{2.2}$$

where

$$J_{\mathrm{FK}}(q) : Q \to \mathbb{R}^{6\times n} \tag{2.3}$$

is the Jacobian of the forward kinematics (2.1) expressed at the tangent space of Q at q. We can compute the Jacobian (2.1) efficiently in a recursive manner as the forward kinematics. Detailed introductions of kinematics are found in textbooks such as Murray, Li, and Sastry (2017) and Lynch and Park (2017).

2.1.1.2 Contact kinematics

Consider that bodies of the rigid body system have contacts with the environment. Then the configuration q of the system involves a certain constraint due to the contacts. For example, if a walking quadruped robot has a *point contact* between its foot and the ground, the position of the foot in the ground coordinate is fixed. That is, q must satisfy

$$p(q) - p_c = 0, \tag{2.4}$$

where $p_c \in \mathbb{R}^3$ is the given contact position. If a walking humanoid robot has a *surface contact* between its foot and the ground, the position and orientation of the foot in the ground coordinate is fixed. That is, q must satisfy

$$\mathrm{Log}(\mathrm{FK}^{-1}(q) \circ \mathrm{FK}_c) = 0. \tag{2.5}$$

where $\mathrm{FK}_c \in SE(3)$ is the given contact position and orientation. $\circ$ denotes the composition operation on $SE(3)$ and $\mathrm{Log}(\cdot)$ is a mapping from $SE(3)$ to Euclidean space $\mathbb{R}^6$ (Solà et al. (2020)). In the following, we describe the stack of these contact constraints as a differentiable function on q:

$$\mathbf{p}(q) = 0. \tag{2.6}$$

Next, let us consider the contact constraint (2.6) over a time interval, e.g., in simulations or optimal control problems (OCPs). In this case, instead of considering (2.6) over the time interval, we can consider the time-derivative of the contact constraint

$$\mathbf{v}(q, v) := \dot{p} = J(q)v = 0 \tag{2.7}$$

over the time interval, where $J(q)$ is the contact Jacobian, i.e., Jacobian of (2.6), provided that the original contact constraint (2.6) is satisfied at a point in the interval. We can further consider an acceleration-level constraint as

$$\mathbf{a}(q, v, a) := \ddot{\mathbf{p}} + 2\alpha\dot{\mathbf{p}} + \beta^2\mathbf{p} = J(q)a + \mathbf{b}(q, v), \tag{2.8}$$

where α and β are weight parameters, and we define

$$\mathbf{b}(q, v) := \dot{J}(q, v)v + 2\alpha J(q)v + \beta^2\mathbf{p}(q). \tag{2.9}$$

Instead of considering (2.6) over the time interval, we can consider (2.8) over the time interval provided that the original contact position constraint (2.6) and velocity constraint (2.7) are satisfied at a point in the interval. The form of (2.8) is called as *Baumgarte's stabilization* method (Baumgarte (1972)) in the sense that we can stabilize the violation of the original constraint (2.6) by setting $\alpha = \beta > 0$ properly (Flores, Machado, Seabra, and Silva (2011)). Note that with $\alpha = \beta = 0$, (2.8) is reduced to the contact acceleration, i.e., twice-time derivative of (2.6), which can cause constraint violations of (2.6) in practice.

2.1.2 Dynamics

Let $q \in Q$, $v \in \mathbb{R}^n$, $a \in \mathbb{R}^n$, $f \in \mathbb{R}^{n_f}$, and $u \in \mathbb{R}^{n_a}$ be the configuration, generalized velocity, acceleration, stack of the contact forces, and torques of the actuated joints, respectively. The equation of motion of the rigid-body system is expressed as

$$M(q)a + h(q,v) - J^{\mathrm{T}}(q)f = S^{\mathrm{T}}u, \tag{2.10}$$

where $M(q) \in \mathbb{R}^{n \times n}$ denotes the inertia matrix, $h(q,v) \in \mathbb{R}^n$ encompasses the Coriolis, centrifugal, and gravitational terms, $J(q) \in \mathbb{R}^{n_f \times n}$ denotes the stack of the contact Jacobians, and $S \in \mathbb{R}^{n_a \times n}$ denotes the selection matrix.

2.1.2.1 Inverse dynamics

The inverse dynamics is the calculation of the torque of a fully-actuated system under the given configuration, velocity, acceleration, and contact forces:

$$\mathrm{ID}(q,v,a,f) := M(q)a + h(q,v) - J^{\mathrm{T}}(q)f. \tag{2.11}$$

The recursive Newton-Euler algorithm (RNEA) (Featherstone (2008)) is an efficient recursive algorithm to compute the inverse dynamics (2.11). It is worth noting that an efficient algorithm is proposed to compute the analytical partial derivatives of the RNEA (Carpentier and Mansard (2018)), which is more efficient than the automatic differentiation counterpart (Giftthaler et al. (2017)).

2.1.2.2 Forward dynamics

The forward dynamics is the calculation of the acceleration from configuration,

$$\mathrm{FD}(q,v,u,f) := M^{-1}(q) \left\{ S^{\mathrm{T}}u + h(q,v) - J^{\mathrm{T}}(q)f \right\}. \tag{2.12}$$

An approach to compute the forward dynamics (2.12) is to first compute the inertia matrix $M(q)$ using the composite rigid body algorithm (CRBA) (Featherstone (2008)) and then compute the matrix inversion $M^{-1}(q)$. A more efficient algorithm for large-scale systems is the articulated body algorithm (ABA) (Featherstone (2008)). An efficient algorithm to compute the analytical partial derivatives of the forward dynamics (2.12) leveraging the analytical partial derivatives of the RNEA is proposed in Carpentier and Mansard (2018).

2.1.2.3 Contact-consistent forward dynamics

In numerical simulations and OCPs, we want to compute the acceleration and contact force for given q, v, and u. The forward dynamics (2.12) is then no longer applicable because f is not given. In this case, by using both (2.11) and (2.8), we can compute a and f for given q, v, and u:

$$\begin{bmatrix} M(q) & J^{\mathrm{T}}(q) \\ J(q) & O \end{bmatrix} \begin{bmatrix} a \\ -f \end{bmatrix} = \begin{bmatrix} S^{\mathrm{T}}u - h(q,v) \\ -\mathbf{b}(q,v) \end{bmatrix}. \tag{2.13}$$

This is called as the contact-consistent forward dynamics or simply as contact dynamics in this thesis.

2.1.2.4 Impulse dynamics

When bodies of the rigid body system collide with the other objects, an impulse change occurs in the generalized velocity of the system. This is expressed by means of Newton's law of impact, which is expressed as

$$M(q)\delta v - J^{\mathrm{T}}(q)\Lambda = 0, \tag{2.14}$$

where $\delta v \in \mathbb{R}^n$ denotes the impulse change in the generalized velocity, and $\Lambda \in \mathbb{R}^{n_f}$ denotes the stack of the impact forces. Herein, we assume a completely inelastic collision, which results in the contact velocity constraints of the form (2.7) immediately after the impulse as

$$\mathbf{v}(q, v, \delta v) := \dot{\mathbf{p}}(q, v + \delta v) = J(q)(v + \delta v) = 0. \tag{2.15}$$

By combining (2.14) and (2.15), we can compute δv and Λ for given q and v:

$$\begin{bmatrix} M(q) & J^{\mathrm{T}}(q) \\ J(q) & O \end{bmatrix} \begin{bmatrix} \delta v \\ -\Lambda \end{bmatrix} = \begin{bmatrix} 0 \\ -J(q)v \end{bmatrix}. \tag{2.16}$$

It is worth noting that the contact position constraints (2.6) must be satisfied at the impact instant.

2.1.2.5 State space representation

The state of the rigid body system is composed of the configuration and generalized velocity as $x := \begin{bmatrix} q^{\mathrm{T}} & v^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}}$. If there is no contact, the discrete-time state equation is given by

$$x_+ = \begin{bmatrix} q \oplus v\Delta\tau \\ v + \mathrm{FD}(q, v, u, \cdot)\Delta\tau \end{bmatrix}, \tag{2.17}$$

where $\Delta\tau$ is the time step and $\oplus$ denotes the addition operator on the manifold Q. If there are contacts, the state equation is given by

$$x_+ = \begin{bmatrix} q \oplus v\Delta\tau \\ v + \begin{bmatrix} I & O \end{bmatrix} \begin{bmatrix} M(q) & J^{\mathrm{T}}(q) \\ J(q) & O \end{bmatrix}^{-1} \begin{bmatrix} S^{\mathrm{T}}u - h(q,v) \\ \mathbf{b}(q,v) \end{bmatrix} \Delta\tau \end{bmatrix} \tag{2.18}$$

and the state-jump equation is given by

$$x_+ = \begin{bmatrix} q \\ v + \begin{bmatrix} I & O \end{bmatrix} \begin{bmatrix} M(q) & J^{\mathrm{T}}(q) \\ J(q) & O \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ -J(q)v \end{bmatrix} \end{bmatrix}. \tag{2.19}$$

2.1.3 Software

Software have been developed to efficiently compute kinematics and dynamics of rigid body systems. `RBDL` (Felis (2017)) is an open-source C++ library implementing kinematics computation, the CRBA, RNEA, and ABA. `RobCoGen` (Frigerio, Buchli, Caldwell, and Semini (2016)) is an open-source C++ code-generation tool supporting kinematics computation, CRBA, RNEA, ABA, and automatic differentiations of these algorithms (Giftthaler et al. (2017)). `Pinocchio` (Carpentier et al. (2019)) is an open-source C++ library for kinematics, CRBA, RNEA, ABA, and the analytical derivatives of the RNEA and ABA (Carpentier and Mansard (2018)).

2.2 Model Predictive Control

2.2.1 Overview

This section introduces mathematical formulations and numerical algorithms of MPC. We particularly focus on numerical optimal control techniques that are efficient for large-scale nonlinear systems. To provide introductory overview of basics of MPC, we consider a continuous-time dynamical system of the form of

$$\dot{x}(t) = f(x(t), u(t)) \tag{2.20a}$$

and inequality constraints

$$g(x(t), u(t)) \leq 0. \tag{2.20b}$$

Note that robotic systems such as robot manipulators and legged robots typically involve switches in dynamics and discontinuous changes in the state (i.e., state jumps) due to contacts. Therefore, the MPC formulations of these systems differ from (2.20a) and (2.20b). The details of such formulations are provided in the following chapters.

In this introductory chapter, we consider the following optimal control problem (OCP).

Problem 2.1. Continuous-time OCP

$$\min_{u(\cdot)} J = V(x(t+T)) + \int_t^{t+T} l(x(\tau), u(\tau)) d\tau \tag{2.21a}$$

$$\text{s.t.} \quad \frac{d}{d\tau} x(\tau) = f(x(\tau), u(\tau)) \quad \tau \in [t, t+T) \tag{2.21b}$$

$$g(x(\tau), u(\tau)) \leq 0 \quad \tau \in [t, t+T), \tag{2.21c}$$

where $V(x)$ and $l(x,u)$ are terminal cost and stage cost, respectively.

The OCP (2.21) is solved at each time t based on the measured or estimated state $x(t)$. After solving the OCP, we input the initial value of the optimal control input to the system. MPC realizes state-feedback control law by repeating this procedure for each state $x(t)$.

2.2.2 Numerical optimal control techniques for fast MPC of large-scale systems

In practice, continuous-time OCP (2.21) is approximated into finite-dimensional optimization problems to be solved numerically via Newton-type methods. Therefore, the solution efficiency of the Newton-type methods is critical for MPC implementation. In the following, we introduce efficient off-the-shelf numerical solution methods of the OCP, which are utilized throughout this thesis.

2.2.2.1 Direct multiple shooting method

We discretize the continuous-time OCP (2.21) into a discrete-time OCP that can be solved by off-the-shelf Newton-type optimization methods. This thesis consistently utilizes the direct multiple shooting (DMS) method (Bock and Plitt (1984)) for discretization which can achieve fast and robust numerical optimization. The characteristics of the DMS method are: 1) it discretizes the continuous-time OCP via explicit integration schemes such as the forward Euler method or Runge-Kutta methods and 2) it explicitly includes the state variables over the horizon in the optimization variables as well as control input. For ease of presentation, we choose the forward Euler method as the integration method in the DMS method. The OCP is then reduced to the following nonlinear program (NLP) problem.

Problem 2.2. NLP

$$\min_{u_0,...,u_{N-1},x_0,...,x_N} J = V(x_N) + \sum_{i=0}^{N-1} l(x_i, u_i)\Delta\tau \tag{2.22a}$$

$$\text{s.t.} \quad x(t) - x_0 = 0 \tag{2.22b}$$

$$x_i + f(x_i, u_i)\Delta\tau - x_{i+1} = 0, \;\; i = \{0, ..., N-1\}, \tag{2.22c}$$

$$g(x_i, u_i) \leq 0, \;\; i = \{0, ..., N-1\}. \tag{2.22d}$$

Other popular methods than the DMS method are the direct single-shooting method and the direct transcription method. The direct single-shooting method regards only the control input sequence as the optimization variables: the state variables are treated as the nonlinear function of the initial state $x(t)$ and the control input sequence $u_0, ..., u_{N-1}$. Although the direct single-shooting method is simple and easy to implement, the DMS method is superior to it in terms of computational time and numerical stability. Moreover, DMS can leverage parallel computation, e.g., in computation of the state equation (2.22c) while the direct single-shooting method cannot. The direct transcription can reduce the number of optimization variables and increase sparsity than the aforementioned shooting methods by representing the state trajectory via certain polynomials. Therefore, the direct collocation method can be faster than these two shooting methods if we implement Newton-type methods via certain linear algebras, e.g., direct matrix inversion of the Newton linear system. However, with some dedicated structure-exploiting algorithms for the Newton linear system such as the Riccati recursion algorithm presented in 2.2.2.4, the DMS method can be faster than the direct collocation method.

2.2.2.2 Primal-dual interior point method

There are two major methods to treat inequality constraints in NLP: the sequential quadratic programming (SQP) method and the primal-dual interior point (PDIP) method (Nocedal and Wright (2006)). SQP method can leverage warm-start in MPC. However, it can lack efficiency when there is a large number of inequality constraints. PDIP method is a promising approach for large-scale problems and allows suboptimal implementation in the sense that the barrier parameter remains moderate value (Wang and Boyd (2010)). Further, PDIP method is superior to the primal interior point method, another interior point method, in terms of numerical stability. For example, PDIP method can treat the infeasible iterates while the primal interior point method cannot.

In the PDIP method, we introduce slack variables $s_0, ..., s_{N-1}$ with barrier parameter $\epsilon > 0$ and reformulate the original NLP problem into the following NLP subproblem without inequality constraints.

Problem 2.3. Barrier NLP subproblem

$$\min_{u_0,...,u_{N-1},x_0,...,x_N,s_0,...,s_{N-1}} \tilde{J} = V(x_N) + \sum_{i=0}^{N-1} l(x_i, u_i)\Delta\tau - \epsilon \sum_{i=0}^{N-1} \ln s_i \tag{2.23a}$$

$$\text{s.t.} \quad x(t) - x_0 = 0 \tag{2.23b}$$

$$x_i + f(x_i, u_i)\Delta\tau - x_{i+1} = 0, \ \ i \in \{0, ..., N-1\}, \tag{2.23c}$$

$$g(x_i, u_i) + s_i = 0, \ \ i \in \{0, ..., N-1\}. \tag{2.23d}$$

In the PDIP method, we repeatedly 1) solve the NLP subproblem (2.23) by a Newton-type method with fixed ϵ and 2) reduce ϵ. By gradually reducing ϵ to zero, the NLP solution becomes close to the solution of the original NLP problem (2.22). To apply Newton-type method for the barrier NLP subproblem (2.23), we introduce the Lagrangian

$$\mathcal{L} := \tilde{J} + \lambda_0^{\mathrm{T}}(x(t) - x_0) + \sum_{i=0}^{N-1} \lambda_{i+1}^{\mathrm{T}}(x_i + f(x_i, u_i)\Delta\tau - x_{i+1}) + \sum_{i=0}^{N-1} \nu_i^{\mathrm{T}}(g(x_i, u_i) + s_i), \tag{2.24}$$

where $\lambda_0, ..., \lambda_N$ and $\nu_0, ..., \nu_{N-1}$ are the Lagrange multipliers. The (perturbed) Karush-Kuhn-Tucker (KKT) conditions of the NLP subproblem are given by the first-order derivatives of the Lagrangian $\mathcal{L}$, that is, (2.23b), (2.23c), (2.23d),

$$V_x(x_N) - \lambda_N = 0, \tag{2.25a}$$

$$-\lambda_i + \lambda_{i+1} + H_x^{\mathrm{T}}(x_i, u_i, \lambda_{i+1})\Delta\tau + g_x^{\mathrm{T}}(x_i, u_i)\nu_i = 0, \ \ i \in \{0, ..., N-1\}, \tag{2.25b}$$

$$H_u^{\mathrm{T}}(x_i, u_i, \lambda_{i+1})\Delta\tau + g_u^{\mathrm{T}}(x_i, u_i)\nu_i = 0, \ \ i \in \{0, ..., N-1\}, \tag{2.25c}$$

and

$$\mathrm{diag}(\nu_i)\ s_i = \epsilon 1, \ \ i \in \{0, ..., N-1\}, \tag{2.25d}$$

where

$$H(x, u, \lambda) := l(x, u) + \lambda^{\mathrm{T}} f(x, u) \tag{2.25e}$$

is the Hamiltonian (Bryson and Ho (1975)). Next, we linearize the KKT conditions with the Newton steps of all optimization variables $\Delta x_0, ..., \Delta x_N$, $\Delta u_0, ..., \Delta u_{N-1}$, $\Delta\lambda_0, ..., \Delta\lambda_N$, $\Delta s_0, ..., \Delta s_{N-1}$, and $\Delta\nu_0, ..., \Delta\nu_{N-1}$. We then obtain

$$x_0 + \Delta x_0 = x(t), \tag{2.26a}$$

$$A_i \Delta x_i + B_i \Delta u_i - \Delta x_{i+1} + \bar{x}_i = 0, \ \ i \in \{0, ..., N-1\}, \tag{2.26b}$$

$$C_i \Delta x_i + D_i \Delta u_i + \Delta s_i + \bar{g}_i = 0, \ \ i \in \{0, ..., N-1\}, \tag{2.26c}$$

$$Q_{xx,N} \Delta x_N - \Delta \lambda_N + \bar{l}_{x,N} = 0, \tag{2.26d}$$

$$Q_{xx,i} \Delta x_i + Q_{xu,i} \Delta u_i + A_i^{\mathrm{T}} \Delta \lambda_{i+1} - \Delta \lambda_i + C_i^{\mathrm{T}} \Delta \nu_i + \bar{l}_{x,i} = 0, \ \ i \in \{0, ..., N-1\}, \tag{2.26e}$$

$$Q_{xu,i}^{\mathrm{T}} \Delta x_i + Q_{uu,i} \Delta u_i + B_i^{\mathrm{T}} \Delta \lambda_{i+1} + D_i^{\mathrm{T}} \Delta \nu_i + \bar{l}_{u,i} = 0, \ \ i \in \{0, ..., N-1\}, \tag{2.26f}$$

and

$$\mathrm{diag}(\nu_i) \Delta s_i + \mathrm{diag}(s_i) \Delta \nu_i + \mathrm{diag}(\nu_i) s_i - \epsilon 1 = 0, \ \ i \in \{0, ..., N-1\}, \tag{2.26g}$$

where we define $A_i := f_x(x_i, u_i)$, $B_i := f_u(x_i, u_i)$, $C_i := g_x(x_i, u_i)$, $D_i := g_u(x_i, u_i)$, $Q_{xx,i} := H_{xx}(x_i, u_i, \lambda_{i+1}) \Delta \tau$, $Q_{xu,i} := H_{xu}(x_i, u_i, \lambda_{i+1}) \Delta \tau$, and $Q_{uu,i} := H_{uu}(x_i, u_i, \lambda_{i+1})$ $\Delta \tau$. Also, we define $\bar{x}_i$, $\bar{g}_i$, $\bar{l}_{x,i}$ and $\bar{l}_{u,i}$ as the right hands of (2.23c)–(2.25c). From (2.26c) and (2.26g), we have

$$\Delta s_i = -C_i \Delta x_i - D_i \Delta u_i - \bar{g}_i, \ \ i \in \{0, ..., N-1\} \tag{2.27a}$$

and

$$\begin{aligned} \Delta \nu_i = & -\mathrm{diag}^{-1}(s_i) \mathrm{diag}(\nu_i) \Delta s_i - \mathrm{diag}^{-1}(s_i)(\mathrm{diag}(\nu_i) s_i - \epsilon 1) \\ = & \ \mathrm{diag}^{-1}(s_i) \mathrm{diag}(\nu_i) C_i \Delta x_i + \mathrm{diag}^{-1}(s_i) \mathrm{diag}(\nu_i) D_i \Delta u_i \\ & + \mathrm{diag}^{-1}(s_i) \mathrm{diag}(\nu_i) \bar{g}_i - \mathrm{diag}^{-1}(s_i)(\mathrm{diag}(\nu_i) s_i - \epsilon 1), \ \ i \in \{0, ..., N-1\}. \end{aligned} \tag{2.27b}$$

By substituting (2.27a) and (2.27b) into (2.25b)–(2.25c), the Newton-step computation is reduced to find $\Delta u_0, ..., \Delta u_{N-1}$, $\Delta x_0, ..., \Delta x_N$, and $\Delta \lambda_0, ..., \Delta \lambda_N$ that satisfy the linear system (2.26a), (2.26b), (2.26d),

$$\tilde{Q}_{xx,i} \Delta x_i + \tilde{Q}_{xu,i} \Delta u_i + A_i^{\mathrm{T}} \Delta \lambda_{i+1} - \Delta \lambda_i + \tilde{l}_{x,i} = 0, \tag{2.28a}$$

and

$$\tilde{Q}_{xu,i}^{\mathrm{T}} \Delta x_i + \tilde{Q}_{uu,i} \Delta u_i + B_i^{\mathrm{T}} \Delta \lambda_{i+1} + \tilde{l}_{u,i} = 0, \tag{2.28b}$$

where

$$\tilde{Q}_{xx,i} := Q_{xx,i} + C_i^{\mathrm{T}} \mathrm{diag}^{-1}(s_i) \mathrm{diag}(\nu_i) C_i, \tag{2.28c}$$

$$\tilde{Q}_{xu,i} := Q_{xu,i} + C_i^{\mathrm{T}} \mathrm{diag}^{-1}(s_i) \mathrm{diag}(\nu_i) D_i, \tag{2.28d}$$

$$\tilde{Q}_{uu,i} := Q_{uu,i} + D_i^{\mathrm{T}} \mathrm{diag}^{-1}(s_i) \mathrm{diag}(\nu_i) D_i, \tag{2.28e}$$

$$\tilde{l}_{x,i} := \bar{l}_{x,i} + C_i^{\mathrm{T}} \left(\mathrm{diag}^{-1}(s_i) \mathrm{diag}(\nu_i) \bar{g}_i - \mathrm{diag}^{-1}(s_i)(\mathrm{diag}(\nu_i) s_i - \epsilon 1) \right), \tag{2.28f}$$

and

$$\tilde{l}_{u,i} := \bar{l}_{u,i} + D_i^{\mathrm{T}} \left(\mathrm{diag}^{-1}(s_i) \mathrm{diag}(\nu_i) \bar{g}_i - \mathrm{diag}^{-1}(s_i)(\mathrm{diag}(\nu_i) s_i - \epsilon 1) \right). \tag{2.28g}$$

After computing $\Delta u_0, ..., \Delta u_{N-1}$, $\Delta x_0, ..., \Delta x_N$, and $\Delta \lambda_0, ..., \Delta \lambda_N$, we compute the remaining Newton steps $\Delta s_0, ..., \Delta s_{N-1}$ and $\Delta \nu_0, ..., \Delta \nu_{N-1}$ from (2.27a) and (2.27b). The computational advantage of the PDIP method is that the structure of the linear system corresponds to the unconstrained linear quadratic OCPs and therefore can be solved efficiently.

After computing all the Newton steps, we determine the step size $\alpha \in (0, 1]$ via *fraction-to-boundary rule* (Nocedal and Wright (2006); Wächter and Biegler (2006)). It gives a maximum step size while keeping $s_0, ..., s_{N-1}$ and $\nu_0, ..., \nu_{N-1}$ to be positive with some margin.

2.2.2.3 Gauss-Newton Hessian approximation

Gauss-Newton Hessian approximation improves computational speed and numerical robustness for certain tracking-type MPC problems. Suppose that the terminal cost and stage cost take the form of

$$V(x) = \phi(x)^{\mathrm{T}} \phi(x), \tag{2.29}$$

and

$$l(x, u) = \phi(x, u)^{\mathrm{T}} \phi(x, u), \tag{2.30}$$

respectively. Gauss-Newton Hessian method then approximates the Hessian matrices as

$$Q_{xx,N}(x_N) = V_{xx}(x_N) \simeq \phi_x(x_N)^{\mathrm{T}} \phi_x(x_N), \tag{2.31}$$

$$\begin{aligned} \begin{bmatrix} Q_{xx,i} & Q_{xu,i} \\ Q_{ux,i} & Q_{uu,i} \end{bmatrix} &= \begin{bmatrix} H_{xx}(x_i, u_i, \lambda_{i+1}) & H_{xu}(x_i, u_i, \lambda_{i+1}) \\ H_{xu}^{\mathrm{T}}(x_i, u_i, \lambda_{i+1}) & H_{uu}(x_i, u_i, \lambda_{i+1}) \end{bmatrix} \Delta\tau \\ &\simeq \begin{bmatrix} \phi_x(x_i, u_i)^{\mathrm{T}} \\ \phi_u(x_i, u_i)^{\mathrm{T}} \end{bmatrix} \begin{bmatrix} \phi_x(x_i, u_i) & \phi_u(x_i, u_i) \end{bmatrix} \Delta\tau. \end{aligned} \tag{2.32}$$

The advantage of the Gauss-Newton Hessian approximation is two-fold. 1) We do not need to compute the twice partial derivatives of $V(x)$, $l(x, u)$, and $f(x, u)$. The twice partial derivative of $f(x, u)$ is particularly time-consuming in ridig body systems as we can see in (2.17)–(2.19). 2) The Hessian matrix of the Newton system is proven to be positive (semi)definite and the Newton-step computation is always successful. Meanwhile, the exact Hessian matrix can be indefinite and the Newton-step computation can be ill-conditioned.

2.2.2.4 Riccati recursion

At each Newton-type iteration, we compute the Newton steps $\Delta x_0, ..., \Delta x_N$, $\Delta u_0, ...,$ Δu_{N-1}, and $\Delta\lambda_0, ..., \Delta\lambda_N$ by solving the linear system (2.26a), (2.26b), (2.26d), (2.28a), and (2.28b). This is one of the most time-consuming parts of solving the NLP, for example, the direct Cholesky factorization typically requires the cubic computational burden $O(N^3)$. The Riccati recursion (Frison (2016); Rao et al. (1998)) can solve the linear problem with linear-time complexity with respect to N.

The Riccati recursion is derived to find a series of matrices P_i and vectors s_i such that

$$\Delta\lambda_i = P_i \Delta x_i - s_i \tag{2.33}$$

holds for all $i \in \{0, ..., N\}$. We start from the terminal stage ($i = N$), where we have (2.33) for $i = N$ with

$$P_N = Q_{xx,N}, \ s_N = -\bar{l}_N. \tag{2.34}$$

At the intermediate stages ($i < N$), suppose that the we have P_{i+1} and s_{i+1} satisfying (2.33). Then, by substituting (2.26b) and (2.33) into (2.28a) and (2.28b), we have (2.33) for $\Delta\lambda_i$ and Δx_i with

$$P_i := F_i - K_i^{\mathrm{T}} G_i K_i, \ s_i := A_i^{\mathrm{T}} (s_{i+1} - P_{i+1} \bar{x}_i) - \bar{l}_{x,i} - H_i k_i, \tag{2.35a}$$

where

$$F_i := Q_{xx,i} + A_i^{\mathrm{T}} P_{i+1} A_i, \tag{2.35b}$$

$$H_i := Q_{xu,i} + A_i^{\mathrm{T}} P_{i+1} B_i, \tag{2.35c}$$

$$G_i := Q_{uu,i} + B_i^{\mathrm{T}} P_{i+1} B_i, \tag{2.35d}$$

and

$$K_i := -G_i^{-1} H_i^{\mathrm{T}}, \ k_i := -G_i^{-1} (B_i^{\mathrm{T}} P_{i+1} \bar{x}_i - B_i^{\mathrm{T}} s_{i+1} + \bar{l}_{u,i}). \tag{2.35e}$$

We can repeat this derivation from $i = N$ to $i = 0$.

The algorithm of the Riccati recursion is composed of backward and forward recursions. For the given Newton system (2.26a), (2.26b), (2.26d), (2.28a), and (2.28b), first, we perform the backward recursion, i.e., we compute P_i, s_i, K_i, and k_i from $i = N$ to $i = 0$ by (2.34)–(2.35e). Next, we perform the forward recursion: we compute Δx_0 from (2.26a), and then we repeatedly compute $\Delta\lambda_i$, Δu_i, and Δx_{i+1} from (2.35a),

$$\Delta u_i = K_i \Delta x_i + k_i, \tag{2.36}$$

and (2.26b), respectively, from $i = 0$ to $i = N$.

2.2.2.5 Summary of Newton-type method for MPC

We summarize the aforementioned numerical optimal control techniques in Algorithm 2.1. The NLP (2.22) starts with initial state $x(t)$, initial guess of the optimization variables, and initial barrier parameter ϵ. The goal is to find a solution such that the KKT error (i.e., l_2-norm of residual in the KKT conditions (2.23b)–(2.25d)) and barrier parameter are smaller than predefined tolerances $\gamma > 0$ and $\epsilon_{\rm tol} > 0$. Note that it is possible to use other criteria for the convergence than the aforementioned one such as a max-norm of the KKT residual (see Nocedal and Wright (2006)). The outer **while** loop (lines 2–25) expresses the NLP subproblem iteration for varying barrier parameter ϵ and inner **while** loop (lines 4–23) does the Newton-type iteration for the fixed barrier parameter. The inner **while** loop consists of the computation of the Newton steps of the PDIP method via the Riccati recursion (lines 4–18) and solution update with the step-size selection via the fraction-to-boundary (lines 19–22). After solving the inner NLP problem with a fixed barrier parameter, that is, if the KKT error is smaller than γ, we reduce the barrier parameter, e.g., via $\epsilon \leftarrow \beta\epsilon$ with some $0 < \beta < 1$. This procedure is repeated until the barrier parameter is smaller than the predefined threshold $\epsilon_{\rm tol}$.

In practice, the computational time is limited under the sampling period of the control system. Therefore, the *suboptimal MPC* is typically employed in which the iterate is utilized as the actual control input before strictly converging to the optimal solution. Suboptimal MPC is often implemented by fixing the number of Newton-type iterations. In interior point methods, this also can be implemented by setting the barrier parameter threshold $\epsilon_{\rm tol}$ moderately large or even fixing the barrier parameter ϵ (Wang and Boyd (2010)).

Algorithm 2.1 Summary of the Newton-type method for the NLP (2.22)

```
Input: Initial state $x(t)$, initial guesses of the optimization variables, initial barrier parameter, and tolerances of KKT error and barrier parameter ($\gamma$, $\epsilon_{\rm tol}$).
Output: Optimal solution $x_0, ..., x_N$, $u_0, ..., u_{N-1}$, $\lambda_0, ..., \lambda_N$, $s_0, ..., s_{N-1}$, and $\nu_0, ..., \nu_{N-1}$
 1: // NLP subproblem iterations.
 2: while $\epsilon > \epsilon_{\rm tol}$ do
 3:     // Newton-type iterations with the fixed barrier parameter.
 4:     while The KKT error is larger than $\gamma$ do
 5:         for $i = 0, \cdots, N$ do in parallel
 6:             Compute matrices and vectors in (2.26a)–(2.26g).
 7:         end for
 8:         // Backward Riccati recursion.
 9:         for $i = N, \cdots, 0$ do in serial
10:             Compute $P_i$, $s_i$, $K_i$, and $k_i$ from (2.35a)–(2.35e).
11:         end for
12:         // Forward Riccati recursion.
13:         for $i = 0, \cdots, N-1$ do in serial
14:             Compute Newton steps $\Delta\lambda_i$, $\Delta u_i$, and $\Delta x_{i+1}$ from (2.33), (2.36), and (2.26b), respectively.
15:         end for
16:         for $i = 0, \cdots, N-1$ do in parallel or serial
17:             Compute Newton steps $\Delta s_0, ..., \Delta s_{N-1}$ and $\Delta\nu_0, ..., \Delta\nu_{N-1}$.
18:         end for
19:         Determine the step size $\alpha \in (0, 1]$ via the fraction-to-boundary rule.
20:         for $i = 0, \cdots, N$ do in parallel or serial
21:             Update variables $x_i$, $u_i$, $\lambda_i$, $s_i$, and $\nu_i$, e.g., as $x_i \leftarrow x_i + \alpha\Delta x_i$.
22:         end for
23:     end while
24:     Reduce the barrier parameter, e.g., via $\epsilon \leftarrow \beta\epsilon$ $(0 < \beta < 1)$.
25: end while
```

Chapter 3

Inverse Dynamics-Based Solution Method of Optimal Control of Rigid Body Systems[1]

3.1 Introduction

Optimal control plays a significant role in motion planning and control such as trajectory optimization (TO) and model predictive control (MPC) (Rawlings et al. (2017)) for rigid-body systems. TO can generate dynamically consistent motion under versatile control objectives and constraints even for highly nonlinear systems such as underactuated systems by solving the optimal control problem (OCP). MPC leverages the same advantages as TO for real-time control by solving the OCP at each sampling time. However, we still have to improve the computational efficiency and numerical robustness of the OCP for rigid-body systems whose dynamics are complicated and highly nonlinear. this The computational time of the OCP for rigid-body systems depends substantially on the computational time of the dynamics and their sensitivities. Most previous researches on the OCP for rigid-body systems (e.g., Koenemann et al. (2015); Mastalli et al. (2020); Neunert et al. (2018)) have been based on forward dynamics, which is a calculation of the generalized acceleration for the given configuration, generalized velocity, and generalized torques. These studies incorporate forward dynamics into the state equation, which is a natural representation, especially for single-shooting methods such as differential dynamic programming (DDP) (Tassa, Erez, and Todorov (2012)) and the iterative linear quadratic regulator

[1]© 2021 IEEE. Reprinted, with permission, from S. Katayama and T. Ohtsuka, "Efficient solution method based on inverse dynamics for optimal control problems of rigid body systems," 2021 IEEE International Conference on Robotics and Automation (ICRA 2021), pp. 2070–2076, 2021.

(iLQR) (Todorov and Li (2005)). They utilize the ABA (Featherstone (1983)), an efficient recursive algorithm to compute forward dynamics, or directly compute the inverse of the joint inertia matrix, in solving the OCP. In contrast, our previous work (Katayama and Ohtsuka (2020)) proposed basing the OCP on inverse dynamics to reduce the computational time compared with that of the OCP based on forward dynamics. Inverse dynamics are calculations of the generalized torques for a given configuration, generalized velocity, and generalized acceleration. We can compute the inverse dynamics with a smaller computational time than with forward dynamics because the RNEA is faster than ABA (Featherstone (2008)). Moreover, we can compute the sensitivities of inverse dynamics faster than those of forward dynamics Carpentier and Mansard (2018); Neunert, Giftthaler, Frigerio, Semini, and Buchli (2016). Therefore, we can reduce the computational cost of the OCP for rigid-body systems by using inverse dynamics instead of forward dynamics, as illustrated numerically in Katayama and Ohtsuka (2020). However, our previous approach of Katayama and Ohtsuka (2020) has a potential drawback when it is utilized for MPC because it is a single-shooting method; i.e., it regards only the acceleration as the decision variable, which means it cannot leverage parallel computing and can lack numerical robustness.

The single-shooting method regards only the control input as decision variables and computes the state on the horizon by simulating the system's dynamics based on the control input at each iteration. Typical examples of the single-shooting method are DDP and iLQR, which solve the linear equation of Newton's method by using Riccati recursion Frison (2016) and are very popular in robotics applications Koenemann et al. (2015); Mastalli et al. (2020); Neunert et al. (2018). In contrast, the multiple-shooting method regards all variables (the state, costate, and control inputs) as optimization variables and can inherently leverage parallel computing by splitting the computation of the residual of the Karush–Kuhn–Tucker (KKT) conditions and the Hessian of the KKT condition into each time stage, which is impossible for the single-shooting methods due to their expensive serial computational parts, such as the simulation of the system's dynamics over the horizon. Even in a single-thread computation, the computational cost of the multiple-shooting method is almost the same as that of the single-shooting method even though it has more optimization variables thanks to structure-exploiting Newton-type methods (Albersmeyer and Diehl (2010); Bock and Plitt (1984); Frison (2016)). Further, the multiple-shooting method is empirically known to converge quickly even if the initial guess of the solution is far from the (local) optimal solution; In contrast, the single-shooting method converges slowly or even diverges in such a situation.

In this chapter, we propose an efficient solution method of the OCP for rigid-body systems based on inverse dynamics and the multiple-shooting method. We treat all variables, including the state, acceleration, and control input torques, as optimization variables, and treat the inverse dynamics as an equality constraint. We eliminate the update of the control input torques from the linear equation of Newton's method by applying condensing for inverse dynamics. The size of the resultant linear equation is the same as that of the multiple-shooting method based on forward dynamics except for the variables related to the passive joints and contacts. Compared with the conventional methods based on forward dynamics, the proposed method reduces the computational cost of the dynamics and their sensitivities by utilizing RNEA and its partial derivatives. In addition, it increases the sparsity of the Hessian of the KKT conditions, which reduces the computational cost, e.g., of Riccati recursion. Note that the inverse dynamics-based formulation is also utilized in a contact-implicit TO (Erez and Todorov (2012); Todorov (2014)) to alleviate the numerical ill-conditioning due to an approximation with smooth contact model. In contrast to these studies, our method is not limited to a specific contact-implicit TO, e.g., it can treat rigid contacts, as illustrated in a numerical experiment.

This chapter is organized as follows. In Section 3.2, we formulate the OCP based on inverse dynamics. Section 3.3 introduces the proposed solution method with condensing of inverse dynamics. Section 3.4 compares the proposed method with state-of-the-art implementations of DDP/iLQR and demonstrates its effectiveness in terms of the computational time and numerical robustness. We conclude in Section 3.5 with a brief summary and mention of future work.

3.2 Optimal Control Problem Based on Inverse Dynamics

3.2.1 Rigid-body systems

Let Q be the configuration manifold of the rigid-body system. Let $q \in Q$, $v \in \mathbb{R}^n$, $a \in \mathbb{R}^n$, $f \in \mathbb{R}^{n_f}$, and $u \in \mathbb{R}^n$ be the configuration, generalized velocity, acceleration, stack of the contact forces, and input torques, respectively. The equation of motion of the rigid-body system is given by

$$M(q)a + h(q, v) - J^{\mathrm{T}}(q)f = u, \tag{3.1}$$

where $M(q) \in \mathbb{R}^{n \times n}$ denotes the inertia matrix, $h(q, v) \in \mathbb{R}^n$ encompasses Coriolis, centrifugal, and gravitational terms, and $J(q) \in \mathbb{R}^{n_f \times n}$ denotes the stack of the contact Jacobians. We write (3.1) as an equality constraint of the inverse dynamics as follows:

$$\mathrm{ID}(q, v, a, f) - u = 0, \tag{3.2}$$

where $\mathrm{ID}(q, v, a, f)$ is defined by the left-hand side of (3.1) in the formulation of the OCP. Note that we can efficiently compute $\mathrm{ID}(q, v, a, f)$ by using RNEA and its partial derivatives by using the partial derivatives of RNEA (Carpentier and Mansard (2018)). We also assume that the input torques u are an n-dimensional vector even when the system is underactuated, as RNEA and its partial derivatives are well-defined only for fully actuated systems. As stated in the next subsection, we treat the underactuated systems by introducing an equality constraint that zeros the elements of u corresponding to passive joints.

3.2.2 Optimal control problem

We consider an OCP with N time stages. The configuration, velocity, acceleration, external forces, and input torques for N stages are denoted as $q_0, ..., q_N \in Q$, $v_0, ..., v_N \in \mathbb{R}^n$, $a_0, ..., a_{N-1} \in \mathbb{R}^n$, $f_0, ..., f_{N-1} \in \mathbb{R}^{n_f}$, $u_0, ..., u_{N-1} \in \mathbb{R}^n$, all of which are regarded as optimization variables to formulate the OCP based on inverse dynamics. We assume the initial state is given by $\bar{q}$ and $\bar{v}$, and then consider the constraints on the initial state

$$\delta(\bar{q}, q_0) = 0, \ \bar{v} - v_0 = 0, \quad \bar{q} \in Q, \ \bar{v} \in \mathbb{R}^n, \tag{3.3}$$

where $\delta(q_1, q_2) \in \mathbb{R}^n$ denotes the subtraction operation of the configurations $q_2 \in Q$ from $q_1 \in Q$ on the manifold Q. The state equation discretized with the forward Euler method is given by

$$\begin{bmatrix} \delta(q_i, q_{i+1}) + v_i \Delta\tau \\ v_i - v_{i+1} + a_i \Delta\tau \end{bmatrix} = 0, \ \ i = 0, ..., N-1, \tag{3.4}$$

where $\Delta\tau$ is the time step of the discretization given by $\Delta\tau = T/N$ with the length of the horizon T. The state equation (3.4) takes a simple form since we consider the equation of the motion of the system (3.1) as an equality constraint (3.2) in the OCP and regard the acceleration a_i as the optimization variables. In general, we can also assume an m_c-dimensional equality constraint,

$$C(q_i, v_i, a_i, u_i, f_i)\Delta\tau = 0, \ \ i = 0, ..., N-1, \tag{3.5}$$

and m_g-dimensional inequality constraint,

$$g(q_i, v_i, a_i, u_i, f_i)\Delta\tau \leq 0, \quad i = 0, ..., N-1. \tag{3.6}$$

Note that we multiply $C(\cdot)$ and $g(\cdot)$ by $\Delta\tau$ in (3.5) and (3.6) so that the proposed formulation will correspond to the continuous-time Euler-Lagrange equations discretized with a time step $\Delta\tau$ (Bryson and Ho (1975)). If the system is underactuated, (3.5) contains the elements of the control input torques corresponding to the passive joints. For example, if the system has a floating base whose joint indices are 1 to 6, $C(q_i, v_i, a_i, u_i, f_i)$ in (3.5) includes $[u_i^{(1)} \; u_i^{(2)} \; u_i^{(3)} \; u_i^{(4)} \; u_i^{(5)} \; u_i^{(6)}]^{\rm T}$, where $u_i^{(j)}$ is the j-th element of u_i.

For the above description of the rigid-body system and constraints, the OCP is given by

$$\min_{q_i, v_i, a_i, u_i, f_i} J = V(q_N, v_N) + \sum_{i=0}^{N-1} l(q_i, v_i, a_i, u_i, f_i)\Delta\tau,$$

subject to (3.2)–(3.6), where $V(q_N, v_N)$ denotes the terminal cost and $l(q_i, v_i, a_i, u_i, f_i)\Delta\tau$ denotes the stage cost. We treat the inequality constraints (3.6) by using the primal-dual interior point (PDIP) method (Nocedal and Wright (2006)), which can treat large-scale and nonlinear constraints efficiently. The inequality constraint (3.6) is then transformed into an equality constraint by introducing slack variables $s_i \in \mathbb{R}^{m_g}$,

$$g(q_i, v_i, a_i, u_i, f_i)\Delta\tau + s_i\Delta\tau = 0, \tag{3.7}$$

with an additional inequality constraint $s_i \geq 0$.

3.2.3 KKT conditions

Next, we derive the KKT conditions, necessary conditions for optimal control (Bryson and Ho (1975); Nocedal and Wright (2006)). Let the set of primal variables at stage i be $y_i := \{q_i \;\; v_i \;\; a_i \;\; f_i \;\; u_i\}$ and the set of primal variables without the control input torques be $\tilde{y}_i := \{q_i \;\; v_i \;\; a_i \;\; f_i\}$ for $i = 0, ..., N-1$. We then introduce the Lagrangian of this OCP with PDIP method,

$$\begin{aligned}
\mathcal{L} = V(q_N, v_N) &+ \sum_{i=0}^{N-1} l(y_i)\Delta\tau + \begin{bmatrix} \lambda_0 \\ \gamma_0 \end{bmatrix}^{\rm T} \begin{bmatrix} \delta(\bar{q}, q_0) \\ \bar{v} - v_0 \end{bmatrix} + \sum_{i=0}^{N-1} \begin{bmatrix} \lambda_{i+1} \\ \gamma_{i+1} \end{bmatrix}^{\rm T} \begin{bmatrix} \delta(q_i, q_{i+1}) + v_i\Delta\tau \\ v_i - v_{i+1} + a_i\Delta\tau \end{bmatrix} \\
&+ \sum_{i=0}^{N-1} \beta_i^{\rm T} ({\rm ID}(\tilde{y}_i) - u_i)\Delta\tau + \sum_{i=0}^{N-1} \mu_i^{\rm T} C(y_i)\Delta\tau + \sum_{i=0}^{N-1} \nu_i^{\rm T} (g(y_i)\Delta\tau + s_i\Delta\tau) \\
&- \sum_{i=0}^{N-1} \epsilon \ln s_i \Delta\tau,
\end{aligned}$$

where λ_i, $\gamma_i \in \mathbb{R}^n$, $\mu_i \in \mathbb{R}^{m_c}$, and $\nu_i \in \mathbb{R}^{m_g}$ denote the Lagrange multipliers with respect to the state equation (3.4), the equality constraint (3.5), and the inequality constraints (3.6), and $\epsilon > 0$ denotes the barrier parameter. The KKT conditions are then given by (3.2)–(3.5), (3.7),

$$\begin{bmatrix} \mathcal{L}_{q_N}^{\mathrm{T}} \\ \mathcal{L}_{v_N}^{\mathrm{T}} \end{bmatrix} = \begin{bmatrix} V_{q_N}^{\mathrm{T}}(q_N, v_N) \\ V_{v_N}^{\mathrm{T}}(q_N, v_N) \end{bmatrix} + \begin{bmatrix} \delta_{q_N}^{\mathrm{T}}(q_{N-1}, q_N)\lambda_N \\ -\gamma_N \end{bmatrix} = 0, \tag{3.8}$$

and the following equations for $i = 0, ..., N-1$,

$$\begin{aligned} \begin{bmatrix} \mathcal{L}_{q_i}^{\mathrm{T}} \\ \mathcal{L}_{v_i}^{\mathrm{T}} \end{bmatrix} = & \begin{bmatrix} l_{q_i}^{\mathrm{T}}(y_i)\Delta\tau \\ l_{v_i}^{\mathrm{T}}(y_i)\Delta\tau \end{bmatrix} + \begin{bmatrix} \delta_{q_i}^{\mathrm{T}}(q_i, q_{i+1}) & O_{n\times n} \\ I_n \Delta\tau & I_n \end{bmatrix} \begin{bmatrix} \lambda_{i+1} \\ \gamma_{i+1} \end{bmatrix} + \begin{bmatrix} \mathrm{ID}_{q_i}^{\mathrm{T}}\Delta\tau(\tilde{y}_i) \\ \mathrm{ID}_{v_i}^{\mathrm{T}}\Delta\tau(\tilde{y}_i) \end{bmatrix} \beta_i \\ & + \begin{bmatrix} C_{q_i}^{\mathrm{T}}\Delta\tau(y_i) \\ C_{v_i}^{\mathrm{T}}\Delta\tau(y_i) \end{bmatrix} \mu_i + \begin{bmatrix} g_{q_i}^{\mathrm{T}}\Delta\tau(y_i) \\ g_{v_i}^{\mathrm{T}}\Delta\tau(y_i) \end{bmatrix} \nu_i + \begin{bmatrix} \delta_{q_i}^{\mathrm{T}}(q_{i-1}, q_i)\lambda_i \\ -\gamma_i \end{bmatrix} = 0, \end{aligned} \tag{3.9}$$

$$\begin{aligned} \begin{bmatrix} \mathcal{L}_{a_i}^{\mathrm{T}} \\ \mathcal{L}_{f_i}^{\mathrm{T}} \end{bmatrix} = & \begin{bmatrix} l_{a_i}^{\mathrm{T}}(y_i)\Delta\tau \\ l_{f_i}^{\mathrm{T}}(y_i)\Delta\tau \end{bmatrix} + \begin{bmatrix} O_{n\times 1} \\ \gamma_{i+1}\Delta\tau \end{bmatrix} + \begin{bmatrix} \mathrm{ID}_{a_i}^{\mathrm{T}}(\tilde{y}_i)\Delta\tau \\ \mathrm{ID}_{f_i}^{\mathrm{T}}(\tilde{y}_i)\Delta\tau \end{bmatrix} \beta_i \\ & + \begin{bmatrix} C_{a_i}^{\mathrm{T}}(y_i)\Delta\tau \\ C_{f_i}^{\mathrm{T}}(y_i)\Delta\tau \end{bmatrix} \mu_i + \begin{bmatrix} g_{a_i}^{\mathrm{T}}(y_i)\Delta\tau \\ g_{f_i}^{\mathrm{T}}(y_i)\Delta\tau \end{bmatrix} \nu_i = 0, \end{aligned} \tag{3.10}$$

$$\mathcal{L}_{u_i}^{\mathrm{T}} = l_{u_i}^{\mathrm{T}}(y_i)\Delta\tau - \beta_i\Delta\tau + C_{u_i}^{\mathrm{T}}(y_i)\mu_i\Delta\tau + g_{u_i}^{\mathrm{T}}(y_i)\nu_i\Delta\tau = 0, \tag{3.11}$$

and

$$\mathrm{diag}(s_i)\nu_i = \epsilon 1, \tag{3.12}$$

where $\epsilon 1$ denotes an m_g-dimensional vector, all of whose elements are ϵ. The last equation (3.12) denotes the complementarity conditions between the slack variable s_i and the Lagrange multiplier ν_i.

3.3 Solution Method of Optimal Control Problem

3.3.1 Linearization for Newton's method

The above KKT conditions are linearized for Newton's method. We apply Gauss-Newton Hessian approximation because the inverse dynamics constraint (3.2) is complicated enough to make it impractical to compute the second-order partial derivatives of (3.2). Accordingly, the Hessian of the Lagrangian at the terminal stage is approximated by the Gauss-Newton-approximated Hessian of the terminal cost and that at the intermediate stage by the Gauss-Newton-approximated Hessian of the stage cost. For example, when the terminal cost and the stage cost take a quadratic form, we have $\mathcal{L}_{q_N q_N} \simeq V_{q_N q_N}$ and $\mathcal{L}_{q_i q_i} \simeq l_{q_i q_i}$. With the approximated Hessian, (3.8) is linearized into a linear equation with respect to the Newton directions $\Delta\lambda_N, \Delta\gamma_N, \Delta q_N, \Delta v_N \in \mathbb{R}^n$ as

$$\begin{bmatrix} \mathcal{L}_{q_N}^{\mathrm{T}} \\ \mathcal{L}_{v_N}^{\mathrm{T}} \end{bmatrix} + \begin{bmatrix} \mathcal{L}_{q_N q_N} & \mathcal{L}_{q_N v_N} \\ \mathcal{L}_{v_N q_N} & \mathcal{L}_{v_N v_N} \end{bmatrix} \begin{bmatrix} \Delta q_N \\ \Delta v_N \end{bmatrix} + \begin{bmatrix} \delta_{q_N}^{\mathrm{T}}(q_{N-1}, q_N)\Delta\lambda_N \\ -\Delta\gamma_N \end{bmatrix} = 0. \tag{3.13}$$

As well, (3.9)–(3.11) are linearized into linear equations with respect to the Newton directions $\Delta\lambda_{i+1}, \Delta\gamma_{i+1}, \Delta q_i, \Delta v_i, \Delta a_i, \Delta u_i, \Delta\beta_i \in \mathbb{R}^n$, $\Delta f_i \in \mathbb{R}^{n_f}$, and $\Delta\mu_i \in \mathbb{R}^{m_c}$. Note that the directions related to the PDIP method, $\Delta s_i, \Delta\nu_i \in \mathbb{R}^{m_g}$, are eliminated explicitly from the linear equations by adding certain terms related to the logarithmic barrier functions to the Hessians (e.g., $\mathcal{L}_{q_i q_i}$) and residuals of the KKT conditions (e.g., $\mathcal{L}_{q_i}$) (Nocedal and Wright (2006); Wächter and Biegler (2006)). By denoting such modified Hessians as $\bar{\mathcal{L}}_{q_i q_i}$ and KKT residuals as $\bar{\mathcal{L}}_{q_i}$, we obtain

$$\begin{aligned} &\begin{bmatrix} \bar{\mathcal{L}}_{q_i}^{\mathrm{T}} \\ \bar{\mathcal{L}}_{v_i}^{\mathrm{T}} \end{bmatrix} + \begin{bmatrix} \bar{\mathcal{L}}_{q_i, y_i} \\ \bar{\mathcal{L}}_{v_i, y_i} \end{bmatrix} \Delta y_i + \begin{bmatrix} \delta_{q_i}^{\mathrm{T}}(q_i, q_{i+1}) & O_{n\times n} \\ I_n \Delta\tau & I_n \end{bmatrix} \begin{bmatrix} \Delta\lambda_{i+1} \\ \Delta\gamma_{i+1} \end{bmatrix} + \begin{bmatrix} \mathrm{ID}_{q_i}^{\mathrm{T}}(\tilde{y}_i)\Delta\tau \\ \mathrm{ID}_{v_i}^{\mathrm{T}}(\tilde{y}_i)\Delta\tau \end{bmatrix} \Delta\beta_i \\ &+ \begin{bmatrix} C_{q_i}^{\mathrm{T}}(y_i)\Delta\tau \\ C_{v_i}^{\mathrm{T}}(y_i)\Delta\tau \end{bmatrix} \Delta\mu_i + \begin{bmatrix} \delta_{q_i}^{\mathrm{T}}(q_{i-1}, q_i)\Delta\lambda_i \\ -\Delta\gamma_i \end{bmatrix} = 0, \end{aligned} \tag{3.14}$$

$$\begin{bmatrix} \bar{\mathcal{L}}_{a_i}^{\mathrm{T}} \\ \bar{\mathcal{L}}_{f_i}^{\mathrm{T}} \end{bmatrix} + \begin{bmatrix} \bar{\mathcal{L}}_{a_i y_i} \\ \bar{\mathcal{L}}_{f_i y_i} \end{bmatrix} \Delta y_i + \begin{bmatrix} O_{n\times 1} \\ \Delta\gamma_{i+1}\Delta\tau \end{bmatrix} + \begin{bmatrix} \mathrm{ID}_{a_i}^{\mathrm{T}}(\tilde{y}_i)\Delta\tau \\ \mathrm{ID}_{f_i}^{\mathrm{T}}(\tilde{y}_i)\Delta\tau \end{bmatrix} \Delta\beta_i + \begin{bmatrix} C_{a_i}^{\mathrm{T}}(y_i)\Delta\tau \\ C_{f_i}^{\mathrm{T}}(y_i)\Delta\tau \end{bmatrix} \Delta\mu_i = 0, \tag{3.15}$$

and

$$\bar{\mathcal{L}}_{u_i}^{\mathrm{T}} + \bar{\mathcal{L}}_{u_i y_i}\Delta y_i - \Delta\beta_i \Delta\tau + C_{u_i}^{\mathrm{T}}(y_i)\Delta\mu_i\Delta\tau = 0. \tag{3.16}$$

The constraints, (3.2)–(3.5) and (3.7), are linearized as

$$\delta(\bar{q}, q_0) + \delta_{q_0}(\bar{q}, q_0)\Delta q_0 = 0, \ \bar{v} - v_0 - \Delta v_0 = 0, \tag{3.17}$$

$$\begin{aligned} &\begin{bmatrix} \delta(q_i, q_{i+1}) + v_i\Delta\tau \\ v_i - v_{i+1} + a_i\Delta\tau \end{bmatrix} + \begin{bmatrix} \delta_{q_i}(q_i, q_{i+1}) & I_n\Delta\tau \\ O_{n\times n} & I_n \end{bmatrix} \begin{bmatrix} \Delta q_i \\ \Delta v_i \end{bmatrix} \\ &+ \begin{bmatrix} O_{n\times 1} \\ \Delta a_i \Delta\tau \end{bmatrix} + \begin{bmatrix} \delta_{q_{i+1}}(q_i, q_{i+1}) & O_{n\times n} \\ O_{n\times n} & -I_n \end{bmatrix} \begin{bmatrix} \Delta q_{i+1} \\ \Delta v_{i+1} \end{bmatrix} = 0, \end{aligned} \tag{3.18}$$

$$\mathrm{ID}_{\tilde{y}_i}(\tilde{y})\Delta\tilde{y}_i - \Delta u_i + \mathrm{ID}(\tilde{y}) - u = 0, \tag{3.19}$$

and

$$C_{\tilde{y}_i}(y_i)\Delta\tilde{y}_i + C_u(y_i)\Delta u_i + C(y_i) = 0. \tag{3.20}$$

Each Newton iteration consists of solving a linear equation that finds Newton directions satisfying (3.13)–(3.20).

3.3.2 Condensing inverse dynamics

Next, we condense the inverse dynamics; i.e., we eliminate Δu_i and $\Delta \beta_i$ from the linear equations (3.13)–(3.20). By substituting the expression of Δu_i and $\Delta \beta_i$ with respect to other Newton directions (3.19) and (3.16) into (3.14)–(3.16) and (3.20), we can obtain a condensed linear equation. For notational simplicity, we introduce the condensed Hessian,

$$\begin{aligned}\tilde{\mathcal{L}}_{z_i w_i} := \bar{\mathcal{L}}_{z_i w_i} + \mathrm{ID}_{z_i}^{\mathrm{T}}(\tilde{y}_i) \bar{\mathcal{L}}_{u_i u_i} \mathrm{ID}_{w_i}(\tilde{y}_i) \\ + \mathrm{ID}_{z_i}^{\mathrm{T}}(\tilde{y}) \bar{\mathcal{L}}_{u_i w_i} + \bar{\mathcal{L}}_{z_i u_i} \mathrm{ID}_{w_i}(\tilde{y})\end{aligned} \tag{3.21}$$

for $z_i, w_i \in \{q_i, v_i, a_i, f_i\}$ and the condensed KKT residual,

$$\tilde{\mathcal{L}}_{z_i}^{\mathrm{T}} := \bar{\mathcal{L}}_{z_i}^{\mathrm{T}} + \mathrm{ID}_{z_i}^{\mathrm{T}}(\tilde{y}) \bar{\mathcal{L}}_{u_i}^{\mathrm{T}} + (\bar{\mathcal{L}}_{z_i, u_i} + \mathrm{ID}_{z_i}^{\mathrm{T}}(\tilde{y}) \bar{\mathcal{L}}_{u_i u_i})(\mathrm{ID}(\tilde{y}) - u_i) \tag{3.22}$$

for $z_i \in \{q_i, v_i, a_i, f_i\}$. We also introduce the condensed Jacobian of the equality constraint,

$$\tilde{C}_{z_i} := C_{z_i} + C_{u_i} \mathrm{ID}_{z_i}(q_i, v_i, a_i, f_i), \tag{3.23}$$

for $z_i \in \{q_i, v_i, a_i, f_i\}$ and the condensed residual of the equality constraint,

$$\tilde{C} := C + C_{u_i}(\mathrm{ID}(q_i, v_i, a_i, f_i) - u_i). \tag{3.24}$$

Then, Δu_i and $\Delta \beta_i$ are eliminated from (3.14)–(3.16) and (3.20):

$$\begin{aligned}\begin{bmatrix} \tilde{\mathcal{L}}_{q_i}^{\mathrm{T}} \\ \tilde{\mathcal{L}}_{v_i}^{\mathrm{T}} \end{bmatrix} + \begin{bmatrix} \tilde{\mathcal{L}}_{q_i y_i} \\ \tilde{\mathcal{L}}_{v_i y_i} \end{bmatrix} \Delta y_i + \begin{bmatrix} \delta_{q_i}^{\mathrm{T}}(q_i, q_{i+1}) & O_{n \times n} \\ I_n \Delta \tau & I_n \end{bmatrix} \begin{bmatrix} \Delta \lambda_{i+1} \\ \Delta \gamma_{i+1} \end{bmatrix} \\ + \begin{bmatrix} C_{q_i}^{\mathrm{T}}(y_i) \\ C_{v_i}^{\mathrm{T}}(y_i) \end{bmatrix} \Delta \mu_i + \begin{bmatrix} \delta_{q_i}^{\mathrm{T}}(q_{i-1}, q_i) \Delta \lambda_i \\ -\Delta \gamma_i \end{bmatrix} = 0,\end{aligned} \tag{3.25}$$

$$\begin{bmatrix} \tilde{\mathcal{L}}_{a_i}^{\mathrm{T}} \\ \tilde{\mathcal{L}}_{f_i}^{\mathrm{T}} \end{bmatrix} + \begin{bmatrix} \tilde{\mathcal{L}}_{a_i y_i} \\ \tilde{\mathcal{L}}_{f_i y_i} \end{bmatrix} \Delta y_i + \begin{bmatrix} O_{n \times 1} \\ \Delta \gamma_{i+1} \Delta \tau \end{bmatrix} + \begin{bmatrix} C_{a_i}^{\mathrm{T}}(y_i) \\ C_{f_i}^{\mathrm{T}}(y_i) \end{bmatrix} \Delta \mu_i = 0, \tag{3.26}$$

and

$$\tilde{C}_{\tilde{y}_i}(y_i) \Delta \tilde{y}_i + \tilde{C}(y_i) = 0. \tag{3.27}$$

After condensing, the linear equation is reduced to find the Newton directions $\Delta \lambda_i, \Delta \gamma_i, \Delta q_i, \Delta v_i, \Delta a_i, \Delta f_i, \Delta \mu_i$ satisfying (3.17), (3.18), (3.13), (3.25)–(3.27).

If the rigid-body system is fully actuated and there are no contacts, the size of the condensed linear equation is the same as that of the multiple-shooting method based on forward dynamics. If it is underactuated and there are no contacts, the size of the condensed linear equation at each time stage is increased from that of the multiple-shooting method based on forward dynamics by only twice the number of the passive joints, since we assume the system is fully actuated and add an equality constraint to zero the control input torques corresponding to the passive joints. If there are contacts, the size of the linear equation also increases depending on the way to treat the contacts.

3.3.3 Algorithm

Algorithm 3.1 is the pseudocode of a single Newton iteration of the proposed method. The first step (lines 1–3) forms the linear equation consisting of (3.17), (3.18), (3.13), and (3.25)–(3.27) by computing the condensed KKT residuals and Hessians, such as $\tilde{\mathcal{L}}_{q_i}$, $\tilde{\mathcal{L}}_{q_i q_i}$, $\tilde{C}$, and $\tilde{C}_{q_i}$. This step is fully parallelizable into each time stage. The second step (line 4–6) computes the directions $\Delta\lambda_i$, $\Delta\gamma_i$, $\Delta\tilde{y}_i$, $\Delta\mu_i$ by solving the linear equation consisting of (3.17), (3.18), (3.13), and (3.25)–(3.27), i.e., inverting the condensed Hessian. In this step, we can utilize various efficient methods tailored for the OCP (D. Kouzoupis and Diehl (2018)), e.g., Riccati recursion (Frison (2016)) and a highly parallelizable method (Deng and Ohtsuka (2019)). The third step (lines 7–9) computes the condensed directions, i.e., Δu_i and $\Delta\beta_i$ from (3.19) and (3.16), and Δs_i and $\Delta\nu_i$ according to the PDIP method (Nocedal and Wright (2006); Wächter and Biegler (2006)). The fourth step (line 10) determines the step size, e.g., by using the fraction-to-boundary rule (Nocedal and Wright (2006); Wächter and Biegler (2006)). Finally, all variables are updated according to the step size (lines 11–16).

The main advantage of the proposed method appears in the first step (lines 1–3). It calls RNEA and the partial derivatives of RNEA $N-1$ times, whereas the forward dynamics-based method computes ABA (or inverse of the joint-inertia matrix) $N-1$ times and the partial derivatives of ABA $N-1$ times. Moreover, the proposed method can reduce the computational cost in the second step (lines 4–6), as our formulation leads to sparsity in the partial derivatives of the state equation. We will explain this point below with the Riccati recursion for computing the Newton directions. Riccati recursion (Frison (2016)) performs a recursive block elimination to compute the Newton directions with the forward Euler discretization (3.4). In Riccati recursion, we regard $[\Delta q_i^{\mathrm{T}} \quad \Delta v_i^{\mathrm{T}}]^{\mathrm{T}}$ as the state variable and $[\Delta a_i^{\mathrm{T}} \quad \Delta f_i^{\mathrm{T}}]^{\mathrm{T}}$ as the control input of a linear quadratic regulator subproblem. We also introduce matrices,

$$A_i := \begin{bmatrix} \delta_{q_i}(q_i, q_{i+1}) & I_n \Delta\tau \\ O_{n\times n} & I_n \end{bmatrix}, \quad B_i := \begin{bmatrix} O_{n\times n} & O_{n\times n_f} \\ \Delta\tau I_n & O_{n\times n_f} \end{bmatrix},$$

Algorithm 3.1 Single Newton iteration of primal-dual interior point method with condensing of inverse dynamics

```
Input: Initial state $\bar{q}, \bar{v}$
Output: $\lambda_0, ..., \lambda_N$, $\gamma_0, ..., \gamma_N$, $y_0, ..., y_{N-1}$, $q_N$, $v_N$, $\mu_0, ..., \mu_{N-1}$, $\beta_0, ..., \beta_{N-1}$, $s_0, ..., s_{N-1}$, $\nu_0, ..., \nu_{N-1}$
 1: for $i = 0, \cdots, N$ do in parallel
 2:     Compute the condensed Hessian and KKT residual.
 3: end for
 4: for $i = 0, \cdots, N$ do in parallel or serial
 5:     Compute the Newton directions $\Delta\lambda_i$, $\Delta\gamma_i$, $\Delta\tilde{y}_i$, $\Delta\mu_i$ by solving the linear equation (3.17), (3.18), (3.13), (3.25)–(3.27).
 6: end for
 7: for $i = 0, \cdots, N-1$ do in parallel
 8:     Compute the condensed Newton directions $\Delta u_i$ and $\Delta\beta_i$ from (3.19), (3.16), and $\Delta s_i$ and $\Delta\nu_i$ according to Wächter and Biegler (2006).
 9: end for
10: Determine the step size $\alpha \in (0, 1]$.
11: for $i = 0, \cdots, N$ do in parallel
12:     Update all variables $\lambda_i$, $\gamma_i$, $y_i$, $\mu_i$ $\beta_i$, $s_i$, $\nu_i$ by
13:     $\lambda_i \leftarrow \lambda_i + \alpha\Delta\lambda_i$, $\gamma_i \leftarrow \gamma_i + \alpha\Delta\gamma_i$, $y_i \leftarrow y_i + \alpha\Delta y_i$,
14:     $\mu_i \leftarrow \mu_i + \alpha\Delta\mu_i$, $\beta_i \leftarrow \beta_i + \alpha\Delta\beta_i$,
15:     $s_i \leftarrow s_i + \alpha\Delta s_i$, $\nu_i \leftarrow \nu_i + \alpha\Delta\nu_i$
16: end for
```

where A_i is composed of the partial derivatives of the state equation with respect to q and v, and B_i is composed of the partial derivatives with respect to a and f. Riccati recursion serially computes $A_i^{\mathrm{T}} P_{i+1} A_i$, $A_i^{\mathrm{T}} P_{i+1} B_i$, $B_i^{\mathrm{T}} P_{i+1} B_i$ for a sequence of matrices P_i that represent the sensitivities of $\Delta\lambda_i$ and $\Delta\gamma_i$ with respect to Δq_i and Δv_i. Thanks to the sparsity of A_i and B_i in the proposed formulation, several matrix products reduce to sums of matrices, and this in turn reduces the computational cost, especially the cost of the serial computational part.

We have developed an open-source C++ software framework for the OCP for rigid-body systems, `robotoc` (Katayama (2020-2022)), that utilizes the above Riccati recursion and ParNMPC, a highly parallelizable method (Deng and Ohtsuka (2019)). `robotoc` uses Eigen (Guennebaud, Jacob, et al. (2010)) for linear algebra and `Pinocchio` (Carpentier et al. (2019)), an efficient C++ library for rigid-body dynamics algorithms, to compute RNEA and its partial derivatives. It also employs the parallel Newton-type method (Deng and Ohtsuka (2019)) with backward Euler discretization method.

3.4 Numerical Experiments

3.4.1 Experimental settings

To evaluate the computational efficiency and numerical robustness of the proposed solution method of the OCP, we compared our solver `robotoc` with `Crocoddyl` (Mastalli et al. (2020)), a highly efficient C++ implementation of DDP/iLQR for rigid-body systems, and Ipopt (Wächter and Biegler (2006)), an off-the-shelf non-linear optimization solver. Both `robotoc` and `Crocoddyl` utilize `Pinocchio` for rigid-body dynamics algorithms, Eigen for linear algebra, and OpenMP (Dagum and Menon (1998)) for parallel computing. Ipopt was customized to solve the OCP for rigid-body systems based on forward dynamics and the multiple-shooting method with parallel computing by using `Pinocchio` and OpenMP. As the algorithm of `robotoc`, we used Riccati recursion to compute the Newton directions (we will refer to this method as inverse dynamics-based Riccati recursion (IDRR) hereafter). Furthermore, we utilized two algorithms from `Crocoddyl`: the standard DDP with a Gauss-Newton Hessian approximation, which is identical to iLQR, and feasible-prone DDP (FDDP) (Mastalli et al. (2020)), a kind of iLQR that improves numerical robustness by modifying the backward pass of iLQR in a multiple-shooting fashion (Giftthaler, Neunert, Stäuble, Buchli, and Diehl (2018)) and utilizes a line-search method based on the Goldstein condition (Nocedal and Wright (2006)). We will refer to the former as iLQR and the latter as FDDP. In Ipopt, we used the Broyden–Fletcher–Goldfarb–Shanno method for the Hessian approximation and Harwell Subroutine Library MA57 to solve the linear subproblems. All experiments were conducted on a laptop with quad-core CPU Intel Core i7-10510U @1.8GHz.

3.4.2 Computational time

First, we compared the computational time per iteration of IDRR, iLQR/FDDP, and Ipopt. In particular, we computed the average computational time over 10,000 trials for systems with various degrees of freedom (DOF), various numbers of time stages N, and various numbers of threads (nproc) with the quadratic terminal cost,

$$V(q, v) = \frac{1}{2} q_e^{\mathrm{T}} Q_q q_e + \frac{1}{2} v_e^{\mathrm{T}} Q_v v_e, \tag{3.28}$$

where $Q_q = I_n$, $Q_v = I_n$, $q_e := q - q_{\mathrm{ref}}$, and $v_e := v - v_{\mathrm{ref}}$ with $q_{\mathrm{ref}}, v_{\mathrm{ref}} \in \mathbb{R}^n$, and the quadratic stage cost,

$$l(q, v, a, u) = \frac{1}{2} q_e^{\mathrm{T}} Q_q q_e + \frac{1}{2} v_e^{\mathrm{T}} Q_v v_e + \frac{1}{2} u_e^{\mathrm{T}} Q_u u_e, \tag{3.29}$$

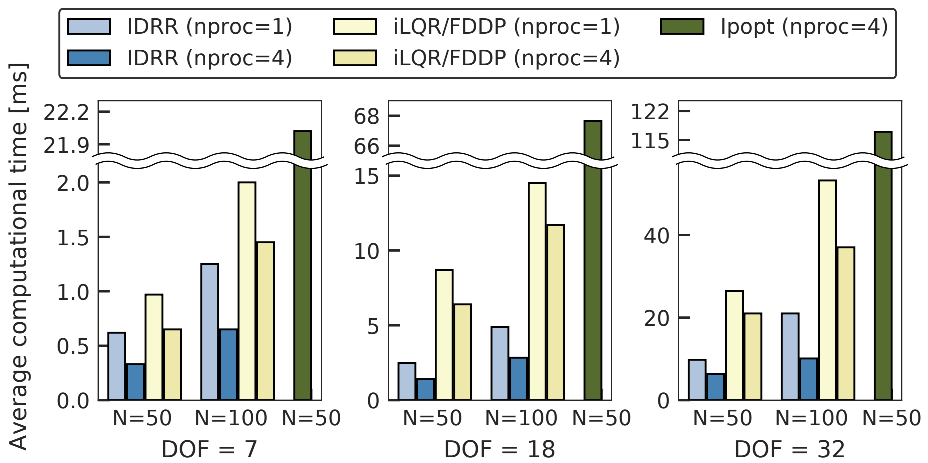


Figure 3.1: Average computational time per iteration [ms] of the proposed method (IDRR), iLQR/FDDP, and Ipopt, for various degrees of freedom (DOF $\in \{7, 14, 32\}$), numbers of time steps ($N \in \{50, 100\}$), and numbers of threads (nproc $\in \{1, 4\}$)

where $Q_u = 0.001 \times I_n$ and $u_e := u - u_{\rm ref}$ with $u_{\rm ref} \in \mathbb{R}^n$. Figure 3.1 shows the computational time per iteration [ms] of each solver. Note that the results of iLQR and FDDP are shown as one because they had almost the same computational time. Moreover, the results of Ipopt are only for the fastest case, i.e., the case with $N = 50$ and nproc $= 4$, because Ipopt was much slower than the other solvers. As shown in the figure, for all combinations of N and number of threads (nproc), our IDRR was faster than iLQR/FDDP and Ipopt. This demonstrates that the inverse dynamics-based formulation can reduce the computational cost compared with the forward dynamics-based formulation. We also found that IDRR became faster as the number of the threads increased, whereas iLQR/FDDP did so only moderately. This is because the proposed method reduces the computational burden of the serial calculation in the Riccati recursion, thanks to its multiple-shooting and sparsity structure, as stated in 3.3.3.

3.4.3 Numerical robustness

Second, we investigated the numerical robustness, i.e., the convergence, of the proposed method through the OCP for KUKA iiwa14, a 7-DOF manipulator. We set the length of the horizon to 1 s and divided it into $N = 50$ steps, i.e., $\Delta\tau = 0.02$[s]. The objective of the OCP is to make the configuration q converge to $q_{\rm ref}$ and the velocity v converge to zero given a random initial state $\bar{q}$ and $\bar{v}$. We used the quadratic terminal cost (3.28) and stage cost (3.29) and set $q_{\rm ref}$ to $[0 \;\; \pi/2 \;\; 0 \;\; \pi/2 \;\; 0 \;\; \pi/2 \;\; 0]^{\rm T}$,

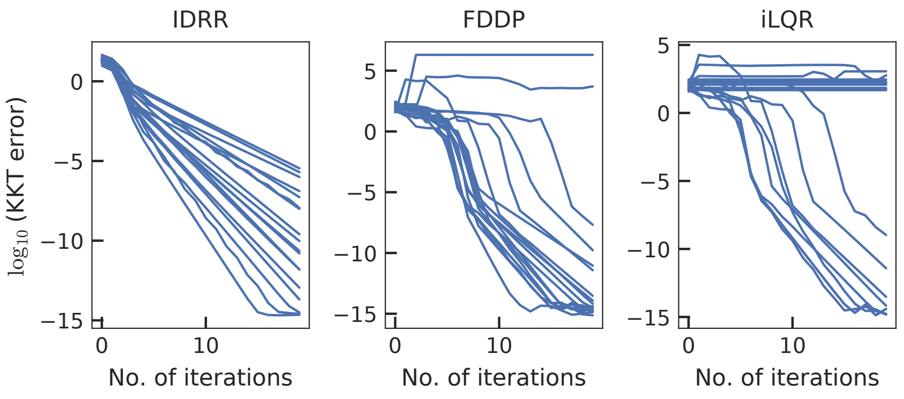


Figure 3.2: $\log_{10}$ scaled KKT errors of the proposed method (IDRR), FDDP, and iLQR over 20 random trials.

v_{ref} to zero, and u_{ref} to the gravity compensation torques at q_{ref}. We did not impose any inequality constraints in this experiment. iLQR and FDDP used a line search and did not use regularization. IDRR did not use either, while Ipopt used both of them. We initialized q_i and v_i in the solution of IDRR, FDDP, and Ipopt by $\bar{q}$ and $\bar{v}$. We randomly selected each element of $\bar{q}$ from $[-1, 1]$ and each element of $\bar{v}$ from $[-10, 10]$. We ran 20 trials for each solver and computed the l^2-norm of the residual of the KKT conditions, which we will refer to as the KKT error hereafter, and the total cost to be minimized. Note that the KKT conditions of IDRR are given by (3.2)–(3.5), (3.7), and (3.8)–(3.12), while those of iLQR/FDDP and Ipopt are composed of different equations based on forward dynamics and/or single-shooting. Figure 3.2 shows the $\log_{10}$-scaled KKT errors of IDRR, FDDP, and iLQR. Results for Ipopt are not shown because it converged rather slowly (over 1000 iterations were needed to reduce the KKT error to under 10^{-10} in most cases) due to the quasi-Newton Hessian approximation. Note as well that, as the equations representing the KKT conditions differ between IDRR and FDDP/iLQR, we cannot directly compare their KKT errors. However, from the graph of FDDP, we can clearly see that FDDP diverged in two trials. In addition, from the graph of iLQR, we can see that in many trials the KKT error could not be completely reduced because the iLQR did not produce a descent direction and the step size became zero. On the other hand, IDRR reduced the KKT error at each iteration in all cases, including those in which FDDP and iLQR failed to converge. This indicates that the proposed method is more robust than single-shooting methods such as iLQR and FDDP.

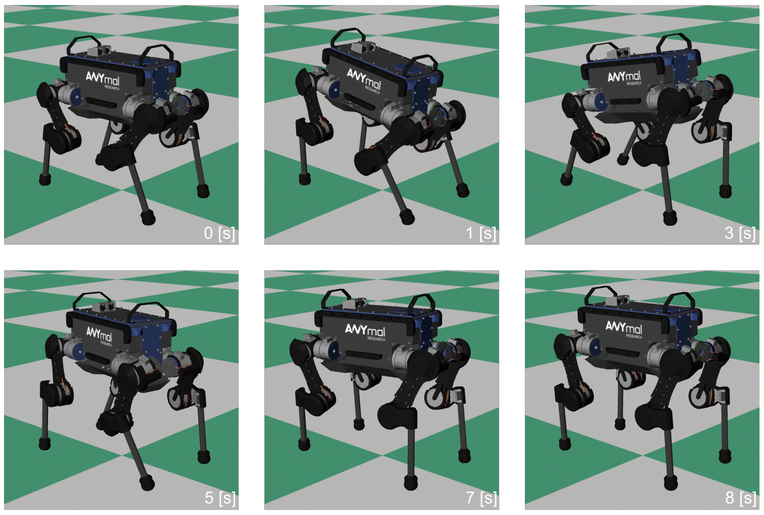


Figure 3.3: Posture control of ANYmal by whole-body MPC controller.

3.4.4 MPC for floating base systems

Finally, we investigated the applicability of the proposed method to floating base systems, which are a kind of underactuated system, by simulating MPC implemented by IDRR for the whole-body control of a quadruped ANYmal. To take the rigid contacts into account, we treated the contact forces as the optimization variables and imposed an equality constraint in the form of Baumgarte's stabilization method (Baumgarte (1972)) for each contact, i.e., a 12-dimensional equality constraint for four contacts. We used a quadratic cost function to track the desired configuration. We also imposed inequality constraints on the joint angle limits, angular velocity limits, and torque limits and linearized friction cones using the PDIP method. We set the length of the horizon to 1 s and divided it into $N = 20$ equal steps. We set the sampling time to 2.5 ms, and the MPC controller updated the solution once per sampling period. We utilized `RaiSim` (Hwangbo, Lee, and Hutter (2018)), an articulated-body simulator with contacts. Figure 3.3 shows the various postures of ANYmal controlled by MPC. As can be seen, the proposed method was able to control the underactuated system. Each control update took around 2.1 ms with four threads on the same laptop as in 3.4.2 and achieved real-time MPC.

3.5 Summary

We proposed an efficient method of solving the OCP for rigid-body systems on the basis of inverse dynamics and the multiple-shooting method. In this method, we regard all variables, including the state, acceleration, and control input torques, as optimization variables and treat the inverse dynamics as an equality constraint. We eliminated the update in the control input torques from the linear equation of Newton's method by applying condensing for inverse dynamics. The size of the resultant linear equation was the same as that of the multiple-shooting method based on forward dynamics except for the variables related to the passive joints and contacts. The proposed method reduces the computational cost of the dynamics and their sensitivities by utilizing RNEA and its partial derivatives. In addition, it increases the sparsity of the Hessian of the KKT conditions, which further reduces the computational cost, e.g., of Riccati recursion. Numerical experiments showed that the proposed method is more than twice as fast as iLQR and a DDP variant based on forward dynamics. They also showed that it is more numerically robust than these conventional methods.

Chapter 4

Lifted Contact Dynamics for Efficient Optimal Control of Rigid Body Systems with Contacts[1]

4.1 Introduction

Trajectory optimization (TO) and model predictive control (MPC) are promising frameworks for motion planning and control of rigid-body systems that make rigid contacts with the environment, e.g., legged robots and robot manipulators, whose dynamics are highly nonlinear and include discontinuities. The computational time required to solve optimal control problems (OCPs) remains a critical constraint for using these two schemes, particularly MPC. The solution methods of such OCPs are generally classified into two types: contact-implicit and event-driven approaches.

Contact-implicit approaches aim to determine the optimal trajectory without specifying the contact sequence in advance. The simplest approach is to approximate the contact forces using spring-damper systems (Neunert et al. (2018)); however, it lacks accuracy and involves stiff optimization problems. The same problem typically occurs in other smooth soft-contact models such as in (Chatzinikolaidis et al. (2020)). More accurate approaches are based on the mathematical programs with complementarity constraints (MPCCs) (Posa et al. (2014)) or bilevel optimization (BO) problems (Carius et al. (2018)). However, the MPCCs inherently lack linear independence constraint qualification, which causes practical and theoretical convergence issues (Nurkanović et al. (2020)). Moreover, the convergence analysis of BO is

[1]© 2022 IEEE. A substantial portion of this chapter including Figures 4.2, 4.3, and 4.5 and Table 4.1 is reprinted, with permission, from S. Katayama and T. Ohtsuka, "Lifted contact dynamics for efficient optimal control of rigid body systems with contacts," 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2022), 2022.

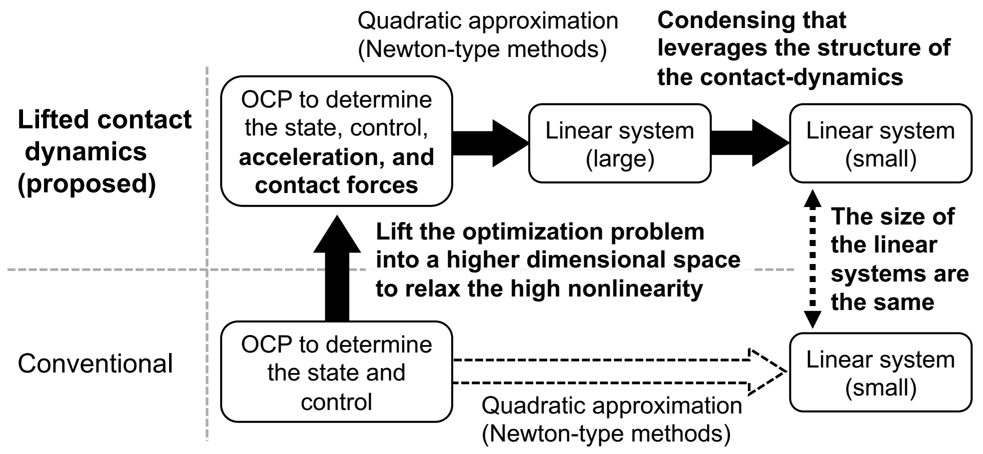

Quadratic approximation (Newton-type methods)
Condensing that leverages the structure of the contact-dynamics
Lifted contact dynamics (proposed)
OCP to determine the state, control, acceleration, and contact forces
Linear system (large)
Linear system (small)
Lift the optimization problem into a higher dimensional space to relax the high nonlinearity
The size of the linear systems are the same
Conventional
OCP to determine the state and control
Linear system (small)
Quadratic approximation (Newton-type methods)

Figure 4.1: Conceptual diagram of the proposed lifted contact dynamics scheme for efficient optimal control of rigid body systems with contacts. The conventional optimal control problem (OCP) is lifted into a higher dimensional one to relax the high nonlinearity. The Newton-type method is carried out efficiently by leveraging the structure of the contact dynamics.

yet to be discussed (Carius et al. (2018)), which renders the practical applications of BO difficult.

In contrast, event-driven approaches formulate the OCPs under predefined contact sequences and model the impacts between the system and the environment explicitly via Newton's law of impact. Therefore, event-driven approaches are more accurate and the continuous-time OCP can be discretized with coarser grids than in contact-implicit methods, which means that they can be faster. Moreover, they consist of only smooth optimization problems; therefore, we can solve them efficiently using the off-the-shelf Newton-type methods for smooth OCPs without being overly cautious with the LICQ. Although event-driven approaches cannot optimize contact sequences as contact-implicit methods can, they can optimize the instants of the impacts under a given contact sequence, as studied in Katayama, Doi, and Ohtsuka (2020); Li and Wensing (2020).

The contact-consistent forward dynamics, a calculation of the acceleration and contact forces using the given configuration, velocity, and torques, which we call *contact dynamics* in this chapter, has been successfully used with Newton-type methods to efficiently solve event-driven OCPs (Budhiraja, Carpentier, Mastalli, and Mansard (2018); Li and Wensing (2020); Mastalli et al. (2020); Schultz and Mombaur (2010)). The contact dynamics formulation is utilized originally in Schultz and Mombaur (2010) with the direct multiple shooting method (DMS) (Bock and Plitt (1984)). Budhiraja et al. (2018) and Li and Wensing (2020) combined contact dynamics with the iterative linear quadratic regulator (iLQR) (Todorov and Li (2005)). Mastalli et al. (2020) improved the approach of Budhiraja et al. (2018) using the efficient analytical derivatives proposed in Carpentier and Mansard (2018) to compute the sensitivities of contact dynamics. However, such an approach still contains high nonlinearity, which results in slow convergence, particularly when there are costs and constraints on the contact forces, e.g., friction cone constraints.

The lifted Newton method is a promising option to relax such high nonlinearity. It involves adding intermediate variables and lifting the optimization problem into a higher-dimensional one (Albersmeyer and Diehl (2010)). For example, it is well known in the context of numerical optimal control and MPC that the multiple shooting method, which considers the state and control input as the optimization variables, has preferred convergence properties over the single shooting method, which considers only the control input as the optimization variable (Bock and Plitt (1984)). However, to the best of our knowledge, no lifting method has previously been used to efficiently treat the contact forces in event-driven OCPs. For example, in our previous study of Chapter 3 (and Katayama and Ohtsuka (2021)), we utilized a certain lifted formulation; however, it may be inefficient when the number of contacts is significant. It is efficient only for systems without contacts or with a few contacts by leveraging inverse dynamics computation. Moreover, since the focus of the previous study of Chapter 3 was on the computational time of rigid body systems without contacts, no discussion or numerical investigation regarding convergence properties under contacts were conducted.

In this chapter, we propose *lifted contact dynamics*, a novel and efficient lifting approach for the optimal control of rigid-body systems with contacts to improve the convergence properties of Newton-type methods. The proposed lifted contact dynamics scheme is illustrated in Fig. 4.1. To relax the high nonlinearity, we consider all variables, including the state, acceleration, contact forces, and control input torques, as the optimization variables and the inverse dynamics and acceleration-level contact constraints as equality constraints. We eliminate the updating of the acceleration, contact forces, and their dual variables from the linear equation to be solved in each Newton-type iteration in an efficient manner. As a result, the computational cost per Newton-type iteration is almost identical to that of the conventional non-lifted Newton-type iteration that embeds contact dynamics in the state equation. This aspect distinguishes this study from our previous inverse dynamics-based algorithm proposed in Chapter 3, which was inefficient for high-dimensional contact constraints.

We also propose a similar lifting method for impulse dynamics, which was not considered in our previous study in Chapter Katayama and Ohtsuka (2021). We conducted numerical experiments on the whole-body optimal control of various quadrupedal gaits subject to the friction cone constraints considered in the interior-point methods. It demonstrated that the proposed method can significantly increase the convergence speed to more than twice that of conventional approaches based on the non-lifted contact dynamics. These investigations on convergence properties under the contacts are included in the contributions of this chapter because it was not discussed in the past literature Katayama and Ohtsuka (2021). The contributions of this paper are then summarized as follows:

- A lifted formulation for OCP of robotic systems with contacts to relax the high nonlinearity.
- Efficient "condensing" algorithms to enable as fast as Newton step computation as the non-lifted counterpart.
- Numerical studies on the practical quadrupedal locomotion problems.

The remainder of this chapter is organized as follows. In Section 4.2, we review the contact and impulse dynamics and conventional formulations of event-driven OCPs. In Section 4.3, we introduce the lifted contact dynamics, a Newton-type method that efficiently condenses the linear equations to be solved in each Newton iteration. Section 4.4 compares the proposed method with existing approaches based on the non-lifted contact dynamics and demonstrates its effectiveness in terms of the convergence properties. Section 4.5 concludes the chapter and outlines future research directions.

Notation: We denote the partial derivatives of a differentiable function with certain variables using a function with subscripts; i.e., $f_x(x)$ denotes $\frac{\partial f}{\partial x}(x)$ and $g_{xy}(x, y)$ denotes $\frac{\partial^2 g}{\partial x \partial y}(x, y)$. We denote an $n \times n$ identity matrix as I_n.

4.2 Overview of Optimal Control Problems of Rigid-Body Systems with Contacts

4.2.1 Contact dynamics

First, we review the components to formulate event-driven direct OCPs, e.g., the contact dynamics and discretized state equations. Let Q be the configuration manifold of the rigid-body system. Let $q \in Q$, $v \in \mathbb{R}^n$, $a \in \mathbb{R}^n$, $f \in \mathbb{R}^{n_f}$, and $u \in \mathbb{R}^{n_a}$ be the configuration, generalized velocity, acceleration, stack of the contact forces, and torques of the actuated joints, respectively. The equation of motion of the rigid-body system is expressed as

$$M(q)a + h(q, v) - J^{\mathrm{T}}(q)f = S^{\mathrm{T}}u, \tag{4.1}$$

where $M(q) \in \mathbb{R}^{n \times n}$ denotes the inertia matrix, $h(q, v) \in \mathbb{R}^n$ encompasses the Coriolis, centrifugal, and gravitational terms, $J(q) \in \mathbb{R}^{n_f \times n}$ denotes the stack of the contact Jacobians, and $S \in \mathbb{R}^{n_a \times n}$ denotes the selection matrix. The evolution of the state $\begin{bmatrix} q^{\mathrm{T}} & v^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}}$ with time step $\Delta\tau > 0$ is expressed as

$$\begin{bmatrix} q^+ \\ v^+ \end{bmatrix} = \begin{bmatrix} q \oplus v\Delta\tau \\ v + a\Delta\tau \end{bmatrix}, \tag{4.2}$$

where $\oplus$ denotes the increment operator on the configuration manifold Q (Solà et al. (2020)). We formulate the OCPs based on the DMS and then consider an equivalent equality constraint:

$$\begin{bmatrix} \delta(q, q^+) + v\Delta\tau \\ v - v^+ + a\Delta\tau \end{bmatrix} = 0, \tag{4.3}$$

where $\delta(q_1, q_2) := q_1 \ominus q_2 \in \mathbb{R}^n$, and $\ominus$ denotes the subtraction operator between the two configurations (Solà et al. (2020)). The system also satisfies the contact constraints of the form

$$\mathbf{p}(q) = 0, \tag{4.4}$$

where $\mathbf{p}(q) \in \mathbb{R}^{n_f}$ is the stack of the positions of the contact frames. In event-driven OCPs, instead of considering (4.4) over a time interval, we typically consider the acceleration-level constraint over the time interval that is generalized into the form of Baumgarte's stabilization method (Baumgarte (1972)):

$$\mathbf{a}(q, v, a) := \ddot{\mathbf{p}} + 2\alpha\dot{\mathbf{p}} + \beta^2\mathbf{p} = J(q)a + \mathbf{b}(q, v), \tag{4.5}$$

where α and β are weight parameters, and we define

$$\mathbf{b}(q, v) := \dot{J}(q, v)v + 2\alpha J(q)v + \beta^2\mathbf{p}(q). \tag{4.6}$$

By setting $\alpha = \beta > 0$, we can stabilize the violation of the original constraint (4.4) over the time interval provided that (4.4) and the equality constraint on the contact velocity,

$$\dot{\mathbf{p}}(q, v) = J(q)v = 0, \tag{4.7}$$

are satisfied at a point on the interval (Flores et al. (2011)). Note that (4.5) is reduced to the equality constraint on the acceleration of the contact frames with

$\alpha = \beta = 0$. By combining (4.1) and (4.5), we obtain the contact dynamics, i.e., the contact-consistent forward dynamics:

$$\begin{bmatrix} M(q) & J^{\mathrm{T}}(q) \\ J(q) & O \end{bmatrix} \begin{bmatrix} a \\ -f \end{bmatrix} = \begin{bmatrix} S^{\mathrm{T}} u - h(q,v) \\ -\mathbf{b}(q,v) \end{bmatrix}. \tag{4.8}$$

In the conventional approaches (Budhiraja et al. (2018); Li and Wensing (2020); Mastalli et al. (2020); Schultz and Mombaur (2010)), we eliminate a and f as functions of q, v, and u from the OCP using (4.8). For example, we substitute a with (4.8) in (4.2) or (4.3) and consider these equations to be the discretized state equation. We can also eliminate a and f from the cost function and the constraints, e.g., the friction cone constraints, using (4.8).

4.2.2 Impulse dynamics

Similar to the contact dynamics, the equation of Newton's law of impact of the systems is expressed as

$$M(q)\delta v - J^{\mathrm{T}}(q)\Lambda = 0, \tag{4.9}$$

where $\delta v \in \mathbb{R}^n$ denotes the impulse change in the generalized velocity, and $\Lambda \in \mathbb{R}^{n_f}$ denotes the stack of the impact forces. The evolution of the state between the impulse and its equivalent equality constraint considered in the DMS are expressed as

$$\begin{bmatrix} q^+ \\ v^+ \end{bmatrix} = \begin{bmatrix} q \\ v + \delta v \end{bmatrix} \tag{4.10}$$

and

$$\begin{bmatrix} \delta(q, q^+) \\ v - v^+ + \delta v \end{bmatrix} = 0, \tag{4.11}$$

respectively. Herein, we assume a completely inelastic collision, which results in the contact velocity constraints of the form (4.7) immediately after the impulse as

$$\mathbf{v}(q, v, \delta v) := \dot{\mathbf{p}}(q, v + \delta v) = J(q)(v + \delta v) = 0. \tag{4.12}$$

The contact position constraint (4.4) for the frames with impacts is also imposed at the impulse instant. By combining (4.9) and (4.12), we obtain the impulse dynamics:

$$\begin{bmatrix} M(q) & J^{\mathrm{T}}(q) \\ J(q) & O \end{bmatrix} \begin{bmatrix} \delta v \\ -\Lambda \end{bmatrix} = \begin{bmatrix} 0 \\ -J(q)v \end{bmatrix}. \tag{4.13}$$

In the conventional formulation, as well as the contact dynamics, δv and Λ are eliminated from the OCP as functions of q and v using (4.13), for example, from (4.10) and (4.11).

4.2.3 Conventional formulation of optimal control

We summarize the conventional formulation of the optimal control of rigid-body systems (Budhiraja et al. (2018); Li and Wensing (2020); Mastalli et al. (2020); Schultz and Mombaur (2010)). We introduce N discretization grids. We define $\mathcal{J} \subset \{0, ..., N-1\}$ which is the set of the impulse stages and define $\mathcal{I} := \{0, ..., N-1\} \setminus \mathcal{J}$. For given user-defined terminal cost $V_f(\cdot)$ and stage costs $l_i(\cdot)$, the conventional OCP is summarized as follows: determine $q_0, ..., q_N \in Q$, $v_0, ..., v_N \in \mathbb{R}^n$, and $\{u_i\}_{i \in \mathcal{I}} \in \mathbb{R}^m$ that minimize a given cost function

$$J := V(x_N) + \sum_{i \in \mathcal{I}} l(x_i, u_i, a_i, f_i) + \sum_{j \in \mathcal{J}} l(x_j, \delta v_i, \Lambda_i), \tag{4.14}$$

where $x_i := \begin{bmatrix} q_i^{\mathrm{T}} & v_i^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}}$ is the state, subject to the state equation obtained by eliminating a_i from (4.2) using (4.8), the state equation at the impulse obtained by eliminating δv_j from (4.10) using (4.13), the contact-position constraint (4.4) at the impulse instant, and the other user-defined equality and inequality constraints. In addition, a_i, f_i, δv_j, and Λ_j are also eliminated as functions of q_i, v_i, and u_i from the cost function (4.14) and the constraints other than the state equations, which increases the nonlinearity. In Schultz and Mombaur (2010), this problem was solved using the DMS, that is, $x_0, ..., x_N$ and $\{u_i\}_{i \in \mathcal{I}}$ were considered as the optimization variables, the state equations were considered as the equality constraints, and (4.14) was represented by $x_0, ..., x_N$ and $\{u_i\}_{i \in \mathcal{I}}$. In Budhiraja et al. (2018); Li and Wensing (2020); Mastalli et al. (2020), this problem was solved using the single shooting method, that is, $x_0, ..., x_N$ were further eliminated from the optimization problem using the state equations, and only $\{u_i\}_{i \in \mathcal{I}}$ were considered as the decision variables, e.g., (4.14) was represented only by $\{u_i\}_{i \in \mathcal{I}}$.

4.3 Lifted Contact Dynamics in Optimal Control

4.3.1 Lifted contact dynamics

We herein present the lifted contact dynamics to relax the high nonlinearity in OCPs, of which conceptual diagram is shown in Fig. 4.1. Let $i \in \mathcal{I}$, $y_i := (q_i, v_i, a_i, f_i)$, and $z_i := (q_i, v_i, a_i)$. We first augment the control input to convert the system into a fully actuated system in the numerical optimization. Without loss of generality, we assume that S is given by $\begin{bmatrix} O & I \end{bmatrix}^{\mathrm{T}}$, and we define $\tilde{u}_i := \begin{bmatrix} u_i^0 \\ u_i \end{bmatrix}$, where $u_i^0 \in \mathbb{R}^{n-n_a}$

denotes the virtual torques on the passive joints. Thus, we can express (4.1) as an equality constraint of the inverse dynamics as follows:

$$\mathrm{ID}(y_i) - \tilde{u}_i = 0, \tag{4.15a}$$

where $\mathrm{ID}(y_i)$ is defined by the left-hand side of (4.1) and can be computed efficiently using the recursive Newton–Euler algorithm (RNEA) (Featherstone (2008)), a fast algorithm for inverse dynamics, with an additional equality constraint

$$u_i^0 = \bar{S}^{\mathrm{T}} \tilde{u}_i = 0, \quad \bar{S} := \begin{bmatrix} I & O \end{bmatrix}^{\mathrm{T}} \in \mathbb{R}^{n \times n - n_a}. \tag{4.15b}$$

We consider a_i and f_i as the optimization variables and (4.1), (4.5), and (4.15b) as the equality constraints. The first-order derivatives of the Lagrangian $\mathcal{L}$ (defined by augmenting the constraints to the cost function (4.14)) with respect to the variables at the stage i, that is, the part of the Karush–Kuhn–Tucker (KKT) conditions associated with the stage i, are given by (4.3), (4.15a), (4.5), and (4.15b),

$$\begin{aligned}\begin{bmatrix} \mathcal{L}_{q_i}^{\mathrm{T}} \\ \mathcal{L}_{v_i}^{\mathrm{T}} \end{bmatrix} &= \begin{bmatrix} l_{q_i}^{\mathrm{T}} \\ l_{v_i}^{\mathrm{T}} \end{bmatrix} + \begin{bmatrix} \delta_{q_i}^{\mathrm{T}}(q_i, q_{i+1}) & O \\ I_n \Delta\tau & I_n \end{bmatrix} \begin{bmatrix} \lambda_{i+1} \\ \gamma_{i+1} \end{bmatrix} \\ &+ \begin{bmatrix} \mathrm{ID}_{q_i}^{\mathrm{T}}(y_i) & \mathbf{a}_{q_i}^{\mathrm{T}}(z_i) \\ \mathrm{ID}_{v_i}^{\mathrm{T}}(y_i) & \mathbf{a}_{v_i}^{\mathrm{T}}(z_i) \end{bmatrix} \begin{bmatrix} \beta_i \\ \mu_i \end{bmatrix} \Delta\tau + \begin{bmatrix} \delta_{q_i}^{\mathrm{T}}(q_{i-1}, q_i)\lambda_i \\ -\gamma_i \end{bmatrix} = 0, \end{aligned} \tag{4.16a}$$

$$\begin{bmatrix} \mathcal{L}_{a_i}^{\mathrm{T}} \\ -\mathcal{L}_{f_i}^{\mathrm{T}} \end{bmatrix} = \begin{bmatrix} l_{a_i}^{\mathrm{T}} \\ -l_{f_i}^{\mathrm{T}} \end{bmatrix} + \begin{bmatrix} \gamma_{i+1}\Delta\tau \\ 0 \end{bmatrix} + \begin{bmatrix} \mathrm{ID}_{a_i}^{\mathrm{T}}(y_i) & \mathbf{a}_{a_i}^{\mathrm{T}}(z_i) \\ -\mathrm{ID}_{f_i}^{\mathrm{T}}(y_i) & -\mathbf{a}_{f_i}^{\mathrm{T}}(z_i) \end{bmatrix} \begin{bmatrix} \beta_i \\ \mu_i \end{bmatrix} \Delta\tau = 0, \tag{4.16b}$$

and

$$\mathcal{L}_{\tilde{u}_i}^{\mathrm{T}} := l_{\tilde{u}_i}^{\mathrm{T}} - \beta_i \Delta\tau + \bar{S}\nu_i \Delta\tau = 0, \tag{4.16c}$$

where $\lambda_{i+1}, \gamma_{i+1}, \beta_i \in \mathbb{R}^n$, $\mu_i \in \mathbb{R}^{n_f}$, and $\nu \in \mathbb{R}^{n-m}$ are the Lagrange multipliers with respect to (4.3), (4.15a), (4.5), and (4.15b), respectively.

In the Newton-type methods, these KKT conditions are linearized into linear equations with respect to the Newton directions Δq_i, Δv_i, Δa_i, Δf_i, Δu_i, Δu_i^0, $\Delta\lambda_{i+1}$, $\Delta\gamma_{i+1}$, $\Delta\beta_i$, $\Delta\mu_i$, and $\Delta\nu_i$. However, this problem is significantly larger than the conventional OCPs based on the non-lifted contact dynamics that considers only Δq_i, Δv_i, Δu_i, $\Delta\lambda_{i+1}$, and $\Delta\gamma_{i+1}$. Thus, we propose an efficient condensing method to reduce the size of the linear equation. We first observe that the equality constraints (4.15a) and (4.5) are linearized as

$$\begin{bmatrix} M(q_i) & J^{\mathrm{T}}(q_i) \\ J(q_i) & O \end{bmatrix} \begin{bmatrix} \Delta a_i \\ -\Delta f_i \end{bmatrix} = - \begin{bmatrix} \mathrm{ID}_{q_i}(y_i) & \mathrm{ID}_{v_i}(y_i) \\ \mathbf{a}_{q_i}(z_i) & \mathbf{a}_{v_i}(z_i) \end{bmatrix} \begin{bmatrix} \Delta q_i \\ \Delta v_i \end{bmatrix} + \begin{bmatrix} \Delta\tilde{u}_i \\ 0 \end{bmatrix} - \begin{bmatrix} \mathrm{ID}(y_i) \\ \mathbf{a}(z_i) \end{bmatrix}. \tag{4.17}$$

Furthermore, we always set $u_i^0 = 0$ and obtain from (4.15b)

$$\Delta u_i^0 = -u_i^0 = 0. \tag{4.18}$$

Therefore, we can express Δa_i and Δf_i using the linear combinations of Δq_i, Δv_i, and Δu_i using (4.17) and (4.18) if we compute

$$\begin{bmatrix} M(q_i) & J^{\mathrm{T}}(q_i) \\ J(q_i) & O \end{bmatrix}^{-1}. \tag{4.19}$$

Next, we observe that (4.16b) and (4.16c) are linearized into

$$\begin{bmatrix} \mathcal{L}_{a_i}^{\mathrm{T}} \\ -\mathcal{L}_{f_i}^{\mathrm{T}} \end{bmatrix} + \begin{bmatrix} \mathcal{L}_{a_i y_i} \\ -\mathcal{L}_{f_i y_i} \end{bmatrix} \Delta y_i + \begin{bmatrix} \Delta \gamma_{i+1} \Delta \tau \\ O \end{bmatrix} + \begin{bmatrix} M(q_i) & J^{\mathrm{T}}(q_i) \\ J(q_i) & O \end{bmatrix} \begin{bmatrix} \Delta \beta_i \\ \Delta \mu_i \end{bmatrix} \Delta \tau = 0 \tag{4.20a}$$

and

$$\mathcal{L}_{\bar{u}_i y_i}^{\mathrm{T}} \Delta y_i - \Delta \beta_i \Delta \tau + \bar{S} \Delta \nu_i \Delta \tau = 0. \tag{4.20b}$$

Therefore, we can also express the Newton directions of the dual variables $\Delta\beta_i$, $\Delta\mu_i$, and $\Delta\nu_i$ with the linear combinations of Δq_i, Δv_i, Δu_i, and $\Delta\gamma_{i+1}$ using (4.20a) and (4.20b) if we compute (4.19). Subsequently, the linear equations to be solved in the Newton-type iterations at stage i are reduced to that with respect to Δq_i, Δv_i, Δu_i, $\Delta\lambda_{i+1}$, and $\Delta\gamma_{i+1}$, defined as

$$\begin{bmatrix} \tilde{F}_{q,i} \\ \tilde{F}_{v,i} \end{bmatrix} + \begin{bmatrix} \delta_{q_i}(q_i, q_{i+1}) & I \Delta \tau \\ \tilde{F}_{vq,i} & \tilde{F}_{vv,i} \end{bmatrix} \begin{bmatrix} \Delta q_i \\ \Delta v_i \end{bmatrix} + \begin{bmatrix} O \\ \tilde{F}_{vu,i} \end{bmatrix} \Delta u_i + \begin{bmatrix} \delta_{q_{i+1}}(q_i, q_{i+1}) & O \\ O & -I \end{bmatrix} \begin{bmatrix} \Delta q_{i+1} \\ \Delta v_{i+1} \end{bmatrix} = 0, \tag{4.21a}$$

$$\begin{aligned} &\begin{bmatrix} \tilde{\mathcal{L}}_{q_i}^{\mathrm{T}} \\ \tilde{\mathcal{L}}_{v_i}^{\mathrm{T}} \end{bmatrix} + \begin{bmatrix} \tilde{\mathcal{L}}_{q_i q_i} & \tilde{\mathcal{L}}_{q_i v_i} \\ \tilde{\mathcal{L}}_{v_i q_i} & \tilde{\mathcal{L}}_{v_i v_i} \end{bmatrix} \begin{bmatrix} \Delta q_i \\ \Delta v_i \end{bmatrix} + \begin{bmatrix} \delta_{q_i}^{\mathrm{T}}(q_{i-1}, q_i) \Delta \lambda_i \\ -\Delta \gamma_i \end{bmatrix} \\ &+ \begin{bmatrix} \delta_{q_i}^{\mathrm{T}}(q_i, q_{i+1}) & \tilde{F}_{vq,i} \\ I_n \Delta \tau & \tilde{F}_{vv,i} \end{bmatrix} \begin{bmatrix} \Delta \lambda_{i+1} \\ \Delta \gamma_{i+1} \end{bmatrix} + \begin{bmatrix} \tilde{\mathcal{L}}_{q_i u_i} \\ \tilde{\mathcal{L}}_{v_i u_i} \end{bmatrix} \Delta u_i = 0, \end{aligned} \tag{4.21b}$$

and

$$\tilde{\mathcal{L}}_{u_i}^{\mathrm{T}} + \begin{bmatrix} \tilde{\mathcal{L}}_{q_i u_i} & \tilde{\mathcal{L}}_{v_i u_i} \end{bmatrix} \begin{bmatrix} \Delta q_i \\ \Delta v_i \end{bmatrix} + \tilde{\mathcal{L}}_{u_i u_i} \Delta u_i + \tilde{F}_{vu,i} \Delta \gamma_{i+1} = 0, \tag{4.21c}$$

where the vectors and matrices with a superscript tilde are derived via simple additions and multiplications of the original Hessians of the Lagrangian, Jacobians of the constraints, and residuals in the KKT conditions. We omit their definitions here because they are excessively long, but they are not hard to derive. Note that our previous approach (Chapter 3 and Katayama and Ohtsuka (2021)) involves solving the linear equation with respect to Δq_i, Δv_i, Δa_i, Δf_i, $\Delta\mu_i$, and $\Delta\nu_i$; thus, it can be inefficient compared with the proposed method. After solving the linear equations and obtaining Δq_i, Δv_i, Δu_i, $\Delta\lambda_{i+1}$, and $\Delta\gamma_{i+1}$, we can easily compute Δa_i, Δf_i, $\Delta\beta_i$, and $\Delta\mu_i$ using (4.17) and (4.20a), which is called an expansion procedure.

4.3.2 Lifted impulse dynamics

Next, we present the lifted impulse dynamics. Let $j \in \mathcal{J}$, $y_j := (q_j, v_j, \delta v_j, \Lambda_j)$, and $z_j := (q_j, v_j, \delta v_j)$. We express (4.9) as an equality constraint:

$$\Gamma(y_j) = 0, \tag{4.22}$$

where $r(y_i)$ is defined by the left-hand side of (4.9). We consider δv_j and Λ_j as the optimization variables and (4.22) and (4.7) as the equality constraints. The first-order derivatives of the Lagrangian $\mathcal{L}$ with respect to the variables at the stage of impulse j, that is, the part of the KKT conditions associated with the impulse stage j, are given by (4.11), (4.22), (4.7),

$$\begin{aligned}\begin{bmatrix} \mathcal{L}_{q_j}^{\mathrm{T}} \\ \mathcal{L}_{v_j}^{\mathrm{T}} \end{bmatrix} &= \begin{bmatrix} l_{q_j}^{\mathrm{T}} \\ l_{v_j}^{\mathrm{T}} \end{bmatrix} + \begin{bmatrix} \delta_{q_j}^{\mathrm{T}}(q_j, q_{j+1}) & O \\ O & I_n \end{bmatrix} \begin{bmatrix} \lambda_{j+1} \\ \gamma_{j+1} \end{bmatrix} \\ &+ \begin{bmatrix} \Gamma_{q_j}^{\mathrm{T}}(y_j) & \mathbf{v}_{q_j}^{\mathrm{T}}(z_j) \\ \Gamma_{v_j}^{\mathrm{T}}(y_j) & \mathbf{v}_{v_j}^{\mathrm{T}}(z_j) \end{bmatrix} \begin{bmatrix} \beta_j \\ \mu_j \end{bmatrix} + \begin{bmatrix} \delta_{q_j}^{\mathrm{T}}(q_{j-1}, q_j)\lambda_j \\ -\gamma_j \end{bmatrix} = 0, \end{aligned} \tag{4.23a}$$

and

$$\begin{bmatrix} \mathcal{L}_{\delta v_j}^{\mathrm{T}} \\ -\mathcal{L}_{\Lambda_j}^{\mathrm{T}} \end{bmatrix} = \begin{bmatrix} l_{\delta v_j}^{\mathrm{T}} \\ -l_{\Lambda_j}^{\mathrm{T}} \end{bmatrix} + \begin{bmatrix} \gamma_{j+1} \\ 0 \end{bmatrix} + \begin{bmatrix} \Gamma_{a_j}^{\mathrm{T}}(\tilde{y}_j) & \mathbf{v}_{\delta v_j}^{\mathrm{T}}(z_j) \\ -\Gamma_{f_j}^{\mathrm{T}}(\tilde{y}_j) & -\mathbf{v}_{f_j}^{\mathrm{T}}(z_j) \end{bmatrix} \begin{bmatrix} \beta_j \\ \mu_j \end{bmatrix} = 0, \tag{4.23b}$$

where $\lambda_{j+1}, \gamma_{j+1}, \beta_j \in \mathbb{R}^n$, and $\mu_j \in \mathbb{R}^{n_f}$ are the Lagrange multipliers with respect to (4.11), (4.22), and (4.7), respectively.

These KKT conditions are also linearized into linear equations with respect to the Newton directions Δq_i, Δv_i, $\Delta \delta v_i$, $\Delta \Lambda_i$, $\Delta \lambda_{i+1}$, $\Delta \gamma_{i+1}$, $\Delta \beta_i$, and $\Delta \mu_i$, which are larger than those of the non-lifted counterpart with respect to Δq_i, Δv_i, $\Delta \lambda_{i+1}$, and $\Delta \gamma_{i+1}$. Thus, we propose an efficient condensing method to reduce the size of the linear equation. We first observe that the equality constraints (4.22) and (4.7) are linearized as

$$\begin{bmatrix} M(q_j) & J^{\mathrm{T}}(q_j) \\ J(q_j) & O \end{bmatrix} \begin{bmatrix} \Delta \delta v_j \\ -\Delta \Lambda_j \end{bmatrix} = - \begin{bmatrix} \Gamma_{q_j}(y_j) & \Gamma_{v_j}(y_j) \\ \mathbf{v}_{q_j}(z_j) & \mathbf{v}_{v_j}(z_j) \end{bmatrix} \begin{bmatrix} \Delta q_j \\ \Delta v_j \end{bmatrix} - \begin{bmatrix} \Gamma(y_j) \\ \mathbf{v}(z_j) \end{bmatrix}. \tag{4.24a}$$

Therefore, we can express $\Delta \delta v_j$ and $\Delta \Lambda_j$ with the linear combinations of Δq_j and Δv_j if we compute (4.19) for q_j. Next, we observe that (4.23b) is linearized into

$$\begin{bmatrix} \mathcal{L}_{\delta v_j}^{\mathrm{T}} \\ -\mathcal{L}_{\Lambda_j}^{\mathrm{T}} \end{bmatrix} + \begin{bmatrix} \mathcal{L}_{\delta v_j y_j} \\ -\mathcal{L}_{\Lambda_j y_j} \end{bmatrix} \Delta y_j + \begin{bmatrix} \Delta \gamma_{j+1} \\ 0 \end{bmatrix} + \begin{bmatrix} M(q_j) & J^{\mathrm{T}}(q_j) \\ J(q_j) & O \end{bmatrix} \begin{bmatrix} \Delta \beta_j \\ \Delta \mu_j \end{bmatrix} = 0. \tag{4.24b}$$

Therefore, we can also express the Newton directions of the dual variables $\Delta \beta_j$ and $\Delta \mu_j$ with the linear combinations of Δq_j, Δv_j, and $\Delta \gamma_{j+1}$ if we solve (4.19) for q_j.

Subsequently, the linear equations to be solved in the Newton-type iterations are reduced with respect to Δq_j, Δv_j, $\Delta \lambda_{j+1}$, and $\Delta \gamma_{j+1}$ and are expressed as

$$\begin{bmatrix} \tilde{F}_{q,j} \\ \tilde{F}_{v,j} \end{bmatrix} + \begin{bmatrix} \delta_{q_j}(q_j, q_{j+1}) & O \\ \tilde{F}_{vq,j} & \tilde{F}_{vv,j} \end{bmatrix} \begin{bmatrix} \Delta q_j \\ \Delta v_j \end{bmatrix} + \begin{bmatrix} \delta_{q_{j+1}}(q_j, q_{j+1}) & O \\ O & -I \end{bmatrix} \begin{bmatrix} \Delta q_{j+1} \\ \Delta v_{j+1} \end{bmatrix} = 0 \quad (4.25a)$$

and

$$\begin{bmatrix} \tilde{\mathcal{L}}_{q_j}^{\mathrm{T}} \\ \tilde{\mathcal{L}}_{v_j}^{\mathrm{T}} \end{bmatrix} + \begin{bmatrix} \tilde{\mathcal{L}}_{q_j q_j} & \tilde{\mathcal{L}}_{q_j v_j} \\ \tilde{\mathcal{L}}_{v_j q_j} & \tilde{\mathcal{L}}_{v_j v_j} \end{bmatrix} \begin{bmatrix} \Delta q_j \\ \Delta v_i \end{bmatrix} + \begin{bmatrix} \delta_{q_j}^{\mathrm{T}}(q_{j-1}, q_j)\Delta\lambda_j \\ -\Delta\gamma_j \end{bmatrix} + \begin{bmatrix} \delta_{q_j}^{\mathrm{T}}(q_j, q_{j+1}) & \tilde{F}_{vq,j} \\ O & \tilde{F}_{vv,j} \end{bmatrix} \begin{bmatrix} \Delta\lambda_{j+1} \\ \Delta\gamma_{j+1} \end{bmatrix} = 0, \quad (4.25b)$$

for which we omit the definitions of the terms with a superscript tilde because they are similar to those of the lifted contact dynamics. After solving the linear equations and obtaining Δq_j, Δv_j, $\Delta \lambda_{j+1}$, and $\Delta \gamma_{j+1}$, we can easily compute $\Delta \delta v_j$, $\Delta \Lambda_j$, $\Delta \beta_j$, and $\Delta \mu_j$ using (4.24a) and (4.24b), which is the expansion procedure of the impulse stage.

4.3.3 Riccati recursion for LQR subproblem

After performing the above condensing, the Newton step computation is reduced to solving a set of linear equations with respect to Δq_i, Δv_i, Δu_i, $\Delta \lambda_i$, and $\Delta \gamma_i$. This can be seen as the KKT conditions of an LQR subproblem with the state $[\Delta q_i^{\mathrm{T}} \ \ \Delta v_i^{\mathrm{T}}]^{\mathrm{T}}$ and control input Δu_i. That is, solving the linear equations after condensing is equivalent to solving the LQR subproblem. We then utilize the Riccati recursion algorithm (Rawlings et al. (2017)), which can solve the LQR subproblem only with $O(N)$ complexity while the direct inversion of the KKT matrix requires an $O(N^3)$ computational burden. For example, DDP uses the Riccati recursion for single-shooting OCPs with a nonlinear forward pass and can efficiently solve very large-scale problems (Budhiraja et al. (2018); Koenemann et al. (2015); Mastalli et al. (2020)).

4.3.4 Primal-dual interior-point method for inequality constraints

OCPs of rigid-body systems can contain many inequality constraints, e.g., joint angle limits, joint angular velocity limits, joint torque limits, and friction cones. To treat such a large number of inequality constraints including nonlinear ones with the Riccati recursion algorithm efficiently, we use the primal-dual interior-point (PDIP) method (Nocedal and Wright (2006)) . In the PDIP method, only certain terms related to the second- and first-order derivatives of the logarithmic barrier functions are added to the Hessians $\mathcal{L}_{y_i y_i}$ and residuals $\mathcal{L}_{y_i}$, respectively, and there are no effects on the

Algorithm 4.1 Computation of the Newton direction using the lifted contact dynamics

```
Input: Initial state $x(t_0)$, the current solution $y_0, ..., y_{N-1}$, $q_N$, $v_N$, $\{u_i\}_{i \in \mathcal{I}}$, $\lambda_0, ..., \lambda_N$,
    $\gamma_0, ..., \gamma_N$, $\beta_0, ..., \beta_{N-1}$, $\mu_0, ..., \mu_{N-1}$, and $\{\nu_i\}_{i \in \mathcal{I}}$
Output: Newton directions $\Delta y_0, ..., \Delta y_{N-1}$, $\Delta q_N$, $\Delta v_N$, $\{\Delta u_i\}_{i \in \mathcal{I}}$, $\Delta \lambda_0, ..., \Delta \lambda_N$,
    $\Delta \gamma_0, ..., \Delta \gamma_N$, $\Delta \beta_0, ..., \Delta \beta_{N-1}$, $\Delta \mu_0, ..., \Delta \mu_{N-1}$, and $\{\Delta \nu_i\}_{i \in \mathcal{I}}$
1: for $i = 0, \cdots, N$ do in parallel
2:     Compute the Hessian of the Lagrangian (e.g., $\mathcal{L}_{q_i q_i}$), Jacobians of the constraints (e.g., $\mathrm{ID}_{q_i}$ and $\mathbf{a}_{q_i}$), and residuals in the KKT conditions (e.g., (4.3), (4.15a), (4.5), (4.15b), and $\mathcal{L}_{q_i}$).
3:     Compute the matrix inversion (4.19).
4:     Form the condensed Hessians, Jacobians, and KKT residuals (e.g., $\tilde{\mathcal{L}}_{q_i q_i}$, $\tilde{F}_{qq,i}$, and $\tilde{\mathcal{L}}_{q_i}$).
5: end for
6: Compute $\Delta q_0, ..., \Delta q_N$, $\Delta v_0, ..., \Delta v_N$, $\{\Delta u_i\}_{i \in \mathcal{I}}$, $\Delta \lambda_0, ..., \Delta \lambda_N$, and $\Delta \gamma_0, ..., \Delta \gamma_N$ by solving the LQR subproblem, e.g., using the Riccati recursion.
7: for $i = 0, \cdots, N-1$ do in parallel
8:     Compute the condensed direction ($\Delta a_i$, $\Delta f_i$, $\Delta \beta_i$, $\Delta \mu_i$, and $\Delta \nu_i$ for $i \in \mathcal{I}$ or $\Delta \delta v_j$, $\Delta \Lambda_j$, $\Delta \beta_j$, and $\Delta \mu_j$ for $j \in \mathcal{J}$).
9: end for
```

proposed condensing method and the LQR subproblem. We can then apply the Riccati recursion algorithm regardless of the inequality constraints. After solving the LQR subproblem and obtaining Δy_i, we can efficiently compute the Newton steps of the slack variables and Lagrange multipliers of the inequality constraints.

4.3.5 Algorithm

We summarize the single Newton iteration, that is, the computation of the Newton direction of the proposed method, in Algorithm 1. First, we compute the Hessians of the Lagrangian, Jacobians of the constraints, and residuals in the KKT conditions (line 2). The modification of the Hessians and KKT residuals due to the PDIP method is also performed in this step. Second, we compute the matrix inversion (4.19) and form the condensed Hessians, Jacobians, and KKT residuals (lines 3 and 4). These two steps are fully parallelizable because of the multiple-shooting formulation. We then solve the LQR subproblem, e.g., using the Riccati recursion (line 6). Finally, we compute the condensed directions from the solution of the LQR subproblem (line 8). The directions of the slack variables and Lagrange multipliers of the PDIP method are also computed in this step.

Table 4.1: Settings of the OCPs for the five quadrupedal gaits

	Walking	Trotting	Pacing	Bounding	Jumping
N	107	57	71	71	71
$\Delta\tau$	0.02	0.02	0.01	0.01	0.01

4.3.6 Comparison with existing methods

Here, we re-summarize the comparison between the proposed method and the existing methods:

4.3.6.1 Comparison with non-lifted formulations

The non-lifted formulations (Budhiraja et al. (2018); Li and Wensing (2020); Mastalli et al. (2020); Schultz and Mombaur (2010)) eliminate a_i, f_i, δv_i, and Λ_i as (nonlinear) functions of q_i, v_i, and u_i. Therefore, the costs and constraints including a_i, f_i, δv_i, and Λ_i (e.g., the friction cone constraints) can be highly nonlinear. The proposed method regard a_i, f_i, δv_i as decision variables to alleviate such high nonlinearities. The computational costs per Newton-type iteration of the proposed method and these methods are similar owing to the proposed efficient condensing procedure.

4.3.6.2 Comparison with inverse dynamics-based algorithm

The inverse dynamics-based method of Chapter 3 and Katayama and Ohtsuka (2021) regards a_i and f_i as the optimization variables: the convergence behavior is identical to that of the proposed method. However, the proposed condensing method can be more efficient than that presented in Katayama and Ohtsuka (2021). The proposed method requires a computational cost of $O(n_a^3)$ for solving the linear system to compute the Newton directions in the Riccati recursion. The method of Katayama and Ohtsuka (2021) requires $O((n+n_f)^3)$, which is inefficient when the numbers of contacts and passive joints (i.e., n_f and $n-n_a$) are large.

4.4 Numerical Experiments: Whole-Body Optimal Control of Quadrupedal Gaits

4.4.1 Experimental settings

To demonstrate the effectiveness of the proposed lifting approach over the conventional non-lifting approaches, we conducted numerical experiments on the whole-body optimal control of quadrupedal robot ANYmal for five gaits, that is, walking, trotting, pacing, bounding, and jumping, subject to the friction cone constraints. We considered the polyhedral-approximated friction cone constraint for each contact force expressed in the world frame $[f_x \ f_y \ f_z]$ as

$$\begin{bmatrix} f_x + \frac{\mu}{\sqrt{2}} f_z \\ -f_x + \frac{\mu}{\sqrt{2}} f_z \\ f_y + \frac{\mu}{\sqrt{2}} f_z \\ -f_y + \frac{\mu}{\sqrt{2}} f_z \\ f_z \end{bmatrix} \geq 0, \tag{4.26}$$

where $\mu > 0$ is the friction coefficient, and we set it as $\mu = 0.7$. The problem settings of the OCPs for the five gaits are shown in Table I. We compared the following five methods:

- DMS-LCD: DMS based on the proposed lifted contact dynamics
- DMS-CD: DMS based on the conventional non-lifted contact dynamics (e.g., used in Schultz and Mombaur (2010))
- DMS-ID: Inverse dynamics-based DMS algorithm (Chapter 3 and Katayama and Ohtsuka (2021))
- FDDP: Feasibility-driven DDP (FDDP), a variant of iLQR that improves the numerical robustness (Mastalli et al. (2020)), based on the conventional non-lifted contact dynamics
- iLQR: iLQR based on the conventional non-lifted contact dynamics

The following were compared in this study: 1) The convergence property between the lifted and non-lifted formulations (DMS-LCD vs. DMS-CD); 2) The CPU time per Newton iteration between the two condensing methods for the lifted formulation (DMS-LCD vs. DMS-ID); 3) The proposed method with a state-of-the-art OCP solver (Mastalli et al. (2020)) (DMS-LCD vs. FDDP and iLQR). DMS-LCD, DMS-CD, and DMS-ID used Gauss–Newton Hessian approximation, the PDIP method method for inequality constraints, and the Riccati recursion to solve the LQR subproblems. FDDP and iLQR used the relaxed barrier function (ReB) method (Hauser and Saccon (2006)), which is a popular constraint-handling approach in DDP-type methods (e.g., used in Li and Wensing (2020)) that enables the iteration to lie outside the feasible region. Note that the ReB method and the PDIP method correspond to the same barrier function under the same barrier parameter and feasible solution (Hauser and

Saccon (2006); Nocedal and Wright (2006)). That is, the five methods consider the same barrier functions for the inequality constraints. We implemented DMS-LCD[2], DMS-ID, and DMS-CD in C++ and used `Pinocchio` (Carpentier et al. (2019)), an efficient C++ library used for rigid-body dynamics and its analytical derivatives, to compute the dynamics and its derivatives of the quadrupedal robot. We used OpenMP for parallel computing (e.g., lines 1–5 of Algorithm 1). For FDDP and iLQR, we used the `Crocoddyl` framework (Mastalli et al. (2020)), which was also implemented in C++, `Pinocchio` for rigid-body dynamics, and OpenMP for parallel computing. FDDP and iLQR consider the Baumgarte's stabilization method of the form in (4.8) as well as DMS-based methods.

We designed the cost functions $V_f(\cdot)$ and $l_i(\cdot)$ in (4.14) as least-square objectives to track the pre-defined feet and center of mass (CoM) trajectories, e.g.,

$$\frac{1}{2}(\phi_i(x_i) - \phi_{i,\mathrm{ref}})^{\mathrm{T}} W (\phi_i(x_i) - \phi_{i,\mathrm{ref}}),$$

where $\phi_i(x_i)$ denotes the stack of the feet and CoM positions. We also imposed simple quadratic weights on the deviation of q_i from the standing pose, v_i, and u_i . We set the stabilization parameters in (4.5) as $\alpha = \beta = 25$. We approximated the contact position constraints (4.4) using a quadratic penalty function for simplicity. DMS-LCD, DMS-CD, and DMS-ID did not use any regularization on the Hessian and used only the fraction-to-boundary-rule (Nocedal and Wright (2006)) for the step-size selection. FDDP and iLQR used an adaptive regularization on the Hessian and a line-search based on the Goldstein condition in the step-size selection, as provided in the `Crocoddyl` solver. We conducted all the experiments for the two barrier parameters ϵ: $\epsilon = 1.0 \times 10^{-1}$ and $\epsilon = 1.0 \times 10^{-4}$, which represent a large and small barrier parameter, respectively. ϵ is fixed during each experiment, which corresponds to suboptimal MPC cases. We set the relaxation parameter of the ReB for each ϵ and each gait as large as possible while keeping the optimal solution satisfying (4.26), which means that the *ideal* ReB (that is, a favorable experimental setting for ReB-based FDDP and iLQR) was considered in the experiments The experiment was conducted on a desktop computer with an octa-core CPU Intel Core i9-9900 @3.10 GHz, and all the algorithms were compiled using the GCC compiler with `-O3 -DNDEBUG -march=native` options. We used eight threads in parallel computing.

[2]Our open-source implementation of DMS-LCD is available online at Katayama (2020-2022)

4.4.2 Results and discussion

Figure 4.2 shows $\log_{10}$ scaled l_2-norms of the KKT residuals with respect to the number of iterations of these five methods over the five quadrupedal gaits with the barrier parameters $\epsilon = 1.0 \times 10^{-1}$ and $\epsilon = 1.0 \times 10^{-4}$. Note that the convergence results of DMS-LCD and DMS-ID are illustrated in the same solid lines because they are identical. As shown in Fig. 4.2, the five methods resulted in a similar convergence when the barrier parameter was large ($\epsilon = 1.0 \times 10^{-1}$). In contrast, when the barrier parameter was small ($\epsilon = 1.0 \times 10^{-4}$), the proposed DMS-LCD converged significantly faster than the other methods, particularly in the aggressive gaits (pacing, bounding, and jumping). This is because, in the interior-point methods, a smaller barrier parameter (i.e., a more accurate solution) corresponds to a slower and more difficult convergence due to the high nonlinearity (Nocedal and Wright (2006)). Nevertheless, the lifting methods (DMS-LCD and DMS-ID) achieved fast convergence even with the small barrier parameter. The faster convergence of DMS-LCD than DMS-CD particularly shows the effectiveness of the lifted formulation.

Figure 4.3 shows the CPU time per iteration and the total CPU time until convergence of the five methods. First, we observed that the CPU times per iteration of DMS-LCD and DMS-CD were almost identical and around 1.5 times faster than those of DMS-ID, which shows the efficiency of the proposed condensing algorithm over the previous one (Katayama and Ohtsuka (2021)). Furthermore, DMS-LCD and DMS-CD were 1.6 to 2 times faster than those of FDDP and iLQR because DMS could leverage parallel computing in the computation of the KKT residual, whereas single shooting methods such as FDDP and iLQR were required to compute the contact and impulse dynamics (4.8) and (4.13) over the horizon with a single thread. We compared the total computational time, and DMS-LCD had the fastest computational time in all the scenarios. It was more than twice as fast as the other non-lifted methods in several scenarios. Figure 4.4 shows the snapshots of the solution trajectory of the jumping gait and Fig. 4.5 shows the time histories of the contact forces of the four feet in the motion. Both figures show that the friction cone constraints (4.26) were active and satisfied during the jumping. A supplemental video including all the five gaits is available at https://youtu.be/jb7gGnblQ7s.

We further conducted several experiments with the various Baumgarte's weight parameters $\alpha = \beta > 0$ in (4.5), the details of which are omitted here owing to the space limitation. We observed that as the weight parameter increased, the proposed method had greater advantages in terms of the convergence speed. In contrast, if the weight parameter was small, the proposed method had no such advantages. Note that a larger weight parameter typically results in a more accurate solution in terms of the original contact position constraints (4.4). This observation confirms that the proposed method can address high nonlinearity because the large Baumgarte's weight parameters cause high nonlinearity (Flores et al. (2011)).

4.5 Summary

We proposed a novel lifting approach for optimal control of rigid-body systems with contacts to improve the convergence properties of Newton-type methods. To relax the high nonlinearity, we considered the acceleration and contact forces as the optimization variables and the inverse dynamics and acceleration-level contact constraints as equality constraints. We eliminated the update of these additional variables from the linear equation for Newton-type method in an efficient manner. As a result, the computational cost per Newton-type iteration is almost identical to that of the conventional non-lifted one that embeds contact dynamics in the state equation. We conducted numerical experiments on the whole-body optimal control of various quadrupedal gaits subject to the friction cone constraints considered in the interior-point methods and demonstrated that the proposed method can significantly increase the convergence speed to more than twice that of the conventional non-lifted approaches.

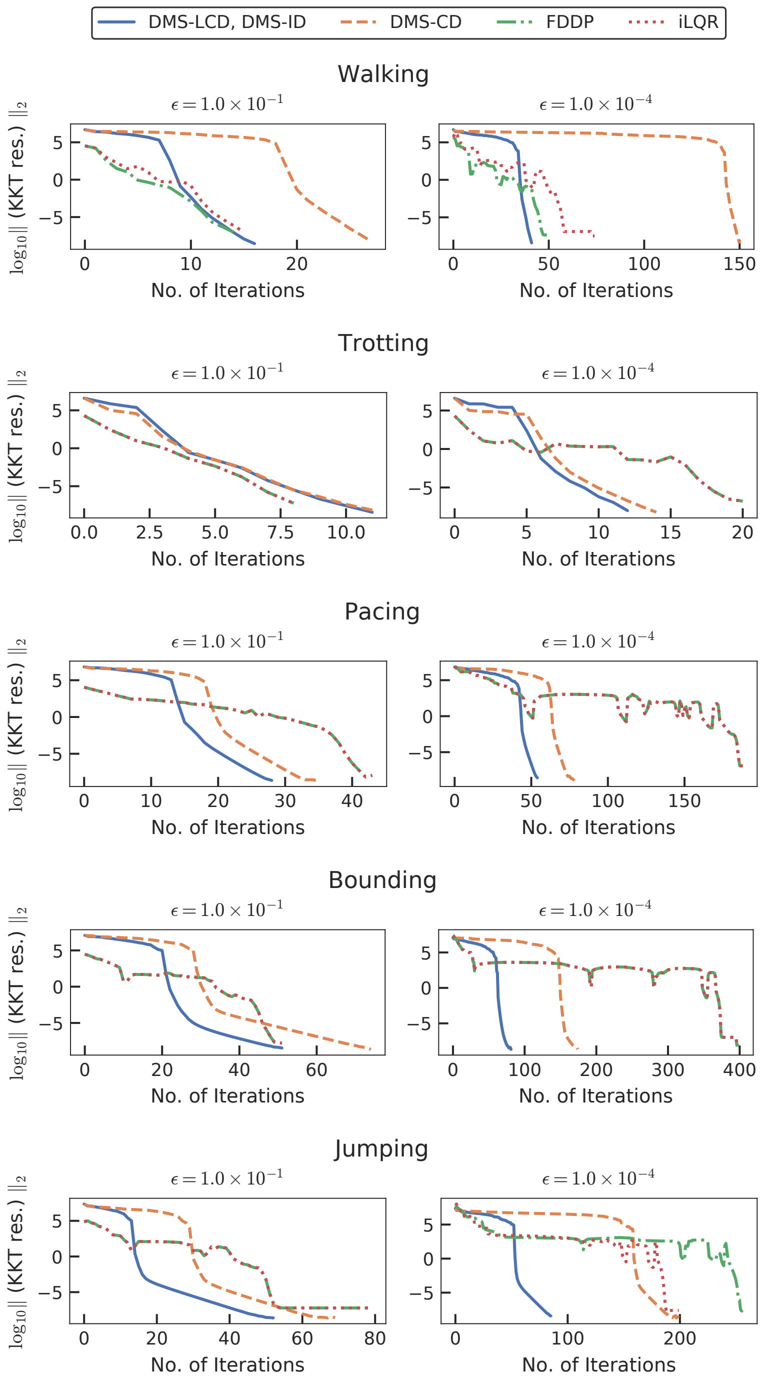


Figure 4.2: $\log_{10}$ scaled l_2-norms of the KKT residuals (KKT res. in short) of DMS-LCD and DMS-ID (solid lines), DMS-CD (dashed lines), FDDP (dash-dotted lines), and iLQR (dotted lines) over the five quadrupedal gaits subject to the friction cone constraints considered in the interior-point methods with the fixed barrier parameters $\epsilon = 1.0 \times 10^{-1}$ and $\epsilon = 1.0 \times 10^{-4}$.

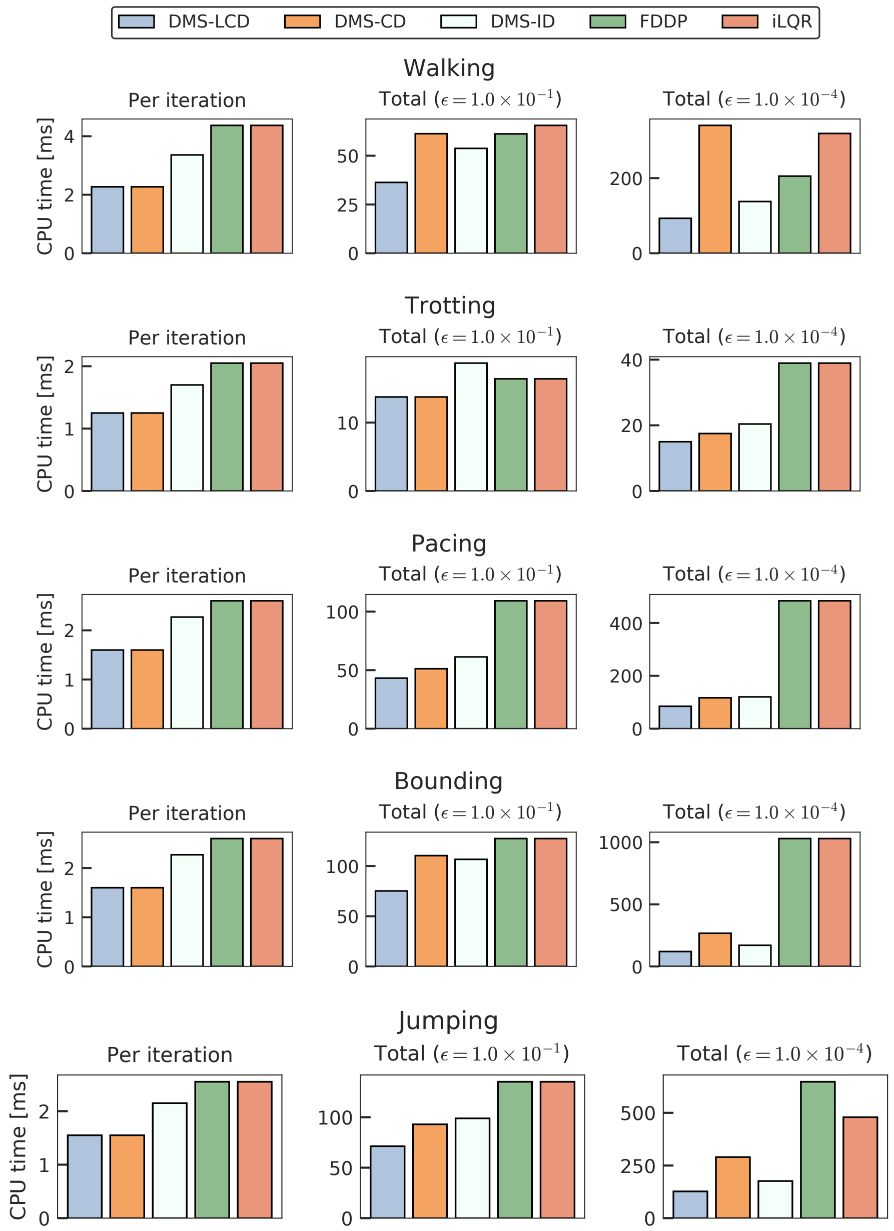


Figure 4.3: CPU time per iteration and the total CPU time until convergence of DMS-LCD, DMS-CD, DMS-ID, FDDP, and iLQR over the five quadrupedal gaits subject to the friction cone constraints considered in the interior-point methods with the fixed barrier parameters $\epsilon = 1.0 \times 10^{-1}$ and $\epsilon = 1.0 \times 10^{-4}$.

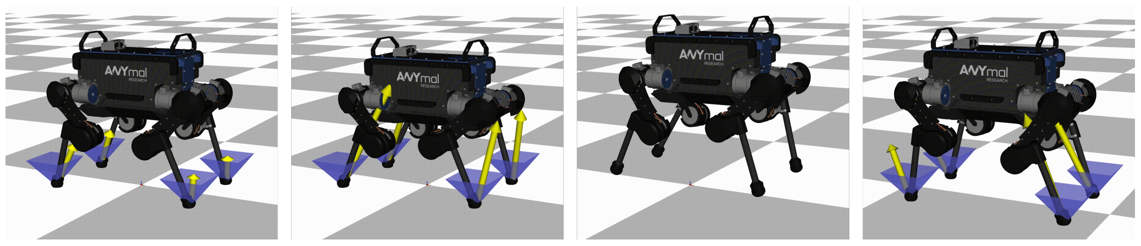


Figure 4.4: Snapshots of the whole-body optimal control of the jumping motion of ANYmal subject to the friction cone constraints. The contact forces are indicated by the yellow arrows and linearized friction cones by the blue polyhedrons.

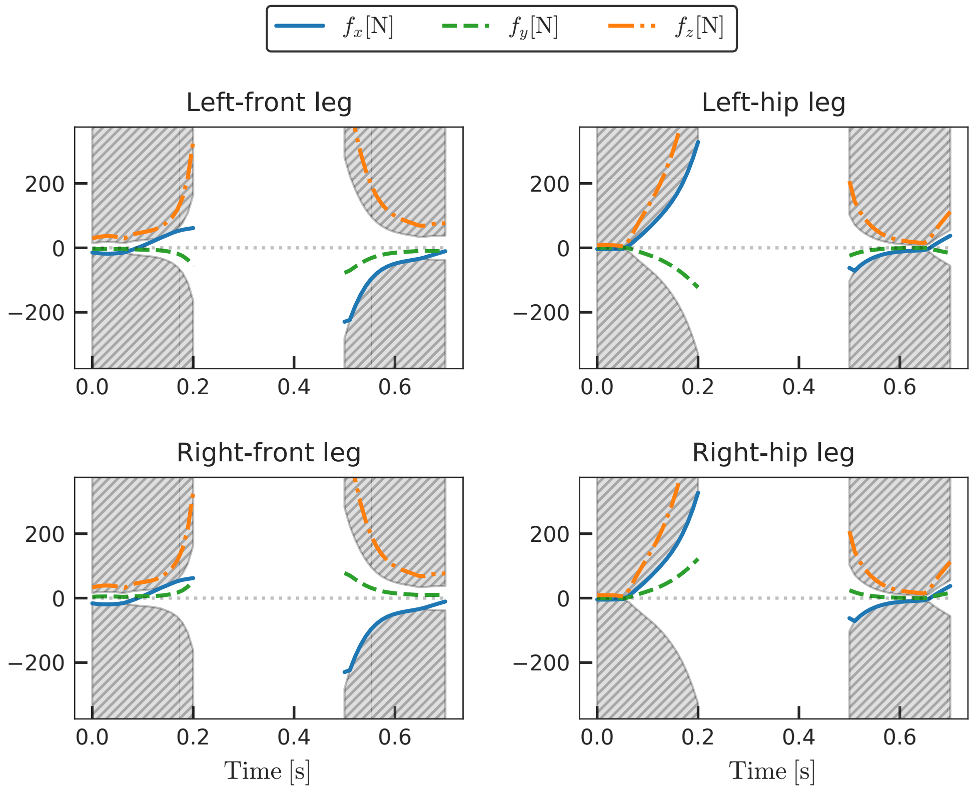


Figure 4.5: Time histories of the contact force expressed in the world frame $[f_x \;\; f_y \;\; f_z]$ of each leg in the jumping motion. The infeasible regions of f_x and f_y due to the friction cone constraints (4.26) are the filled gray hatches. The infeasible region of $f_z \geq 0$ is the lower-half space of each plot.

Chapter 5

Efficient Riccati Recursion for Optimal Control Problems with Pure-State Equality Constraints[1]

5.1 Introduction

Optimal control underlies the motion planning and control of dynamical systems such as trajectory optimization (TO) and model predictive control (MPC) (Rawlings et al. (2017)). TO achieves versatile and dynamically consistent planning by solving optimal control problems (OCPs). MPC leverages the same advantages as TO in real-time control by solving an OCP online within a particular sampling interval. It is essential, particularly for MPC, to solve direct OCPs within a short computational time, even if they involve highly complicated dynamics, a large dimensional state, and a long horizon.

Newton-type methods are the most practical methods used for solving OCPs in terms of the convergence speed. One of the most efficient algorithms that implement the Newton-type methods to solve both the single-shooting and multiple-shooting OCPs of large-scale systems is the Riccati recursion algorithm (Frison (2016); Rawlings et al. (2017)). The Riccati recursion algorithm scales only linearly with respect to the number of discretization grids of the horizon, in contrast to the direct methods (i.e., methods applying the Cholesky decomposition directly to the entire Hessian matrix) that scale cubically. For example, it was successfully applied to solve OCPs even for significantly complex systems such as legged robots with large degrees of

[1]© 2022 IEEE. This chapter is reprinted, with permission, from S. Katayama and T. Ohtsuka, "Efficient Riccati recursion for optimal control problems with pure-state equality constraints," 2022 American Control Conference (ACC 2022), pp. 3579–3586, 2022.

freedom within very short computational times (Koenemann et al. (2015); Mastalli et al. (2020); Neunert et al. (2018)).

However, there is a drawback of the Riccati recursion algorithm: it cannot efficiently treat pure-state equality constraints, which often arise, for example, in waypoint constraints, terminal constraints, switching constraints in hybrid systems such as legged robots (Farshidian, Neunert, Winkler, Rey, and Buchli (2017); Li and Wensing (2020)), and inequality constraints handled by active-set methods (Sideris and Rodriguez (2011)). Sideris and Rodriguez (2011) extended the Riccati recursion algorithm for pure-state equality constraints and illustrated its effectiveness over the direct method for certain quadratic programming problems. However, the computational time of this method scales cubically with respect to the total dimension of pure-state equality constraints over the horizon, and it is inefficient when the total dimension is large. Giftthaler and Buchli (2017) proposed a projection approach to treat pure-state equality constraints with the Riccati recursion algorithm efficiently. However, this approach can only treat the equality constraints whose relative degree is 1, for example, velocity-level constraints of second-order systems, and cannot treat position-level constraints for such systems, which is a very common and practical class of constraints.

Other popular constraint-handling methods used with the Riccati recursion algorithm are the penalty function method and the augmented Lagrangian (AL) method. For example, Farshidian et al. (2017) used the penalty function method and Li and Wensing (2020) used the AL method to treat pure-state equality constraints representing the switching constraints arising in OCPs involving quadrupedal gaits. A transformation of the linear-quadratic OCP into a dual problem for the efficient Riccati recursion (Axehill (2005)) also used the penalty function method to treat the pure-state constraints. However, the penalty function method practically yields only the approximated solution, as illustrated by the numerical results obtained in Farshidian et al. (2017). The AL method can treat constraints better than the penalty function method, for example, it converges to the optimal solution even if the penalty parameter remains at a finite value. However, it generally lacks convergence speed compared with the Newton-type methods that achieve superlinear or quadratic convergence. The AL method essentially achieves linear convergence and it yields superlinear convergence only if the penalty parameter goes to infinity (Bertsekas (2016)), which is an impractical assumption. For example, Li and Wensing (2020) used the AL method to consider the switching constraints in an OCP of quadruped bouncing motion. The AL method required a large number of the iterations (up to 300), although a simple

2D robot model was used and only one cycle of the bouncing motion was considered, which led to very low-dimensional (only four dimensions in all) pure-state equality constraints.

In this chapter, we propose a novel approach to efficiently treat pure-state equality constraints in OCPs using a Riccati recursion algorithm. The proposed method transforms a pure-state equality constraint into a mixed state-control constraint such that the constraint is expressed by variables at a certain previous time stage. We show a relationship between an OCP with the original pure-state constraint (the original OCP) and an OCP with the transformed mixed state-control constraint (the transformed OCP); if the solution satisfies the first-order necessary conditions (FONC) and/or second-order sufficient conditions (SOSC) of the transformed OCP, then the solution also satisfies the FONC and/or SOSC of the original OCP. Therefore, if we find a solution that satisfies the SOSC of the transformed OCP, it is a local minimum of the original OCP. We then derive a Riccati recursion algorithm to solve the transformed OCP with linear time complexity in the grid number of the horizon, in contrast with the previous approach of Sideris and Rodriguez (2011) that scales cubically with respect to the total dimension of pure-state equality constraints. Moreover, because the proposed method is in substance a Newton's method for an optimization problem with equality constraints, the proposed method achieves superlinear or quadratic convergence, which distinguishes our approach from the penalty function method and the AL method in terms of convergence speed. We present numerical experiments of the whole-body optimal control of quadrupedal gaits that involve pure-state equality constraints owing to contact switches, which represent the position-level constraints of a second-order system, and demonstrate the effectiveness of the proposed method over the existing approaches, that is, the approach of Sideris and Rodriguez (2011) and the AL method.

This chapter is organized as follows. In Section 5.2, we transform an OCP with a pure-state equality constraint into an OCP with a mixed state-control equality constraint. In Section 5.3, we derive a Riccati recursion algorithm to apply Newton's method efficiently to the OCP with transformed mixed state-control equality constraints. In Section 5.4, the theoretical properties of the proposed transformation of OCPs are described. In Section 5.5, the proposed method is compared with existing methods and its effectiveness is demonstrated in terms of computational time and convergence speed. In Section 5.6, we conclude with a brief summary and mention of future work.

Notation: We describe the partial derivatives of a differentiable function with respect to certain variables using a function with subscripts; that is, $f_x(x)$ denotes $\frac{\partial f}{\partial x}(x)$ and $g_{xy}(x,y)$ denotes $\frac{\partial^2 g}{\partial x \partial y}(x,y)$.

5.2 Transformation of Optimal Control Problem with Pure-State Equality Constraints

5.2.1 Original optimal control problem

We consider the following discrete-time OCP: Find the state $x_0, ..., x_N \in \mathbb{R}^{n_x}$ and the control input $u_0, ..., u_{N-1} \in \mathbb{R}^{n_u}$ minimizing the cost function

$$J = V(x_N) + \sum_{i=0}^{N-1} L(x_i, u_i) \tag{5.1a}$$

subject to the state equation

$$x_i + f(x_i, u_i) - x_{i+1} = 0, \;\; i \in \{0, ..., N-1\}, \tag{5.1b}$$

a pure-state equality constraint

$$\phi(x_k) = 0, \; \phi(x_k) \in \mathbb{R}^{n_c}, \tag{5.1c}$$

and the initial state constraint

$$x_0 - \bar{x} = 0, \; \bar{x} \in \mathbb{R}^{n_x}. \tag{5.1d}$$

In the following, we assume the form of the state as $x_i = \begin{bmatrix} q_i^{\mathrm{T}} & v_i^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}}$, where $q_i \in \mathbb{R}^n$ and $v_i \in \mathbb{R}^n$ represent the generalized coordinates and velocity of the system, respectively, and assume the form of the state equation as follows:

$$f(x_i, u_i) := \begin{bmatrix} f^{(q)}(x_i) \\ f^{(v)}(x_i, u_i) \end{bmatrix}, \;\; f^{(q)}(x_i), \; f^{(v)}(x_i, u_i) \in \mathbb{R}^n. \tag{5.2}$$

We also assume that $k \geq 2$, $n_u \leq n$, $n_c \leq n$, and that the constraint (5.1c) depends only on the generalized coordinate, that is, its Jacobian is expressed as follows:

$$\phi_x(x_k) = \begin{bmatrix} \phi_q(q_k) & O \end{bmatrix}, \;\; \phi_q(q_k) \in \mathbb{R}^{n_c \times n}. \tag{5.3}$$

The state equation (5.2) mainly represents a second-order Lagrangian system with n degrees of freedom, and a constraint (5.1c), whose Jacobian is of form (5.3) represents a position-level constraint (that is, the relative degree of the constraint with respect to the control input is 2), which is a very common and practical class of problem settings.

5.2.2 Transformation of optimal control problem

To solve the aforementioned OCP efficiently, we transform the original pure-state equality constraint (5.1c) into a mixed state-control equality constraint that is equivalent to (5.1c) as long as (5.1b) is satisfied. If (5.1b) is satisfied,

$$x_k = x_{k-2} + f(x_{k-2}, u_{k-2}) + f(x_{k-2} + f(x_{k-2}, u_{k-2}), u_{k-1})$$

holds and therefore

$$\phi(x_{k-2} + f(x_{k-2}, u_{k-2}) + f(x_{k-2} + f(x_{k-2}, u_{k-2}), u_{k-1})) = 0 \tag{5.4}$$

is equivalent to (5.1c). Furthermore, because $\phi(\cdot)$ only depends on the generalized coordinate, (5.4) is equivalent to

$$\phi(x_{k-2} + f(x_{k-2}, u_{k-2}) + g(x_{k-2} + f(x_{k-2}, u_{k-2}))) = 0, \tag{5.5a}$$

where we define

$$g(x) := \begin{bmatrix} f^{(q)}(x) \\ 0 \end{bmatrix}. \tag{5.5b}$$

Therefore, the constraint (5.5a) is equivalent to (5.1c) if (5.1b) is satisfied. Therefore, we consider (5.5a) instead of (5.1c) in the following. We herein summarize the original and transformed OCPs as follows:

Problem 5.1. - Original OCP: *Find the solution $x_0, ..., x_N$, $u_0, ..., u_{N-1}$ minimizing (5.1a) subject to (5.1b)–(5.1d).*

Problem 5.2. - Transformed OCP: *Find the solution $x_0, ..., x_N$, $u_0, ..., u_{N-1}$ minimizing (5.1a) subject to (5.1b), (5.1d), and (5.5a).*

In fact, we have the following relations between the transformed OCP and the original OCP: If the solution $x_0, ..., x_N$, $u_0, ..., u_{N-1}$ satisfies the FONC of the transformed OCP, it also satisfies the FONC of the original OCP. If the solution $x_0, ..., x_N$, $u_0, ..., u_{N-1}$ satisfies the SOSC of the transformed OCP, it also satisfies the SOSC of the original OCP. Therefore, if we find the solution $x_0, ..., x_N$, $u_0, ..., u_{N-1}$ that satisfies the SOSC of the transformed OCP, it is a local minimum of the original OCP. We show these theoretical points later in Section 5.4.

It should be noted that it is trivial to apply the proposed approach to constraints of relative degree 1: we transform (5.1c) into a mixed state-control constraint represented by x_{k-1} and u_{k-1} in the same manner as mentioned before. Therefore, our approach comprises the approach of Giftthaler and Buchli (2017) that also involves constraint transformation but can treat only constraints of relative degree 1. The difference between our approach and that of Giftthaler and Buchli (2017) is that we introduce the constraint transformation in the original nonlinear problem and leverage the structure of the system (5.1b), whereas in approach of Giftthaler and Buchli (2017), constraint transformation is introduced in the linear subproblem arising in the Newton-type iterations without any assumptions in the state equation. As a result, only our approach can treat the practically important constraints of relative degree 2 with a theoretical justification for the transformation.

It should also be noted that the proposed approach is completely different from the classical transformation of pure-state equality constraints in continuous-time OCPs by considering their derivatives with respect to time, for example, in Section 3.4 of (Bryson and Ho (1975)). To explain this difference, we consider that there is a pure-state constraint of the form of (5.1c) over a time interval. The classical method then transforms the pure-state constraint over the interval into a combination of the pure-state equality constraints (5.1c) and $\frac{d}{dt}\phi(x) = 0$ at a point on the interval and the mixed state-control constraint $\frac{d^2}{dt^2}\phi(x) = 0$ over the interval, where $\frac{d}{dt}$ and $\frac{d^2}{dt^2}$ yield a kind of Lie derivative. Therefore, the classical method still needs to consider the pure-state equality constraints, whereas our approach transforms all pure-state equality constraints into the corresponding mixed state-control constraints.

5.2.3 Optimality conditions

We derive the optimality conditions, known as FONC, of the transformed OCP. We first define the Hamiltonian

$$H(x, u, \lambda) := L(x, u) + \lambda^{\mathrm{T}} f(x, u)$$

and

$$\tilde{H}(x, u, \lambda, \nu) := H(x, u, \lambda) + \nu^{\mathrm{T}} \phi(x + f(x, u) + g(x + f(x, u))).$$

We also define the intermediate time stages in which the constraint is not active as $\bar{I} := \{1, ..., k-3, k-1, ..., N-1\}$. The optimality conditions are then derived as follows (Bryson and Ho (1975)):

$$V_x^{\mathrm{T}}(x_N) - \lambda_N = 0, \tag{5.6}$$

$$H_x^{\mathrm{T}}(x_i, u_i, \lambda_{i+1}) + \lambda_{i+1} - \lambda_i = 0 \tag{5.7}$$

and

$$H_u^{\mathrm{T}}(x_i, u_i, \lambda_{i+1}) = 0 \tag{5.8}$$

for $i \in \bar{I}$,

$$\begin{aligned}&\tilde{H}_x^{\mathrm{T}}(x_{k-2}, u_{k-2}, \lambda_{k-1}, \nu) + \lambda_{k-1} - \lambda_{k-2} \\ &= H_x^{\mathrm{T}}(x_{k-2}, u_{k-2}, \lambda_{k-1}) + \lambda_{k-1} - \lambda_{k-2} + (I + f_x^{\mathrm{T}}(x_{k-2}, u_{k-2}))(I + g_x^{\mathrm{T}})\phi_x^{\mathrm{T}}\nu = 0,\end{aligned} \tag{5.9}$$

and

$$\begin{aligned}&\tilde{H}_u^{\mathrm{T}}(x_{k-2}, u_{k-2}, \lambda_{k-1}, \nu) \\ &= H_u^{\mathrm{T}}(x_{k-2}, u_{k-2}, \lambda_{k-1}) + f_u^{\mathrm{T}}(x_{k-2}, u_{k-2})(I + g_x^{\mathrm{T}})\phi_x^{\mathrm{T}}\nu = 0,\end{aligned} \tag{5.10}$$

where $\lambda_0, ..., \lambda_N$ are the Lagrange multipliers with respect to (5.1d) and (5.1b), and ν is that with respect to (5.5a). It should be noted that we have omitted the arguments from ϕ_x and g_x in (5.9) and (5.10).

5.3 Riccati Recursion

5.3.1 Linearization for Newton's method

To apply Newton's method for the transformed OCP, we linearize the optimality conditions. It should be noted that we adopt the direct multiple shooting method (Bock and Plitt (1984)), that is, we consider $x_0, ..., x_N$, $u_0, ..., u_{N-1}$, $\lambda_0, ..., \lambda_N$, and ν as the optimization variables.

5.3.1.1 Terminal stage

At the terminal stage ($i = N$), we have

$$Q_{xx,N}\Delta x_N - \Delta\lambda_N + \bar{l}_{x,N} = 0, \tag{5.11}$$

where we define $Q_{xx,N} := V_{xx}(x_N)$. Further, we define $\bar{l}_{x,N}$ using the left-hand side of (5.6).

5.3.1.2 Intermediate stages without equality constraint

In the intermediate stages without an equality constraint ($i \in \bar{I}$), we have

$$A_i\Delta x_i + B_i\Delta u_i - \Delta x_{i+1} + \bar{x}_i = 0, \tag{5.12a}$$

$$Q_{xx,i}\Delta x_i + Q_{xu,i}\Delta u_i + A_i^{\mathrm{T}}\Delta\lambda_{i+1} - \Delta\lambda_i + \bar{l}_{x,i} = 0, \tag{5.12b}$$

and

$$Q_{xu,i}^{\mathrm{T}}\Delta x_i + Q_{uu,i}\Delta u_i + B_i^{\mathrm{T}}\Delta\lambda_{i+1} + \bar{l}_{u,i} = 0, \tag{5.12c}$$

where we define $A_i := I + f_x(x_i, u_i)$, $B_i := f_u(x_i, u_i)$, $Q_{xx,i} := H_{xx}(x_i, u_i, \lambda_{i+1})$, $Q_{xu,i} := H_{xu}(x_i, u_i, \lambda_{i+1})$, $Q_{uu,i} := H_{uu}(x_i, u_i, \lambda_{i+1})$. Further, we define $\bar{x}_i$, $\bar{l}_{x,i}$, and $\bar{l}_{u,i}$ using the left-hand sides of (5.1b), (5.7), and (5.8), respectively.

5.3.1.3 Intermediate stage with an equality constraint

At the intermediate stages with an equality constraint ($i = k-2$), we have (5.12a) for $i = k-2$ and have

$$Q_{xx,k-2}\Delta x_{k-2} + Q_{xu,k-2}\Delta u_{k-2} + A_{k-2}^{\mathrm{T}}\Delta\lambda_{k-1} - \Delta\lambda_{k-2} + C^{\mathrm{T}}\Delta\nu + \bar{l}_{x,k-2} = 0, \tag{5.13a}$$

$$Q_{xu,k-2}^{\mathrm{T}}\Delta x_{k-2} + Q_{uu,k-2}\Delta u_{k-2} + B_{k-2}^{\mathrm{T}}\Delta\lambda_{k-1} + D^{\mathrm{T}}\Delta\nu + \bar{l}_{u,k-2} = 0, \tag{5.13b}$$

and

$$C\Delta x_{k-2} + D\Delta u_{k-2} + \bar{\phi} = 0, \tag{5.13c}$$

where we define $Q_{xx,k-2} := \tilde{H}_{xx}(x_{k-2}, u_{k-2}, \lambda_{k-1}, \nu)$, $Q_{xu,k-2} := \tilde{H}_{xu}(x_{k-2}, u_{k-2}, \lambda_{k-1}, \nu)$, $Q_{uu,k-2} := \tilde{H}_{uu}(x_{k-2}, u_{k-2}, \lambda_{k-1}, \nu)$, $C := \phi_x(I + g_x)A_{k-2}$, $D := \phi_x(I + g_x)B_{k-2}$. We further define $\bar{l}_{x,k-2}$, $\bar{l}_{u,k-2}$, and $\bar{\phi}$ using the left-hand sides of (5.9), (5.10), and (5.5a), respectively.

5.3.1.4 Initial stage

Finally, we have

$$\Delta x_0 + x_0 - \bar{x} = 0. \tag{5.14}$$

It should be noted that we can apply the Gauss-Newton Hessian approximation, which improves the computational speed when the constraints (5.1b) and (5.5a) are too complicated for their second-order derivatives to be computed. $Q_{xx,N}$ and $Q_{xx,i}$, $Q_{xu,i}$, and $Q_{uu,i}$ for $i \in \{0, ..., N-1\}$ are then approximated using only the cost function (5.1a) and do not depend on the Lagrange multipliers.

5.3.2 Derivation of Riccati recursion

We derive a Riccati recursion algorithm to solve the linear equations for Newton's method (5.11)–(5.14) efficiently. As the standard Riccati recursion algorithm (Frison (2016); Rawlings et al. (2017)), our goal is to derive a series of matrices P_i and vectors s_i such that

$$\Delta\lambda_i = P_i\Delta x_i - s_i \tag{5.15}$$

holds.

5.3.2.1 Terminal stage

At the terminal stage ($i = N$), we have

$$P_N = Q_{xx,N}, \ s_N = -\bar{l}_N. \tag{5.16}$$

In the forward recursion, we have Δx_N, and we compute $\Delta \lambda_N$ from (5.15).

5.3.2.2 Intermediate stages without an equality constraint

At the intermediate stages without an equality constraint ($i \in \bar{I}$), we have the following standard backward Riccati recursion (Frison (2016); Rawlings et al. (2017)) for given P_{i+1} and s_{i+1} satisfying (5.15):

$$F_i := Q_{xx,i} + A_i^{\mathrm{T}} P_{i+1} A_i, \tag{5.17a}$$

$$H_i := Q_{xu,i} + A_i^{\mathrm{T}} P_{i+1} B_i, \tag{5.17b}$$

$$G_i := Q_{uu,i} + B_i^{\mathrm{T}} P_{i+1} B_i, \tag{5.17c}$$

$$K_i := -G_i^{-1} H_i^{\mathrm{T}}, \ k_i := -G_i^{-1} (B_i^{\mathrm{T}} P_{i+1} \bar{x}_i - B_i^{\mathrm{T}} s_{i+1} + \bar{l}_{u,i}), \tag{5.17d}$$

and

$$P_i := F_i - K_i^{\mathrm{T}} G_i K_i, \ s_i := A_i^{\mathrm{T}} (s_{i+1} - P_{i+1} \bar{x}_i) - \bar{l}_{x,i} - H_i k_i. \tag{5.17e}$$

In the forward recursion, for a particular value of Δx_i, we compute Δu_i from

$$\Delta u_i = K_i \Delta x_i + k_i, \tag{5.17f}$$

$\Delta \lambda_i$ from (5.15), and Δx_{i+1} from (5.12a).

5.3.2.3 Intermediate stage with an equality constraint

At the intermediate stage with an equality constraint ($i = k - 2$), we first define (5.17a)–(5.17e) for $i = k - 2$ for the specific values of P_{k-1} and s_{k-1} that satisfies (5.15). We then have the relations that are used in the forward recursion for $k - 1$ as follows:

$$\begin{bmatrix} \Delta u_{k-2} \\ \Delta \nu \end{bmatrix} = \begin{bmatrix} K_{k-2} \\ M \end{bmatrix} \Delta x_{k-2} + \begin{bmatrix} k_{k-2} \\ m \end{bmatrix}, \tag{5.18a}$$

where we define

$$\begin{bmatrix} K_{k-2} \\ M \end{bmatrix} := - \begin{bmatrix} G_{k-2} & D^{\mathrm{T}} \\ D & O \end{bmatrix}^{-1} \begin{bmatrix} H_{k-2}^{\mathrm{T}} \\ C \end{bmatrix} \tag{5.18b}$$

and

$$\begin{bmatrix} k_{k-2} \\ m \end{bmatrix} := - \begin{bmatrix} G_{k-2} & D^{\mathrm{T}} \\ D & O \end{bmatrix}^{-1} \begin{bmatrix} B_{k-2}^{\mathrm{T}} P_{k-1} \bar{x}_{k-2} - B_{k-2}^{\mathrm{T}} s_{k-1} + \bar{l}_{u,k-2} \\ \bar{\phi} \end{bmatrix}. \tag{5.18c}$$

We then obtain the backward recursions

$$P_{k-2} := F_{k-2} - \begin{bmatrix} K_{k-2}^{\mathrm{T}} & M^{\mathrm{T}} \end{bmatrix} \begin{bmatrix} G_{k-2} & D^{\mathrm{T}} \\ D & O \end{bmatrix} \begin{bmatrix} K_{k-2} \\ M \end{bmatrix} \tag{5.18d}$$

and

$$s_{k-2} := A_{k-2}^{\mathrm{T}}(s_{k-1} - P_{k-1}\bar{x}_{k-2}) - \bar{l}_{x,k-2} - H_{k-2}k_{k-2} - C^{\mathrm{T}}m. \tag{5.18e}$$

5.3.3 Algorithm, convergence, and computational analysis

We summarize the single Newton iteration, that is, the computation of the Newton direction for a particular solution, using the proposed Riccati recursion algorithm (Algorithm 5.1). In the first step, we formulate the linear equations of Newton's method, that is, we compute the coefficient matrices and residuals of (5.11)–(5.14) (line 1). This step can leverage parallel computing. Second, we perform the backward Riccati recursion and compute P_i and z_i for $i \in \{0, ..., N\}$ (lines 4–11). Third, we perform the forward Riccati recursion and compute the Newton directions for all the variables (lines 12–20).

Because the proposed method is in substance a Newton's method for an optimization problem with equality constraints, it achieves superlinear or quadratic convergence, for example, by Proposition 4.4.3 of Bertsekas (2016), which distinguishes the convergence behavior of the proposed method from that of the AL method, popularly used to treat the pure-state equality constraints with the Riccati recursion algorithm. The AL method achieves superlinear convergence only if the penalty parameter goes to infinity, which is an impractical assumption; otherwise, its convergence rate is just linear.

It should be noted that we can trivially apply the proposed method to OCPs with multiple pure-state equality constraints on the horizon. When there are multiple time stages involving constraint (5.1c) on the horizon, we compute the coefficient matrices and residuals in (5.13a)–(5.13c) for each of the time stages in line 1 of Algorithm 5.1, apply line 7 of Algorithm 5.1 for each of the time stages in the backward Riccati recursion, and apply line 15 of Algorithm 5.1 for each of the time stages in the forward Riccati recursion.

The proposed method is particularly efficient when there are several stages with pure-state equality constraints on the horizon. Suppose that there is an $n_{c,i}$-dimensional pure-state equality constraint at each time stage i of the horizon ($n_{c,i} = 0$ if there is no constraint at stage i). The proposed method then computes the inverse of a matrix whose size is $(n_u + n_{c,i}) \times (n_u + n_{c,i})$ at each time stage in the backward recursion. In contrast, the previous approach of Sideris and Rodriguez (2011) requires

Algorithm 5.1 Computation of Newton direction using proposed Riccati recursion

```
Input: Initial state $x(t_0)$, the current solution $x_0, ..., x_N$, $u_0, ..., u_{N-1}$, and Lagrange
    multipliers $\lambda_0, ..., \lambda_N$, $\nu$.
Output: Newton directions $\Delta x_0$, ..., $\Delta x_N$, $\Delta u_0$, ..., $\Delta u_{N-1}$, $\Delta\lambda_0$, ..., $\Delta\lambda_N$, and $\Delta\nu$.
 1: for $i = 0, \cdots, N$ do in parallel
 2:     Computes the matrices and residuals in (5.11)–(5.14).
 3: end for
 4: Compute $P_N$ and $z_N$ from (5.16).
 5: for $i = N, \cdots, 0$ do in serial
 6:     if $i = k - 2$ then
 7:         Computes $P_{k-2}$ and $z_{k-2}$ from (5.17a)–(5.17c), (5.18b), (5.18c), and
    (5.18e).
 8:     else
 9:         Computes $P_i$ and $z_i$ from (5.17a)–(5.17e).
10:     end if
11: end for
12: Compute $\Delta x_0$ from (5.14).
13: for $i = 0, \cdots, N - 1$ do in serial
14:     if $i = k - 2$ then
15:         Compute $\Delta u_{k-2}$, $\Delta\nu$, $\Delta\lambda_{k-2}$, and $\Delta x_{k-1}$ from (5.18a), (5.15), and (5.12a).
16:     else
17:         Compute $\Delta u_i$, $\Delta\lambda_i$, and $\Delta x_{i+1}$ from (5.17f), (5.15), and (5.12a).
18:     end if
19: end for
20: Compute $\Delta\lambda_N$ from (5.15).
```

the computation of the inverse of a matrix of size $(\sum_{i=0}^{N} n_{c,i}) \times (\sum_{i=0}^{N} n_{c,i})$. Broadly speaking, the computational burden of the proposed method with respect to the grid number N is $O(N)$, whereas that of the approach in Sideris and Rodriguez (2011) is $O(N^3)$.

It should be noted that it is easy to apply the proposed method to the single-shooting methods, which are popular in robotic applications (Farshidian et al. (2017); Koenemann et al. (2015); Li and Wensing (2020); Neunert et al. (2018)), by considering only $u_0, ..., u_{N-1}$ and ν as the decision variables. In the single-shooting methods, before line 1 of Algorithm 1, we first compute $x_0, ..., x_N$ based on $x(t_0)$ and $u_0, ..., u_{N-1}$ using the state equation (5.1b) sequentially. Further, we compute $\lambda_N, ..., \lambda_0$ using (5.6), (5.7), and (5.9) based on $u_0, ..., u_{N-1}$, ν, and $x_0, ..., x_N$, respectively, in the backward recursion (lines 5–11 of Algorithm 1). We can then compute the Newton directions $\Delta u_0, ..., \Delta u_{N-1}$ and $\Delta\nu$ using the same (or a similar) forward recursion (lines 12–20 of Algorithm 1).

5.4 Theoretical Properties of Optimal Control Problem Transformation

We show the theoretical relationships between the transformed OCP and the original OCP in this section. The first theorem concerns a stationary point of the transformed OCP and a stationary point of the original OCP.

Theorem 5.1. *Suppose that $x_0, ..., x_N$, $u_0, ..., u_{N-1}$, $\lambda_0, ..., \lambda_N$, and ν satisfy the FONC of the transformed OCP. Then, there exist the Lagrange multipliers $\lambda_0^*, ..., \lambda_N^*$ and ν^* such that $x_0, ..., x_N$, $u_0, ..., u_{N-1}$, $\lambda_0^*, ..., \lambda_N^*$, and ν^* satisfy the FONC of the original OCP.*

Proof. First, we define the intermediate time stages without the active constraints of the original OCP as $\tilde{I} := \{0, ..., k-1, k+1, ..., N-1\}$. The FONC of the original OCP is then expressed by (5.1b)–(5.1d), (5.6), and (5.7) for $i \in \tilde{I}$,

$$H_x^{\mathrm{T}}(x_k, u_k, \lambda_{k+1}^*) + \phi_x^{\mathrm{T}} \nu^* + \lambda_{k+1}^* - \lambda_k^* = 0, \tag{5.19}$$

and (5.8) for $i \in \{0, ..., N-1\}$, in which (5.1b)–(5.1d) are trivially satisfied because $x_0, ..., x_N$ and $u_0, ..., u_{N-1}$ satisfy the FONC of the transformed OCP. It should be noted that because $x_0, ..., x_N$ and $u_0, ..., u_{N-1}$ satisfy (5.1b) and $\phi_x(\cdot)$ only depends on the generalized coordinate,

$$\phi_x(x_k) = \phi_x(x_{k-2} + f(x_{k-2}, u_{k-2}) + g(x_{k-2} + f(x_{k-2}, u_{k-2}))) \tag{5.20}$$

holds. Therefore, we describe both the left- and right-hand sides of (5.20) as ϕ_x in this proof. Let

$$\lambda_i^* = \lambda_i, \ \ i \in \{0, ..., k-2, k+1, ..., N\}. \tag{5.21}$$

Then, (5.6), (5.7) for $i \in \{0, ..., k-3, k+1, ..., N\}$, and (5.8) for $i \in \{0, ..., k-2, k+1, ..., N\}$ of the original OCP are reduced to those of the transformed OCP and are, therefore, satisfied. Furthermore, let

$$\nu^* = \nu, \ \lambda_k^* = \lambda_k + \phi_x^{\mathrm{T}} \nu^*, \ \lambda_{k-1}^* = \lambda_{k-1} + (I + g_x^{\mathrm{T}}) \phi_x^{\mathrm{T}} \nu^*. \tag{5.22}$$

Then, (5.7) and (5.8) for $i = k-2, k-1$ and (5.19) of the original OCP are also reduced to (5.9), (5.10), (5.7) and (5.8) for $i = k-1$, and (5.7) for $i = k$ of the transformed OCP, respectively, noting that $\phi_x f_x(x_{k-1}, u_{k-1}) = \phi_x g_x$ and $\phi_x f_u(x_{k-1}, u_{k-1}) = O$, and are, therefore, satisfied, which completes the proof. □

From the proof of Theorem 4.1, we can obtain the Lagrange multipliers at a stationary point of the original OCP corresponding to those of the transformed OCP. The following theorem concerns the sufficiency of the optimality.

Theorem 5.2. *Suppose that $x_0, ..., x_N$, $u_0, ..., u_{N-1}$, $\lambda_0, ..., \lambda_N$, and ν satisfy the SOSC of the transformed OCP. Then, there exist the Lagrange multipliers $\lambda_0^*, ..., \lambda_N^*$, and ν^* such that $x_0, ..., x_N$, $u_0, ..., u_{N-1}$, $\lambda_0^*, ..., \lambda_N^*$, and ν^* satisfy the SOSC of the original OCP.*

Proof. Because $x_0, ..., x_N$, $u_0, ..., u_{N-1}$, $\lambda_0, ..., \lambda_N$, and ν also satisfy the FONC of the transformed OCP, we have $\lambda_0^*, ..., \lambda_N^*$ and ν^* defined by (5.21) and (5.22) such that $x_0, ..., x_N$, $u_0, ..., u_{N-1}$, $\lambda_0^*, ..., \lambda_N^*$, and ν^* satisfy the FONC of the original OCP from Theorem 4.1. From the assumption of the SOSC of the transformed OCP, we have

$$\delta x_N^{\mathrm{T}} Q_{xx,N} \delta x_N + \sum_{i=0}^{N-1} \begin{bmatrix} \delta x_i^{\mathrm{T}} \\ \delta u_i^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} Q_{xx,i} & Q_{xu,i} \\ Q_{ux,i}^{\mathrm{T}} & Q_{uu,i} \end{bmatrix} \begin{bmatrix} \delta x_i \\ \delta u_i \end{bmatrix} > 0 \tag{5.23}$$

for arbitrary δx_i and δu_i satisfying $\delta x_0 = 0$, $(I + f_{x,i})\delta x_i + f_{u,i}\delta u_i - \delta x_{i+1} = 0$ for $i \in \{0, ..., N-1\}$ and $\phi_x (I + g_x)(I + f_{x,k-2})\delta x_{k-2} + \phi_x (I + g_x) f_{u,k-2} \delta u_{k-2} = 0$, where we describe $f_{x,i} := f_x(x_i, u_i)$ and $f_{u,i} := f_u(x_i, u_i)$ for $i \in \{0, ..., N-1\}$. We introduce the Hessians of the original OCP as $Q_{xx,N}^* := V_{xx}(x_N) = Q_{xx,N}$ and $Q_{xx,i}^* := H_{xx}(x_i, u_i, \lambda_{i+1}^*)$, $Q_{xu,i}^* := H_{xu}(x_i, u_i, \lambda_{i+1}^*)$, and $Q_{uu,i}^* := H_{uu}(x_i, u_i, \lambda_{i+1}^*)$ for $i \in \tilde{I}$. Further, we introduce $Q_{xx,k}^* := H_{xx}(x_k, u_k, \lambda_{k+1}^*, \nu^*)$, $Q_{xu,k}^* := H_{xu}(x_k, u_k, \lambda_{k+1}^*, \nu^*)$, and $Q_{uu,k}^* := H_{uu}(x_k, u_k, \lambda_{k+1}^*, \nu^*)$. We can then complete the proof if

$$\delta x_N^{*\mathrm{T}} Q_{xx,N}^* \delta x_N^* + \sum_{i=0}^{N-1} \begin{bmatrix} \delta x_i^{*\mathrm{T}} \\ \delta u_i^{*\mathrm{T}} \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} Q_{xx,i}^* & Q_{xu,i}^* \\ Q_{ux,i}^{*\mathrm{T}} & Q_{uu,i}^* \end{bmatrix} \begin{bmatrix} \delta x_i^* \\ \delta u_i^* \end{bmatrix} > 0 \tag{5.24}$$

holds for arbitrary δx_i^* and δu_i^* satisfying $\delta x_0^* = 0$, $(I + f_{x,i})\delta x_i^* + f_{u,i}\delta u_i^* - \delta x_{i+1}^* = 0$ for $i \in \{0, ..., N-1\}$ and $\phi_x \delta x_k^* = 0$. First, we can see that the subspace of the feasible variations δx_i^* and δu_i^* is identical to that of δx_i and δu_i because (5.20) holds. Then, we consider δx_i^* and δu_i^* as being identical to δx_i and δu_i. Next, by substituting (5.21) and (5.22) into the Hessians of the original OCP, we obtain $Q_{xx,i}^* = Q_{xx,i}$, $Q_{xu,i}^* = Q_{xu,i}$, and $Q_{uu,i}^* = Q_{uu,i}$ for $i \in \{0, ..., k-3, k+1, ..., N-1\}$, $Q_{xx,k}^* = Q_{xx,k} + \nu \cdot \phi_{xx}$, $Q_{xu,k}^* = Q_{xu,k}$, $Q_{uu,k}^* = Q_{uu,k}$, $Q_{xx,k-1}^* = Q_{xx,k-1} + (\phi_x^{\mathrm{T}} \nu) \cdot f_{xx,k-1}$, $Q_{xu,k-1}^* = Q_{xu,k-1} + (\phi_x^{\mathrm{T}} \nu) \cdot f_{xu,k-1}$, $Q_{uu,k-1}^* = Q_{uu,k-1} + (\phi_x^{\mathrm{T}} \nu) \cdot f_{uu,k-1}$,

$$\begin{aligned} Q_{xx,k-2}^* = Q_{xx,k-2} &- (I + f_{x,k-2}^{\mathrm{T}})((\phi_x^{\mathrm{T}} \nu) \cdot g_{xx})(I + f_{x,k-2}) \\ &- (I + f_{x,k-2}^{\mathrm{T}})(I + g_x^{\mathrm{T}})(\nu \cdot \phi_{xx})(I + g_x)(I + f_{x,k-2}), \end{aligned}$$

$$\begin{aligned}Q^*_{xu,k-2} &= Q_{xu,k-2} - (I + f^{\mathrm{T}}_{x,k-2})((\phi^{\mathrm{T}}_x \nu) \cdot g_{xx}) f_{u,k-2} \\ &\quad - (I + f^{\mathrm{T}}_{x,k-2})(I + g^{\mathrm{T}}_x)(\nu \cdot \phi_{xx})(I + g_x) f_{u,k-2},\end{aligned}$$

and

$$Q^*_{uu,k-2} = Q_{uu,k-2} - f^{\mathrm{T}}_{u,k-2}(I + g^{\mathrm{T}}_x)(\nu \cdot \phi_{xx})(I + g_x) f_{u,k-2},$$

where $f_{xx,i} := f_{xx}(x_i, u_i)$, $f_{xu,i} := f_{xu}(x_i, u_i)$, $f_{uu,i} := f_{uu}(x_i, u_i)$, and the notation "$\cdot$" denotes vector–tensor multiplication. Since the FONC of the original OCP holds and $\nu \cdot \phi_{xx} = \begin{bmatrix} \nu \cdot \phi_{qq} & O \\ O & O \end{bmatrix}$, we have $(\nu \cdot \phi_{xx})(I + f_{x,k-1}) = (\nu \cdot \phi_{xx})(I + g_x)$ and $(\nu \cdot \phi_{xx}) f_{u,k-1} = O$, which yields $\delta x^{\mathrm{T}}_k (\nu \cdot \phi_{xx}) \delta x_k = \delta x^{\mathrm{T}}_{k-1} (I + g^{\mathrm{T}}_x)(\nu \cdot \phi_{xx})(I + g_x) \delta x_{k-1}$. In addition, from the structures of (5.2), (5.3), and (5.5b), we have $(\phi^{\mathrm{T}}_x \nu) \cdot f_{xx,k-1} = (\phi^{\mathrm{T}}_x \nu) \cdot g_{xx}$, $(\phi^{\mathrm{T}}_x \nu) \cdot f_{xu,k-1} = O$, and $(\phi^{\mathrm{T}}_x \nu) \cdot f_{uu,k-1} = O$. By substituting these relations, the left-hand side of (5.24) is reduced to the left-hand side of (5.23), which completes the proof. □

We summarize the property of the proposed transformation in the next proposition:

Proposition 5.3. *Suppose that $x_0, ..., x_N$, $u_0, ..., u_{N-1}$, $\lambda_0, ..., \lambda_N$, and ν satisfy the SOSC of the transformed OCP. Then, the solution $x_0, ..., x_N$, $u_0, ..., u_{N-1}$ is a strict local minimum of the original OCP.*

Proof. Because the solution $x_0, ..., x_N$, $u_0, ..., u_{N-1}$ with the Lagrange multipliers satisfies the SOSC of the original OCP, as indicated by Theorem 4.2, the solution $x_0, ..., x_N$, $u_0, ..., u_{N-1}$ is a strict local minimum of the original OCP. □

5.5 Numerical Experiments on Whole-Body Quadrupedal Gaits Optimization

5.5.1 Experimental settings

To demonstrate the effectiveness of the proposed method over existing methods, we conducted numerical experiments on the whole-body optimal control of a quadrupedal robot ANYmal for various gaits. The equation of motion of the full 3D model of the quadrupedal robot is of the form of (5.2). Moreover, a pure-state constraint whose Jacobian is of the form (5.3) is imposed just before each impact between the leg and the ground, which is termed the switching constraint (Farshidian et al. (2017); Li and Wensing (2020)). We compare the following three Riccati recursion algorithms based on the direct-multiple shooting method and Gauss-Newton Hessian approximation with various constraint handling methods:

- The proposed method
- The Riccati recursion with pure-state constraints (Sideris and Rodriguez (2011))
- The AL method (Nocedal and Wright (2006))

We implemented these three algorithms in C++ and used Pinocchio (Carpentier et al. (2019)), an efficient C++ library used for rigid-body dynamics and its analytical derivatives, to compute the dynamics and its derivatives of the quadrupedal robot. We used OpenMP (Dagum and Menon (1998)) for parallel computing (e.g., line 1 of Algorithm 1) and four threads through the following experiments. To consider the practical situation, we also imposed inequality constraints on the joint angle limits, joint angular velocity limits, and joint torque limits of each joint. We used the primal-dual interior point method (Wächter and Biegler (2006)) with fixed barrier parameters for the inequality constraints. None of the three methods used line search; they only used the fraction-to-boundary rule (Wächter and Biegler (2006)) for step-size selection. We fixed the instants of the impact between the robot and the ground in the following experiments and did not treat them as optimization variables as in Farshidian et al. (2017); Li and Wensing (2020), to focus on the evaluation of the constraint-handling methods. All experiments were conducted on a laptop with a hexa-core CPU Intel Core i9-8950HK @2.90 GHz.

In the following two experiments, we considered that the OCP converges when the l_2-norm of the residuals in the Karush—Kuhn–Tucker (KKT) conditions, which we refer to as the KKT error, becomes smaller than a prespecified threshold. The KKT conditions are composed of the FONC and primal and dual residuals in the inequality constraints. For example, the KKT conditions of the proposed method are composed of (5.1b), (5.1d), (5.5a), (5.6)–(5.10), and the residuals in the inequality constraints. The KKT conditions of the Riccati recursion with pure-state constraints Sideris and Rodriguez (2011) and the AL method are only slightly different depending on the method used to treat pure-state equality constraints.

5.5.2 Trotting gait for different numbers of steps

First, we evaluated the performances of the three methods for different total dimensions of pure-state equality constraints by considering the trotting gaits of ANYmal with different numbers of trotting steps. A six-dimensional (three-dimensional for

Table 5.1: Settings of OCPs for each number of trotting steps

No. of trotting steps	2	4	6	8	10
Horizon length T	1.55	2.55	3.55	4.55	5.55
No. of grids N	35	59	83	107	131
Total dim. of (5.1c)	12	24	36	48	60

each impact leg) pure-state equality constraint (switching constraint) was imposed on the OCPs for each trotting step. We chose the number of trotting steps from 2, 4, 6, 8, and 10 and measured the CPU time per Newton iteration and the total CPU time until convergence (we chose 1.0×10^{-10} as the convergence tolerance of the KKT errors). The settings used for the OCPs (horizon length T, number of grids N, and total dimension of equality constraints (5.1c)) are listed in Table I. We carefully tuned the parameters of the AL method (Nocedal and Wright (2006)). For example, we chose the initial penalty parameter as $p = 5$ and the penalty update value as $\beta = 8$; that is, the AL method updates the penalty parameter as $p \leftarrow \beta p$ when the KKT error excluding the constraint violation (5.1c) is smaller than a tolerance that is also tuned carefully (Nocedal and Wright (2006)).

Figure 5.1 depicts the CPU time per Newton iteration (left figure) and the total CPU time until convergence (right figure) of each method. As depicted in the left figure of Fig. 5.1, the CPU time per Newton iteration in the proposed method was almost the same as that in the AL method, whereas the Riccati recursion with pure-state constraints (Sideris and Rodriguez (2011)) took more computational time when compared with the other two methods. The right figure of Fig. 5.1 also indicates that the proposed method achieved the fastest convergence.

The proposed method took exactly the same number of iterations (approximately 20) until convergence as the Riccati recursion with pure-state constraints (Sideris and Rodriguez (2011)) in all the cases. Therefore, the proposed method was faster than it in terms of the total CPU time as in the case of the per Newton iteration. The AL method was significantly slower than the other two methods with respect to the total CPU time because it required approximately 80 iterations in all the cases, although we carefully tuned the AL algorithm.

5.5.3 Trotting, jumping, and running gait problems

Next, we investigated the performances of the proposed method in three different quadrupedal gaits: trotting, jumping, and running gaits, of which the jumping and

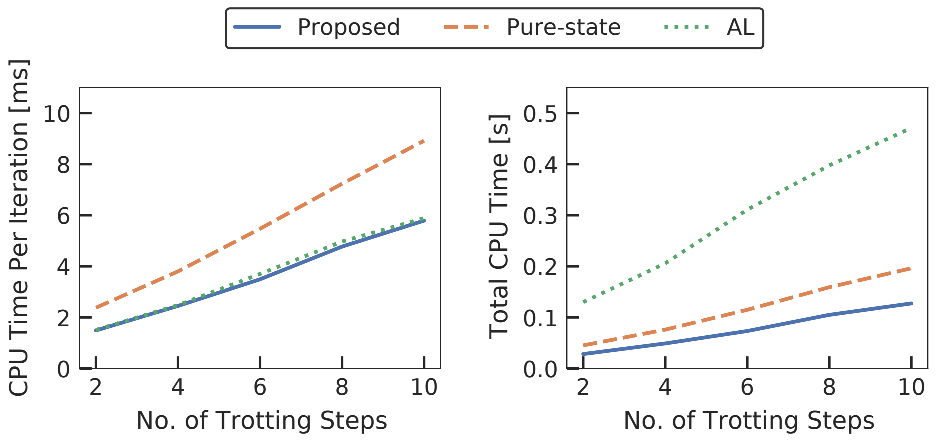


Figure 5.1: (Left) CPU time per Newton iteration and (right) total CPU time until convergence for different numbers of trotting steps in the proposed method (Proposed), Riccati recursion with pure-state constraints (Sideris and Rodriguez (2011)) (pure state), and the AL method (AL).

running gaits are particularly highly nonlinear and complicated problems. Each jumping step imposes a 12-dimensional pure-state equality constraint, and each running step imposes a 6-dimensional one. We summarize the settings of each problem (horizon length T, number of grids N, number of steps, total dimension of equality constraints (5.1c), tolerance of convergence, and initial penalty parameter of the AL method $p_{\rm init}$) in Table II. As done in the preceding example, we carefully tuned the parameters of the AL method, that is, $p_{\rm init}$ and the update rule of the penalty and the Lagrange multiplier, for each problem. We measured the KKT errors with respect to the number of iterations, and the total number of iterations and CPU time until convergence.

Figure 5.2 depicts the $\log_{10}$-scaled KKT error of each method for the three gait problems with respect to the number of iterations. We can see that the convergence behavior of the proposed method was almost the same as that of the Riccati recursion with pure-state constraints (Sideris and Rodriguez (2011)). In contrast, the AL method resulted in significantly slow convergence because it needs to update the penalty parameter and Lagrange multiplier to reduce the constraint violation, which we can see in the peaks in the KKT error of the AL method in Fig. 5.2. Figure 5.3 indicates the number of iterations and total CPU time until convergence of each method. We can see that the total number of iterations of the proposed method was almost the same as that of the Riccati recursion with pure-state constraints (Sideris and Rodriguez (2011)), whereas the AL method required a significantly large num-

Table 5.2: Settings of OCPs for trotting, jumping, and running gaits.

Gait type	Trotting	Jumping	Running
Horizon length T	6.05	5	7
No. of grids N	143	107	346
No. of steps	11	3	26
Total dim. of (5.1c)	66	36	156
KKT tolerance	1.0×10^{-10}	1.0×10^{-10}	1.0×10^{-8}
p_{init}	5	1000	5

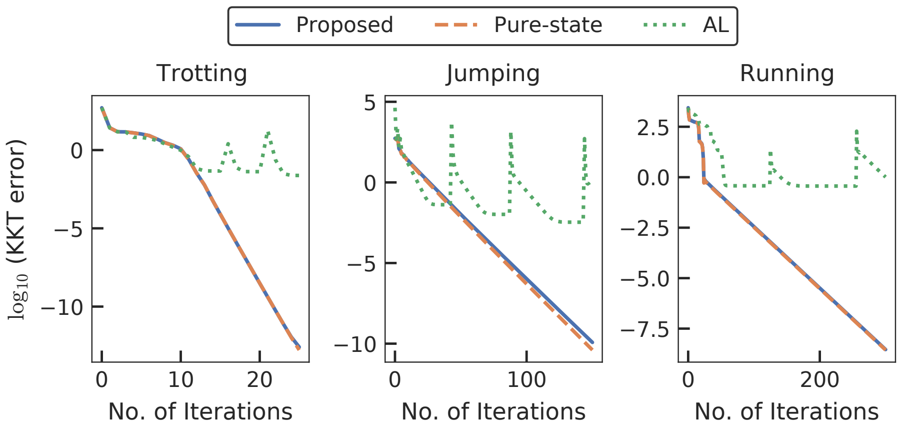


Figure 5.2: $\log_{10}$-scaled KKT errors of the proposed method (Proposed), the Riccati recursion with pure-state constraints (Sideris and Rodriguez (2011)) (Pure-state), and the AL method (AL) for the three gait problems with respect to the number of iterations. It should be noted that the KKT errors include the violations of the switching constraints. The graphs of the AL method have peaks when the penalty parameter and Lagrange multipliers are updated.

ber of iterations. In addition, as each iteration of the proposed method was faster than that of the Riccati recursion with pure-state constraints (Sideris and Rodriguez (2011)), as in the previous experiment, the proposed method achieved the fastest convergence.

A supplemental video including these gaits is available at https://youtu.be/uX1_58QvPUg.

5.6 Summary

We proposed a novel approach to efficiently treat pure-state equality constraints in OCPs with a Riccati recursion algorithm. The proposed method transforms a pure-state equality constraint into a mixed state-control constraint such that the constraint

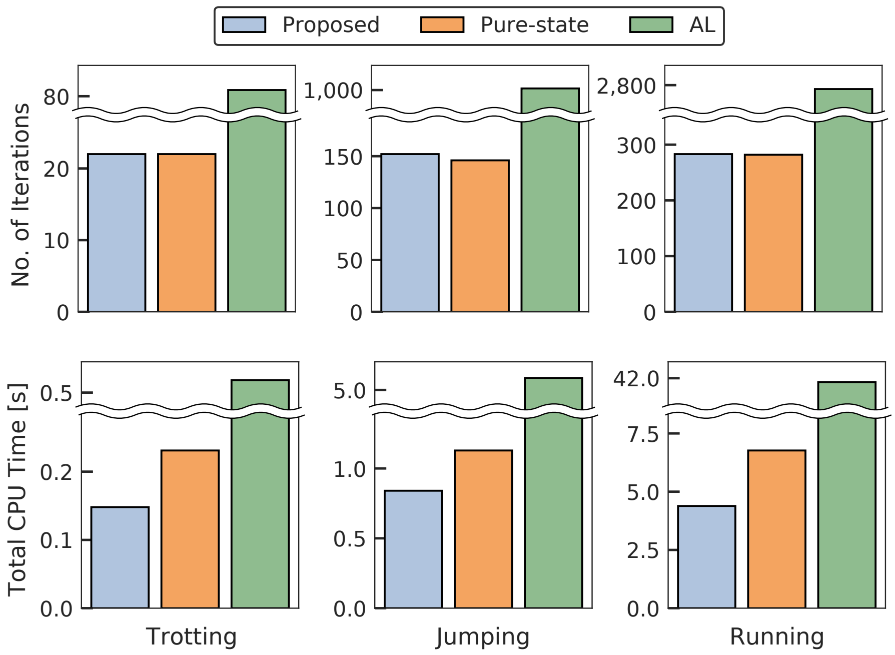


Figure 5.3: Number of iterations and total CPU time until convergence of the proposed method (Proposed), the Riccati recursion with pure-state constraints (Sideris and Rodriguez (2011)) (Pure-state), and the AL method (AL) for the three gaits problems.

is expressed by variables at a certain previous time stage. We derived a Riccati recursion algorithm to solve the transformed OCP with linear time complexity in the grid number of the horizon, in contrast to the previous approach (Sideris and Rodriguez (2011)), which scaled cubically with respect to the total dimension of the pure-state equality constraints. Because the proposed method is essentially a Newton's method for an optimization problem with equality constraints, the proposed method achieves superlinear or quadratic convergence, which distinguishes our approach from the penalty function method and the AL method in terms of the convergence property. We showed that if the solution satisfies the FONC and/or SOSC of the transformed OCP, then the solution also satisfies the FONC and/or SOSC of the original OCP. Therefore, if we find a solution that satisfies the SOSC of the transformed OCP, it is a local minimum of the original OCP. We performed numerical experiments on the whole-body optimal control of quadrupedal gaits that involve pure-state equality constraints owing to contact switches and demonstrated the effectiveness of the proposed method over the approach of Sideris and Rodriguez (2011) and the AL method.

Chapter 6

Structure-Exploiting Newton-Type Method for Optimal Control of Switched Systems

6.1 Introduction

Switched systems are a class of hybrid systems consisting of a finite number of subsystems and switching laws of active subsystems. Many practical control systems are modeled as switched systems, such as real-world complicated process systems (Bürger, Zeile, Altmann-Dieses, Sager, and Diehl (2019)), automotive systems with gear shifts (Robuschi, Zeile, Sager, and Braghin (2021)), and robotic systems with rigid contacts (Farshidian, Kamgarpour, et al. (2017); Li and Wensing (2020)). It is difficult to solve the optimal control problems (OCPs) of such systems because these OCPs generally involve mixed-integer nonlinear programs (MINLPs) (Belotti et al. (2013)). Therefore, model predictive control (MPC) (Rawlings et al. (2017)), in which OCPs must be solved in real-time, is particularly difficult for switched systems. One of the most practical approaches for solving the OCPs of switched systems is the combinatorial integral approximation (CIA) decomposition method (Bürger et al. (2019); Sager, Jung, and Kirches (2011)). CIA decomposition relaxes the MINLP into a nonlinear program (NLP) in which the binary variables are relaxed into continuous variables. Subsequently, the integer variables are approximately reconstructed from the relaxed NLP solution by solving a mixed-integer linear program. However, vanishing constraints, which typically model mode-dependent path constraints, raise numerical issues due to the violation of the linear independence constraint qualification (LICQ) (Jung, Kirches, and Sager (2013)). A similar smoothing approach is used for discrete natures when solving OCPs for mechanical systems with rigid contacts, which are often approximately formulated as mathematical programs with complementarity constraints (MPCCs) (Posa et al. (2014); Yunt (2011)). However, this case has the same LICQ problem as the vanishing constraints, requiring a great deal of computational time to alleviate the numerical ill-conditioning. Furthermore, it often suffers from undesirable stationary points (Nurkanović et al. (2020)).

Another tractable and practical approach for the optimal control of switched systems is to fix the switching (mode) sequence. For example, in robotics applications, a high-level planner computes the feasible contact sequence taking perception into account. Subsequently, the sequence is provided to a lower-level optimal control-based dynamic motion planner or MPC controller (Grandia, Taylor, Ames, and Hutter (2021); Jenelten et al. (2020); Kuindersma et al. (2016)). The lower-level dynamic planner or MPC then discovers the optimal switching times and optimal control input. An advantage of this approach over the CIA decomposition and MPCC approaches is that the optimization problem is smooth and can therefore be efficiently solved without suffering from the LICQ problem. For example, Johnson and Murphey (2011) and Stellato, Ober-Blöbaum, and Goulart (2017) proposed efficient Newton-type methods for OCPs of switched systems with autonomous subsystems. That is, the switched systems only included a switching signal but no continuous control input, which is called the switching-time optimization (STO) problem. The STO approach can be more efficient than the CIA decomposition because the STO problem can be formulated as a smooth and tractable NLP, as numerically shown in Stellato et al. (2017). However, once the switched system includes a continuous control input, the efficient methods of Johnson and Murphey (2011) and Stellato et al. (2017) cannot be applied. To the best of our knowledge, there is no efficient numerical method for OCPs of such systems. As a result, many real-world robotic applications are limited to focusing on dynamic motion planning with fixed switching instants (Di Carlo et al. (2018); Farshidian, Jelavic, et al. (2017); Mastalli et al. (2020)).

The two-stage approach has been studied for the OCPs of switched systems with continuous control input (Farshidian, Kamgarpour, et al. (2017); Fu and Zhang (2021); Li and Wensing (2020); Xu and Antsaklis (2002); Xu and Antsaklis (2004)). A general two-stage approach was proposed in Xu and Antsaklis (2002) and Xu and Antsaklis (2004). In this approach, the optimization problem to determine the control input and switching instants was decomposed into an STO problem with a fixed control input (upper-level problem) and standard OCP that only determined the control input with fixed switching instants (lower-level problem). In this framework, off-the-shelf OCP solvers can be used for a lower-level problem. Xu and Antsaklis

(2002) and Xu and Antsaklis (2004) used an indirect OCP solver to compute an accurate solution of a lower level problem. Farshidian, Kamgarpour, et al. (2017) and Li and Wensing (2020) formulated a lower-level problem as a direct OCP and solved it using off-the-shelf Newton-type methods. However, these studies still lack convergence speed because each of these two stages does not take the other stage into account when solving its own optimization problem. As a result, the application examples of Farshidian, Kamgarpour, et al. (2017) and Li and Wensing (2020) were limited to off-line computation for the trajectory optimization problems of simplified robot models. Moreover, they could not guarantee the local convergence or address inequality constraints. Fu and Zhang (2021) was the first study that guaranteed convergence with a finite number of iterations and treated inequality constraints within a two-stage framework. However, it still required extensive computational time until convergence, even for a very simple linear quadratic example.

Other approaches simultaneously optimized the switching instants and other variables, such as the state and control input (Betts (2010); Katayama et al. (2020); Katayama and Ohtsuka (2021b); Patterson and Rao (2014)). A multi-phase trajectory optimization Betts (2010); Patterson and Rao (2014) naturally incorporated the STO problem into the direct transcription, solving the NLP to simultaneously determine all variables (including the state, control input, and switching instants) using general-purpose off-the-shelf NLP solvers, such as Ipopt (Wächter and Biegler (2006)). However, the computational speed of general-purpose linear solvers used in off-the-shelf NLPs can typically be further improved, particularly for large-scale systems, because they have a certain sparsity structure (Rao et al. (1998); Wang and Boyd (2010); Zanelli et al. (2020)). In Katayama et al. (2020), we applied the simultaneous approach with the direct single-shooting method and achieved real-time MPC for a simple walking robot using the Newton-Krylov type method (Ohtsuka (2004)). In Katayama and Ohtsuka (2021b), we formulated an NLP using the direct multiple-shooting method (Bock and Plitt (1984)) and proposed a Riccati recursion algorithm for the NLP, which achieved faster computational time compared with the two-stage methods. However, these methods lacked the convergence guarantee due to the irregular discretization of the continuous-time OCP, which changed the problem structure throughout the Newton-type iterations. As a result, the method from Katayama and Ohtsuka (2021b) could only converge when the initial guess of the switching instants was close to the optimal one in the numerical example.

This chapter proposes an efficient Newton-type method for optimal control of switched systems under a given mode sequence. First, a mesh-refinement-based approach is proposed to discretize the continuous-time OCP using the direct multiple-shooting method (Bock and Plitt (1984)) to formulate an NLP that facilitates the local convergence of the Newton-type methods. Second, a dedicated efficient structure-exploiting algorithm (Riccati recursion algorithm (Rao et al. (1998))) is proposed for the Newton-type method because the sparsity structure of the NLP is different from the standard OCP. The proposed method computes each Newton step with linear time-complexity of the total number of the discretization grids as the standard Riccati recursion algorithm. Additionally, it can always solve the Karush-Kuhn-Tucker (KKT) systems arising in the Newton-type method if the solution is sufficiently close to a local minimum, so that the second-order sufficient condition (SOSC) holds. This is in contrast to some general quadratic programming (QP) solvers that cannot treat the proposed formulation because the Hessian matrix is inherently indefinite. Third, a modification on the reduced Hessian matrix is proposed to enhance the convergence using the nature of the Riccati recursion algorithm as the dynamic programming (Bertsekas (2005)) for a QP subproblem. Two numerical experiments are conducted to demonstrate the efficiency of the proposed method: a comparison with off-the-shelf NLP solvers and testing the whole-body optimal control of quadrupedal gaits. The comparison with off-the-shelf NLP solvers showed that the proposed method could solve the OCPs that the sequential quadratic programming (SQP) method with qpOASES (Ferreau et al. (2014)) or OSQP (Stellato et al. (2020)) failed to solve and was up to two orders of magnitude faster than a general NLP solver Ipopt (Wächter and Biegler (2006)). The whole-body optimal control of quadrupedal gaits showed that the proposed method achieved the whole-body MPC of robotic systems with rigid contacts.

The remainder of this chapter is organized as follows. A mesh-refinement-based discretization method of the continuous-time OCP is presented in Section 6.2. The KKT system to be solved to compute the Newton-step is discussed in Section 6.3 and the Riccati recursion algorithm to solve the KKT system and its convergence properties are described in Section 6.4. The reduced Hessian modification using the Riccati recursion algorithm is also described in this section. The above formulations and Riccati recursion algorithm are extended to switched systems with state jumps and switching conditions in Section 6.4, representing robotic systems with contacts. A numerical comparison of the proposed method with off-the-shelf NLP solvers and examples of whole-body optimal control of a quadrupedal robot are presented in Section 6.6. Finally, a brief summary and mention of future work are presented in Section 6.7.

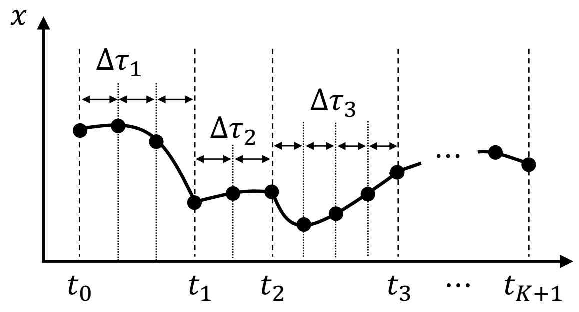


Figure 6.1: Proposed discretization method for the OCPs of switched systems.

Notation and preliminaries: The Jacobians and Hessians of a differentiable function using certain vectors are described as follows: $\nabla_x f(x)$ denotes $\left(\frac{\partial f}{\partial x}\right)^{\mathrm{T}}(x)$, and $\nabla_{xy} g(x, y)$ denotes $\frac{\partial^2 g}{\partial x \partial y}(x, y)$. A diagonal matrix whose elements are a vector x is denoted as $\mathrm{diag}(x)$. A vector of an appropriate size with all components presented by $\alpha \in \mathbb{R}$ is denoted as $\alpha 1$. All functions are assumed to be twice-differentiable.

6.2 Problem Formulation

We consider a switched system consisting of $K + 1$ $(K > 0)$ subsystems, which is expressed as:

$$\dot{x}(t) = f_k(x(t), u(t)), \quad t \in [t_{k-1}, t_k], \quad k \in \mathcal{K}. \tag{6.1}$$

The system also contains constraints, which are expressed as:

$$g_k(x(t), u(t)) \leq 0, \quad t \in [t_{k-1}, t_k], \quad k \in \mathcal{K}, \tag{6.2}$$

where $x(t) \in \mathbb{R}^{n_x}$ denotes the state, $u(t) \in \mathbb{R}^{n_u}$ denotes the control input, $f_k : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_x}$, and $g_k : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_g}$. $\mathcal{K} := \{1, ..., K + 1\}$ denotes the given indices of the active subsystems. Note that the index k is also referred to as a *phase* in this chapter. t_0 and t_{K+1} denote the fixed initial and terminal times of the horizon, respectively. t_k, $k \in \{1, ..., K\}$ denotes the switching instant from phase k to phase $k + 1$. The OCP of the switched system for a given initial state $x(t_0) \in \mathbb{R}^{n_x}$ is expressed as:

$$\min_{u(\cdot), t_1, ..., t_K} J = V_f(x(t_{K+1})) + \sum_{k=1}^{K+1} \int_{t_{k-1}}^{t_k} l_k(x(\tau), u(\tau)) d\tau \tag{6.3a}$$

$$\begin{aligned} \text{s.t.} \quad & (6.1),\ (6.2), \\ & t_{k-1} + \underline{\Delta}_k \leq t_k, \quad k \in \{1, ..., K\}, \end{aligned} \tag{6.3b}$$

where $\underline{\Delta}_k \geq 0,\ k \in \mathcal{K}$ is the minimum dwell-time.

Next, a discretization method and mesh-refinement-based solution approach are proposed for the OCP to guarantee the local convergence. The OCP is discretized with the direct multiple-shooting method (Bock and Plitt (1984)) based on the forward Euler method so that all the time-steps in phase $k \in \mathcal{K}$ are equal. Note that it is easy to extend the proposed method for higher-order explicit integration schemes to the direct multiple-shooting method, such as the fourth-order explicit Runge-Kutta method, as long as the Riccati recursion algorithm can be applied to the integration schemes. We introduce N grid points over the horizon, the discretized state $X := \{x_0, ..., x_N\}$, discretized control input $U := \{u_0, ..., u_{N-1}\}$, and switching instants $T := \{t_1, ..., t_K\}$. The NLP is then expressed as:

$$\min_{X,U,T} J = V_f(x_N) + \sum_{k \in \mathcal{K}} \sum_{i \in \mathcal{I}_k} l_k(x_i, u_i) \Delta\tau_k \tag{6.4a}$$

$$\begin{aligned} \text{s.t.} \quad & x_0 - \bar{x} = 0, && (6.4b) \\ & x_i + f_k(x_i, u_i)\Delta\tau_k - x_{i+1} = 0, \quad i \in \mathcal{I}_k,\ k \in \mathcal{K}, && (6.4c) \\ & g_k(x_i, u_i) \leq 0, \quad i \in \mathcal{I}_k,\ k \in \mathcal{K}, && (6.4d) \\ & t_{k-1} + \underline{\Delta}_k - t_k \leq 0, \quad k \in \{1, ..., K\}, && (6.4e) \end{aligned}$$

where $\mathcal{I}_k$ is the set of stage indices at phase k (that is, the set of time stages where the subsystem equation $f_k(x_i, u_i)$ is active). $\Delta\tau_k$ is the time step at phase k, defined as:

$$\Delta\tau_k := \frac{t_k - t_{k-1}}{N_k}, \quad k \in \{1, ..., K\}. \tag{6.4f}$$

Note that the last stage N is not included in any $\mathcal{I}_k$ and that the initial state condition is lifted as (6.4b) for an efficient MPC implementation (Diehl et al. (2005)).

The continuous-time OCP (6.3) is solved using an adaptive mesh-refinement approach (Betts (2010)), which consists of a solution of the NLP (6.4) using a Newton-type method and mesh-refinement due to the changes of $\Delta\tau_k$, as shown in Algorithm 6.1. After solving the NLP, the size of the discretization steps $\Delta\tau_k$ is checked for each $k \in \mathcal{K}$. If the step is too large, that is, if it exceeds the specified threshold $\Delta\tau_{\max}$, the solution at phase k may be not accurate. Therefore, the number of grids for phase k is increased. Conversely, if the step is too small ($\Delta\tau < \Delta\tau_{\min}$), the number of grids for phase k is reduced to decrease the computational time of the next NLP step. The algorithm terminates if some criteria (for example, l_2 norm of the KKT residual), which is denoted as "NLP error" in Algorithm 6.1, is smaller than a predefined threshold $\epsilon > 0$ and the discretization steps pass the checks. By appropriately

Algorithm 6.1 Adaptive mesh-refinement approach for the optimal control of switched systems (6.3)

```
Input: The initial state $x(t_0)$, initial guess of the solution $X$, $U$, $T$, initial guess
    of the Lagrange multipliers, maximum and minimum discretization step sizes
    $\Delta\tau_{\max} > \Delta\tau_{\min} > 0$, and convergence tolerance $\epsilon > 0$.
Output: Optimal solution $X$, $U$, $T$.
1: while NLP error $> \epsilon$ do
2:     if $\Delta\tau_k < \Delta\tau_{\min}$ then
3:         Mesh-refinement for phase $k$ (decrease grids).
4:     end if
5:     Solve the NLP (6.4) using the Newton-type method.
6:     if $\Delta\tau_k > \Delta\tau_{\max}$ then
7:         Mesh-refinement for phase $k$ (increase grids).
8:     end if
9: end while
```

choosing $\Delta\tau_{\min}$, the NLP dimension is almost constant throughout Algorithm 6.1. This is an advantage of the proposed method over direct transcription methods, such as Patterson and Rao (2014), whose NLP dimension is unknown before being solved.

It is worth noting that the NLP (6.4) is smooth and its structure does not change within each NLP step. Therefore, the Newton-type method for the NLP (6.4) is always tractable.

Remark 6.1. *It is trivial to show the local convergence of Newton-type methods for the NLP (6.4) under some reasonable assumptions (Nocedal and Wright (2006)). Moreover, if the solution guess after the mesh-refinement of Algorithm 6.1 is sufficiently close to a local minimum of the NLP after the mesh-refinement (that is, the mesh-refinement is sufficiently accurate), then the local convergence of the overall Algorithm 6.1 is also guaranteed. In contrast, previous methods cannot guarantee the local convergence, such as the two-stage approaches (Farshidian, Kamgarpour, et al. (2017); Li and Wensing (2020); Xu and Antsaklis (2002); Xu and Antsaklis (2004)) and simultaneous approaches with the other discretization methods (Katayama et al. (2020); Katayama and Ohtsuka (2021b)).*

Note that it is assumed throughout this chapter that the state equations (6.1), inequality constraints (6.2), and cost function (6.3a) do not depend on the time. That is, they are time invariant for notational simplicity. However, it is easy to extend the proposed method to time-varying cases, where the time of each grid i that depends on the switching times $t_1, ..., t_K$ in (6.1), (6.2), and (6.3a) must be taken into account. Therefore, additional sensitivities of (6.1), (6.2), and (6.3a) with respect to the switching times are introduced in the KKT conditions and systems, which are derived in the next section. The formulations and methods of this chapter can be directly applied because such additional sensitivities do not change the NLP structure.

6.3 KKT System for Newton-Type Method

Next, the KKT system is derived to compute the Newton step of the NLP (6.4). The inequality constraints are treated with the primal-dual interior point method (Nocedal and Wright (2006); Wächter and Biegler (2006)). That is, the slack variables $z_0, ..., z_{N-1} \in \mathbb{R}^{n_g}$ and $w_1, ..., w_K \in \mathbb{R}$ are introduced for (6.4d) and (6.4e), respectively. The equality constraints are then considered, which are expressed as:

$$r_{g,i} := g_k(x_i, u_i) + z_i = 0, \;\; i \in \mathcal{I}_k, \; k \in \mathcal{K}, \tag{6.5a}$$

$$r_{\Delta,k} := t_{k-1} + \underline{\Delta}_k - t_k + w_k = 0, \;\; k \in \{1, ..., K\} \tag{6.5b}$$

instead of the inequality constraints (6.4d) and (6.4e). The barrier functions $-\epsilon \sum_{i=0}^{N-1} \ln z_i - \epsilon \sum_{k=1}^{K} \ln w_k$ are also added to the cost function (6.4a), where $\epsilon > 0$ is the barrier parameter. The perturbed KKT conditions (Nocedal and Wright (2006)) are obtained by introducing the Lagrange multipliers. $\lambda_0, ..., \lambda_N \in \mathbb{R}^{n_x}$ are introduced as the Lagrange multipliers with respect to (6.4b) and (6.4c), $\nu_0, ..., \nu_{N-1} \in \mathbb{R}^{n_g}$ are with respect to (6.5a), and $\upsilon_0, ..., \upsilon_{K+1} \in \mathbb{R}$ are with respect to (6.5b). The perturbed KKT conditions are expressed as:

$$r_{x,N} := \nabla_x V_f(x_N) - \lambda_N = 0, \tag{6.6a}$$

$$r_{x,i} := \nabla_x H_k(x_i, u_i, \lambda_{i+1}) \Delta\tau_k + \nabla_x g_k^{\mathrm{T}}(x_i, u_i) \nu_i + \lambda_{i+1} - \lambda_i = 0, \quad i \in \mathcal{I}_k, \, k \in \mathcal{K}, \tag{6.6b}$$

$$r_{u,i} := \nabla_u H_k(x_i, u_i, \lambda_{i+1}) \Delta\tau_k + \nabla_u g_k^{\mathrm{T}}(x_i, u_i) \nu_i = 0, i \in \mathcal{I}_k, \, k \in \mathcal{K}, \tag{6.6c}$$

$$r_{z,i} := \mathrm{diag}(z_i) \nu_i - \epsilon 1 = 0, \quad i \in \mathcal{I}_k, \; k \in \mathcal{K}, \tag{6.6d}$$

$$r_{w,k} := w_k \upsilon_k - \epsilon = 0, \quad k \in \mathcal{K}, \tag{6.6e}$$

and

$$\frac{1}{N_k} \sum_{i \in \mathcal{I}_k} H_k(x_i, u_i, \lambda_{i+1}) - \frac{1}{N_{k+1}} \sum_{i \in \mathcal{I}_{k+1}} H_{k+1}(x_i, u_i, \lambda_{i+1}) + \upsilon_k - \upsilon_{k+1} = 0, \quad k \in \{2, ..., K+1\}, \tag{6.6f}$$

where

$$H_k(x_i, u_i, \lambda_{i+1}) := l_k(x_i, u_i) + \lambda_{i+1}^{\mathrm{T}} f_k(x_i, u_i) \tag{6.6g}$$

is the Hamiltonian at phase $k \in \mathcal{K}$. Next, the KKT system is derived to compute the Newton steps of all variables, that is, $\Delta x_0, ..., \Delta x_N \in \mathbb{R}^{n_x}$, $\Delta u_0, ..., \Delta u_{N-1} \in \mathbb{R}^{n_u}$ $\Delta t_1, ..., \Delta t_K \in \mathbb{R}$, and $\Delta \lambda_0, ..., \Delta \lambda_N \in \mathbb{R}^{n_x}$. Note that the KKT system is herein considered as the standard primal-dual interior point method (Nocedal and Wright (2006); Wächter and Biegler (2006)), in which the Newton directions regarding the inequality constraints are eliminated (that is, $\Delta z_0, ..., z_{N-1}, \Delta \nu_0, ..., \Delta \nu_{N-1} \in \mathbb{R}^{n_g}$ and $\Delta w_0, ..., \Delta w_{K+1}, \Delta \upsilon_0, ..., \Delta \upsilon_{K+1} \in \mathbb{R}$). The KKT system of interest is then expressed as:

$$\Delta x_0 + x_0 - \bar{x} = 0, \tag{6.7a}$$

$$A_i \Delta x_i + B_i \Delta u_i + f_i (\Delta t_k - \Delta t_{k-1}) - \Delta x_{i+1} + \bar{x}_i = 0, i \in \mathcal{I}_k, \quad k \in \mathcal{K}, \tag{6.7b}$$

$$\begin{aligned} Q_{xx,i} \Delta x_i + Q_{xu,i} \Delta u_i + A_i^{\mathrm{T}} \Delta \lambda_{i+1} - \Delta \lambda_i + h_{x,i} (\Delta t_k - \Delta t_{k-1}) + \bar{l}_{x,i} = 0, \\ i \in \mathcal{I}_k, \quad k \in \mathcal{K}, \end{aligned} \tag{6.7c}$$

$$Q_{xu,i}^{\mathrm{T}} \Delta x_i + Q_{uu,i} \Delta u_i + B_i^{\mathrm{T}} \Delta \lambda_{i+1} + h_{u,i} (\Delta t_k - \Delta t_{k-1}) + \bar{l}_{u,i} = 0, \quad i \in \mathcal{I}_k, \quad k \in \mathcal{K}, \tag{6.7d}$$

$$\begin{aligned} &\frac{1}{N_k} \sum_{i \in \mathcal{I}_k} \left(h_{x,i}^{\mathrm{T}} \Delta x_i + h_{u,i}^{\mathrm{T}} \Delta u_i + f_i^{\mathrm{T}} \Delta \lambda_{i+1} + \bar{h}_i \right) \\ &- \frac{1}{N_{k+1}} \sum_{i \in \mathcal{I}_{k+1}} \left(h_{x,i}^{\mathrm{T}} \Delta x_i + h_{u,i}^{\mathrm{T}} \Delta u_i + f_i^{\mathrm{T}} \Delta \lambda_{i+1} + \bar{h}_i \right) \\ &+ Q_{tt,k} (\Delta t_k - \Delta t_{k-1}) + \bar{q}_{t,k} = 0, \quad k \in \{1, ..., K\}, \end{aligned} \tag{6.7e}$$

and

$$Q_{xx,N} \Delta x_N - \Delta \lambda_N + \bar{l}_{x,N} = 0, \tag{6.7f}$$

where

$$Q_{xx,i} := \nabla_{xx} H_k (x_i, u_i, \lambda_{i+1}) \Delta \tau_k + \nabla_x g_k^{\mathrm{T}} (x_i, u_i) \mathrm{diag}(z_i)^{-1} \mathrm{diag}(\nu_i) \nabla_x g_k^{\mathrm{T}} (x_i, u_i),$$

$$Q_{xu,i} := \nabla_{xu} H_k (x_i, u_i, \lambda_{i+1}) \Delta \tau_k + \nabla_x g_k^{\mathrm{T}} (x_i, u_i) \mathrm{diag}(z_i)^{-1} \mathrm{diag}(\nu_i) \nabla_u g_k^{\mathrm{T}} (x_i, u_i),$$

$$Q_{uu,i} := \nabla_{uu} H_k (x_i, u_i, \lambda_{i+1}) \Delta \tau_k + \nabla_u g_k^{\mathrm{T}} (x_i, u_i) \mathrm{diag}(z_i)^{-1} \mathrm{diag}(\nu_i) \nabla_u g_k^{\mathrm{T}} (x_i, u_i),$$

$$\bar{l}_{x,i} := r_{x,i} + \nabla_x g_k^{\mathrm{T}} (x_i, u_i) \mathrm{diag}(z_i)^{-1} (\mathrm{diag}(\nu_i) r_{g,i} - r_{z,i}),$$

$$\bar{l}_{u,i} := r_{u,i} + \nabla_u g_k^{\mathrm{T}} (x_i, u_i) \mathrm{diag}(z_i)^{-1} (\mathrm{diag}(\nu_i) r_{g,i} - r_{z,i}),$$

$$Q_{tt,k} := w_k^{-1} \upsilon_k, \quad \bar{q}_{t,k} := w_k^{-1} (\upsilon_k r_{\Delta,k} - r_{w,k}),$$

$$A_i := I + \nabla_x f_k (x_i, u_i) \Delta \tau_k, \quad B_i := \nabla_x f_k (x_i, u_i) \Delta \tau_k,$$

$$f_i := \frac{1}{N_k} f_k (x_i, u_i), \quad h_i := \frac{1}{N_k} H_k (x_i, u_i, \lambda_i),$$

and

$$h_{x,i} := \frac{1}{N_k} \nabla_x H_k(x_i, u_i, \lambda_i), \quad h_{u,i} := \frac{1}{N_k} \nabla_u H_k(x_i, u_i, \lambda_i).$$

In addition, $\bar{x}_i$ is residual in (6.4c). Note that some of the phase index k is omitted from the KKT system equations (6.7) because the matrices and vectors (other than the Newton steps in (6.7)) are fixed once they are computed and therefore do not depend on the phase index k to solve the KKT system.

Note that the KKT system (6.7) is equivalent to the KKT conditions of a QP subproblem

$$\begin{aligned} \min_{\substack{\Delta u_0, ..., \Delta u_{N-1} \\ \Delta t_1, ..., \Delta t_{K+1}}} \sum_{k \in \mathcal{K}} \sum_{i \in \mathcal{I}_k, i \neq N} \frac{1}{2} & \left\{ \begin{bmatrix} \Delta t_k - \Delta t_{k-1} \\ \Delta x_i \\ \Delta u_i \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} 0 & h_{x,i}^{\mathrm{T}} & h_{u,i}^{\mathrm{T}} \\ h_{x,i} & Q_{xx,i} & Q_{xu,i} \\ h_{u,i} & Q_{ux,i} & Q_{uu,i} \end{bmatrix} \begin{bmatrix} \Delta t_k - \Delta t_{k-1} \\ \Delta x_i \\ \Delta u_i \end{bmatrix} \right. \\ & \left. + \begin{bmatrix} \bar{h}_i \\ \bar{l}_{x,i} \\ \bar{l}_{u,i} \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} \Delta t_k - \Delta t_{k-1} \\ \Delta x_i \\ \Delta u_i \end{bmatrix} \right\} \\ + \sum_{k \in \mathcal{K}} & \left\{ \frac{1}{2} Q_{tt,k} \Delta t_k^2 - Q_{tt,k} \Delta t_k \Delta t_{k-1} + \bar{q}_{t,k} \Delta t_k \right\} + \Delta x_N^{\mathrm{T}} Q_{xx,N} \Delta x_N + \bar{l}_{x,N}^{\mathrm{T}} \Delta x_N \\ \text{s.t. } & (6.7a), (6.7b), \end{aligned} \tag{6.8}$$

which is a quadratic approximation of the NLP (6.4). Subsequently, $\Delta\lambda_0, ..., \Delta\lambda_N$ can be regarded as the Lagrange multiplier of the QP with respect to (6.7a) and (6.7b) (Nocedal and Wright (2006)).

Remark 6.2. *The Hessian matrix of (6.8) is inherently indefinite, which makes solving the KKT system (6.7) difficult when using off-the-shelf QP solvers because they typically require a positive definite Hessian matrix. This can be explained with a block diagonal of the Hessian matrix (6.8), expressed as:*

$$\begin{bmatrix} 0 & h_{x,i}^{\mathrm{T}} & h_{u,i}^{\mathrm{T}} \\ h_{x,i} & Q_{xx,i} & Q_{uu,i} \\ h_{u,i} & Q_{xu,i}^{\mathrm{T}} & Q_{uu,i} \end{bmatrix}. \tag{6.9}$$

This is indefinite due to the off-diagonal terms $h_{x,i}$ and $h_{u,i}$, even when: $\begin{bmatrix} Q_{xx,i} & Q_{uu,i} \\ Q_{xu,i}^{\mathrm{T}} & Q_{uu,i} \end{bmatrix} \succ O$.

6.4 Riccati Recursion to Solve KKT Systems

In this section, a Riccati recursion algorithm is presented to compute the Newton step of the NLP (6.4) by solving the KKT system (6.7). The sparsity structure of the

KKT system (6.7) is no longer the same as the standard OCP, which prevents applying the off-the-shelf efficient Newton-type algorithms for OCPs (Rao et al. (1998); Wang and Boyd (2010); Zanelli et al. (2020)). Moreover, as stated in Remark 6.2, the Hessian matrix of the KKT system is indefinite and unsolvable using general QP solvers. Motivated by these problems, we propose a Riccati recursion algorithm that efficiently solves the KKT system, specifically with $O(N)$ computational time, whenever the SOSC holds. Moreover, a reduced Hessian modification method is proposed to enhance the convergence when the SOSC does not hold, that is, when the reduced Hessian matrix is indefinite. As the standard Riccati recursion (Rao et al. (1998)), the proposed method is composed of backward and forward recursions, which recursively eliminate the variables from the KKT system (6.7) backward in time and recursively compute the Newton step forward in time, respectively.

6.4.1 Backward recursion

In the backward recursion, the Newton steps are recursively eliminated from stages N to 0. Specifically, expressions of Δx_{i+1}, Δu_i, and $\Delta \lambda_i$ are derived at each stage $i \in I_k$ for Δx_i, Δt_k, and Δt_{k-1}, respectively, from (6.7b)–(6.7d). Moreover, variables are recursively eliminated from (6.7e) using these expressions. When the stage of interest changes from k to $k-1$, Δt_k is further eliminated from (6.7e). That is, an expression of Δt_k is derived with respect to Δx_{i_k} and Δt_{k-1}, where $i_k := \min \mathcal{I}_k$. This can be seen as the extension of the standard Riccati recursion that also recursively derives expressions of Δx_{i+1}, Δu_i, and $\Delta \lambda_i$ with respect to Δx_i (Rao et al. (1998); Rawlings et al. (2017)).

6.4.1.1 Terminal stage

From (6.7f) at stage N, we obtain:

$$\Delta \lambda_N = P_N \Delta x_N - s_N, \tag{6.10a}$$

where

$$P_N = Q_{xx,N}, \quad s_N = -\bar{l}_{x,N}. \tag{6.10b}$$

6.4.1.2 Intermediate stages

Consider $i, i+1 \in \mathcal{I}_k$, $k \in \mathcal{K}$. Suppose that we have the expression of $\Delta \lambda_{i+1}$ with respect to Δx_{i+1}, Δt_k, and Δt_{k-1} as

$$\Delta \lambda_{i+1} = P_{i+1} \Delta x_{i+1} - s_{i+1} + \Psi_{i+1} (\Delta t_k - \Delta t_{k-1}) + \Phi_{i+1} (-\Delta t_k), \tag{6.11a}$$

where $P_{i+1} \in \mathbb{R}^{n_x \times n_x}$ and $s_{i+1},\ \Psi_{i+1},\ \Phi_{i+1} \in \mathbb{R}^{n_x}$. Suppose also that we have the equations (6.7e) for k and $k-1$ in which Δx_j for $j \geq i+2$, Δu_j for $j \geq i+1$, and $\Delta \lambda_j$ for $j \geq i+1$ are eliminated, that is, the equations (6.7e) for k and $k-1$ are reduced to

$$\begin{aligned}&\sum_{j \in \mathcal{I}_k,\, j \leq i} \left(h_{x,j}^{\mathrm{T}} \Delta x_j + h_{u,j}^{\mathrm{T}} \Delta u_j + f_j^{\mathrm{T}} \Delta \lambda_{j+1} + \bar{h}_j \right) \\ &+ \Psi_{i+1}^{\mathrm{T}} \Delta x_{i+1} + \xi_{i+1}(\Delta t_k - \Delta t_{k-1}) + \chi_{i+1}(-\Delta t_k) + \eta_{i+1} \\ &- \Phi_{i+1}^{\mathrm{T}} \Delta x_{i+1} - \chi_{i+1}(\Delta t_k - \Delta t_{k-1}) - \rho_{i+1}(-\Delta t_k) - \iota_{i+1} = 0, \end{aligned} \tag{6.11b}$$

and

$$\begin{aligned}&\sum_{j \in \mathcal{I}_{k-1}} \left(h_{x,j}^{\mathrm{T}} \Delta x_j + h_{u,j}^{\mathrm{T}} \Delta u_j + f_j^{\mathrm{T}} \Delta \lambda_{j+1} + \bar{h}_j \right) \\ &- \sum_{j \in \mathcal{I}_K,\, j \leq i} \left(h_{x,j}^{\mathrm{T}} \Delta x_j + h_{u,j}^{\mathrm{T}} \Delta u_j + f_j^{\mathrm{T}} \Delta \lambda_{j+1} + \bar{h}_j \right) \\ &- \Psi_{i+1}^{\mathrm{T}} \Delta x_{i+1} - \xi_{i+1}(\Delta t_k - \Delta t_{k-1}) - \chi_{i+1}(-\Delta t_k) - \eta_{i+1} = 0, \end{aligned} \tag{6.11c}$$

where $\xi_{i+1}, \chi_{i+1}, \rho_{i+1}, \eta_{i+1}, \iota_{i+1} \in \mathbb{R}$. Note that $\Psi_{i+1} = 0$, $\Phi_{i+1} = 0$, $\xi_{i+1} = 0$, $\chi_{i+1} = 0$, $\rho_{i+1} = 0$, $\eta_{i+1} = 0$, and $\iota_{i+1} = 0$ at stage $i = N-1$. Moreover, $\Psi_{i+1} = 0$, $\xi_{i+1} = 0$, $\chi_{i+1} = 0$, and $\eta_{i+1} = 0$ at stages $i = \min \mathcal{I}_k - 1$ and $k \in \{2, ..., K\}$, as explained in 6.4.1.3. First, the following equations are introduced:

$$F_i := Q_{xx,i} + A_i^{\mathrm{T}} P_{i+1} A_i, \tag{6.12a}$$

$$H_i := Q_{xu,i} + A_i^{\mathrm{T}} P_{i+1} B_i, \tag{6.12b}$$

$$G_i := Q_{uu,i} + B_i^{\mathrm{T}} P_{i+1} B_i, \tag{6.12c}$$

$$\psi_{x,i} := h_{x,i} + A_i^{\mathrm{T}} P_{i+1} f_i + A_i^{\mathrm{T}} \Psi_{i+1}, \tag{6.12d}$$

$$\psi_{u,i} := h_{u,i} + B_i^{\mathrm{T}} P_{i+1} f_s + B_i^{\mathrm{T}} \Psi_{i+1}, \tag{6.12e}$$

and

$$\phi_{x,i} := A_i^{\mathrm{T}} \Phi_{i+1}, \ \ \phi_{u,i} := B_i^{\mathrm{T}} \Phi_{i+1}. \tag{6.12f}$$

Second, $\Delta \lambda_{i+1}$ and Δx_{i+1} are eliminated from (6.7d) using (6.11a) and (6.7b). Subsequently, we can express Δu_i using Δx_i, Δt_k, and Δt_{k-1} as:

$$\Delta u_i = K_i \Delta x_i + k_i + T_i(\Delta t_k - \Delta t_{k-1}) + W_i(-\Delta t_k), \tag{6.13a}$$

where

$$K_i := -G_i^{-1} H_i^{\mathrm{T}}, \tag{6.13b}$$

$$k_i := -G_i^{-1}(B_i^{\mathrm{T}} P_{i+1} \bar{x}_i - B_i^{\mathrm{T}} z_{i+1} + \bar{l}_{u,i}), \tag{6.13c}$$

and

$$T_i := -G_i^{-1} \psi_{u,i}, \ \ W_i := -G_i^{-1} \phi_{u,i}. \tag{6.13d}$$

Third, $\Delta\lambda_{i+1}$, Δx_{i+1}, and Δu_i are eliminated from (6.7c) using (6.11a), (6.7b), and (6.13a), respectively. As a result, $\Delta\lambda_i$ using Δx_i, Δt_k, and Δt_{k-1} is expressed as:

$$\Delta\lambda_i = P_i \Delta x_i - s_i + \Psi_i(\Delta t_k - \Delta t_{k-1}) + \Phi_i(-\Delta t_k), \tag{6.14a}$$

where

$$P_i := F_i - K_i^{\mathrm{T}} G_i K_i, \tag{6.14b}$$

$$s_i := -\left\{ \bar{l}_{x,i} + A_i^{\mathrm{T}}(P_{i+1}\bar{x}_i - s_{i+1}) + H_i k_i \right\}, \tag{6.14c}$$

and

$$\Psi_i := \psi_{x,i} + K_i \psi_{u,i}, \ \ \Phi_i := \phi_{x,i} + K_i \phi_{u,i}. \tag{6.14d}$$

Moreover, by eliminating $\Delta\lambda_{i+1}$, Δx_{i+1}, and Δu_i from (6.11b) and (6.11c) using (6.11a), (6.7b), and (6.13a), we obtain:

$$\begin{aligned} &\sum_{j \in \mathcal{I}_k,\, j \le i-1} \left(h_{x,j}^{\mathrm{T}} \Delta x_j + h_{u,j}^{\mathrm{T}} \Delta u_j + f_j^{\mathrm{T}} \Delta\lambda_{j+1} + \bar{h}_j \right) \\ &+ \Psi_i^{\mathrm{T}} \Delta x_i + \xi_i(\Delta t_k - \Delta t_{k-1}) + \chi_i(-\Delta t_k) + \eta_i \\ &- \Phi_i^{\mathrm{T}} \Delta x_i - \chi_i(\Delta t_k - \Delta t_{k-1}) - \rho_i(-\Delta t_k) - \iota_i = 0 \end{aligned} \tag{6.15a}$$

and

$$\begin{aligned} &\sum_{j \in \mathcal{I}_{k-1}} \left(h_{x,j}^{\mathrm{T}} \Delta x_j + h_{u,j}^{\mathrm{T}} \Delta u_j + f_j^{\mathrm{T}} \Delta\lambda_{j+1} + \bar{h}_j \right) \\ &- \sum_{j \in \mathcal{I}_K,\, j \le i-1} \left(h_{x,j}^{\mathrm{T}} \Delta x_j + h_{u,j}^{\mathrm{T}} \Delta u_j + f_j^{\mathrm{T}} \Delta\lambda_{j+1} + \bar{h}_j \right) \\ &- \Psi_i^{\mathrm{T}} \Delta x_i - \xi_i(\Delta t_k - \Delta t_{k-1}) - \chi_i(-\Delta t_k) - \eta_i = 0, \end{aligned} \tag{6.15b}$$

where

$$\xi_i := \xi_{i+1} + f_i^{\mathrm{T}}(P_{i+1} f_i + 2\Psi_{i+1}) + \psi_{u,i}^{\mathrm{T}} T_i, \tag{6.16a}$$

$$\eta_i := \eta_{i+1} + \bar{h}_i + f_i^{\mathrm{T}}(P_{i+1}\bar{x}_i - s_{i+1}) + \Psi_{i+1}^{\mathrm{T}} \bar{x}_i + \psi_{u,i}^{\mathrm{T}} k_i, \tag{6.16b}$$

$$\chi_i := \chi_{i+1} + \Phi_{i+1}^{\mathrm{T}} f_i + \psi_{u,i}^{\mathrm{T}} W_i, \tag{6.16c}$$

$$\rho_i := \rho_{i+1} + \phi_{u,i}^{\mathrm{T}} W_i, \tag{6.16d}$$

and

$$\iota_i := \iota_{i+1} + \Phi_{i+1}^{\mathrm{T}} \bar{x}_i + \phi_{u,i}^{\mathrm{T}} k_i. \tag{6.16e}$$

Therefore, we have equations (6.14a), (6.15a), and (6.15b) for stage i in the same form as equations (6.11) for stage $i+1$.

6.4.1.3 Phase transition stages

Here, we consider phase $k \in \{1, ..., K\}$ and stage $i_k := \min \mathcal{I}_k$. In this stage, the phase of interest changes from k to $k-1$ when $k > 1$, and the backward recursion terminates when $k = 1$. The equations (6.11) until i_k are expressed as:

$$\Delta\lambda_{i_k} = P_{i_k}\Delta x_{i_k} - s_{i_k} + \Psi_{i_k}(\Delta t_k - \Delta t_{k-1}) + \Phi_{i_k}(-\Delta t_k), \tag{6.17a}$$

$$\begin{aligned}&\Psi_{i_k}^{\mathrm{T}}\Delta x_{i_k} + \xi_{i_k}(\Delta t_k - \Delta t_{k-1}) + \chi_{i_k}(-\Delta t_k) + \eta_{i_k} \\ &- \Phi_{i_k}^{\mathrm{T}}\Delta x_{i_k} - \chi_{i_k}(\Delta t_k - \Delta t_{k-1}) - \rho_{i_k}(-\Delta t_k) - \iota_{i_k} = 0,\end{aligned} \tag{6.17b}$$

and

$$\begin{aligned}&\sum_{j\in\mathcal{I}_{k-1}} \left(h_{x,j}^{\mathrm{T}}\Delta x_j + h_{u,j}^{\mathrm{T}}\Delta u_j + f_j^{\mathrm{T}}\Delta\lambda_{j+1} + \bar{h}_j\right) - \Psi_{i_k}^{\mathrm{T}}\Delta x_{i_k} \\ &- \xi_{i_k}(\Delta t_k - \Delta t_{k-1}) - \chi_{i_k}(-\Delta t_k) - \eta_{i_k} = 0.\end{aligned} \tag{6.17c}$$

Note that $\Delta t_{k-1} = \Delta t_0 = 0$ when $k = 1$. Therefore, from (6.17b), Δt_k is determined as:

$$\Delta t_k = -\sigma_{i_k}^{-1}(\Psi_{i_k} - \Phi_{i_k})^{\mathrm{T}}\Delta x_{i_k} - \sigma_{i_k}^{-1}(\xi_{i_k} - \chi_{i_k})(-\Delta t_{k-1}) - \sigma_{i_k}^{-1}(\eta_{i_k} - \iota_{i_k}). \tag{6.18}$$

Here, the following is defined:

$$\sigma_{i_k} := \xi_{i_k} - 2\chi_{i_k} + \rho_{i_k}. \tag{6.19}$$

The backward recursion is completed when $k = 1$ because $i_1 = \min \mathcal{I}_1 = 0$. When $k > 1$, (6.18) is further substituted into (6.17a) and (6.17c), which produces:

$$\Delta\lambda_{i_K} = \tilde{P}_{i_K}\Delta x_{i_K} - \tilde{s}_{i_K} + \tilde{\Psi}_{i_K}(-\Delta t_{k-1}) \tag{6.20a}$$

and

$$\sum_{i\in\mathcal{I}_{K-1}} \left(h_{x,i}^{\mathrm{T}}\Delta x_i + h_{u,i}^{\mathrm{T}}\Delta u_i + f_i^{\mathrm{T}}\Delta\lambda_{i+1} + \bar{h}_i\right) - \tilde{\Phi}_{i_k}^{\mathrm{T}}\Delta x_{i_k} - \tilde{\rho}_{i_k}(-\Delta t_{k-1}) - \tilde{\iota}_{i_k} = 0, \tag{6.20b}$$

where

$$\tilde{P}_{i_k} := P_{i_k} - \sigma_{i_k}^{-1}(\Psi_{i_k} - \Phi_{i_k})(\Psi_{i_k} - \Phi_{i_k})^{\mathrm{T}}, \tag{6.20c}$$

$$\tilde{s}_{i_k} := s_{i_k} + \sigma_{i_k}^{-1}(\eta_{i_k} - \iota_{i_k})(\Psi_{i_k} - \Phi_{i_k}), \tag{6.20d}$$

$$\tilde{\Phi}_{i_k} := \Psi_{i_k} - \sigma_{i_k}^{-1}(\xi_{i_k} - \chi_{i_k})(\Psi_{i_k} - \Phi_{i_k}), \tag{6.20e}$$

$$\tilde{\rho}_{i_k} := \xi_{i_k} - \sigma_{i_k}^{-1} (\xi_{i_k} - \chi_{i_k})^2, \tag{6.20f}$$

and

$$\tilde{\iota}_{i_k} := \eta_{i_k} - \sigma_{i_k}^{-1} (\xi_{i_k} - \chi_{i_k})(\eta_{i_k} - \iota_{i_k}). \tag{6.20g}$$

Therefore, together with an equation (6.7e) for $k-1$, we obtain equations (6.14a), (6.15a), and (6.15b) for stage i_k in the same form as equations (6.11) for stage $i+1$ with $P_{i+1} = \tilde{P}_{i_k}$, $s_{i+1} = \tilde{s}_{i_k}$, $\Psi_{i+1} = 0$, $\Phi_{i+1} = \tilde{\Psi}_{i_k}$, $\xi_{i+1} = 0$, $\chi_{i+1} = 0$, $\rho_{i+1} = \tilde{\rho}_{i_k}$, $\eta_{i+1} = 0$, and $\iota_{i+1} = \tilde{\iota}_{i_k}$.

6.4.2 Forward recursion

After the backward recursion up to the initial stage ($i = 0$), all Newton directions are recursively computed from stage 0 to N forward in time using the results of the backward recursion. First, the initial state direction Δx_0 is computed from (6.7a). Second, the direction of the first switching time Δt_1 is computed from (6.18) with $\Delta t_0 = 0$. Third, $\Delta \lambda_i$, Δu_i, and Δx_{i+1} are computed forward in time in each phase k using (6.11a), (6.13a), and (6.7b), respectively. Fourth, after this procedure, Δt_{k+1} is computed using (6.18) until stage $i_{k+1} - 1 = \min \mathcal{I}_{k+1} - 1$. The third and forth steps are repeated from phase $k = 1$ to $k = K+1$, and the forward recursion is completed by computing $\Delta \lambda_N$ using (6.10a).

6.4.3 Properties of proposed Riccati recursion

Next, we demonstrate that the proposed Riccati recursion always successfully solves the KKT system (6.7) if the solution is sufficiently close to the local minimum. This is in contrast to general QP solvers that fail to solve the KKT system because the Hessian matrix is inherently indefinite, as stated in Remark 6.2. We first explain that the proposed backward recursion is the same as dynamic programming for the QP subproblem (6.8), which is a quadratic approximation of the NLP (6.4). The following lemma states the equivalence of the dynamic programming and the proposed Riccati recursion:

Lemma 6.1. *Consider a phase $k \in \mathcal{K}$ and stage $i \in \mathcal{I}_k$. Suppose that $G_j \succ O$ for all $j \geq i$, and $\sigma_{i_l} > 0$ for all $l > k$, where G_j and σ_{i_l} are defined in (6.12c) and (6.19), respectively. Then, the cost-to-go function of stage i of dynamic programming for the*

QP (6.8) is expressed as:

$$\frac{1}{2}\begin{bmatrix} -\Delta t_k \\ \Delta t_k - \Delta t_{k-1} \\ \Delta x_i \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} \rho_i & \chi_i & \Phi_i^{\mathrm{T}} \\ \chi_i & \xi_i & \Psi_i^{\mathrm{T}} \\ \Phi_i & \Psi_i & P_i \end{bmatrix} \begin{bmatrix} -\Delta t_k \\ \Delta t_k - \Delta t_{k-1} \\ \Delta x_i \end{bmatrix} + \begin{bmatrix} \iota_i \\ \eta_i \\ -s_i \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} -\Delta t_k \\ \Delta t_k - \Delta t_{k-1} \\ \Delta x_i \end{bmatrix}, \tag{6.21}$$

where ξ_i, ρ_i, Φ_i, Ψ_i, P_i, ι_i, η_i, and s_i are defined by the Riccati recursion algorithm presented in subsection 6.4.1. Moreover, the subproblem of the dynamic programming to determine Δu_i is represented by

$$\begin{aligned} \min_{\Delta u_i} \frac{1}{2} & \begin{bmatrix} -\Delta t_k \\ \Delta t_k - \Delta t_{k-1} \\ \Delta x_i \\ \Delta u_i \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} Q_{tt,i} & h_{x,i}^{\mathrm{T}} & h_{u,i}^{\mathrm{T}} \\ h_{x,i} & Q_{xx,i} & Q_{xu,i} \\ h_{u,i} & Q_{ux,i} & Q_{uu,i} \end{bmatrix} \begin{bmatrix} -\Delta t_k \\ \Delta t_k - \Delta t_{k-1} \\ \Delta x_i \\ \Delta u_i \end{bmatrix} \\ & + \begin{bmatrix} \bar{h}_i \\ \bar{l}_{x,i} \\ \bar{l}_{u,i} \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} -\Delta t_k \\ \Delta t_k - \Delta t_{k-1} \\ \Delta x_i \\ \Delta u_i \end{bmatrix} \\ & + \frac{1}{2} \begin{bmatrix} -\Delta t_k \\ \Delta t_k - \Delta t_{k-1} \\ \Delta x_{i+1} \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} \rho_{i+1} & \chi_{i+1} & \Phi_{i+1}^{\mathrm{T}} \\ \chi_{i+1} & \xi_{i+1} & \Psi_{i+1}^{\mathrm{T}} \\ \Phi_{i+1} & \Psi_{i+1} & P_{i+1} \end{bmatrix} \begin{bmatrix} -\Delta t_k \\ \Delta t_k - \Delta t_{k-1} \\ \Delta x_{i+1} \end{bmatrix} \\ & + \begin{bmatrix} \iota_{i+1} \\ \eta_{i+1} \\ -s_{i+1} \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} -\Delta t_k \\ \Delta t_k - \Delta t_{k-1} \\ \Delta x_{i+1} \end{bmatrix} \\ \text{s.t.} \quad & A_i \Delta x_i + B_i \Delta u_i + f_i(\Delta t_k - \Delta t_{k-1}) - \Delta x_{i+1} + \bar{x}_i = 0, \end{aligned} \tag{6.22}$$

and determining Δt_k is expressed as:

$$\min_{\Delta t_k} \ (6.21). \tag{6.23}$$

Proof. The proof is achieved by induction. At the terminal stage ($i = N$), the cost-to-go function is represented by (6.21) with $\Psi_N = \Phi_N = 0$ and $\xi_N = \chi_N = \rho_N = \eta_N = \iota_N = 0$. Next, suppose that we have the cost-to-go function (6.21) of stage $i+1$. Then, it is clear that the subproblem of the dynamic programming is represented by (6.22). Δu_i can be uniquely determined from (6.22) as (6.13) because $G_i \succ O$. The cost-to-go function of stage i is represented by (6.21), where P_i, s_i, Ψ_i, Φ_i, ξ_i, χ_i, ρ_i, η_i, and ι_i are defined as (6.12), (6.13), (6.14), and (6.16). When $i = i_k := \min I_k$, Δt_k can be further uniquely determined by solving (6.23) under the assumption $\sigma_{i_k} > 0$ after determining Δu_{i_k} and obtaining the cost-to-go function of stage i_k (6.21). Then, the cost-to-go function of stage i_k is in the form of (6.21), where P_{i_k}, s_{i_k}, Φ_{i_k}, ρ_{i_k}, and

ι_{i_k} are defined as (6.20), $\Psi_{i_k} = 0$, $\xi_{i_k} = 0$, $\chi_{i_k} = 0$, and $\eta_{i_k} = 0$, respectively, which completes the proof. □

Note that the equivalence cannot be shown without the positive definiteness of G_i and σ_{i_k}: if they are not positive definite, a unique solution does not exist and the cost-to-go function is not defined. The following theorem is obtained based on Lemma 6.1. Note that the discussion herein is restricted to the exact Hessian matrix to analyze the SOSC.

Theorem 6.1. *We suppose that the SOSC and LICQ hold at the current iterate and consider that the exact Hessian matrix is used. Then, G_i, as defined in (6.12c), is positive definite for all $i \in \{0, ..., N-1\}$. Moreover, σ_{i_k}, as defined in (6.19), satisfies $\sigma_{i_k} > 0$ for all $k \in \{0, ..., K\}$.*

Proof. The QP subproblem (6.8) with an exact Hessian matrix must have a unique global solution because the SOSC and LICQ hold (Nocedal and Wright (2006)). Therefore, the dynamic programming subproblems (6.22) and (6.23) must have unique solutions. Subsequently, the Hessian matrix with respect to Δu_i in (6.22) must be positive definite, and the quadratic term with respect to Δt_k in (6.23) must be positive. Then, Lemma 6.1 recursively shows that the Hessian matrix is G_i with respect to Δu_i. Additionally, the lemma also shows that the quadratic term with respect to Δt_k is σ_{i_k}, which completes the proof. □

Theorem 6.1 indicates that G_i^{-1} can always be efficiently computed using Cholesky factorizations if the current iterate is sufficiently close to a local minimum. This fact also leads to the local convergence of the proposed method for NLP (6.4) under the SOSC and LICQ. The proof for this is omitted because it is trivial.

6.4.4 Reduced Hessian modification via Riccati recursion

The recused Hessian matrix can be indefinite when the solution is not sufficiently close to a local minimum, such that the SQSC does not hold. Subsequently, the KKT matrix is no longer invertible and the local convergence is not guaranteed. Efficient Cholesky factorization cannot be used to compute G_i^{-1}. Therefore, an algorithmic modification of the Riccati recursion is proposed to make the algorithm numerically robust and efficient for such situations, which can be considered a modification on the reduced Hessian matrix. To consider practical situations, Hessian approximations are allowed in the following, while Theorem 6.1 analyzes the exact Hessian matrix. First, we introduce the following practical assumption:

Assumption 6.1. *$\begin{bmatrix} Q_{xx,i} & Q_{xu,i} \\ Q_{xu,i}^{\mathrm{T}} & Q_{uu,i} \end{bmatrix} \succeq O$ and $Q_{uu,i} \succ O$ for all $i \in \{0, ..., N-1\}$, $Q_{tt,k} \geq 0$ for all $k \in \{0, ..., K\}$, and $Q_{xx,N} \succeq O$.*

This assumption is easily satisfied with the Gauss-Newton Hessian approximation or, more generally, with sequential convex programming (Messerer, Baumgärtner, and Diehl (2021)) for $\nabla_{xx} H_k(x_i, u_i, \lambda_{i+1})$, $\nabla_{xu} H_k(x_i, u_i, \lambda_{i+1})$, and $\nabla_{uu} H_k(x_i, u_i, \lambda_{i+1})$. Under Assumption 6.1, a unique solution of the dynamic programming subproblem (6.22) exists (that is, $G_i \succ O$) and $P_i \succeq O$ if $P_{i+1} \succeq O$, which is the same discussion as the dynamic programming for standard linear quadratic OCPs (Bertsekas (2005)). The solution to (6.23) also exists if $\sigma_{i_k} > 0$. Based on these observations, a modification of the Riccati recursion algorithm is proposed, whereby $\tilde{P}_{i_k}$ is updated at each phase-transition stage $i_k = \min \mathcal{I}_k$. This is expressed as:

$$\tilde{P}_{i_k} = P_{i_k} \tag{6.24}$$

instead of (6.20c). Then, $\tilde{P}_{i_k} \succeq O$, as long as $P_{i_k} \succeq O$. Note that a different method of (6.24) can be used to make $\tilde{P}_{i_k} \succeq O$, which eliminates the negative curvature from P_{i_k} through an eigenvalue decomposition (Quirynen et al. (2014)). However, this requires much more computational time than (6.24). Nevertheless, the computationally cheap modification (6.24) works surprisingly well in practice. We also modify σ_{i_k} instead of (6.19), which is expressed as:

$$\tilde{\sigma}_{i_k} = \begin{cases} \sigma_{i_k} & \sigma_{i_k} > \sigma_{\min} \\ |\sigma_{i_k}| + \bar{\sigma} & \sigma_{i_k} \leq \sigma_{\min} \end{cases}, \tag{6.25}$$

where $\sigma_{\min} \geq 0$ and $\bar{\sigma} > 0$. A practical rule to choose $\sigma_{\min}$ and $\bar{\sigma}$ is as follows. We empirically observed that numerical ill-conditioning in the Newton-type method produced a large Δt_s, that is, a too small σ_{i_k} in (6.18). From this observation, a desired maximum absolute value of Δt_s was chosen as $\Delta t_{s,\max} > 0$. Subsequently, $\sigma_{\min}$ and $\bar{\sigma}$ were chosen such that the absolute value of $\sigma_{i_k}^{-1}(\eta_{i_k} - \iota_{i_k})$ did not exceed $\Delta t_{s,\max}$. This is expressed as:

$$\sigma_{\min} = \bar{\sigma} = \Delta t_{s,\max}^{-1} |\eta_{i_k} - \iota_{i_k}| \tag{6.26}$$

for each $k \in \mathcal{K}$. For example, it was discovered that choosing $\Delta t_{s,\max}$ from a range of 0.1 to 1.0 worked well in practice.

The following Theorem claims that the proposed algorithmic modification makes the reduced Hessian matrix of the KKT system (6.7) positive definite and the KKT matrix invertible:

Theorem 6.2. *Suppose that Assumption 6.1 holds. Then, the reduced Hessian matrix of the KKT system (6.7) with the modifications (6.24) and (6.25) is positive definite. If, in addition, the LICQ holds, then the KKT matrix is invertible.*

Proof. Under Assumption 6.1 and with the Hessian modifications (6.24) and (6.25), $P_i \succeq O$ for all $i = 0, ..., N$, $G_i \succ O$ for all $i = 0, ..., N-1$, and $\sigma_{i_k} > 0$ for all $k = 1, ..., K$ hold. Therefore, the QP subproblems of the dynamic programming (6.22) and (6.23) always have unique global solutions. Theorem 16.2 of Nocedal and Wright (2006) completes the first claim. The second claim then directly follows from Theorem 16.1 of Nocedal and Wright (2006). $\square$

Remark 6.3. *The KKT matrix is always invertible under Assumption 6.1 and the proposed reduced Hessian modifications. Therefore, the local convergence of the Newton-type method is ensured if the resultant reduced Hessian is sufficiently close to the exact one (Messerer et al. (2021); Nocedal and Wright (2006)) and the LICQ holds.*

6.4.5 Algorithm

The single Newton iteration using the proposed Riccati recursion algorithm is summarized in Algorithm 6.2. After computing the KKT system (6.7) (line 1), the proposed method recursively eliminates the Newton steps from the KKT system (6.7) in the backward recursion (lines 3–14) and recursively computes the Newton steps in the forward recursion (lines 16–27). Finally, the Newton steps of the slack variables and Lagrange multipliers are computed from the other Newton steps (line 28) according to the primal-dual interior point method (Nocedal and Wright (2006); Wächter and Biegler (2006)). Calculations in the proposed algorithm, such as matrix inversions or matrix-matrix multiplications, do not grow with respect to the length of the horizon N. Therefore, the computational time of the proposed method is $O(N)$ as the standard Riccati recursion algorithm (Rao et al. (1998)).

6.5 State Jumps and Switching Conditions

In this section, the proposed NLP formulations and Riccati recursion algorithm are further extended to switched systems involving state jumps and switching conditions, which have not yet been considered. Some classes of switched systems involve state jumps at the same time as the switch, which is expressed as:

$$x(t_k) = f_j(x(t_k-)), \quad f_j : \mathbb{R}^{n_x} \to \mathbb{R}^{n_x}, \tag{6.27}$$

where t_k- is the instant immediately before t_k. We also consider that there is a stage cost on the state immediately before the state jump. That is, it is assumed that $l_j(x(t_k-))$ is added to the cost function (6.3a). Moreover, such switches are often state-dependent, that is, the switch occurs if the state satisfies some condition (hereafter called the switching condition), which is expressed as:

$$e(x(t_k-)) = 0, \;\; e : \mathbb{R}^{n_x} \to \mathbb{R}^{n_e}. \tag{6.28}$$

For example, a legged robot can be modeled as a system with switches involving state jumps and switching conditions. When the distance between the robot's foot and the ground becomes zero (the switching condition), the generalized velocity instantly changes due to the impact (the state jump) (Farshidian, Kamgarpour, et al. (2017); Li and Wensing (2020)).

If there is a state jump at the kth switch, a new grid point i_k- is introduced corresponding to time t_k-, which is added to the KKT conditions and expressed as:

$$x_{i_k} = f_j(x_{i_k-}) \tag{6.29a}$$

and

$$\nabla_x l_j(x_{i_k-}) + \nabla_x f_j^{\mathrm{T}}(x_{i_k-})\lambda_{i_k} - \lambda_{i_k-} = 0. \tag{6.29b}$$

The KKT systems regarding the state jump are therefore expressed as:

$$\Delta x_{i_k} = A_{i_k-}\Delta x_{i_k-} + \bar{x}_{i_k-} \tag{6.30a}$$

and

$$\Delta \lambda_{i_k-} = Q_{xx,i_k-}\Delta x_{i_k-} + A_{i_k-}^{\mathrm{T}}\Delta\lambda_{i_k} + \bar{l}_{x,i_k-}, \tag{6.30b}$$

where $A_{i_k-} := \nabla_x f_j(x_{i_k-})$ and Q_{xx,i_k-} is the Hessian of the Lagrangian with respect to x_{i_k-}. $\bar{x}_{i_k}$ and $\bar{l}_{x,i_k-}$ are the residuals in (6.29a) and (6.29b), respectively. When (6.30a) and (6.30b) are considered in the backward recursion, we have the equations until stage $i_k = \min \mathcal{I}_k$. That is, the equations are in the form of (6.20a), (6.20b), and (6.7e), with the phase index k replaced with $k-1$. By eliminating $\Delta\lambda_{i_k}$ and Δx_{i_k} from these equations using (6.30a) and (6.30b), we then obtain:

$$\Delta\lambda_{i_k-} = P_{i_k-}\Delta x_{i_k-} + \Phi_{i_k-}(-\Delta t_{k-1}) - s_{i_k-}, \tag{6.31a}$$

$$\begin{aligned}\sum_{i \in \mathcal{I}_{k-1}} &\left(h_{x,i}^{\mathrm{T}}\Delta x_i + h_{u,i}^{\mathrm{T}}\Delta u_i + f_i^{\mathrm{T}}\Delta\lambda_{i+1} + \bar{h}_i\right) \\ &- \Phi_{i_k-}^{\mathrm{T}}\Delta x_{i_k-} - \xi_{i_k-}(\Delta t_{k-1}) - \eta_{i_k-} = 0,\end{aligned} \tag{6.31b}$$

and

$$\sum_{i \in \mathcal{I}_{k-2}} \left(h_{x,i}^{\mathrm{T}} \Delta x_i + h_{u,i}^{\mathrm{T}} \Delta u_i + f_i^{\mathrm{T}} \Delta \lambda_{i+1} + \bar{h}_i \right) \\ - \sum_{i \in \mathcal{I}_{k-1}} \left(h_{x,i}^{\mathrm{T}} \Delta x_i + h_{u,i}^{\mathrm{T}} \Delta u_i + f_i^{\mathrm{T}} \Delta \lambda_{i+1} + \bar{h}_i \right) = 0, \tag{6.31c}$$

where

$$P_{i_k -} := Q_{xx, i_k -} + A_{i_k -}^{\mathrm{T}} \tilde{P}_{i_k} A_{i_k -}, \tag{6.31d}$$

$$\Phi_{i_k -} := A_{i_k -}^{\mathrm{T}} \tilde{\Phi}_{i_k}, \tag{6.31e}$$

$$s_{i_k -} := A_{i_k -}^{\mathrm{T}} (\tilde{s}_{i_k} - \tilde{P}_{i_k} \bar{x}_{i_k -}) - \bar{l}_{x, i_k -}, \tag{6.31f}$$

and

$$\xi_{i_k -} = \tilde{\xi}_{i_k}, \quad \eta_{i_k -} = \tilde{\eta}_{i_k} + \tilde{\Phi}_{i_k}^{\mathrm{T}} \bar{x}_{i_k -}. \tag{6.31g}$$

Therefore, the same equation as (6.14) is achieved at stage i_k-, and we can proceed to stage $i_k - 1$. In the forward recursion, $\Delta x_{i_k -}$ is computed from $\Delta x_{i_k - 1}$ and $\Delta u_{i_k - 1}$ using (6.7b) at stage $i_k - 1$. Then, Δx_{i_k} and $\Delta \lambda_{i_k -}$ are computed from $\Delta x_{i_k -}$ according to (6.30a) and (6.14a) for $i = i_k-$ with $\Psi_{i_k} = 0$.

6.5.1 Switching conditions

The switching conditions are typically presented as pure-state equality constraints, which are difficult to treat efficiently with the Riccati recursion algorithm (specifically with the $O(N)$ computational burden) as discussed in Chapter 5. In this section, we assume that the state is partitioned into the generalized coordinate and velocity as $x = \begin{bmatrix} q^{\mathrm{T}} & v^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}}$, $q, v \in \mathbb{R}^n$ and the state equation of each subsystem is expressed as:

$$f(x, u) := \begin{bmatrix} f^{(q)}(x) \\ f^{(v)}(x, u) \end{bmatrix}, \tag{6.32}$$

where $f^{(q)} : \mathbb{R}^{n_x} \to \mathbb{R}^n$ and $f^{(v)} : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^n$. Moreover, we assume that the pure-state equality constraints representing the switching conditions have a relative degree of two, that is, the Jacobian of (6.28) is expressed as:

$$\nabla_x e(x) = \begin{bmatrix} \nabla_q e(q) & O \end{bmatrix}, \quad \nabla_q e(q) \in \mathbb{R}^{n_e \times n}. \tag{6.33}$$

These assumptions mainly represent the position-level constraints on mechanical systems, such as robotic systems with rigid contacts. To perform the Newton-type method with $O(N)$ complexity under these assumptions, we did not consider the pure-state equality constraint

$$e(x_{i_k} -) = 0 \tag{6.34}$$

directly as Sideris and Rodriguez (2011). Instead, the constraint was transformed into a mixed state-input constraint at the two-stage grid point before the switch (see Chapter 5), which is expressed as:

$$e_i := e\left(x_i + f(x_i, u_i)\Delta\tau_k + g(x_i + f(x_i, u_i)\Delta\tau_k)\Delta\tau_k\right) = 0, \tag{6.35}$$

where i is two-stage before the grid i_k-, that is, i satisfies $i+2 = i_k-$. The Lagrange multiplier is introduced with respect to (6.35), $\zeta_i \in \mathbb{R}^{n_e}$. The KKT conditions regarding this stage are then represented by (6.4c), (6.35), $\tilde{r}_{x,i} := r_{x,i} + C_i^{\mathrm{T}}\zeta_i = 0$, and $\tilde{r}_{u,i} := r_{u,i} + D_i^{\mathrm{T}}\zeta_i = 0$, where $C_i := \nabla_x e(\cdot)$ and $D_i := \nabla_u e(\cdot)$. Furthermore, the KKT conditions of (6.6f) related to t_{k-1} are replaced with:

$$E_i^{\mathrm{T}}\zeta_i + \frac{1}{N_{k-1}}\sum_{j\in\mathcal{I}_{k-1}} H_{k-1}(x_j, u_j, \lambda_{j+1}) - \frac{1}{N_k}\sum_{i\in\mathcal{I}_k} H_k(x_j, u_j, \lambda_{j+1}) + \upsilon_{k-1} - \upsilon_k = 0$$

and

$$\begin{aligned}&\frac{1}{N_{k-2}}\sum_{j\in\mathcal{I}_{k-2}} H_{k-2}(x_j, u_j, \lambda_{j+1}) - E_i^{\mathrm{T}}\zeta_i \\ &- \frac{1}{N_{k-1}}\sum_{i\in\mathcal{I}_{k-1}} H_{k-1}(x_j, u_j, \lambda_{j+1}) + \upsilon_{k-2} - \upsilon_{k-1} = 0,\end{aligned}$$

where $E_i := \frac{1}{N_{k-1}}\nabla_{t_k} e(\cdot)$. The KKT systems regarding this stage are then represented by (6.7b) and expressed as:

$$C_i\Delta x_i + D_i\Delta u_i + E_i(\Delta t_k - \Delta t_{k-1}) + \bar{e}_i = 0, \tag{6.36a}$$

$$Q_{xx,i}\Delta x_i + Q_{xu,i}\Delta u_i + A_i^{\mathrm{T}}\Delta\lambda_{i+1} + C_i^{\mathrm{T}}\Delta\zeta_i - \Delta\lambda_i + h_{x,i}(\Delta t_k - \Delta t_{k-1}) + \tilde{l}_{x,i} = 0, \tag{6.36b}$$

and

$$Q_{xu,i}^{\mathrm{T}}\Delta x_i + Q_{uu,i}\Delta u_i + B_i^{\mathrm{T}}\Delta\lambda_{i+1} + D_i^{\mathrm{T}}\Delta\zeta_i + h_{u,i}(\Delta t_k - \Delta t_{k-1}) + \tilde{l}_{u,i} = 0, \tag{6.36c}$$

where

$$\tilde{l}_{x,i} := \tilde{r}_{x,i} + \nabla_x g^{\mathrm{T}}(x_i, u_i)\mathrm{diag}(z_i)^{-1}(\mathrm{diag}(\nu_i)r_{g,i} - r_{z,i})$$

and

$$\tilde{l}_{u,i} := \tilde{r}_{u,i} + \nabla_u g^{\mathrm{T}}(x_i, u_i)\mathrm{diag}(z_i)^{-1}(\mathrm{diag}(\nu_i)r_{g,i} - r_{z,i}).$$

In addition, $\bar{e}_i$ is residual in (6.35). In the backward recursion at stage i, we have (6.11a),

$$\begin{aligned}&\sum_{j\in\mathcal{I}_{k-1},\, j\leq i}\left(h_{x,j}^{\mathrm{T}}\Delta x_j + h_{u,j}^{\mathrm{T}}\Delta u_j + f_j^{\mathrm{T}}\Delta\lambda_{j+1} + \bar{h}_j\right) \\ &+ E_i^{\mathrm{T}}(\Delta\zeta_i + \zeta_i) + \Psi_{i+1}^{\mathrm{T}}\Delta x_{i+1} + \xi_{i+1}(\Delta t_{k-1} - \Delta t_{k-2}) \\ &+ \chi_{i+1}(-\Delta t_{k-1}) + \eta_{i+1} - \Phi_{i+1}^{\mathrm{T}}\Delta x_{i+1} \\ &- \chi_{i+1}(\Delta t_{k-1} - \Delta t_{k-2}) - \rho_{i+1}(-\Delta t_{k-1}) - \iota_{i+1} = 0,\end{aligned} \tag{6.37a}$$

and

$$\begin{aligned}
&\sum_{j \in \mathcal{I}_{k-2}} \left(h_{x,j}^{\mathrm{T}} \Delta x_j + h_{u,j}^{\mathrm{T}} \Delta u_j + f_j^{\mathrm{T}} \Delta \lambda_{j+1} + \bar{h}_j \right) \\
&- \sum_{j \in \mathcal{I}_{K-1},\, j \leq i} \left(h_{x,j}^{\mathrm{T}} \Delta x_j + h_{u,j}^{\mathrm{T}} \Delta u_j + f_j^{\mathrm{T}} \Delta \lambda_{j+1} + \bar{h}_j \right) \\
&- \Psi_{i+1}^{\mathrm{T}} \Delta x_{i+1} - \xi_{i+1} (\Delta t_{k-1} - \Delta t_{k-2}) - \chi_{i+1} (-\Delta t_{k-1}) - \eta_{i+1} + E_i^{\mathrm{T}} (\Delta \zeta_i + \zeta_i) = 0.
\end{aligned} \tag{6.37b}$$

Δu_i and $\Delta \zeta_i$ are then eliminated, resulting in:

$$\begin{bmatrix} \Delta u_i \\ \Delta \zeta_i \end{bmatrix} = \begin{bmatrix} K_i^{\mathrm{T}} \\ M_i \end{bmatrix} \Delta x_i + \begin{bmatrix} T_i \\ L_i \end{bmatrix} (\Delta t_{k-1} - \Delta t_{k-2}) + \begin{bmatrix} W_i \\ N_i \end{bmatrix} (-\Delta t_{k-1}) + \begin{bmatrix} k_i \\ m_i \end{bmatrix}, \tag{6.38a}$$

where

$$\begin{bmatrix} K_i \\ M_i \end{bmatrix} := - \begin{bmatrix} G_i & D_i^{\mathrm{T}} \\ D_i & \end{bmatrix}^{-1} \begin{bmatrix} H_i^{\mathrm{T}} \\ C_i \end{bmatrix}, \tag{6.38b}$$

$$\begin{bmatrix} T_i \\ L_i \end{bmatrix} := - \begin{bmatrix} G_i & D_i^{\mathrm{T}} \\ D_i & \end{bmatrix}^{-1} \begin{bmatrix} \psi_{u,i} \\ E_i \end{bmatrix}, \tag{6.38c}$$

$$\begin{bmatrix} W_i \\ N_i \end{bmatrix} := - \begin{bmatrix} G_i & D_i^{\mathrm{T}} \\ D_i & \end{bmatrix}^{-1} \begin{bmatrix} \phi_{u,i} \\ 0 \end{bmatrix}, \tag{6.38d}$$

and

$$\begin{bmatrix} k_i \\ m_i \end{bmatrix} := - \begin{bmatrix} G_i & D_i^{\mathrm{T}} \\ D_i & \end{bmatrix}^{-1} \begin{bmatrix} B_i{}^{\mathrm{T}} (P_{i+1} \bar{x}_i - s_i) + \tilde{l}_{u,i} \\ \bar{e}_i \end{bmatrix}. \tag{6.38e}$$

Therefore, we obtain the form equations of (6.14a), (6.15a), and (6.15b) for stage i with the following:

$$P_i = F_i - \begin{bmatrix} K_i \\ M_i \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} G_i & D_i^{\mathrm{T}} \\ D_i & \end{bmatrix} \begin{bmatrix} K_i \\ M_i \end{bmatrix}, \tag{6.39a}$$

$$s_i = - \left\{ \tilde{l}_{x,i} + A_i^{\mathrm{T}} (P_{i+1} \bar{x}_i - s_{i+1}) + H_i k_i + C_i^{\mathrm{T}} m_i \right\}, \tag{6.39b}$$

and

$$\Psi_i = \psi_{x,i} + \begin{bmatrix} K_i \\ M_i \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} \psi_{u,i} \\ E_i \end{bmatrix}, \quad \Phi_i = \phi_{x,i} + \begin{bmatrix} K_i \\ M_i \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} \phi_{u,i} \\ 0 \end{bmatrix}. \tag{6.39c}$$

$$\xi_i = \xi_{i+1} + f_i^{\mathrm{T}} (P_{i+1} f_i + 2 \Psi_{i+1}) + \psi_{u,i}^{\mathrm{T}} T_i + E_i^{\mathrm{T}} L_i, \tag{6.40a}$$

$$\eta_i = \eta_{i+1} + \bar{h}_i + E_i^{\mathrm{T}} \zeta_i + f_i^{\mathrm{T}} (P_{i+1} \bar{x}_i - s_{i+1}) + \Psi_{i+1}^{\mathrm{T}} \bar{x}_i + \psi_{u,i}^{\mathrm{T}} k_i + E_i^{\mathrm{T}} m_i, \tag{6.40b}$$

$$\chi_i = \chi_{i+1} + \Phi_{i+1}^{\mathrm{T}} f_i + \psi_{u,i}^{\mathrm{T}} W_i + E_i^{\mathrm{T}} N_i, \tag{6.40c}$$

$$\rho_i = \rho_{i+1} + \phi_{u,i}^{\mathrm{T}} W_i, \tag{6.40d}$$

and

$$\iota_i = \iota_{i+1} + \Phi_{i+1}^{\mathrm{T}} \bar{x}_i + \phi_{u,i}^{\mathrm{T}} k_i. \tag{6.40e}$$

At the forward recursion, Δu_i and $\Delta \zeta_i$ are computed using (6.38a) instead of only computing Δu_i as (6.13a).

6.6 Numerical Experiments

6.6.1 Comparison with off-the-shelf solvers

6.6.1.1 Problem settings

The effectiveness of the proposed method was demonstrated through two numerical experiments. The first experiment was a comparison with off-the-shelf NLP solvers on the switched system, consisting of three nonlinear subsystems treated in Xu and Antsaklis (2002); Xu and Antsaklis (2004). The dynamics of the subsystems are expressed as:

$$f_1(x,u) = \begin{bmatrix} x_1 + u_1 \sin(x_1) \\ -x_2 - u_1 \cos(x_2) \end{bmatrix},$$

$$f_2(x,u) = \begin{bmatrix} x_2 + u_1 \sin(x_2) \\ -x_1 - u_1 \cos(x_1) \end{bmatrix},$$

and

$$f_3(x,u) = \begin{bmatrix} -x_1 - u_1 \sin(x_1) \\ x_2 + u_1 \cos(x_2) \end{bmatrix}.$$

The stage cost is expressed as:

$$l_k(x,u) = \frac{1}{2}||x - x_{\rm ref}||^2 + ||u||^2, \ \ k \in \{1,2,3\}.$$

In addition, the terminal cost is expressed as:

$$V_f(x) = \frac{1}{2}||x - x_{\rm ref}||^2,$$

where $x_{\rm ref} = [1, \ -1]^{\rm T}$. The initial and terminal times of the horizon are denoted by $t_0 = 0$ and $t_3 = 3$, respectively. The initial state is denoted by $x(t_0) = [2, \ 3]^{\rm T}$.

Benchmarks were set for the solution times of the NLP (6.4) example against the Ipopt (Wächter and Biegler (2006)) and a SQP methods with qpOASES (Ferreau et al. (2014)), both of which were used through the CasADi (Andersson, Gillis, Horn, Rawlings, and Diehl (2019)) interface. The mesh-refinement was not considered in this comparison in order to focus on a pure comparison of the abilities to solve the NLP problems. That is, only the single NLP step of Algorithm 6.1 was considered. Ipopt was used with the default settings of CasADi. For example, MUMPS (Amestoy, Duff, Koster, and L'Excellent (2001)) was used to solve the linear systems. The built-in SQP solver of CasADi was used with the exact Hessian matrix and qpOASES as the backend QP solver. The Hessian regularization was enabled within the qpOASES options and the Hessian type was set to "indefinite". The proposed method was written in C++, used Eigen (Guennebaud et al. (2010)) for the linear algebra, and

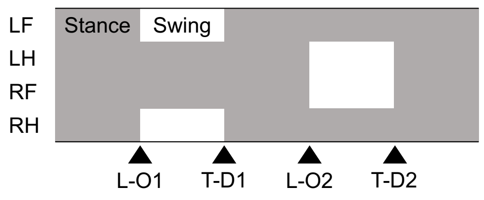
Table 6.1: Average computational time (ms) of each solver for different total number of grids

N	Proposed	Ipopt	SQP with qpOASES
10	0.08	4.3	failed
50	0.27	24.6	failed
100	0.47	45.1	failed
500	1.93	127	failed

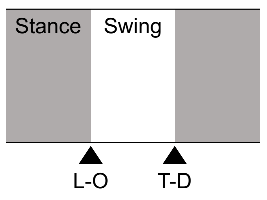
used an exact Hessian matrix and reduced Hessian modifications (6.24) and (6.25) throughout the experiments. $\sigma_{\min}$ and $\bar{\sigma}$ were chosen using (6.26) with $\Delta t_{k,\max} = 0.5$. The proposed method only used the fraction-to-boundary rule (Nocedal and Wright (2006); Wächter and Biegler (2006)) for the step size selection and monotonically decreased the barrier parameter ϵ when the l_2 norm of the KKT residual (in the perturbed KKT conditions (6.6)) was smaller than 0.1. The solution times were compared for these algorithms with the total number of grids set as N = 10, 50, 100, and 500. N was divided into N_1, N_2, and N_3, so that each phase had an almost equal number of discretization grids. The values of $[N_1, N_2, N_3]$ were set to $[4, 3, 3]$, $[17, 17, 16]$, $[34, 33, 33]$, and $[167, 167, 166]$ for N = 10, 50, 100, and 500, respectively. The minimum dwell-times were set as $\underline{\Delta}_1 = \underline{\Delta}_2 = \underline{\Delta}_3 = 0.01$ in the constraints (6.4e) to avoid ill-conditioned problems. The initial guess of the solution was set as $x_0 = x_1 = \cdots = x_N = x(t_0)$, $u_0 = u_1 = \cdots u_{N-1} = 0$, $t_1 = 1.0$, and $t_2 = 2.0$ for all solvers. It was assumed that the proposed method converged when the solution satisfied the default convergence criteria of Ipopt (max norm of the residual in the unperturbed KKT conditions (Nocedal and Wright (2006); Wächter and Biegler (2006))). All solvers were run 100 times for each N on a laptop with a quad-core CPU Intel Core i7-10510U @1.80 GHz, and the average computational time until convergence was measured.

6.6.1.2 Results

The average computational times of each solver for the different total number of grids N are listed in Table 6.1. As shown in the table, the proposed method converged the fastest of all the cases, and was up to two orders of magnitude faster than Ipopt. The proposed method was extremely fast because the dimension of the control input of each subsystem was 1 and it did not involve any explicit matrix inversions. The SQP method with qpOASES failed to converge for all cases, even the regularization and



(a) Trotting



(b) Jumping

Figure 6.2: Contact patterns of trotting and jumping motions. Gray and white cells indicate the intervals where the leg is standing and swinging, respectively. LF, LH, RF, and RH denote left-front, left-hind, right-front, and right-hind foot in contact, respectively. Black triangles indicate the switches (lift-off: L-O, touch-down: T-D).

indefinite-Hessian mode were enabled. Note that another QP solver OSQP (Stellato et al. (2020)) was also tested, but it failed to treat the indefinite Hessian matrix.

The total computational time of Algorithm 6.1 until convergence was also measured, including the mesh-refinement steps. The thresholds for the discretization step sizes were $\Delta\tau_{\max} = 0.35$, 0.065, 0.035, and 0.0065 for cases $N = 10$, 50, 100, and 500, respectively. The total computational times of Algorithm 6.1 for $N = 10$, 50, 100, and 500, were 0.13, 0.57, 0.92, and 4.3 ms, respectively. The mesh-refinements were only conducted a few times in each case, and a similar solution was achieved compared to that of the continuous-time counterpart reported in Xu and Antsaklis (2004). As expected, the solution accuracy improved as N increased.

6.6.2 Whole-body optimal control of quadrupedal gaits

6.6.2.1 Problem settings

Numerical experiments on the whole-body optimal control of a quadrupedal robot ANYmal (Hutter et al. (2017)) were conducted to demonstrate the efficiency of the proposed method in practical and complicated examples. A trotting motion with a step length of 0.15 m and aggressive jumping motion with a jump length of 0.8 m also investigated. The contact patterns of the trotting and jumping motions are shown in Fig. 6.2. The state equation of the robot switched based on the combination of the support feet, that is, it switched when the feet lifted off from the ground or touched down onto the ground. Therefore, the trotting motion had four switches and the jump motion had two switches over the horizon, as shown in Fig. 6.2. The lift-off did not include the state jump or the switching condition, but the touch-down included both (Farshidian, Kamgarpour, et al. (2017); Li and Wensing (2020); Schultz and Mombaur (2010)). The switching condition were treated using the proposed method in Section

6.5.1: the switching conditions were position constraints on the foot whose Jacobians took the form of (6.33) and the state equation of the robot under an acceleration-level rigid-contact constraint took the form of (6.32). The details of the above formulations can be found in Li and Wensing (2020); Schultz and Mombaur (2010). Note also that the acceleration-level contact-consistent dynamics of the robot were lifted to improve the convergence speed without changing the structure of the KKT system (6.7) (Katayama and Ohtsuka (2021a)). To consider a practical situation, the limitations that were imposed on the joint angles, velocities, and torques were considered to be the inequality constraints. The polyhedral-approximated friction cone constraint was also considered for each contact force expressed in the world frame $[f_x \;\; f_y \;\; f_z]$ as:

$$\begin{bmatrix} f_x + \frac{\mu}{\sqrt{2}} f_z \\ -f_x + \frac{\mu}{\sqrt{2}} f_z \\ f_y + \frac{\mu}{\sqrt{2}} f_z \\ -f_y + \frac{\mu}{\sqrt{2}} f_z \\ f_z \end{bmatrix} \geq 0, \tag{6.41}$$

where $\mu > 0$ is the friction coefficient, which was set as $\mu = 0.7$. Note that the friction cone constraint (6.41) also switched depending on the active subsystems (that is, depending on the combination of the support feet). The initial and terminal times of the horizon were set as $t_0 = 0$ and $t_5 = 1.0$ for the trotting problem and $t_0 = 0$ and $t_2 = 1.7$ for the jumping problem, respectively.

To design the stage cost of the trotting motion, the state $x \in \mathbb{R}^{36}$ was partitioned into the generalized coordinate and velocity $[q^{\mathrm{T}} \;\; v^{\mathrm{T}}]^{\mathrm{T}}$, $q, v \in \mathbb{R}^{18}$. Additionally, the position (i.e., forward kinematics) of the foot $i \in \{$Left-front, Left-hip, Right-front, Right-hip$\}$ was considered in the world frame $p_i(q) \in \mathbb{R}^3$. The stage cost was then designed for the trotting motion, which is expressed as:

$$\begin{aligned} &\frac{1}{2}(q - q_{\mathrm{ref}})^{\mathrm{T}} W_q (q - q_{\mathrm{ref}}) + \frac{1}{2}(v - v_{\mathrm{ref}})^{\mathrm{T}} W_v (v - v_{\mathrm{ref}}) + \frac{1}{2} a^{\mathrm{T}} W_a a \\ &+ \sum_{i \in \{\text{Swing legs}\}} \frac{1}{2}(p_i(q) - p_{i,\mathrm{ref}})^{\mathrm{T}} W_p (p_i(q) - p_{i,\mathrm{ref}}), \end{aligned} \tag{6.42}$$

where $a \in \mathbb{R}^{18}$ is the generalized acceleration, $q_{\mathrm{ref}}, v_{\mathrm{ref}} \in \mathbb{R}^{18}$ and $p_{i,\mathrm{ref}} \in \mathbb{R}^3$ are reference values, and $W_q, W_v, W_a \in \mathbb{R}^{18 \times 18}$ and $W_p \in \mathbb{R}^{3 \times 3}$ are diagonal weight matrices. q_{ref} was chosen as the generalized coordinate of the standing pose of the robot. We set the desired translational velocity of the floating base of the robot as v_{ref} and set the other element of v_{ref} as zero. The plane coordinate of $p_{i,\mathrm{ref}}$ was set as a middle point of the two successive predefined ground locations of the foot, and its vertical coordinate was set as the desired height of the swing foot. The last term in (6.42) changed depending on the swing legs, that is, the stage cost also switched as the active subsystem. The terminal cost $V_f(x_N)$ and impulse cost $l(x_{i_k-})$ were set as the sums of the first and second terms of (6.42), respectively. Similarly, the stage cost was designed for the jumping motions, which is expressed as:

$$\frac{1}{2}(q - q_{\rm ref})^{\rm T} W_q (q - q_{\rm ref}) + \frac{1}{2} v^{\rm T} W_v v + \frac{1}{2} a^{\rm T} W_a a. \tag{6.43}$$

In addition, the terminal cost $V_f(x_N)$ and impulse cost $l(x_{i_k-})$ were set as the sums of the first and second terms of (6.43), respectively, but with different weight parameters. The minimum-dwell times in (6.4e) for the trotting motion were set as $\underline{\Delta}_1 = \underline{\Delta}_3 = \underline{\Delta}_5 = 0.02$ and $\underline{\Delta}_2 = \underline{\Delta}_4 = 0.2$. The minimum-dwell times were set for the jumping motion as $\underline{\Delta}_1 = \underline{\Delta}_2 = 0.15$ and $\underline{\Delta}_3 = 0.65$. A large $\underline{\Delta}_3$ was set in the jumping motion to sufficiently observe the whole-body motion after touch-down. This was because the optimizer tended to make the touch-down time very close to the end of the horizon to minimize the overall cost of the OCP without a large $\underline{\Delta}_3$.

The proposed algorithms were implemented in C++ and Eigen was used for the linear algebra. The efficient Pinocchio C++ library was used (Carpentier et al. (2019)) for rigid-body dynamics (Featherstone (2008)) and its analytical derivatives (Carpentier and Mansard (2018)) in order to compute the dynamics and its derivatives of the quadrupedal robot. The Gauss-Newton Hessian approximation was used to avoid computing the second-order derivatives of the rigid-body dynamics. The reduced Hessian modifications (6.24) and (6.25) were used in the proposed Riccati recursion. $\sigma_{\rm min}$ and $\bar{\sigma}$ were chosen using (6.26) with $\Delta t_{k,\rm max} = 0.1$. Only the fraction-to-boundary rule (Nocedal and Wright (2006); Wächter and Biegler (2006)) was used for the step size selection. The barrier parameter was fixed at $\epsilon = 1.0 \times 10^{-3}$, which is a common suboptimal MPC setting (Wang and Boyd (2010)). OpenMP (Dagum and Menon (1998)) was used for parallel computing of the KKT system (6.7) in stage-wise and eight threads through the following experiments. These two experiments were conducted on a desktop computer with an octa-core CPU Intel Core i9-9900 @3.10 GHz. $\Delta\tau_{\rm max} = 0.02$ was used for the mesh refinement. The total number of horizon grids were fixed to $N = 50$ and $N = 90$ for the trotting and jumping problems, respectively. That is, when grids were added to the mesh-refinement phase, the same number of grids were removed from the other phases. The proposed method converged when the l_2 norm of the KKT residual was smaller than a sufficiently small threshold 1.0×10^{-7} and each $\Delta\tau_k$ was smaller than $\Delta\tau_{\rm max}$. Mesh refinement was performed when the l_2 norm of the KKT residual became moderately small (smaller

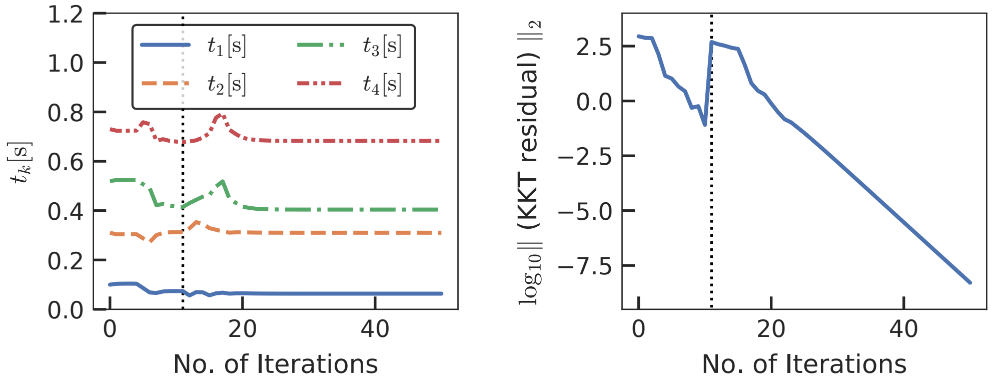


Figure 6.3: Convergence of the proposed method for the trotting motion of a quadrupedal robot: (a) switching instants and (b) l_2 norm of the residual in the perturbed KKT conditions over the iterations. Vertical dotted lines indicate that the mesh refinement was carried out.

than 0.1). This implementation is available online as a part of our efficient optimal control solvers for robotic systems, called robotoc (Katayama (2020-2022)).

6.6.2.2 Results

The convergence results of the trotting and jumping motions are shown in Figs. 6.3 and 6.4, respectively. For the trotting motion that included a mesh-refinement step, the proposed method converged after 46 iterations. The total computational time was 60 ms (1.3 ms per Newton iteration). For the jumping motion that included two mesh-refinement steps, the proposed method converged after 147 iterations. The total computational time was 308 ms (2.1 ms per Newton iteration). The snapshots of the solution trajectory of the aggressive jumping motion are shown in Fig. 6.5, and the time histories of the contact forces of the four feet in the jumping motion are shown in Fig. 6.6. We can see that the contact forces lie inside the friction cone constraints throughout the jumping motion. These two figures show that the friction cone constraints (6.41) were active at time intervals immediately before and after the jumping, and the solution strictly satisfied the constraints.

The above results showed that the proposed method could achieve fast convergence and computational times per Newton iteration, even for large-scale (36-dimensional state, 12-dimensional control input) and complicated problems. In MPC, the shorter length of the horizon and smaller number of discretization grids can typically be considered, and warm-starting can be leveraged. For example, it can be expected that the computational time per Newton iteration for the trotting problem would be

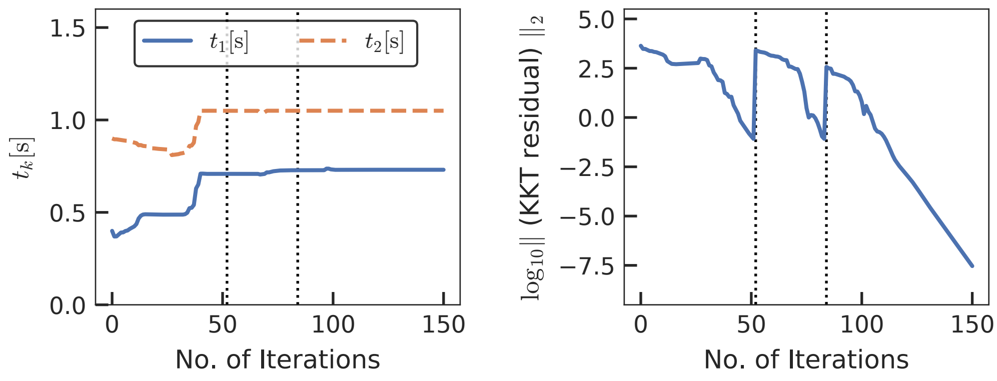


Figure 6.4: Convergence of the proposed method for the jumping motion of the quadrupedal robot: (a) switching instants and (b) l_2 norm of the residual in the perturbed KKT conditions over the iterations. Vertical dotted lines indicate that the mesh refinement was carried out.

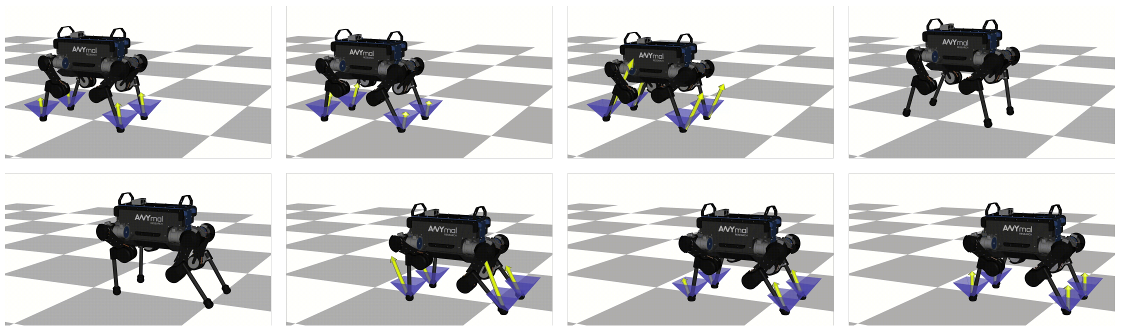


Figure 6.5: Snapshots of solution trajectory of the whole-body optimal control of ANYmal's 0.8 m aggressive jumping motion. The yellow arrows and blue polyhedrons represent the contact forces and friction cone constraints, respectively, which are not illustrated while the robot is flying.

less than half of 1.3 ms if the number of grids was reduced to 20. Therefore, the proposed method is a promising approach that can achieve the whole-body MPC of robotics systems with rigid contacts within a milliseconds-range sampling time.

6.7 Summary

This chapter proposed an efficient Newton-type method for optimal control of switched systems under a given mode sequence. A direct multiple-shooting method with adaptive mesh refinement was formulated to guarantee the local convergence of the Newton-type method for the NLP. A dedicated structure-exploiting Riccati recursion algorithm was proposed that performed the Newton-type method for the NLP with the linear time-complexity of the total number of discretization grids. Moreover, the

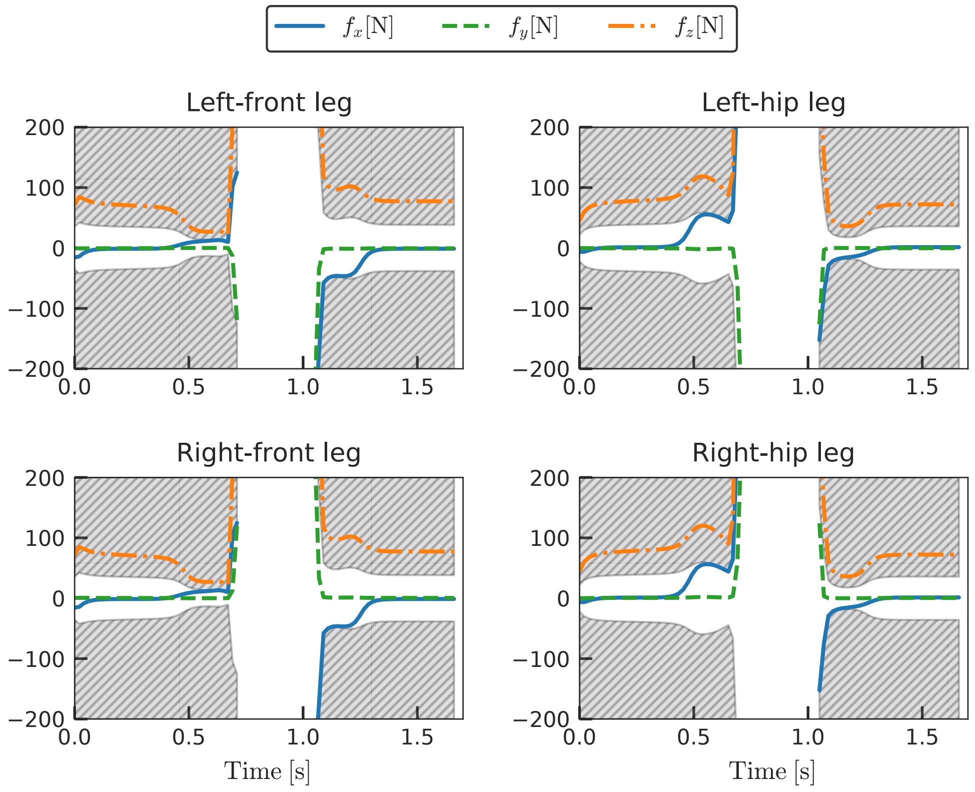


Figure 6.6: Time histories of the contact force expressed in the world frame $[f_x \ f_y \ f_z]$ of each leg in the jumping motion. The infeasible regions of f_x (solid lines) and f_y (dashed lines) due to the friction cone constraints are the filled gray hatches. The infeasible region of $f_z \geq 0$ (dash-dotted lines) is in the lower-half of each plot.

proposed method could always solve the KKT systems if the solution was sufficiently close to a local minimum. Conversely, general QP solvers cannot be guaranteed to solve the KKT systems because the Hessian matrix is inherently indefinite. Moreover, to enhance the convergence, a modification on the reduced Hessian matrix was proposed using the nature of the Riccati recursion algorithm as the dynamic programming for a QP subproblem. A numerical comparison was conducted with off-the-shelf solvers and demonstrated that the proposed method was up to two orders of magnitude faster. Whole-body optimal control of quadrupedal gaits was also investigated and it was demonstrated that the proposed method could achieve the whole-body MPC of robotic systems with rigid contacts.

A possible future extension of the proposed method is OCPs with free-final time, including minimum-time OCPs (Bryson and Ho (1975)), because the NLP structure of such problems is expected to be similar to the proposed formulation.

Algorithm 6.2 Computation of Newton step via proposed Riccati recursion

```
Input: Initial state $x(t_0)$ and the current iterate $x_0$, ..., $x_N$, $x_s$, $u_0$, ..., $u_{N-1}$, $u_s$, $\lambda_0$, ..., $\lambda_N$, $\lambda_s$, and $t_s$.
Output: Newton directions $\Delta x_0$, ..., $\Delta x_N$, $\Delta x_s$, $\Delta u_0$, ..., $\Delta u_{N-1}$, $\Delta u_s$, $\Delta\lambda_0$, ..., $\Delta\lambda_N$, $\Delta\lambda_s$, and $\Delta t_s$.
 1: Compute the KKT system (6.7).
 2: // Backward recursion
 3: Compute $P_N$ and $z_N$ from (6.10).
 4: for $k = K+1, \cdots, 1$ do
 5:     for $i = \max \mathcal{I}_k, \cdots, \min \mathcal{I}_k$ do
 6:         Compute $K_i$, $k_i$, $T_i$, and $W_i$ from (6.13) .
 7:         Compute $P_i$, $z_i$, $\Psi_i$, and $\Phi_i$ from (6.14).
 8:         Compute $\xi_i$, $\xi_i$, $\rho_i$, $\eta_i$, and $\iota_i$, from (6.16).
 9:     end for
10:     if $k < K+1$ then
11:         Compute $\tilde{P}_{i_k}$, $\tilde{s}_{i_k}$, $\tilde{\Phi}_{i_k}$, $\tilde{\rho}_{i_k}$, and $\tilde{\iota}_{i_k}$ from (6.20) optionally with modifications (6.24) and (6.25).
12:     end if
13:     Set $\Psi_{i_k}$, $\xi_{i_k}$, $\chi_{i_k}$, and $\iota_{i_k}$ to zero.
14: end for
15: // Forward recursion
16: Compute $\Delta x_0$ from (6.7a).
17: Compute $\Delta t_1$ from (6.18).
18: for $k = 1, \cdots, K+1$ do
19:     for $i = \min \mathcal{I}_k, \cdots, \max \mathcal{I}_k$ do
20:         Compute $\Delta u_i$ and $\Delta\lambda_i$ from (6.13) and (6.14a), respectively.
21:         Compute $\Delta x_{i+1}$ from (6.7b)
22:     end for
23:     if $k < K+1$ then
24:         Compute $\Delta t_{k+1}$ from (6.18)
25:     end if
26: end for
27: Compute $\Delta\lambda_N$ from (6.10).
28: Compute $\Delta z_i$, $\Delta\nu_i$, $\Delta w_i$, and $\Delta v_i$ for $i = 0, ..., N-1$ from the other Newton steps.
```

Chapter 7

Whole-Body Model Predictive Control with Rigid Contacts via Online Switching Time Optimization[1]

7.1 Introduction

A fundamental difficulty in controlling robotic systems lies in discrete events involved in their dynamics. Essential tasks of robots such as manipulation and locomotion involve contacts with the environment. Making and breaking contacts result in switches of their dynamics and impulsive changes in their state. Therefore, robot systems are inherently modeled as hybrid dynamical systems. Furthermore, some classes of robots such as quadrupedal robots and humanoid robots are underactuated. Owing to these difficulties, most analytical model-based controllers focus on only particular classes of problems. For example, the hybrid zero dynamics controller is a successful approach focusing on bipedal locomotion (Reher and Ames (2021); Westervelt et al. (2003)). However, once we consider more complicated cases: acrobatic jumping, walking on uneven terrain, and multi-contact situations in the real world, such analytical construction of controllers are still difficult.

Model predictive control (MPC) (Rawlings et al. (2017)), which solves trajectory optimization problems online, is expected to be a unified model-based approach to control the robotic systems with contacts; hence, if we can solve the MPC problem

[1]© 2022 IEEE. A substantial portion of this chapter including Figures 7.1–7.8 is reprinted, with permission, from S. Katayama and T. Ohtsuka, "Whole-body model predictive control with rigid contacts via online switching time optimization," 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2022), 2022.

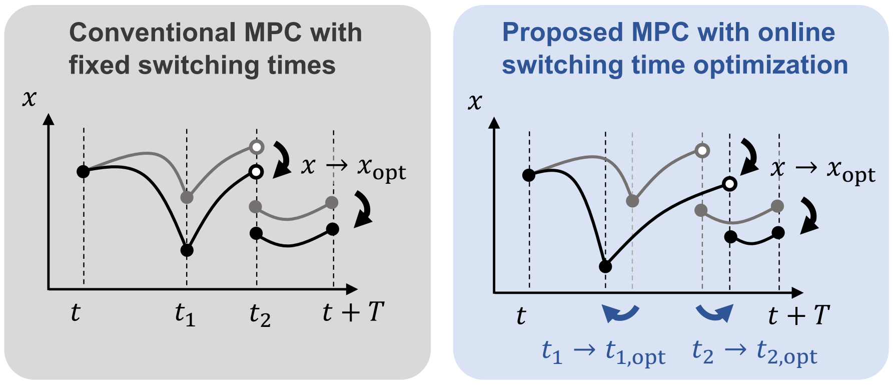

Conventional MPC with fixed switching times
x
$x \to x_{\mathrm{opt}}$
t
t_1
t_2
$t+T$
Proposed MPC with online switching time optimization
x
$x \to x_{\mathrm{opt}}$
t
$t+T$
$t_1 \to t_{1,\mathrm{opt}}$
$t_2 \to t_{2,\mathrm{opt}}$

Figure 7.1: Conceptual diagram of comparison between (left) the conventional model predictive control (MPC) with fixed switching times and (right) the proposed MPC with online switching time optimization. The proposed MPC optimizes the switching times t_1, t_2 and trajectory x simultaneously, while the conventional MPC solely optimizes the trajectory under the fixed switching times.

for the hybrid systems online, we can realize the robot control systems considering the future discrete event (e.g., interactions with the environment). However, the hybrid MPC problem is generally formulated as a mixed-integer nonlinear program (MINLP), which is an NP-hard problem owing to its combinatorial nature (Belotti et al. (2013)). A remarkable approach that can avoid the combinatorial nature is the contact-implicit trajectory optimization (CITO) (Yunt (2011)), which is often formulated as mathematical programs with complementarity constraints (MPCC) (Posa et al. (2014)) or bi-level optimization problems (Carius et al. (2018)). However, MPCC inherently lacks constraint qualifications such as the linear independence constraint qualification (LICQ), and this leads to numerical ill-conditioning. In addition, numerical alleviations for the LICQ problem often cause undesirable stationary points (Nurkanović et al. (2020)); thus, it is difficult to even solve MPCC off-line. Bi-level optimization is also difficult to solve in general, for example, even a linear bi-level optimization problem is NP-hard (Hansen et al. (1992)). There are other heuristic approximation methods toward CITO (Chatzinikolaidis et al. (2020); Neunert et al. (2018); Todorov (2014)), which can be seen as kinds of soft contact models. However, once the problem involves rigid contacts, these approximation methods can cause non-physical artifacts (Acosta et al. (2022)) or their convergence performance can be significantly decreased.

Another tractable and still practical approach is to fix the contact sequence. In practical situations, a robot has a high-level planner that computes the feasible contact sequence, considering information from perception systems (e.g., safe regions in terrain for foot placements) (Kuindersma et al. (2016); Mastalli et al. (2020)). A low-level motion planner or the MPC controller then optimizes the trajectory, based on the contact sequence. In this case, the low-level optimization problem is defined, to determine the trajectory and instants of each discrete event (making or breaking the contacts). The optimization problem to determine the instants of discrete events is particularly known as the switching time optimization (STO) problem. However, it has been difficult to solve the optimal control problem (OCP) involving the STO problem efficiently (Fu and Zhang (2021); Xu and Antsaklis (2004)). Accordingly, the application of the STO approach remains the offline trajectory optimization of simplified robot models (Farshidian, Kamgarpour, et al. (2017); Li and Wensing (2020)). Therefore, recent successful applications of MPC to robotic systems have still been limited to the case with the fixed switching instants (Li, Frei, and Wensing (2021); Wolfslag et al. (2020)). However, in practical situations, the behavior of the robot can be different from the predicted one, and the pre-computed switching instants can become infeasible or unreasonable during control, due to model mismatches, disturbances, and dynamic changes in the environment that are not predicted in advance. In such cases, the control performance deteriorates or the MPC optimizer fails to converge to a feasible solution.

It is worth noting that there are heuristics to adapt the contact timings online (Caron and Pham (2017); Smaldone, Scianca, Lanari, and Oriolo (2021)). However, such heuristics can lack versatility; the resultant motions can be conservative and cannot treat multi-objective cost and constraints (for example, the approach of Caron and Pham (2017) does not consider the friction cone constraints). Moreover, the MPC optimizer can still fail to determine a feasible solution with the heuristic timings.

In this chapter, we propose a whole-body MPC of robots with rigid contacts using the online STO. Figure 7.1 illustrates the conceptual diagram of the comparison between the existing MPC framework and proposed MPC. We utilize an efficient Newton-type algorithm for the OCPs involving STO problems proposed in Chapter 6. The algorithm enables us to efficiently optimize the trajectory and switching times simultaneously. However, the applicability of the method for MPC is not discussed in Chapter 6, it only demonstrates offline trajectory optimization problems involving the STO problem. This chapter implements MPC utilizing the STO algorithm, and proposes several ways to improve the numerical robustness of the MPC, e.g., heuristic regularization and minimum dwell-time constraints. We conducted numerical simulations on dynamic jumping of a quadrupedal robot and demonstrated that the proposed whole-body MPC with online STO successfully controls the dynamic motions, while conventional MPC with fixed switching times cannot find a feasible solution, thereby failing in the control. This advantage is observed in various simulation settings, which demonstrates that the proposed method extends the ability of MPC for robots with rigid contacts. We further conducted hardware experiments of our MPC with online STO on quadruped robot Unitree A1 (Unitree Robotics (n.d.)), and demonstrated that the proposed MPC with online STO achieved dynamic control of a real robotic system that involves disturbances and model mismatches.

The contributions of this chapter are then summarized as follows:

- To the best of our knowledge, this is the first study that realizes the whole-body MPC of robots with rigid contacts using the online STO.
- We conducted simulation studies on dynamic jumping control of a quadrupedal robot, and demonstrated that the proposed whole-body MPC with the online STO extends the ability of MPC for robots with rigid contacts.
- We conducted hardware experiments of our MPC with online STO on quadruped robot Unitree A1.

The remainder of this chapter is organized as follows: Section 7.2 models the robot dynamics with rigid contacts as a switched system and formulates its OCP. Section 7.3 describes the MPC with online STO for the OCP described in Section 7.2. Section 7.4 demonstrates the effectiveness of the proposed MPC with online STO over the conventional MPC with fixed switching times, through numerical simulation of the whole-body control of a quadruped robot jumping. Section 7.5 conducts hardware experiments on the quadrupedal robot Unitree A1 and demonstrates that the proposed method achieves dynamic motions on the real robot. Finally, a brief summary and mention of future studies are presented in Section 7.6.

7.2 Optimal Control Problem Formulation

7.2.1 Rigid body systems with rigid contacts

First, we model a rigid body system with rigid contacts (a robot having contacts with the environment) as a switched system, and formulate its OCP, as originally presented in Schultz and Mombaur (2010) and further studied in Budhiraja et al. (2018); Li and Wensing (2020); Mastalli et al. (2020). To formulate an OCP of the switched systems without the combinatorial nature or complementarity constraints, we assume that the contact sequence (the sequence of the active contacts) is given, for example, it is provided by a higher-level planner. Let $q, v, a \in \mathbb{R}^n$, $f \in \mathbb{R}^{n_f}$, and $u \in \mathbb{R}^m$ be the configuration, generalized velocity, acceleration, stack of the contact forces, and joint torques, respectively. Note that the presented formulation can consider the configuration space including SE(3) to model the floating base as Katayama and Ohtsuka (2021a), while this chapter assumes the Euclidian configuration space for notational simplicity. Also note that the contact dimension n_f alters depending on combinations of the active contacts. The equation of motion of the rigid-body system is then expressed as:

$$M(q)a + h(q, v) - J^{\mathrm{T}}(q)f = S^{\mathrm{T}}u, \tag{7.1}$$

where $M(q) \in \mathbb{R}^{n \times n}$ denotes the inertia matrix, $h(q, v) \in \mathbb{R}^n$ encompasses the Coriolis, centrifugal, and gravitational terms, $J(q) \in \mathbb{R}^{n_f \times n}$ denotes the stack of the contact Jacobians, and $S \in \mathbb{R}^{m \times n}$ denotes the selection matrix. Because the contact sequence is provided, we can treat the contact constraints as a bilateral constraint of the form

$$\mathbf{p}(q) = 0, \tag{7.2}$$

where $\mathbf{p}(q) \in \mathbb{R}^{n_f}$ is the stack of the positions (and includes rotations for active surface contacts) of the active contact frames. Furthermore, instead of considering (7.2) over a time interval, we consider the acceleration-level constraint over the time interval (Baumgarte (1972)):

$$\mathbf{a}(q, v, a) := \ddot{\mathbf{p}} + 2\alpha\dot{\mathbf{p}} + \beta^2\mathbf{p} = J(q)a + \mathbf{b}(q, v), \tag{7.3a}$$

where α and β are weight parameters, and we define

$$\mathbf{b}(q, v) := \dot{J}(q, v)v + 2\alpha J(q)v + \beta^2\mathbf{p}(q). \tag{7.3b}$$

Then the original position constraint (7.2) is satisfied over the time interval provided that (7.2) and the equality constraint on the contact velocity,

$$\dot{\mathbf{p}}(q, v) = J(q)v = 0, \tag{7.4}$$

are satisfied at some point (Flores et al. (2011)). Equation (7.3) is reduced to twice the time derivative of (7.2) with $\alpha = \beta = 0$; however, the constraint violation of the original position constraint (7.2) can be accumulated because of the numerical computation. $\alpha = \beta > 0$ is typically chosen to stabilize the violation of the original constraint (7.2). By combining (7.1) and (7.3), we obtain the contact-consistent forward dynamics:

$$\begin{bmatrix} a \\ -f \end{bmatrix} = \begin{bmatrix} M(q) & J^{\mathrm{T}}(q) \\ J(q) & O \end{bmatrix}^{-1} \begin{bmatrix} S^{\mathrm{T}}u - h(q, v) \\ -\mathbf{b}(q, v) \end{bmatrix}. \tag{7.5}$$

We define the state vector and state equation as

$$x := \begin{bmatrix} q \\ v \end{bmatrix}, \tag{7.6}$$

and we have a state equation in the form of

$$\dot{x} = f(x, u) := \begin{bmatrix} v \\ a(x, u) \end{bmatrix}, \tag{7.7}$$

where $a(x, u) \in \mathbb{R}^n$ is a function of x and u as defined in (7.5).

We would like to emphasize that the state equation (7.7) switches depending on the combination of active contacts, which we also refer to *contact mode* in the following. That is, the rigid body system with contacts is a switched system. We express the state equations (7.7) for each contact mode as

$$\dot{x} = f_k(x, u),$$

where k denotes the index of the contact mode. We refer to k as the index of an active subsystem of the switched system in the following.

When the bodies of the system and environment collide, the generalized velocity of the system alters according to the equation of Newton's law of impact:

$$M(q) \delta v - J^{\mathrm{T}}(q) \Lambda = 0, \tag{7.8}$$

where $\delta v \in \mathbb{R}^n$ denotes the impulsive change in the generalized velocity and $\Lambda \in \mathbb{R}^{n_f}$ denotes the stack of the impact forces. The evolution of the state between the impact is expressed as

$$\begin{bmatrix} q^+ \\ v^+ \end{bmatrix} = \begin{bmatrix} q^- \\ v^- + \delta v \end{bmatrix}, \tag{7.9}$$

where $q^-, v^- \in \mathbb{R}^n$ denote the configuration and velocity immediately before the impulse and $q^+, v^+ \in \mathbb{R}^n$ do immediately after the impulse, respectively. The contact position constraint (7.2) for the frames with the impact is also imposed at the impulse instant, which corresponds to switching or guard conditions of the hybrid systems (Reher and Ames (2021); Westervelt et al. (2003)). We assume a completely inelastic collision, in that the contact velocity constraints (7.4) holds for the velocity immediately after the impulse $v = v^- + \delta v$, i.e.,

$$\mathbf{v}(q^-, v^-, \delta v) := \dot{\mathbf{p}}(q^-, v^- + \delta v) = J(q^-)(v^- + \delta v) = 0. \tag{7.10}$$

By combining (7.8) and (7.10), we obtain the impulse dynamics:

$$\begin{bmatrix} \delta v \\ -\Lambda \end{bmatrix} = \begin{bmatrix} M(q^-) & J^{\mathrm{T}}(q^-) \\ J(q^-) & O \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ -J(q^-) v^- \end{bmatrix}. \tag{7.11}$$

With these settings, the state jump equation is expressed as

$$x^+ = \psi(x^-) = \begin{bmatrix} q^- \\ v^- + \delta v^-(x^-) \end{bmatrix}, \tag{7.12}$$

where $\delta v^-(x^-) \in \mathbb{R}^n$ is a function of x^-, as defined in (7.11). Note that the impulse forces occur only for bodies that have impacts with the environment. Therefore, (7.2) and (7.12) change depending on the contact modes immediately before and after the collision ($k-1$ and k), which are expressed as

$$\mathbf{p}_{k-1,k}(x^-) = 0$$

and

$$x^+ = \psi_{k-1,k}(x^-),$$

respectively.

7.2.2 Inequality constraints

The robot system involves several physical limitations, such as joint position, velocity, and torque limits. Furthermore, contact force must lie inside the friction cone; otherwise, the resultant solution can be physically-infeasible (for example, the solution allows negative-direction normal contact forces). We thus introduce polyhedral-approximated friction cone constraint for each contact force, and it can be expressed in the world frame $[f_x \;\; f_y \;\; f_z]$ as:

$$\begin{bmatrix} f_x + \frac{\mu}{\sqrt{2}} f_z \\ -f_x + \frac{\mu}{\sqrt{2}} f_z \\ f_y + \frac{\mu}{\sqrt{2}} f_z \\ -f_y + \frac{\mu}{\sqrt{2}} f_z \\ f_z \end{bmatrix} \geq 0, \tag{7.13}$$

where $\mu > 0$ is the friction coefficient. Note that the friction cone constraint (7.13) is solely imposed for each active contact. Therefore, the collection of the inequality constraints imposed at a time instant also switches depending on the contact situations. We summarize the collection of the inequality constraints for each contact mode (that is, for each active subsystem k) as

$$g_k(x, u) \leq 0.$$

7.2.3 Optimal control problem of switched systems

We herein summarize the modeling of the rigid body system as a switched system and formulate an OCP for the system. Without loss of generality, let $\mathcal{K} = \{1, ..., K+1\}$ $(K > 0)$ be the given indices of active subsystems over the horizon of the OCP, i.e., $\mathcal{K}$ is the given contact sequence. The continuous-time OCP is provided as follows. We consider a switched system comprising $K+1$ subsystems

$$\dot{x}(t) = f_k(x(t), u(t)), \quad t \in [t_{k-1}, t_k), \quad k \in \mathcal{K}, \tag{7.14a}$$

with state jumps and switching conditions

$$x(t_k) = \psi_{k-1,k}(x(t_k-)), \; p_{k-1,k}(x(t_k-)) = 0, \quad j \in \mathcal{K}_j, \tag{7.14b}$$

and constraints

$$g_k(x(t), u(t)) \leq 0, \quad t \in [t_{k-1}, t_k], \quad k \in \mathcal{K}, \tag{7.14c}$$

where $\mathcal{K}_j \subset \mathcal{K}$ denotes the set of contact modes that involve impacts between the system and environment at the switch from $k \in \mathcal{K}_j$ to $k+1$. Subsequently, $\mathcal{K} - \mathcal{K}_j$ denotes the indices that do not involve state jumps, i.e., the switch from $k \in \mathcal{K} - \mathcal{K}_j$ to $k+1$ only involve breaking the contacts. Note that we also refer to the index k as *phase* in this chapter. t_0 and t_{K+1} denote the fixed initial and terminal times of the horizon and t_k, $k \in \{1, ..., K\}$ denotes the switching instant from phase k to phase $k+1$. For a given initial state $x(t_0) \in \mathbb{R}^{n_x}$, we consider the OCP of the switched system of the form of

$$\min_{u(\cdot), t_1, ..., t_K} J = V_f(x(t_{K+1})) + \sum_{k=1}^{K+1} \int_{t_{k-1}}^{t_k} l_k(x(\tau), u(\tau)) d\tau \tag{7.15a}$$

$$\begin{aligned} \text{s.t.} \quad & (7.14a) - (7.14c), \\ & t_{k-1} + \underline{\Delta}_k \leq t_k, \quad k \in \mathcal{K} \end{aligned} \tag{7.15b}$$

where $V_f(\cdot)$ and $l_k(\cdot)$ are user-defined terminal cost and stage cost for phase k, respectively. $\underline{\Delta}_k \geq 0$, $k \in \mathcal{K}$ is the minimum dwell-time, i.e., the last constraints in (7.15b) are the minimum dwell-time constraints.

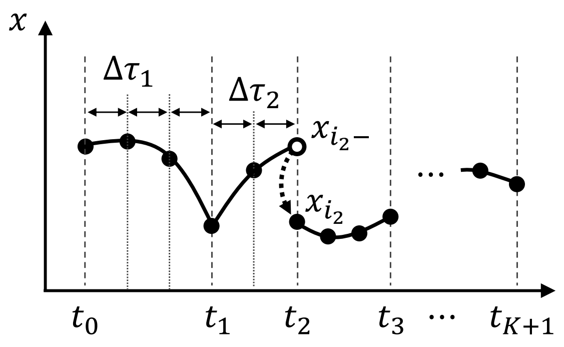


Figure 7.2: Conceptual diagram of the proposed discretization method of the continuous-time OCP of a switched system. Solid lines illustrate the continuous transition of the state. The switch from phase 2 to phase 3 involves a state jump (i.e., $2 \in \mathcal{K}_j$), which is illustrated as a dotted curve arrow. A white circle indicates a new grid representing the state immediately before the state jump.

7.3 Model Predictive Control with Online Switching Time Optimization

7.3.1 Direct multiple shooting method with mesh-refinement

The OCP (7.15) includes the switching instants as the optimization variables, as well as the state and control trajectory. The first key point for efficient and robust numerical computation is the discretization method of the continuous-time OCP. We discretize the continuous-time OCP using the direct multiple shooting (DMS) method (Bock and Plitt (1984)) to equalize the all-time steps in a phase. A conceptual diagram of the discretization method is illustrated in Fig. 7.2. We introduce N grid points over the horizon, discretized state $X := \left\{ x_0, ..., x_N, x_{i_k}^-, ..., x_{i_j}^- \right\}$, discretized control input $U := \{u_0, ..., u_{N-1}\}$, and switching instants $T := \{t_1, ..., t_K\}$. The resultant nonlinear program (NLP) is given as

$$\min_{X,U,T} J = V_f(x_N) + \sum_{k\in\mathcal{K}} \sum_{i\in\mathcal{I}_k} l_k(x_i, u_i)\Delta\tau_k \tag{7.16a}$$

$$\begin{aligned}
\text{s.t.} \quad & x_0 - \bar{x} = 0, && (7.16\text{b})\\
& x_i + f_k(x_i, u_i)\Delta\tau_k - x_{i+1} = 0, \quad i \in \mathcal{I}_k, \ k \in \mathcal{K}, && (7.16\text{c})\\
& x_{i_k} = f_j(x_{i_k-}), \ p_j(x_{i_k-}) = 0, \quad k \in \mathcal{K}_j, && (7.16\text{d})\\
& g_k(x_i, u_i) \leq 0, \quad i \in \mathcal{I}_k, \ k \in \mathcal{K}, && (7.16\text{e})\\
& t_{k-1} + \underline{\Delta}_k - t_k \leq 0, \quad k \in \{1, ..., K\}, && (7.16\text{f})
\end{aligned}$$

where $\Delta\tau_k$ is the time step at phase k, defined as

$$\Delta\tau_k := \frac{t_k - t_{k-1}}{N_k}, \quad k \in \{1, ..., K\}. \tag{7.16g}$$

An advantage over the two-stage methods (Xu and Antsaklis (2004)), which have been applied to robot systems in Farshidian, Kamgarpour, et al. (2017); Li and Wensing (2020), of the NLP formulation (7.16) is that the local convergence of the Newton-type method for the NLP (7.16) is guaranteed. After solving the NLP (7.16), each discretization step-size $\Delta\tau_k$ can change before solving it because the switching times are optimized. Therefore, after the NLP, we check the step-size $\Delta\tau_k$ for the all-phase $k \in \mathcal{K}$. If it is too large, we perform mesh-refinement to increase the accuracy of the solution in terms of the continuous-time counterpart, i.e., we add grids on the phase k where $\Delta\tau_k$ is large.

7.3.2 Riccati recursion to compute Newton step

We solve the NLP (7.16) using primal-dual interior point method (Nocedal and Wright (2006)) with Gauss-Newton Hessian approximation. The Newton-step computation (solution of a linear system) is then reduced to that of the unconstrained OCP by eliminating the Newton-step of the slack and dual variables of the inequality constraints. Subsequently, the Riccati recursion algorithm for the OCP of switched systems proposed in Katayama and Ohtsuka (2021c) is adopted for the Newton-step computation. The method has the following three characteristics:

- Its computational time is linear-time complexity with regards to the number of the discretization grids N. The per Newton-step computation is approximately the same computational cost as the conventional Riccati recursion algorithm for Newton-type methods of unconstrained OCPs, which is efficient for large scale systems, therefore, it is often adopted for complicated robotic applications (Budhiraja et al. (2018); Farshidian, Kamgarpour, et al. (2017); Li and Wensing (2020); Mastalli et al. (2020)).

- It only requires the positive semi-definiteness of the reduced Hessian matrix obtained by multiplying the null-space matrix of the Jacobian of constraints to the Hessian matrix. Therefore, the Riccati recursion algorithm can treat the NLP (7.16) whose Hessian matrix is inherently indefinite. Conversely, some general QP solvers require the positive definiteness of the Hessian matrix and cannot treat the NLP (7.16).

- It improves the numerical stability by modifying the reduced Hessian matrix positive definite without additional computational burden.

We treat the pure-state constraints (7.2) efficiently within the Riccati recursion algorithm using a constraint-transformation of Katayama and Ohtsuka (2022). Furthermore, we lift the contact-consistent forward dynamics (7.5) and impulse dynamics (7.11) to relax the high nonlinearity of the NLP and improve convergence property, without increasing the additional computational burden (Katayama and Ohtsuka (2021a)).

7.3.3 Heuristic regularization to improve convergence property

Even with the Hessian modification of the Riccati recursion algorithm (Katayama and Ohtsuka (2021c)), the Newton-step computation can be ill-conditioned owing to the non-convex nature of the NLP. We then heuristically introduce a large penalty on the update of the switching times when the iterate is not close to a local minimum. That is, we add a large constant to the diagonal element of the Hessian matrix corresponding twice to the partial derivatives with respect to a switching time.

7.3.4 Minimum dwell-time constraints

The minimum dwell-time constraints (7.15b) play an important role in practice. We observe that the convergence speed increases as the minimum dwell-time $\underline{\Delta}_k$ increases, because it reduces search ranges of t_k. However, too large $\underline{\Delta}_k$ can reduce the optimality and ill condition the problem. Therefore, we have to carefully choose $\underline{\Delta}_k$.

Moreover, in MPC implementation, the optimized switching times can be procrastinated depending on the problem settings (e.g., the given contact sequence and cost function). That is, with certain problem settings, the cost of the switching is high and the switching times always lie near the terminal of the horizon. As a result, predicted discrete events (i.e., making or breaking the contacts) never happen on the real system. This issue is illustrated in the upper figure of Fig. 7.3. To prevent this phenomenon in MPC, at each sampling time, we gradually increase the minimum-dwell time of the last phase $\underline{\Delta}_{K+1}$, which is illustrated in the lower figure of Fig. 7.3. For example, at each sampling time, we increase $\underline{\Delta}_{K+1}$ as

$$\underline{\Delta}_{K+1}^{\text{new}} \leftarrow \underline{\Delta}_{K+1} + (\text{sampling period}). \tag{7.17}$$

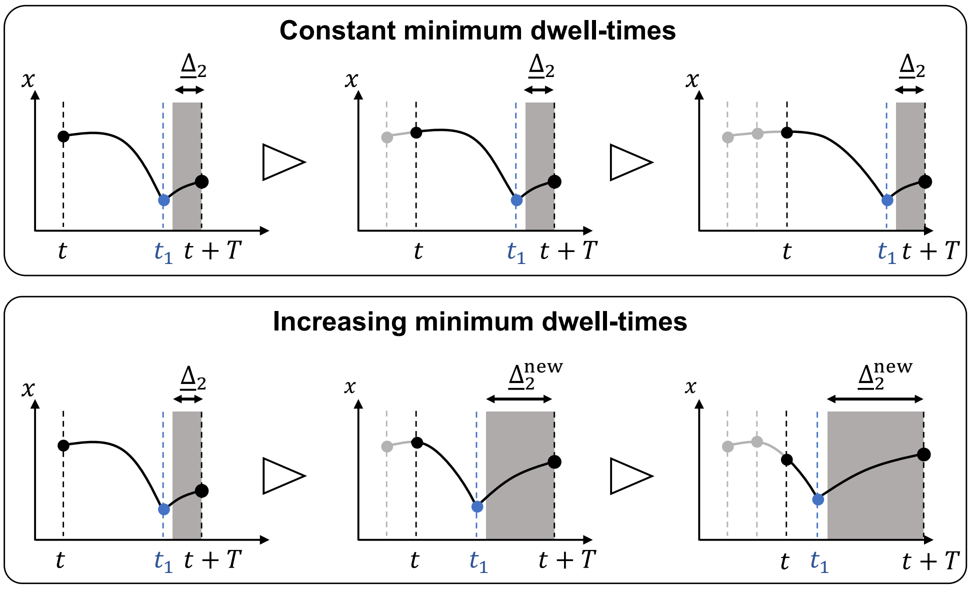

Constant minimum dwell-times
x
$\underline{\Delta}_2$
t
t_1
$t+T$
Increasing minimum dwell-times
$\underline{\Delta}_2^{\mathrm{new}}$

Figure 7.3: Effects of increasing the minimum dwell-time constraints. The upper figure illustrates a situation where the switching time t_1 is kept at the end of the horizon owing to the fixed minimum dwell-time. The lower figure illustrates a solution for this problem by increasing the minimum dwell-time $\underline{\Delta}_{K+1}$. t denotes the sampling instant, i.e., the initial time of the horizon, and T denotes the length of the horizon.

7.3.5 Software implementation

We implement the proposed MPC algorithm as an open-source software `robotoc`, an efficient optimal control solver for robotic systems (Katayama (2020-2022)), which is written in C++, uses Eigen (Guennebaud et al. (2010)) for linear algebra, OpenMP (Dagum and Menon (1998)) for stage-wise parallel computation of the cost, constraints, and their derivatives of the NLP, and Pinocchio (Carpentier et al. (2019)) for the rigid body dynamics and its derivatives computations. The details of the software are presented in Appendix A.

7.4 Simulation Study: Comparison to Conventional MPC with Fixed Contact Timings

7.4.1 Experimental settings

We demonstrated the effectiveness of the proposed whole-body MPC with online STO over the conventional whole-body MPC with the fixed switching instants through numerical simulations. We refer to the former as *MPC-STO* and the latter just as *MPC* in the following. We considered the 0.6 m jumping of quadrupedal robot

Unitree A1, a torque-controlled quadruped robot whose degree of freedom (DOF) is 12.

We designed the cost functions $V_f(\cdot)$ and $l_k(\cdot)$ as simple quadratic weights on the deviation of the configuration from the reference standing pose, generalized velocity, and acceleration that is defined as the function of x and u as (7.5). That is, we did not impose any costs on the pre-defined jump trajectory but just specified landing contact locations by the contact constraints (7.2)–(7.3b). The jumping motions were induced mainly by these contact constraints.

We imposed inequality constraints comprising the joint position, velocity, torque limits, and the polyhedral-approximated friction cone constraints (7.13). We set the MPC settings as follows: the horizon length was 0.8 s, the number of the discretization grids N was 20, the number of the Newton-type iterations per sampling time was 2. We fixed the barrier parameter of the primal-dual interior point method as 1.0×10^{-3} throughout the simulation, which is a common and practical strategy of suboptimal MPC (Wang and Boyd (2010)). We provided the same initial guess of the solution including the switching instants to the two MPC controllers. We investigated the control results for various initial guesses of the switching instants, i.e., lift-off and touch-down times. This investigation corresponds to the situation where pre-computed optimal switching time is not appropriate owing to disturbances. We ran the simulations on physical simulator PyBullet (Coumans and Bai (2016–2022)) and the MPC-STO and MPC at a 400 Hz sampling rate (2.5 ms sampling period). In MPC-STO, we updated the minimum-dwell time constraints according to (7.17), for each initial guesses of the switching instants.

7.4.2 Results

Figure 7.4 illustrates the snapshots of the simulation results in which the initial guesses of the lift-off and touch-down times are given by 0.2 s and 0.5 s, respectively. The upper five pictures are taken from the proposed whole-body MPC controller labeled as *MPC-STO*. The lower five pictures are from the proposed whole-body MPC controller labeled as *MPC*. As illustrated in Fig. 7.4, the conventional MPC failed in whole-body jumping control of the quadruped, whereas our MPC-STO succeeded. Figure 7.5 illustrates the predicted switching times of both the proposed MPC-STO and conventional STO at each simulation time. As illustrated in Fig. 7.5, the proposed method optimized the switching times immediately, which led to success in the control of the dynamic jumping motion.

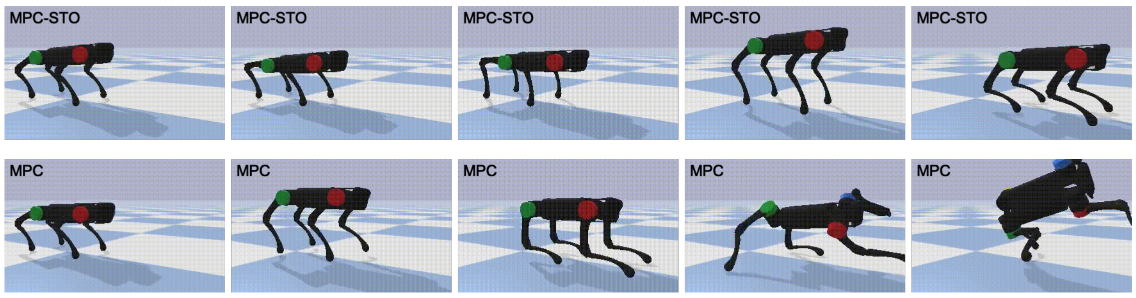


Figure 7.4: Snapshots of the 0.6 m jumping control simulations of quadruped robot Unitree A1 by whole-body MPC-STO (proposed) and whole-body MPC (conventional), in which the initial guesses of lift-off and touch-down times are given by 0.2 s and 0.5 s, respectively. The upper successful simulation (labeled as MPC-STO) is with the proposed whole-body MPC controller with online STO. The lower failing simulation (labeled as MPC) is with the conventional whole-body MPC controller with fixed switching times. Pictures with the same column are taken from the same simulation instant. A supplemental video is found at https://youtu.be/SureDVDFbfM.

Figure 7.6 illustrates the control results of the proposed MPC-STO and conventional MPC controllers for various initial guesses of the switching times. The proposed MPC-STO succeeded more than twice as many cases as the conventional MPC, which demonstrates that the proposed method extends the ability of whole-body MPC. This means that the proposed MPC-STO can achieve various control even though there are large disturbances that make the pre-computed switching instants meaningless. The average computational time of the control updates at each sampling time of both MPC and MPC-STO was approximately 1.3 ms, i.e, each Newton-type iteration took 0.65 ms, on octa-core CPU Intel Core i9-11900H @2.50 GHz with six threads in the parallel computation of the DMS. In the failure cases of MPC-STO, the solver could not converge to the optimal solution owing to the limited number of Newton-type iterations per sampling time (2 Newton-type iterations). If we can reduce the computational time per Newton-type iteration, the controller can take a greater number of the iterations, and we then expect that MPC-STO can succeed even with worse initial guesses. Therefore, improving the computational speed is still an important future study.

Note that the present MPC-STO can easily be applied to other kinds of dynamic motions. To show this, we further conducted simulations of several jumping motions. We used the almost same simple cost function as the above experiments: we only changed the reference values of the quadratic cost on the configuration. Figure 7.7 shows the snapshots of dynamic lateral jumping, jumping to back, and rotational jumping by the proposed whole-body MPC controller. Our MPC algorithm with

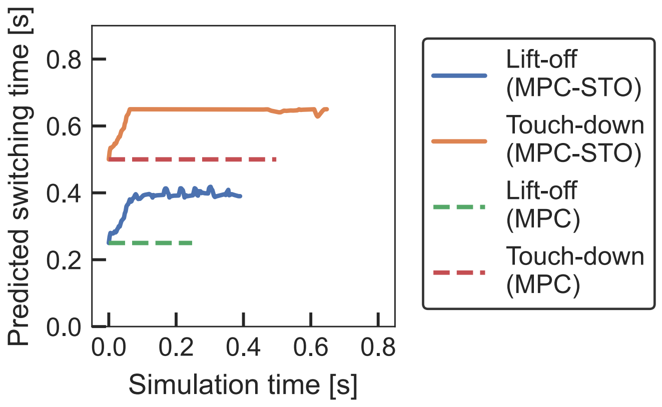


Figure 7.5: Predicted switching times at each simulation time of whole-body MPC-STO (proposed) and whole-body MPC (conventional) for whole-body jumping control of quadruped robot Unitree A1 in which the initial guesses of lift-off and touch-down times are given by 0.2 s and 0.5 s, respectively.

online STO successfully controlled these motions only with the simple cost function without efforts to construct complicated cost functions specialized to each motion or heuristics to determine the switching times.

7.5 Hardware Experiments on Quadrupedal Robot Unitree A1

7.5.1 Experimental settings

We further validated the proposed MPC on the real hardware of the quadrupedal robot Unitree A1. We applied the proposed MPC for successive dynamic jumping control of the quadruped robot. The overall control framework is illustrated in Fig. 7.8. We implemented a state estimator to estimate the state of the floating base using encoders and an inertial measurement unit (IMU). Specifically, we estimate the contact state from joint torque measurements (Camurri et al. (2017)) and then estimate the state of the floating base by fusing IMU measurements and the contact leg kinematics (Hartley et al. (2020)). We also implemented a cascaded controller using the whole-body MPC-STO and joint PD controllers to compensate model-mismatches such as joint frictions and estimation errors. That is, we send the desired joint angle and joint velocity computed by the MPC-STO as well as the feedforward joint torques ($q_{\mathrm{J,cmd}}$, $\dot{q}_{\mathrm{J,cmd}}$, and τ_{ff} in Fig. 7.8, respectively) to the build-in joint PD controller of the robot. We also utilize the state-feedback gain obtained by the Riccati recursion

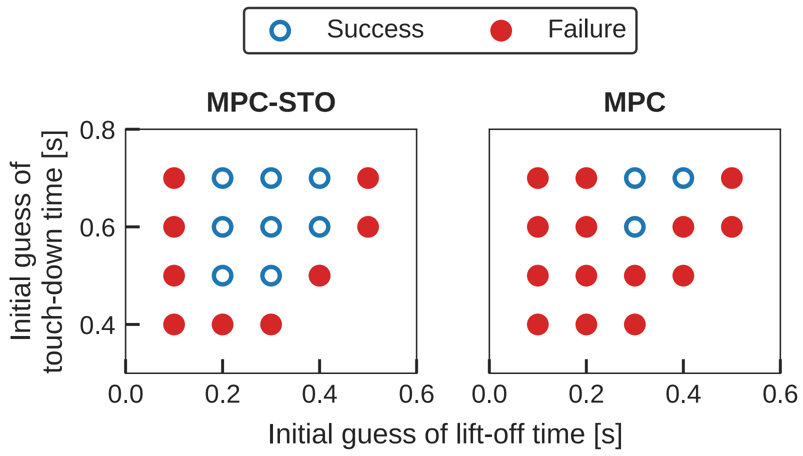


Figure 7.6: Simulation results (success or failure) of the whole-body MPC-STO (proposed) and whole-body MPC (conventional) for whole-body jumping control of quadruped robot Unitree A1 for various initial guesses of the lift-off and touch-down times.

algorithm of the MPC-STO as the joint PD gains (K_P and K_D in Fig. 7.8) because it can be seen as the optimal state-feedback gain of a linear-quadratic approximation of the NLP (7.16).

The MPC-STO and state estimator are run at 400 Hz on off-board laptop Ubuntu, with the PREEMPT RT kernel on octa-core CPU Intel Core i9-11900H @2.50 GHz with six threads in the parallel computing for the DMS in the MPC. The joint PD controller is run on the onboard PC of Unitree A1.

7.5.2 Results

Figure 7.9 illustrates the snapshots of jumping motions via the proposed whole-body MPC controller. As illustrated in the figure, the proposed method successfully realizes the successive dynamic jumps of the quadruped robot. Figure 7.10 shows the time-histories of measurements and commands of three joints in the LH-leg. It showed that each motor could not exactly track the commanded position, velocity, and torques provided by the proposed MPC due to the un-modeled factors such as joint frictions, i.e., disturbances. In addition, there could be state-estimation errors and modeling errors in the robot models. Nevertheless, the proposed controller achieved the control thanks to the fast MPC feedback and cascaded joint PD controller.

Future work is to realize longer jumps. One of the difficulties in jumping control comes from the state estimation; in the flying phase, the leg odometry is not available and state estimation becomes inaccurate. This is a common issue among legged

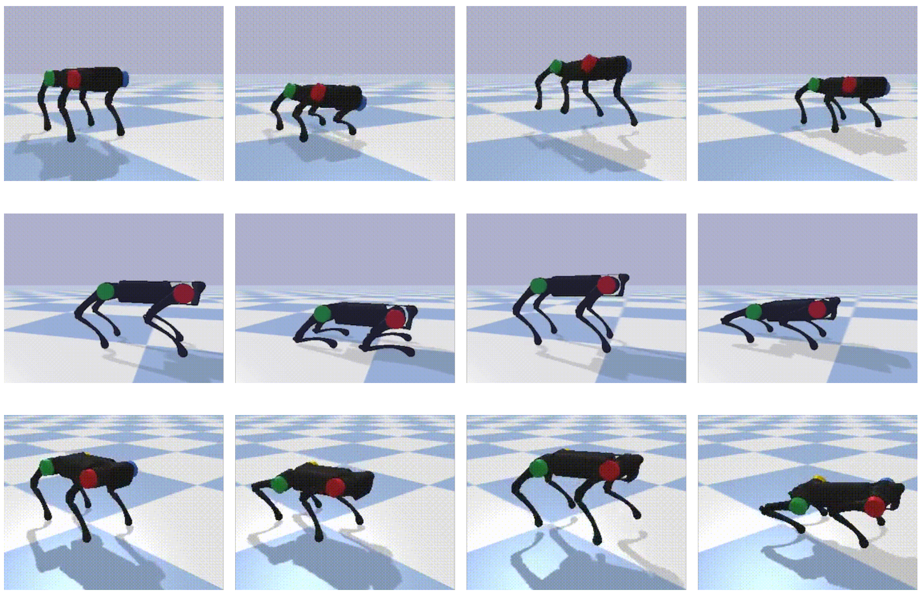


Figure 7.7: Snapshots of dynamic 0.4 m lateral jumping (upper), 0.3 m jumping to back (middle), and 30-degree yaw-rotational jumping (lower) control of quadrupedal robot Unitree A1 by whole-body MPC-STO.

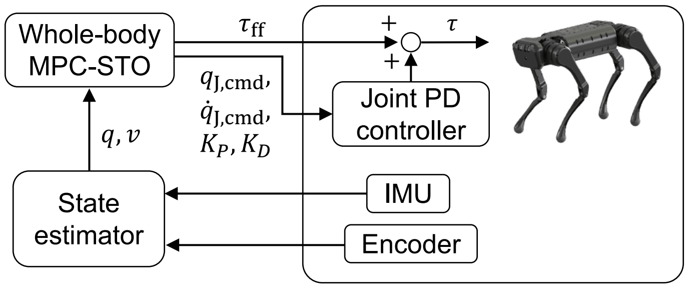


Figure 7.8: Block diagram of the whole-body MPC-STO control framework of quadruped robot Unitree A1.

motions with a flight phase. Improving the robustness of the MPC for this type of problem is our future study.

7.6 Summary

In this chapter, we presented a whole-body MPC of robotic systems with rigid contacts under a given contact sequence using the online STO. We treated the robot dynamics with rigid contacts as a switched system and formulated an OCP of switched systems to implement the MPC. We utilized the efficient solution algorithm for the MPC involving the STO problem that optimizes the switching times and trajectory simultaneously. The present algorithm is efficient to enable online optimization, unlike the

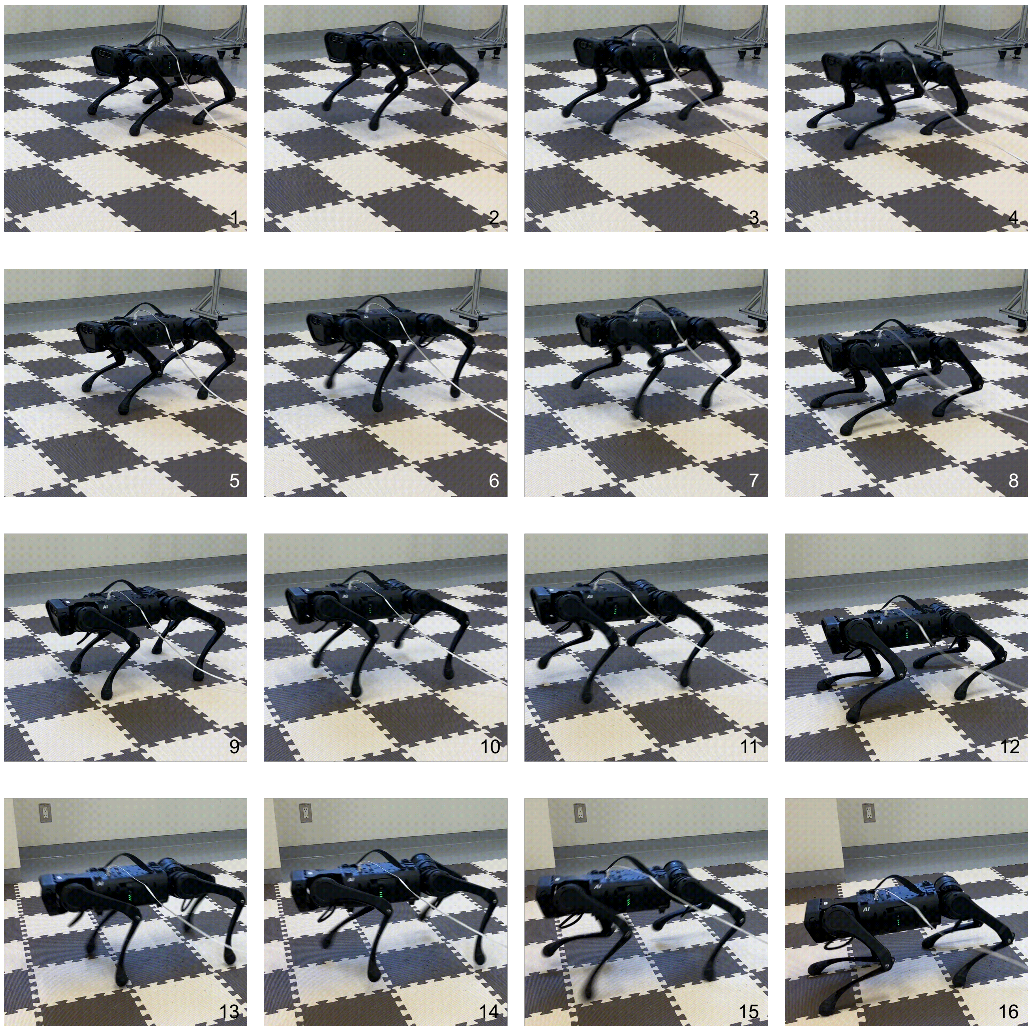


Figure 7.9: Snapshots of successive four-times dynamic jumping of quadruped robot Unitree A1 by the proposed whole-body MPC-STO. A supplemental video is found at https://youtu.be/SureDVDFbfM.

existing methods because of their inefficiency. We demonstrated the effectiveness of the proposed MPC with online STO over the conventional MPC with fixed switching times, through numerical simulation on dynamic jumping motions of a quadruped robot. Accordingly, the proposed MPC succeeded in the control more than twice as many cases as the conventional MPC. We further conducted hardware experiments on the quadrupedal robot Unitree A1 and demonstrated that the proposed method achieves dynamic motions on the real robot.

A possible limitation of our method is that it requires the pre-defined contact sequence and contact locations. The Methodology for their determination and synthesis, with the proposed STO strategy are included in future studies.

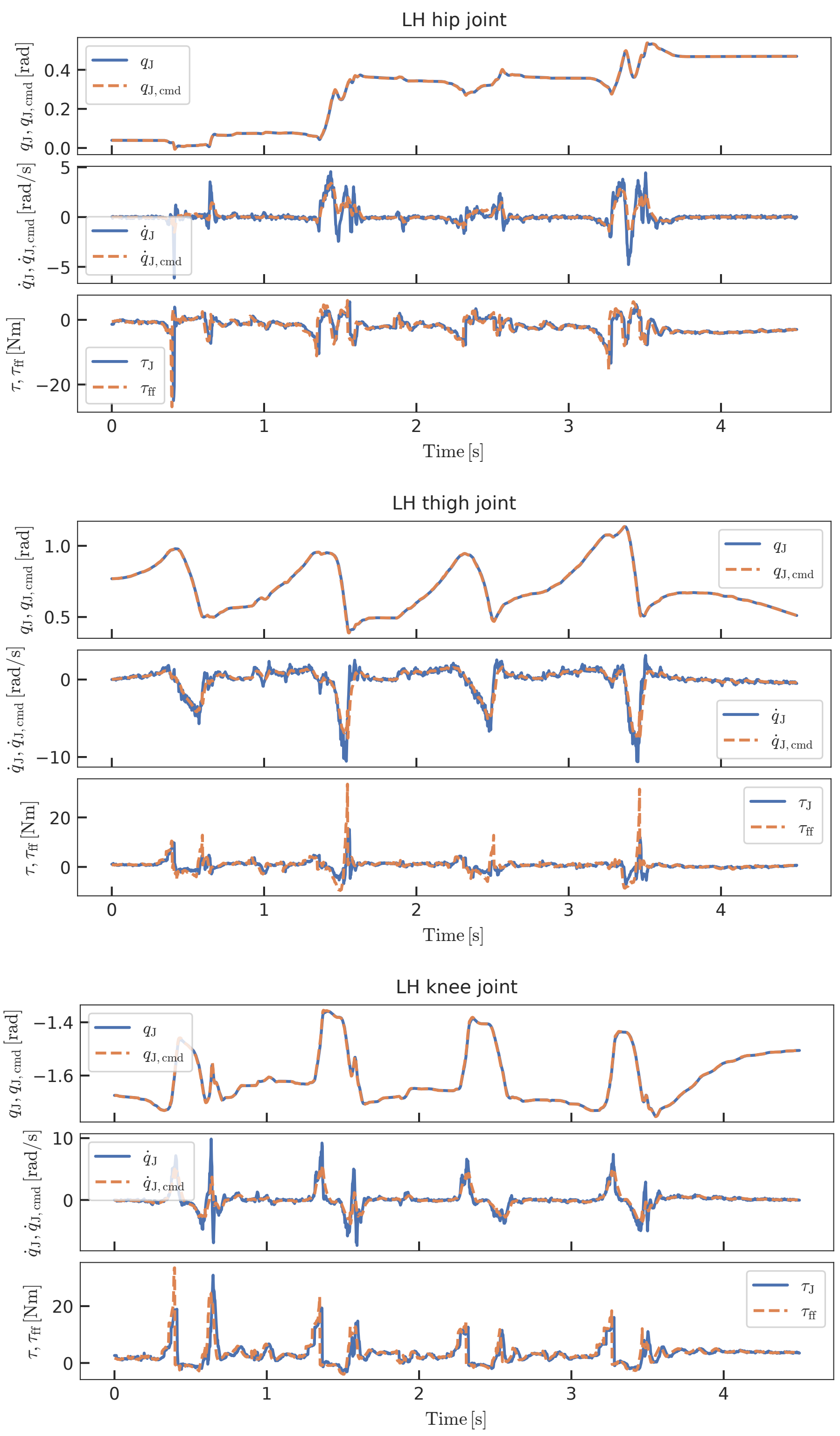


Figure 7.10: Time histories of joint measurements and commands in the LH leg in the successive jumping control by the proposed whole-body MPC-STO.

Chapter 8

Conclusions

8.1 Summary of Contributions

In this thesis, we have proposed algorithm developments toward fast model predictive control (MPC) of robotic systems with rigid contacts. MPC is expected to be a unified control approach that can realize versatile, efficient, and dynamic motions of robotic systems with rigid contacts. However, it has been challenging because they are fast and large-scale nonlinear systems, and contacts cause switches in dynamics and state jumps. This thesis has consistently tackled these challenges to establish dedicated fast MPC algorithms for robotic systems.

In Chapter 3, we have proposed an inverse-dynamics-based solution method for optimal control problems (OCPs). It improves the computational speed of each Newton-type iteration by leveraging rigid-body inverse dynamics algorithms. In Chapter 4, we have proposed lifted contact dynamics to improve the convergence properties by relaxing the high nonlinearity. The approach realizes efficient lifting by leveraging the structure of the rigid-body dynamics with contacts. In Chapter 5, we have proposed a constraint-transformation approach to treat pure-state equality constraints in OCPs, which arise in OCPs of robots with contacts. It enables us to treat the constraints with the Riccati recursion algorithm efficiently. In Chapter 6, we have proposed a Riccati-based fast and robust Newton-type algorithm to optimize the switching times and trajectory simultaneously. The algorithm computes the Newton-step linear time complexity with respect to the horizon. It also improves the numerical robustness by modifying reduced Hessian efficiently. In Chapter 7, we have implemented whole-body MPC built on top of the aforementioned approaches and demonstrated the effectiveness of the proposed approaches by numerical simulation and hardware experiments of a quadrupedal robot. These experiments demonstrate that the proposed MPC achieves milliseconds-range computational time and highly dynamic motion control.

8.2 Discussion and Future Work

Finally, we discuss the limitations of our approaches and future work for these problems.

8.2.1 Application to a wider variety of robotic problems

In Chapter 7, we have successfully applied the proposed MPC to quadrupedal locomotion over flat floors both on the physical simulation and real-world hardware. However, challenges in the control of robotic systems vary depending on the hardware platform, working environment, and desired tasks. Because the proposed MPC is based on whole-body dynamics and rigid-contact models, it can potentially treat a wide range of problems. It is therefore included in our future work to investigate the effectiveness of the proposed method in a wide variety of robotic applications. In more detail, the effectiveness of the whole-body dynamics and rigid contact model must be investigated through comparison with MPC based on different formulations such as SRBD/centroidal models (Di Carlo et al. (2018); Farshidian, Jelavic, et al. (2017); Sleiman et al. (2021)) and approximated contact models (Koenemann et al. (2015); Neunert et al. (2018)) over various robotic problems.

An example of such robotic applications other than quadrupedal locomotion is humanoid (bipedal) control. Humanoid robots involve surface contacts, which have more complicated physical constraints (e.g., *contact wrench cone* (Caron, Pham, and Nakamura (2015))) than the friction cone of the point contacts of quadruped locomotion. Further, while some quadrupedal robots such as MIT mini cheetah (Katz et al. (2019)) are designed so that each limb is light compared with the total weight, this is not the case for typical humanoid robots. Therefore, the whole-body dynamics-based MPC is expected to work remarkably better than the SRBD-based MPC for humanoid robots. However, the humanoid robots often have large degrees of freedom, e.g., typically have more than twice as much as these of typical quadrupedal robots. Therefore, real-time optimization of MPC is still challenging for humanoid robots as the existing research (Dantec et al. (2021)) took a huge computational for whole-body MPC. A possible approach is to introduce an appropriate model-reduction, e.g., fixing some redundant joints that have little effect on the performance in MPC.

Another example is loco-manipulation (Ferrolho, Ivan, Merkt, Havoutis, and Vijayakumar (2022); Sleiman et al. (2021); Wolfslag et al. (2020)), which is a multi-contact problem for humanoid robots and quadrupedal robots equipped with robot arms. In this problem, the robot must conduct manipulation tasks while simultaneously leveraging its locomotion skills. It can involve complicated contact models depending on the manipulation problems. Further, manipulation requires various whole-body postures, in which the SRBD-model becomes inaccurate (Sleiman et al. (2021)). Therefore, the whole-body dynamics-based MPC is expected to be also valuable for this problem.

8.2.2 Improving robustness

The simulation and hardware experiments in Chapter 7 assume the perfect knowledge of the environment, e.g., we assume that the terrain is flat. However, in practice, there are uncertainties in the environment model. For example, there are steps and rough terrains in the real world. Moreover, it is difficult to predict friction coefficients accurately. Also, the ground can be soft and the impact is not completely inelastic, which is different from our rigid-contact assumption. These uncertainties can affect the performance of our MPC controller. There are several possible research directions to tackle this problem and we introduce them in the following.

Robust and stochastic MPC: First, we can use robust (tube or min-max) nonlinear MPC (Köhler, Soloperto, Müller, and Allgöwer (2020); Raimondo, Limon, Lazar, Magni, and ndez Camacho (2009); Zanelli, Frey, Messerer, and Diehl (2021)) or stochastic nonlinear MPC techniques (Mesbah, Streif, Findeisen, and Braatz (2014)). These methods can prevent the violation of inequality constraints under certain uncertainties. Further, analysis of the robust stability of these methods can be a key perspective for designing the cost function of MPC under uncertainties. However, applying these methods is still challenging due to their computational inefficiency, particularly for fast and large-scale robotic systems. Therefore, algorithm developments of robust and stochastic MPC are future work in these approaches.

Machine learning techniques: The second possible approach to improve the robustness of MPC is introducing machine-learning techniques. In particular, *domain randomaization* is a well-known technique in *sim-to-real* reinforcement learning (RL) to improve the robustness against parameter uncertainties (Tobin et al. (2017); W. Zhao, Queralta, and Westerlund (2020)). Recently, these are incredible robotic applications of RL by this methods (Andrychowicz et al. (2020); Miki et al. (2022)).

It is in principle possible to use the same methodology for our MPC controller. For example, we can automatically tune the parameters of MPC such as weight parameters in the cost function and length of the horizon by using the *Bayesian optimization* (BO) (Muratore, Eilers, Gienger, and Peters (2021)). It is also possible to apply RL methods such as Q-learning or actor-critic (Sutton and Barto (2018)) to tune MPC parameters by regarding the MPC as a function approximator of both policy and the action value-function (Bøhn, Gros, Moe, and Johansen (2021); Gros and Zanon (2019)). These methods are also valuable in that they automate the troublesome tuning of MPC parameters.

Another possible approach is to incorporate neural networks (NN) into the MPC. A most intuitive way is to introduce the value-function approximation as a terminal cost of MPC. It is proved that, if the terminal cost is exactly the same as the value-function, the control policy of finite-horizon MPC yields the optimal policy for infinite-horizon MPC, which is an *ideal* control policy for a given reward or stage cost (Gros and Zanon (2019)). Several researchers have pointed out the usage of the value-function approximation in RL as the terminal or stage costs of MPC (Erez, Tassa, and Todorov (2012); Hoeller, Farshidian, and Hutter (2020); Karnchanachari, Valls, Hoeller, and Hutter (2020); Lowrey, Rajeswaran, Kakade, Todorov, and Mordatch (2019)).

A challenge in applying the aforementioned RL methods for complicated robotic MPC problems is that the stage cost, a candidate of the reward function of the RL methods, of robotic MPC problems is typically time-varying. Further, the dynamics switch depending on the contact mode. Therefore, a practical way to apply RL methods for tuning MPC parameters or value-function approximation in the robotic problems is non-trivial and included in future work.

Incorporating contact information feedback A key point both in the above two approaches is to incorporate the feedback of the contact information. Typical robots have force sensors in the end-effectors or torque sensors in joints. In such cases, we can measure or estimate the contact forces. The contact force values or estimations also include information such as whether the contacts in end-effectors are active or not. In typical model-based methods including MPC, such information is only used to estimate the floating base of the robot Camurri et al. (2017) and it is ignored in the controller. This is also the case for value-based machine learning approaches including the existing combination of MPC and RL (Erez et al. (2012); Hoeller et al. (2020); Karnchanachari et al. (2020); Lowrey et al. (2019)). In contrast, recent successful RL applications to robotic problems such as Andrychowicz et al. (2020) and Miki et al. (2022) leverages the contact information with policy iterations.

Therefore, incorporating contact information feedback is an important future work both in model-based approaches (robust and stochastic MPC) and machine-learning approaches (e.g., value-function learning).

Appendix A

`robotoc`: Open-Source Software for Whole-Body Model Predictive Control

A.1 Introduction

In the main chapters of this thesis, we have developed algorithms to efficiently solve optimal control problems (OCPs) of robotic systems toward real-time whole-body model predictive control (MPC). However, its implementation is complicated. Typical MPC problems are often implemented via existing software such as general-purpose nonlinear programming (NLP) modeling tool `CasADi` (Andersson et al. (2019)) or general nonlinear MPC tool `acados` (Verschueren et al. (2021)), both of which rely on symbolic expressions in formulating the OCPs. However, such tools are impractical for complicated robotic MPC problems because it is difficult to represent whole-body dynamics or full-body kinematics by symbolic expressions. In addition, direct symbolic or automatic differentiation of the forward kinematics and inverse/forward dynamics typically lack efficiency compared with the dedicated recursive algorithms such as recursive kinematic Jacobian computation (Lynch and Park (2017)) or analytical derivatives of the dynamics (Carpentier and Mansard (2018)). Moreover, since the robotic systems are switched systems, the problem structures (e.g., dimensions of the control input, state equation, cost function, and constraints) change dynamically during control, which further makes it difficult to implement MPC for robotic systems with the general NLP or MPC tools.

To interface these difficulties, we have developed an open-source software framework `robotoc`, an efficient optimal control framework for robotic systems (Katayama (2020-2022)) based on the algorithms presented throughout this thesis. `robotoc` aims to solve the OCPs defined as (7.1)–(7.16). It is consistently written in C++ for efficiency together with Python interfaces for ease of prototyping. Note that there are other software for numerical optimal control and MPC of robotic systems, such as `ocs2` (Farshidian (2017-2022)) and `Crocoddyl` (Mastalli et al. (2020)). Compared with these existing frameworks, our `robotoc` employs advanced algorithms that have been developed so far in this thesis. The characteristics of `robotoc`, particularly compared with these existing frameworks, are summarized as follows:

- Direct multiple-shooting method (Bock and Plitt (1984)) for time-discretization. This is more efficient and robust than the single-shooting methods in `ocs2` and `Crocoddyl` (specifically, the variants of differential dynamic programming (Jacobson and Mayne (1970))).
- Primal-dual interior point method for inequaliy constraints (Wächter and Biegler (2006)). `robotoc` can treat nonlinear inequality constraints regorously and efficiently. In contrast, `ocs2` and `Crocoddyl` only treat them as soft constraints and cannot guarantee the feasibility of the solution.
- Riccati recursion (Rao et al. (1998)) / ParNMPC (Deng and Ohtsuka (2019)) to compute the Newton steps.
- The inverse dynamics-based method presented in Chapter 3 for robotic systems without contacts or a floating base.
- The lifted contact dynamics scheme presented in Chapter 4 for robotic systems with contacts.
- Pure-state constraint handling by the constraint-transformation presented in Chapter 5 for the position-level switching constraints due to contacts. This is treated by the augmented Lagrangian method in `ocs2` and is approximated by a penalty function method in `Crocoddyl`.
- The Riccati recursion for the switching time optimization (STO) problems presented in Chapter 6. With this algorithm, only our framework can achieve efficient and robust optimization of the switching times and the trajectory simultaneously.

In the remainder of this appendix, we first introduce an overview of the interfaces of our framework `robotoc`. Next, we present some implementation details that make `robotoc` efficient. Finally, we provide some robotic application examples of `robotoc`.

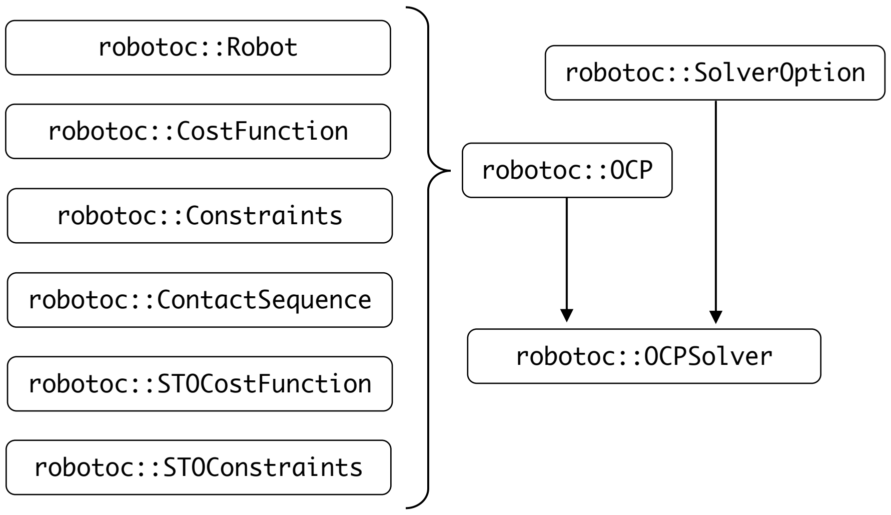


Figure A.1: Overview of interfaces of `robotoc`.

A.2 Interface Overview

In this section, we give an overview of the interfaces of `robotoc`, which is illustrated in Fig. A.1. The interfaces are available both in C++ and Python.

A.2.1 `robotoc::Robot`

`robotoc::Robot` is a robot model and describes the kinematics and dynamics of the robot. We construct the robot model `robotoc::Robot` from an URDF (universal robot description format) file. If the robot involves contacts, we further specify the contact frames, contact types, and parameters (the time step or weight parameters) of the Baumgarte's stabilization method (Baumgarte (1972); Flores et al. (2011)). The below shows an example to construct a quadruped robot model.

```
const std::string path_to_urdf = ...;
const std::vector<std::string> contact_frames{"LF_FOOT", "LH_FOOT", "
    RF_FOOT", "RH_FOOT"};  // frame names in the URDF file
const std::vector<robotoc::ContactType> contact_types(contact_frames.
    size(), robotoc::ContactType::PointContact);
const double baumgarte_time_step = 0.05;
robotoc::Robot robot(path_to_urdf, robotoc::BaseJointType::FloatingBase,
contact_frames, contact_types, baumgarte_time_step);
};
```

Listing A.1: C++ example of constructing a quadruped robot

A.2.2 robotoc::CostFunction

In robotoc, we create the cost function, robotoc::CostFunction, by combining various cost components. Each of the cost components inherits the interface class robotoc::CostFunctionComponentBase and users can readily implement new cost component. The followings are examples of new cost components with C++ and Python.

```
class NewCostComponent final : public robotoc::CostFunctionComponentBase
    {
// override methods to compute the cost value, and derivatives, and
   Gauss-Newton Hessian
};
```

Listing A.2: C++ example of a new cost component

```
class NewCostComponent(robotoc.CostFunctionComponentBase):
def __init__(self, ...):
super().__init__()
...
# override methods to compute the cost value, and derivatives, and Gauss
   -Newton Hessian
```

Listing A.3: Python example of a new cost component

Further, robotoc provides basic and useful cost components such as regarding the configuration-space variables, the task-space positions, and the CoM positions. The following shows an example of constructing the cost function with the provided cost components in C++.

```
auto config_cost = std::make_shared<robotoc::ConfigurationSpaceCost>(
   robot);
auto task_space_cost = std::make_shared<robotoc::TaskSpace3DCost>(robot,
    "ee_frame", task_3d_ref);
auto com_cost = std::make_shared<robotoc::CoMCost>(robot, com_ref);
auto cost = std::make_shared<robotoc::CostFunction>();
cost->push_back(config_cost);
cost->push_back(task_space_cost);
cost->push_back(com_cost);
```

Listing A.4: C++ example of the cost function

A.2.3 robotoc::Constraints

As well as the cost function, we create the constraints, robotoc::Constraints, by combining various constraint components. Each of the constraint components inherits the interface class robotoc::ConstraintComponentBase and users can readily implement new constraint component. The followings are examples of new cost components with C++ and Python.

```
class NewConstraintComponent final : public robotoc::
    CostFunctionComponentBase {
// override methods to compute the constraint and its derivatives
};
```

Listing A.5: C++ example of a new constraint component

```
class NewConstraintComponent(robotoc.ConstraintFunctionComponentBase):
def __init__(self, ...):
super().__init__()
...
# override methods to compute the constraint and its derivatives
```

Listing A.6: Python example of a new constraint component

Further, `robotoc` provides basic and useful constraint components such as joint position limits, joint velocity limits, joint torque limits, and friction cones. The following shows an example of constructing the constraints with the provided constraint components in C++.

```
auto joint_position_lower = std::make_shared<robotoc::
    JointPositionLowerLimit>(robot);
auto joint_position_upper = std::make_shared<robotoc::
    JointPositionUpperLimit>(robot);
auto joint_velocity_lower = std::make_shared<robotoc::
    JointVelocityLowerLimit>(robot);
auto joint_velocity_upper = std::make_shared<robotoc::
    JointVelocityUpperLimit>(robot);
auto joint_torques_lower = std::make_shared<robotoc::
    JointTorquesLowerLimit>(robot);
auto joint_torques_upper = std::make_shared<robotoc::
    JointTorquesUpperLimit>(robot);
const double friction_coeff = 0.6;
auto friction_cone = std::make_shared<robotoc::FrictionCone>(robot,
    friction_coeff);
auto constraints = std::make_shared<robotoc::Constraints>();
constraints->push_back(joint_position_lower);
constraints->push_back(joint_position_upper);
constraints->push_back(joint_velocity_lower);
constraints->push_back(joint_velocity_upper);
constraints->push_back(joint_torques_lower);
constraints->push_back(joint_torques_upper);
constraints->push_back(friction_cone);
```

Listing A.7: C++ example of the constraints

A.2.4 `robotoc::ContactSequence`

The contact sequence, `robotoc::ContactSequence`, describes the sequence of the contact status. The contact status, `robotoc::ContactStatus`, includes the information of the contacts such as which candidates of the contact frames have active contacts and the positions of the active contacts. For each contact phase, we add the contact status into the contact sequence. When adding a new contact status, we input the instant of the switch from an old contact phase to the new contact phase. The below shows an example of a quadrupedal trotting.

```cpp
auto contact_sequence = std::make_shared<robotoc::ContactSequence>(robot
   );
auto contact_status_standing = robot.createContactStatus();
// configure contact status...
contact_sequence->init(contact_status_standing);
contact_sequence->push_back(contact_status_lhrf_swing, swing_start_time)
   ;
contact_sequence->push_back(contact_status_rhlf_swing, swing_start_time+
   swing_time);
...
```

Listing A.8: C++ example of the contact sequence

If we want to optimize the switching times, i.e., for the switching time optimization (STO), we specify it when we add the contact status to the contact sequence as

```cpp
const bool enable_sto = true;
contact_sequence->push_back(contact_status_lhrf_swing, swing_start_time,
    enable_sto);
contact_sequence->push_back(contact_status_rhlf_swing, swing_start_time+
   swing_time, enable_sto);
...
```

Listing A.9: C++ example of the contact sequence for the STO

For a given contact sequence, the OCP solver of `robotoc` automatically constructs the structure of the OCP such as the dimension of the contact forces, the dynamics of the robot (i.e., the state equation), state jumps, switching constraints, cost function, and constraints (e.g., friction cones for active contacts). That is, in `robotoc`, we do not need to care about complicated problem structures of OCPs of robotic systems.

A.2.5 `robotoc::STOCostFunction` and `robotoc::STOConstraints`

If we want to optimize the switching times, we need to define the cost and constraints regarding the switching times. We call the cost and constraints associated with the switching times as *STO cost* and the constraints as *STO constraints* in the following. The STO cost function, `robotoc::STOCostFunction`, is constructed with the cost component on the switching times (currently no implementations of the cost component for the STO problems are provided). The STO constraints, `robotoc::STOConstraints`, is a collection of the minimum dwell-time constraints of each phase. The below shows a C++ example of the STO cost and STO constraints.

```
auto sto_cost = std::make_shared<robotoc::STOCostFunction>();
const std::vector<double> min_dwell_times = ...;
auto sto_constraints = std::make_shared<robotoc::STOConstraints>(
    min_dwell_times);
```

Listing A.10: C++ example of the STO cost and STO constraints

A.2.6 **`robotoc::OCP`** and **`robotoc::OCPSolver`**

There are three types OCPs and OCP solvers in `robotoc`:

- `robotoc::OCP`, `robotoc::OCPSolver`: The OCP solver for a robotic system with/without a floating base or contacts.
- `robotoc::UnconstrOCP`, `robotoc::UnconstrOCPSolver`: The OCP solver for a robotic system without a floating base or contacts (i.e., unconstrained dynamics).
- `robotoc::UnconstrParNMPC`, `robotoc::UnconstrParNMPCSolver`: The OCP solver for a robotic system without a floating base or contacts (i.e., unconstrained dynamics) with ParNMPC algorithm (Deng and Ohtsuka (2019)).

Here, we provide an example of `robotoc::OCP` and `robotoc::OCPSolver`. The definition of the OCP, `robotoc::OCP`, is constructed by passing the aforementioned robot model, cost function, and constraints. The OCP solver is then constructed by passing the OCP and solver options, `robotoc::SolverOptions`, which includes, e.g., termination criteria. The following example illustrates it.

```
robotoc::OCP ocp(robot, cost, constraints, contact_sequence, T, N);
auto solver_options = robotoc::SolverOptions::defaultOptions();
robotoc::OCPSolver ocp_solver(ocp, solver_options, nthreads);
```

Listing A.11: C++ example of the OCP and OCP solver

If we want to optimize the switching times as well as the state and control input trajectory, we further have to pass the STO cost and constraints to the OCP as follows.

```
robotoc::OCP ocp(robot, cost, constraints, sto_cost, sto_constraints,
    contact_sequence, T, N);
auto solver_options = robotoc::SolverOptions::defaultOptions();
robotoc::OCPSolver ocp_solver(ocp, solver_options, nthreads);
```

Listing A.12: C++ example of the OCP and OCP solver for a STO problem

The OCP is solved for given initial time and initial state. The following shows an example of solving an OCP.

```
const double t = ...; // initial time
Eigen::VectorXd q = ...; // initial configuration
Eigen::VectorXd v = ...; // initial velocity
ocp_solver.setSolution("q", q); // warm start
ocp_solver.setSolution("v", v); // warm start
ocp_solver.initConstraints(t); // initialize the constraints
ocp_solver.solve(t, q, v); // solves the OCP
```

Listing A.13: C++ example of solving an OCP

A.3 Implementation Details

Basic linear algebra Overall linear algebra are based on `Eigen` (Guennebaud et al. (2010)). We can easily obtain high-performance matrix calculations by enabling vectorization with `-march=native` compilation command. In `robotoc`, we pre-allocate memories of matrices and vectors of `Eigen` to let dynamic memory allocations be as small as possible during solving the OCPs. We also utilize `Eigen::LLT`, a fast matrix inversion of symmetric positive-definite (SPD) matrices by Cholesky factorization. We consistently use Gauss-Newton Hessian approximation and therefore we face only matrix inversions of SPD matrices via `Eigen::LLT`.

Rigid body kinematics and dynamics computation We rely on the `Pinocchio` (Carpentier et al. (2019)) for efficient rigid body kinematics and dynamics computation. We compute the forward kinematics, kinematics Jacobian, inverse dynamics, and derivatives of the dynamics by using `Pinocchio`. `Pinocchio` also provides classes and functions to treat $SE(3)$ in the numerical optimization, which is useful, e.g., when considering the cost of the task-space.

Parallel computation with direct multiple shooting method The Newton-type iteration of a nonlinear program (NLP) consists of three steps: 1) computation of the residual of the Karush–Kuhn–Tucker (KKT) conditions and KKT matrix, 2) computation of the Newton steps (via Riccati recursion or ParNMPC algorithm in `robotoc`), 3) step size selection (fraction-to-boundary rule in `robotoc`) and update the iterate. In `robotoc`, we fully parallelize step 1 stage-wise, which is one of the most time-consuming parts of Newton-type iterations, thanks to the direct multiple shooting method. We utilize `OpenMP` (Dagum and Menon (1998)) for the parallel computation. In `robotoc::UnconstrParNMPCSolver`, parallel computing is

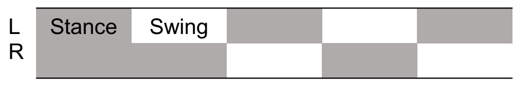


Figure A.2: The contact pattern of the iCub walking. L denotes the left foot in contact and R does the right foot in contact.

also applied for step 2, which enables fast Newton-step computation with a larger number of processors in exchange for convergence speed and robustness.

A.4 Application Examples

Here, we introduce application examples of `robotoc`. Note that implementations of the following examples can be found in Katayama (2020-2022).

A.4.1 Whole-body MPC of a humanoid robot walking

A.4.1.1 Problem settings and MPC design

The first example was a whole-body MPC of a walking control of a humanoid robot iCub (Natale et al. (2017)). iCub has 28 degrees of freedom (DOF), which is still too large to implement real-time whole-body MPC. Therefore, we considered a robot model whose joints in the upper half of the body are fixed. We then treated the robot model that has the 18 DOF, i.e., the configuration of the lies in $SE(3) \times \mathbb{R}^{12}$ and the control input is 12-dimensional. It has a floating base and can have two surface contacts with two feet. The objective of the bipedal walking control was to track the reference linear and angular CoM velocity commands that are expressed in the local coordinate of the floating base.

In this example, we designed the walking control based on a predefined constant walking period. That is, we did not consider optimizing the switching times. First, we designed the contact sequence with the predefined footstep timings to track the reference linear and angular CoM velocity. The contact pattern is illustrated in Fig. A.2. Successively, we designed the cost function with a simple cost on the configuration-space variables, costs on the swing foot trajectory, and cost on the CoM trajectory based on the contact sequence. We imposed the constraints with joint limits such as joint position limits, joint velocity limits, and joint torque limits. We also imposed contact wrench cone constraints (Caron et al. (2015)) for the two surface contacts on the two feet. We set the friction coefficient as 0.4 and the rectangular size of the wrench cone as 0.05 [m] and 0.025 [m], respectively. We set the length of the MPC horizon as 0.7 s, which is the same as the walking period of the robot.

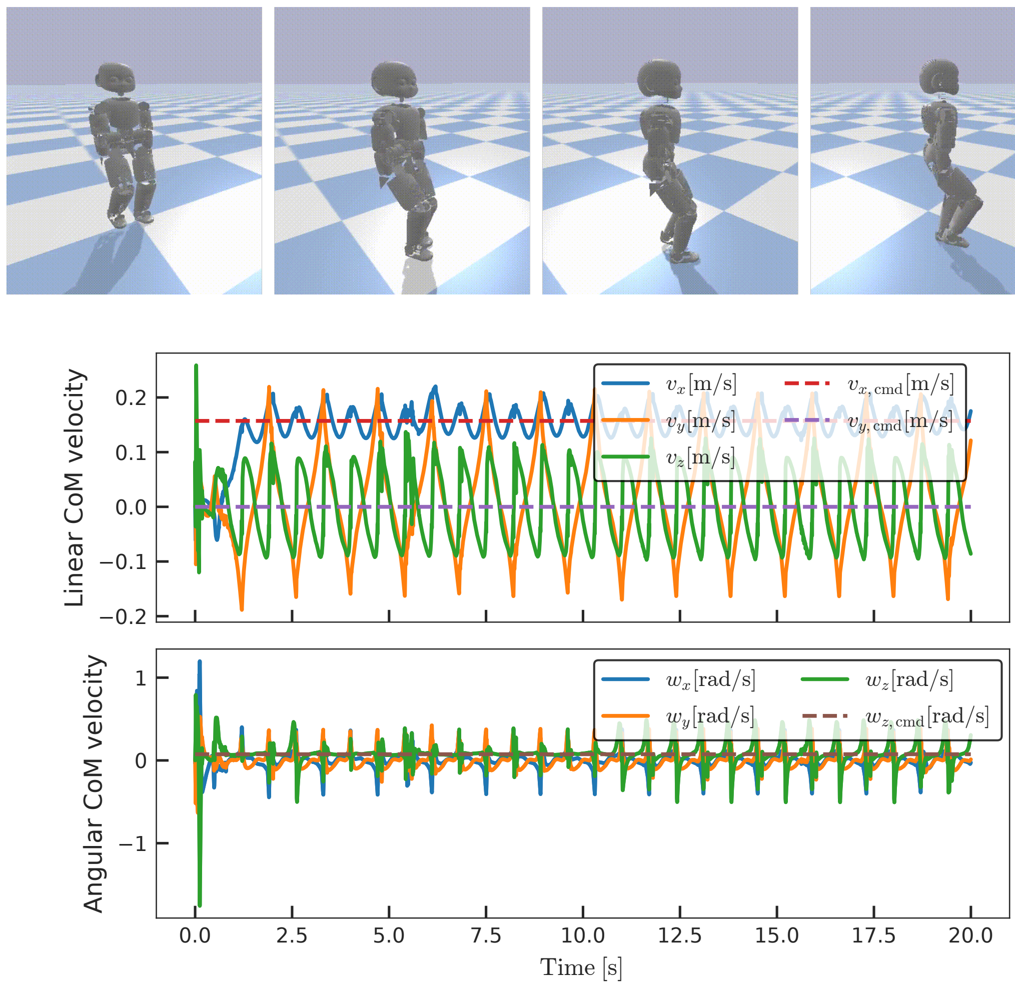
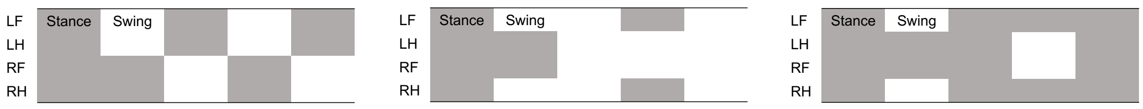


Figure A.3: (Upper) snapshots of walking control of a humanoid robot iCub by the `robotoc`'s whole-body MPC and (lower) time histories of the linear and angular CoM velocities of the robot (solid lines) and reference command velocities (dashed lines) both of which are expressed in the local coordinate of the floating base.

We also set the discretization number of the horizon from 20 to 22, which can vary during the control depending on the number of discrete events on the horizon.

A.4.1.2 Results

We simulated the proposed MPC on the physical simulator `Pybullet` (Coumans and Bai (2016–2022)) with 400 Hz control frequency. Figure A.3 shows the snapshots of the iCub walking by our whole-body MPC implemented with `robotoc` and the time histories of the linear and angular CoM velocity of the robot. As shown in Fig. A.3, the proposed MPC realized walking control that tracked the reference linear and angular CoM velocity commands. Our MPC controller performed a Newton-type



(a) Pace (b) Flying-trot (c) Trot

Figure A.4: The contact pattern of the (a) pace gait, (b) flying trot gait, and (c) trot of A1. LF, LH, RF, and RH denote left-front, left-hind, right-front, and right-hind foot in contact, respectively.

iteration at each sampling period of 2.5 ms and it took almost 0.5 ms on octa-core CPU Intel Core i9-9900 @3.10 GHz with six threads in the parallel computing. Therefore, our framework enables real-time whole-body MPC with a sufficient margin.

A.4.2 Whole-body MPC of quadruped robot gaits

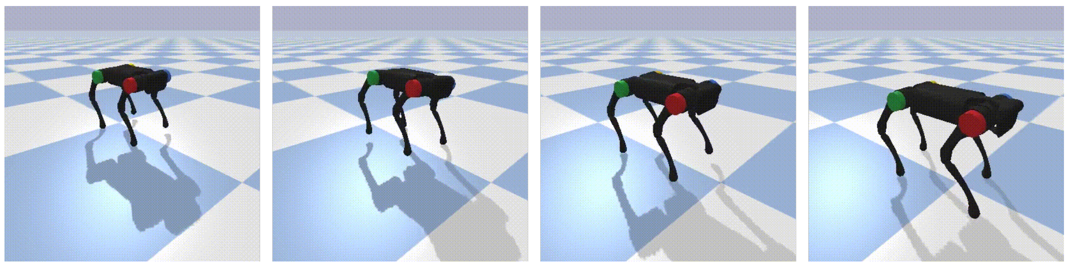
A.4.2.1 Problem settings and MPC design

The second example is a whole-body MPC of various gaits of a quadruped robot Unitree A1 (Unitree Robotics (n.d.)). The robot model has an 18 DOF, i.e., its state lies in $SE(3) \times \mathbb{R}^{12}$ and it has a 12-dimensional control input. It has a floating base and can have four point contacts with four feet. The objective was to control the quadrupedal gaits to track the reference linear and angular velocity commands.

As in the previous example, we designed the contact sequence, cost function, and constraints based on a predefined constant walking period for each gait. That is, we first designed the contact sequence with the predefined footstep timings to track the reference linear and angular CoM velocity. We then designed the cost function with a simple cost on the configuration-space variables, costs on the swing foot trajectory, and cost on the CoM trajectory based on the contact sequence. We also imposed the constraints with joint limits such as joint position limits, joint velocity limits, joint torque limits, and friction cone constraints. We set the length of the MPC horizon as 0.5 s and the discretization number of the horizon from 18 to 20, the latter of which can vary during the control depending on the number of discrete events on the horizon.

A.4.2.2 Results

We simulated the proposed MPC for pace gait, flying trot gait, and trot gait on rough terrain on `Pybullet` with a 400 Hz control frequency. The contact patterns of these gaits are shown in Fig. A.4. Figure A.5 shows the snapshots of the pace gait by our whole-body MPC implemented with `robotoc` and the time histories of the linear



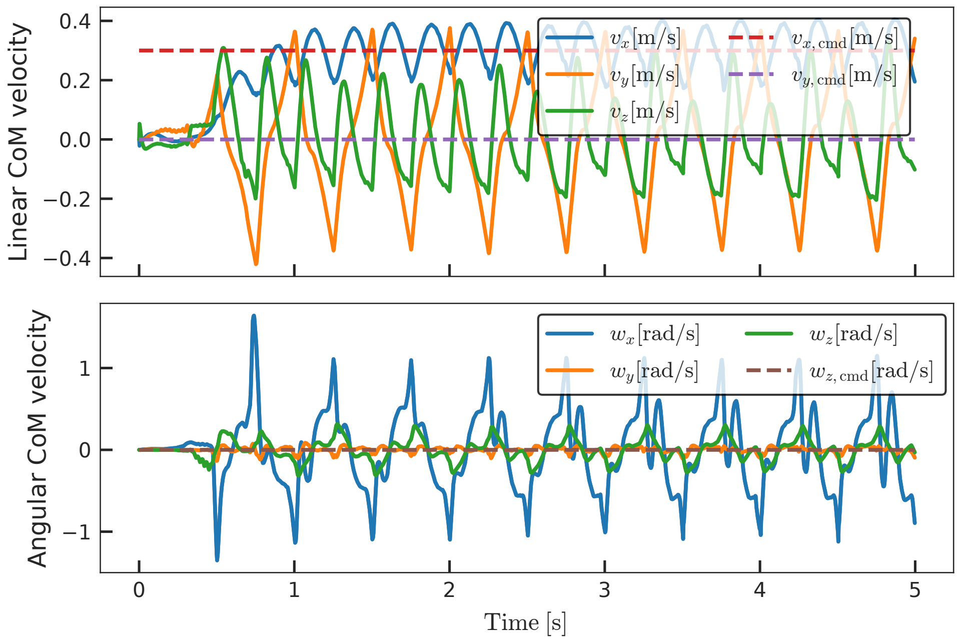
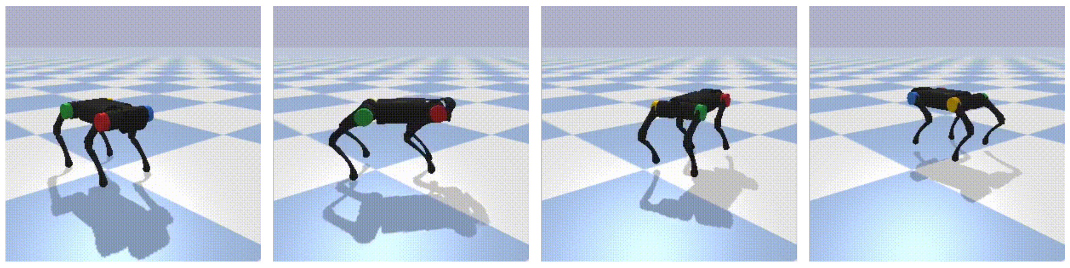


Figure A.5: (Upper) snapshots of pace gait control of a quadruped robot A1 by the robotoc's whole-body MPC and (lower) time histories of the linear and angular CoM velocities of A1 expressed in the body local coordinate (solid lines) and reference command velocities (dashed lines).

and angular CoM velocity of the robot. Figures A.6 and A.7 show the results of the flying trot gait and trot gait on rough terrain, respectively. As shown in these figures, the proposed MPC realized various gaits even under the disturbances (i.e., the rough terrain) that tracked the reference linear and angular CoM velocity commands. Our MPC controller performed two Newton-type iterations at each sampling period of 2.5 ms and it took almost 1.1 ms on octa-core CPU Intel Core i9-9900 @3.10 GHz with six threads in the parallel computing. Therefore, our framework can realize real-time whole-body MPC with a sufficient margin.



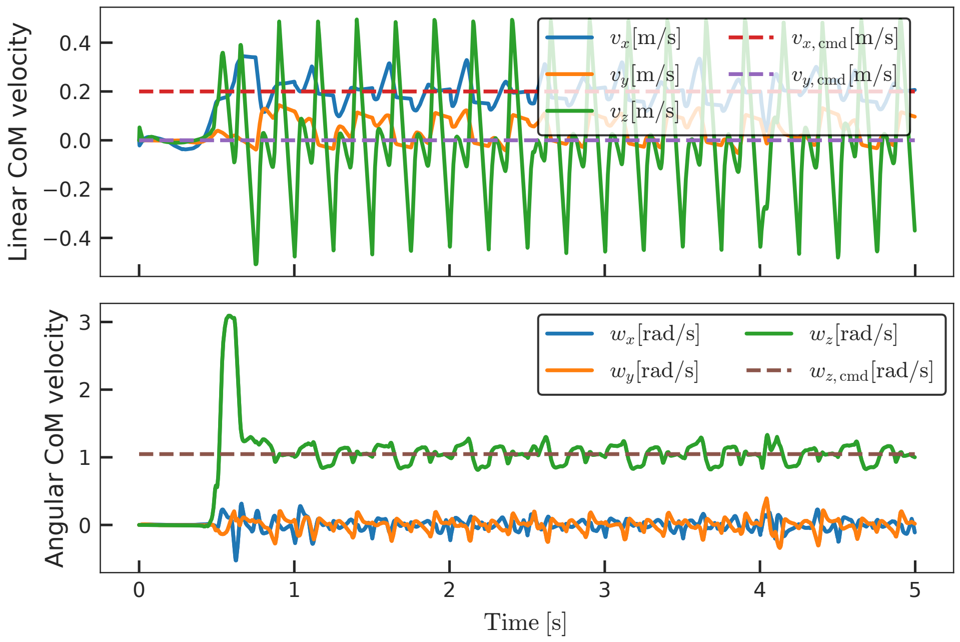
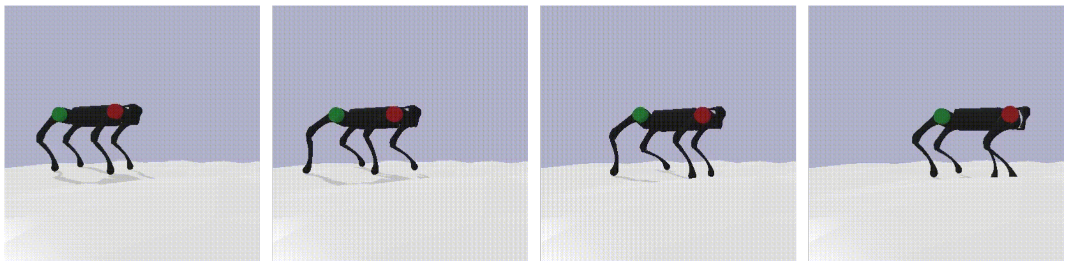


Figure A.6: (Upper) snapshots of flying-trot gait control of a quadruped robot A1 by the `robotoc`'s whole-body MPC and (lower) time histories of the linear and angular CoM velocities of A1 expressed in the body local coordinate (solid lines) and reference command velocities (dashed lines).

A.5 Summary

In this appendix, we have presented `robotoc`, our software framework for optimal control and whole-body MPC of robotic systems. Our framework provides efficient and robust algorithms presented throughout this thesis while it also gives the interfaces that mitigate complicated configuration of OCPs by users. We have reviewed interfaces that are available both in C++ and Python. We also have introduced implementation details of `robotoc`, which enables efficient numerical computations. Finally, we demonstrated practical complicated robotic examples via `robotoc`, of which implementations are also available in Katayama (2020-2022), and showed that our framework can achieve real-time whole-body MPC for a variety of problems.



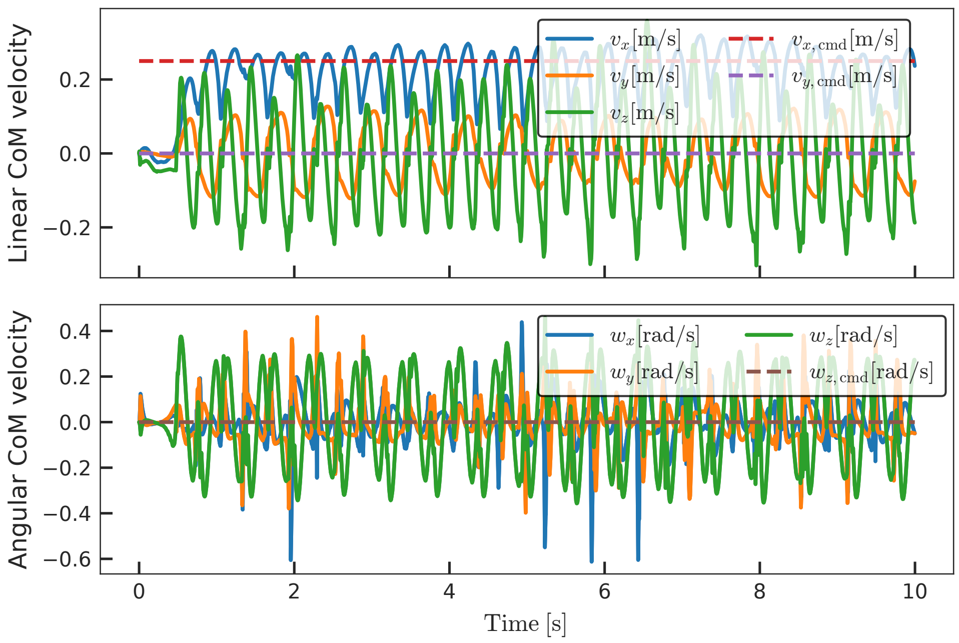


Figure A.7: (Upper) snapshots of trot gait control of a quadruped robot A1 on rough terrain by the robotoc's whole-body MPC and (lower) time histories of the linear and angular CoM velocities of A1 expressed in the body local coordinate (solid lines) and reference command velocities (dashed lines).

References

Abe, Y., Da Silva, M., & Popović, J. (2007). Multiobjective control with frictional contacts. In *2007 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (pp. 249–258).

Acosta, B., Yang, W., & Posa, M. (2022). *Validating robotics simulators on real-world impacts* (Vol. 7) (No. 3).

Agility Robotics. (n.d.). *Cassie website.* https://www.agilityrobotics.com/robots.

Albersmeyer, J., & Diehl, M. (2010). The lifted Newton method and its application in optimization. *SIAM Journal on Optimization*, *20*(3), 1655–1684.

Ames, A. D., Galloway, K., Sreenath, K., & Grizzle, J. W. (2014). Rapidly exponentially stabilizing control Lyapunov functions and hybrid zero dynamics. *IEEE Transactions on Automatic Control*, *59*(4), 876–891.

Ames, A. D., & Powell, M. (2013). Towards the unification of locomotion and manipulation through control Lyapunov functions and quadratic programs. In *Control of Cyber-Physical Systems* (pp. 219–240). Springer.

Amestoy, P., Duff, I. S., Koster, J., & L'Excellent, J.-Y. (2001). A fully asynchronous multifrontal solver using distributed dynamic scheduling. *SIAM Journal on Matrix Analysis and Applications*, *23*(1), 15–41.

Andersson, J. A. E., Gillis, J., Horn, G., Rawlings, J. B., & Diehl, M. (2019). CasADi – A software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, *11*(1), 1–36.

Andrychowicz, M., Baker, B., Chociej, M., Józefowicz, R., McGrew, B., Pachocki, J., Petron, A., Plappert, M., Powell, G., Ray, A., Schneider, J., Sidor, S., Tobin, J., Welinder, P., Weng, L., & Zaremba, W. (2020). Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, *39*(1), 3–20.

Axehill, D. (2005). *Applications of integer quadratic programming in control and communication* (Unpublished doctoral dissertation). Institutionen för systemteknik.

Barrau, A., & Bonnabel, S. (2016). The invariant extended Kalman filter as a stable observer. *IEEE Transactions on Automatic Control*, *62*(4), 1797–1812.

Baumgarte, J. (1972). Stabilization of constraints and integrals of motion in dynamical systems. *Computer Methods in Applied Mechanics and Engineering*, *1*(1), 1–16.

Belotti, P., Kirches, C., Leyffer, S., Linderoth, J., Luedtke, J., & Mahajan, A. (2013). Mixed-integer nonlinear optimization. *Acta Numerica*, *22*, 1—131.

Bertsekas, D. P. (2005). *Dynamic Programming and Optimal Control* (3rd ed., Vol. I). Athena Scientific.

Bertsekas, D. P. (2016). *Nonlinear Programming* (3rd ed.). Athena Scientific.

Betts, J. T. (2010). *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming* (2rd ed.). Cambridge University Press.

Bjelonic, M., Grandia, R., Harley, O., Galliard, C., Zimmermann, S., & Hutter, M. (2020). Whole-body MPC and online gait sequence generation for wheeled-legged robots. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 8388–8395).

Bledt, G., & Kim, S. (2019). Implementing regularized predictive control for simultaneous real-time footstep and ground reaction force optimization. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 6316–6323).

Bledt, G., Wensing, P. M., & Kim, S. (2017). Policy-regularized model predictive control to stabilize diverse quadrupedal gaits for the MIT cheetah. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 4102–4109).

Bloesch, M., Hutter, M., Hoepflinger, M. A., Leutenegger, S., Gehring, C., Remy, C. D., & Siegwart, R. (2013). State estimation for legged robots-consistent fusion of leg kinematics and IMU. In *Robotics: Science and systems* (Vol. 17, pp. 17–24). MIT Press.

Bock, H., & Plitt, K. (1984). A multiple shooting algorithm for direct solution of optimal control problems. In *the 9th IFAC World Congress* (pp. 1603–1608).

Bøhn, E., Gros, S., Moe, S., & Johansen, T. A. (2021). Optimization of the model predictive control meta-parameters through reinforcement learning. *arXiv preprint arXiv:2111.04146*.

Boston Dynamics. (n.d.-a). *Atlas website.* https://www.bostondynamics.com/atlas.

Boston Dynamics. (n.d.-b). *Spot website.* https://www.bostondynamics.com/products/spot.

Bretl, T. (2006). Motion planning of multi-limbed robots subject to equilibrium constraints: The free-climbing robot problem. *The International Journal of Robotics Research*, *25*(4), 317–342.

Bryson, A. E., & Ho, Y.-C. (1975). *Applied Optimal Control: Optimization, Estimation, and Control.* CRC Press.

Budhiraja, R., Carpentier, J., Mastalli, C., & Mansard, N. (2018). Differential dynamic programming for multi-phase rigid contact dynamics. In *2018 IEEE-RAS International Conference on Humanoid Robots (Humanoids).*

Bürger, A., Zeile, C., Altmann-Dieses, A., Sager, S., & Diehl, M. (2019). Design, implementation and simulation of an MPC algorithm for switched nonlinear systems under combinatorial constraints. *Journal of Process Control*, *81*, 15–30.

Camurri, M., Fallon, M., Bazeille, S., Radulescu, A., Barasuol, V., Caldwell, D. G., & Semini, C. (2017). Probabilistic contact estimation and impact detection for state estimation of quadruped robots. *IEEE Robotics and Automation Letters*, *2*(2), 1023–1030.

Camurri, M., Ramezani, M., Nobili, S., & Fallon, M. (2020). Pronto: A multi-sensor state estimator for legged robots in real-world scenarios. *Frontiers in Robotics and AI*, *7*, 68.

Carius, J., Ranftl, R., Koltun, V., & Hutter, M. (2018). Trajectory optimization with implicit hard contacts. *IEEE Robotics and Automation Letters*, *3*(4), 3316–3323.

Carius, J., Ranftl, R., Koltun, V., & Hutter, M. (2019). Trajectory optimization for legged robots with slipping motions. *IEEE Robotics and Automation Letters*, *4*(3), 3013–3020.

Caron, S., & Pham, Q.-C. (2017). When to make a step? Tackling the timing problem in multi-contact locomotion by TOPP-MPC. In *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)* (pp. 522–528).

Caron, S., Pham, Q.-C., & Nakamura, Y. (2015). Stability of surface contacts for humanoid robots: Closed-form formulae of the contact wrench cone for rectangular support areas. In *2015 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 5107–5112).

Carpentier, J., & Mansard, N. (2018). Analytical derivatives of rigid body dynamics algorithms. In *Robotics: Science and Systems (RSS 2018)* (p. hal-01790971v2f).

Carpentier, J., Saurel, G., Buondonno, G., Mirabel, J., Lamiraux, F., Stasse, O., & Mansard, N. (2019). The Pinocchio C++ library – A fast and flexible implementation of rigid body dynamics algorithms and their analytical derivatives. In *International Symposium on System Integration (SII)* (pp. 614–619).

Chatzinikolaidis, I., You, Y., & Li, Z. (2020). Contact-implicit trajectory optimization using an analytically solvable contact model for locomotion on variable ground. *IEEE Robotics and Automation Letters*, *5*(4), 6357–6364.

Cheng, X., Huang, E., Hou, Y., & Mason, M. T. (2021). Contact mode guided sampling-based planning for quasistatic dexterous manipulation in 2D. In *2021 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 6520–6526).

Coumans, E., & Bai, Y. (2016–2022). *Pybullet, a python module for physics simulation for games, robotics and machine learning.* http://pybullet.org.

Dagum, L., & Menon, R. (1998). OpenMP: An industry-standard API for shared-memory programming. *IEEE Computational Science & Engineering*, *5*(1), 46-–55.

Dai, H., Valenzuela, A., & Tedrake, R. (2014). Whole-body motion planning with centroidal dynamics and full kinematics. In *2014 IEEE-RAS International Conference on Humanoid Robots (Humanoids)* (pp. 295–302).

Dantec, E., Budhiraja, R., Roig, A., Lembono, T., Saurel, G., Stasse, O., Fernbach, P., Tonneau, S., Vijayakumar, S., Calinon, S., Taix, M., & Mansard, N. (2021). Whole body model predictive control with a memory of motion: Experiments on a torque-controlled talos. In *2021 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 8202–8208).

Dantec, E., Taix, M., & Mansard, N. (2022). First order approximation of model predictive control solutions for high frequency feedback. *IEEE Robotics and*

Automation Letters, *7*(2), 4448–4455.

Deits, R., & Tedrake, R. (2014). Footstep planning on uneven terrain with mixed-integer convex optimization. In *2014 IEEE-RAS International Conference on Humanoid Robots (Humanoids)* (pp. 279–286).

Deng, H., & Ohtsuka, T. (2019). A parallel Newton-type method for nonlinear model predictive control. *Automatica*, *109*, 108560.

Di Carlo, J., Wensing, P. M., Katz, B., Bledt, G., & Kim, S. (2018). Dynamic locomotion in the MIT cheetah 3 through convex model-predictive control. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 1–9).

Diehl, M., Bock, H. G., & Schlöder, J. P. (2005). A real-time iteration scheme for nonlinear optimization in optimal feedback control. *SIAM Journal on control and optimization*, *43*(5), 1714–1736.

Diehl, M., Bock, H. G., Schlöder, J. P., Findeisen, R., Nagy, Z., & Allgöwer, F. (2002). Real-time optimization and nonlinear model predictive control of processes governed by differential-algebraic equations. *Journal of Process Control*, *12*(4), 577–585.

Diehl, M., Ferreau, H. J., & Haverbeke, N. (2009). Efficient numerical methods for nonlinear MPC and moving horizon estimation. In *Nonlinear Model Predictive Control* (pp. 391–417). Springer.

Ding, Y., Pandala, A., & Park, H.-W. (2019). Real-time model predictive control for versatile dynamic motions in quadrupedal robots. In *2019 International Conference on Robotics and Automation (ICRA)* (pp. 8484–8490).

D. Kouzoupis, A. Z., G. Frison, & Diehl, M. (2018). Recent advances in quadratic programming algorithms for nonlinear model predictive control. *Vietnam Journal of Mathematics*, *46*(4), 863–882.

Englsberger, J., Ott, C., & Albu-Schäffer, A. (2015). Three-dimensional bipedal walking control based on divergent component of motion. *IEEE Transactions on Robotics*, *31*(2), 355–368.

Erez, T., Tassa, Y., & Todorov, E. (2012). Infinite-horizon model predictive control for periodic tasks with contacts. *Robotics: Science and systems VII*, 73.

Erez, T., & Todorov, E. (2012). Trajectory optimization for domains with contacts using inverse dynamics. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems* (pp. 4914–4919).

Fankhauser, P., Bloesch, M., Gehring, C., Hutter, M., & Siegwart, R. (2014). Robot-centric elevation mapping with uncertainty estimates. In *Mobile Service Robotics* (pp. 433–440). World Scientific.

Fankhauser, P., Bloesch, M., & Hutter, M. (2018). Probabilistic terrain mapping for mobile robots with uncertain localization. *IEEE Robotics and Automation Letters (RA-L)*, *3*(4), 3019–3026.

Farshidian, F. (2017-2022). *ocs2.* Retrieved from https://github.com/leggedrobotics/ocs2

Farshidian, F., Jelavic, E., Satapathy, A., Giftthaler, M., & Buchli, J. (2017). Real-time motion planning of legged robots: A model predictive control approach. In *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Hu-*

manoids) (pp. 577–584).

Farshidian, F., Kamgarpour, M., Pardo, D., & Buchli, J. (2017). Sequential linear quadratic optimal control for nonlinear switched systems. In *the 20th IFAC World Congress* (Vol. 50, pp. 1463–1469).

Farshidian, F., Neunert, M., Winkler, A. W., Rey, G., & Buchli, J. (2017). An efficient optimal planning and control framework for quadrupedal locomotion. In *2017 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 93–100).

Featherstone, R. (1983). The calculation of robot dynamics using articulated-body inertias. *The International Journal of Robotics Research*, *2*(1), 13–30.

Featherstone, R. (2008). *Rigid Body Dynamics Algorithms*. Springer.

Felis, M. L. (2017). RBDL: an efficient rigid-body dynamics library using recursive algorithms. *Autonomous Robots*, *41*(2), 495–511.

Fernbach, P., Tonneau, S., & Taïx, M. (2018). CROC: Convex resolution of centroidal dynamics trajectories to provide a feasibility criterion for the multi contact planning problem. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 1–9).

Ferreau, H. J., Kirches, C., Potschka, A., Bock, H. G., & Diehl, M. (2014). qpOASES: A parametric active-set algorithm for quadratic programming. *Mathematical Programming Computation*, *6*(4), 327–363.

Ferrolho, H., Ivan, V., Merkt, W., Havoutis, I., & Vijayakumar, S. (2022). RoLoMa: Robust loco-manipulation for quadruped robots with arms. *arXiv preprint arXiv:2203.01446*.

Florence, P. R., Manuelli, L., & Tedrake, R. (2018). Dense object nets: Learning dense visual object descriptors by and for robotic manipulation. *arXiv preprint arXiv:1806.08756*.

Flores, P., Machado, M., Seabra, E., & Silva, M. (2011). A parametric study on the Baumgarte stabilization method for forward dynamics of constrained multibody systems. *Journal of Computational and Nonlinear Dynamics*, *6*, 011019.

Frasch, J. V., Sager, S., & Diehl, M. (2015). A parallel quadratic programming method for dynamic optimization problems. *Mathematical Programming Computation*, *7*(3), 289–329.

Frigerio, M., Buchli, J., Caldwell, D. G., & Semini, C. (2016). RobCoGen: a code generator for efficient kinematics and dynamics of articulated robots, based on Domain Specific Languages. *Journal of Software Engineering for Robotics (JOSER)*, *7*(1), 36–54.

Frison, G. (2016). *Algorithms and methods for high-performance model predictive control* (Unpublished doctoral dissertation). Technical University of Denmark.

Frison, G., & Diehl, M. (2020). HPIPM: A high-performance quadratic programming framework for model predictive control. *IFAC-PapersOnLine*, *53*(2), 6563–6569.

Fu, J., & Zhang, C. (2021). Optimal control of path-constrained switched systems with guaranteed feasibility. *IEEE Transactions on Automatic Control*.

Giftthaler, M., & Buchli, J. (2017). A projection approach to equality constrained iterative linear quadratic optimal control. In *2017 IEEE-RAS 17th International*

Conference on Humanoid Robotics (Humanoids) (pp. 61–66).

Giftthaler, M., Neunert, M., Stäuble, M., Buchli, J., & Diehl, M. (2018). A family of iterative Gauss-Newton shooting methods for nonlinear optimal control. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 1–9).

Giftthaler, M., Neunert, M., Stäuble, M., Frigerio, M., Semini, C., & Buchli, J. (2017). Automatic differentiation of rigid body dynamics for optimal control and estimation. *Advanced Robotics*, *31*(22), 1225–1237.

Goebel, R., Sanfelice, R. G., & Teel, A. R. (2012). *Hybrid Dynamical Systems.* Princeton University Press.

Grandia, R., Farshidian, F., Ranftl, R., & Hutter, M. (2019). Feedback MPC for torque-controlled legged robots. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 4730–4737).

Grandia, R., Taylor, A. J., Ames, A. D., & Hutter, M. (2021). Multi-layered safety for legged robots via control barrier functions and model predictive control. In *2021 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 8352–8358).

Grimminger, F., Meduri, A., Khadiv, M., Viereck, J., Wüthrich, M., Naveau, M., Berenz, V., Heim, S., Widmaier, F., Flayols, T., Fiene, J., Badri-Spröwitz, A., & Righetti, L. (2020). An open torque-controlled modular robot architecture for legged locomotion research. *IEEE Robotics and Automation Letters*, *5*(2), 3650–3657.

Gros, S., & Zanon, M. (2019). Data-driven economic NMPC using reinforcement learning. *IEEE Transactions on Automatic Control*, *65*(2), 636–648.

Guennebaud, G., Jacob, B., et al. (2010). *Eigen v3.* Retrieved from http://eigen.tuxfamily.org

Han, X.-F., Laga, H., & Bennamoun, M. (2019). Image-based 3D object reconstruction: State-of-the-art and trends in the deep learning era. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *43*(5), 1578–1604.

Hansen, P., Jaumard, B., & Savard, G. (1992). New branch-and-bound rules for linear bilevel programming. *SIAM Journal on Scientific and Statistical Computing*, *13*(5), 1194-–1217.

Hartley, R., Ghaffari, M., Eustice, R. M., & Grizzle, J. W. (2020). Contact-aided invariant extended Kalman filtering for robot state estimation. *The International Journal of Robotics Research*, *39*(4), 402–430.

Hauser, J., & Saccon, A. (2006). A barrier function method for the optimization of trajectory functionals with constraints. In *45th IEEE Conference on Decision and Control* (pp. 864–869).

Hoeller, D., Farshidian, F., & Hutter, M. (2020). Deep value model predictive control. In *Conference on Robot Learning* (pp. 990–1004).

Hoheisel, T., Kanzow, C., & Schwartz, A. (2013). Theoretical and numerical comparison of relaxation methods for mathematical programs with complementarity constraints. *Mathematical Programming*, *137*(1), 257–288.

Howell, T. A., Jackson, B. E., & Manchester, Z. (2019). ALTRO: A fast solver for constrained trajectory optimization. In *2019 IEEE/RSJ International Conference* *on Intelligent Robots and Systems (IROS)* (pp. 7674–7679).

Hutter, M., Gehring, C., Lauber, A., Gunther, F., Bellicoso, C. D., Tsounis, V., Fankhauser, P., Diethelm, R., Bachmann, S., Bloesch, M., Kolvenbach, H., Bjelonic, M., Isler, L., & Meyer, K. (2017). ANYmal - toward legged robots for harsh environments. *Advanced Robotics*, *31*(17), 918–931.

Hwangbo, J., Lee, J., & Hutter, M. (2018). Per-contact iteration method for solving contact dynamics. *IEEE Robotics and Automation Letters*, *3*(2), 895–902.

Ishihara, K., Itoh, T. D., & Morimoto, J. (2019). Full-body optimal control toward versatile and agile behaviors in a humanoid robot. *IEEE Robotics and Automation Letters*, *5*(1), 119–126.

Jacobson, D. H., & Mayne, D. Q. (1970). *Differential Dynamic Programming*. Elsevier Publishing Company.

Jenelten, F., Miki, T., Vijayan, A. E., Bjelonic, M., & Hutter, M. (2020). Perceptive locomotion in rough terrain – online foothold optimization. *IEEE Robotics and Automation Letters*, *5*, 5370–5376.

Johnson, E. R., & Murphey, T. D. (2011). Second-order switching time optimization for nonlinear time-varying dynamic systems. *IEEE Transactions on Automatic Control*, *56*(8), 1953–1957.

Jung, M. N., Kirches, C., & Sager, S. (2013). On perspective functions and vanishing constraints in mixed-integer nonlinear optimal control. In G. R. M. Jünger (Ed.), *Facets of Combinatorial Optimization* (pp. 387–417). Springer.

Kajita, S., Kanehiro, F., Kaneko, K., Fujiwara, K., Harada, K., Yokoi, K., & Hirukawa, H. (2003). Biped walking pattern generation by using preview control of zero-moment point. In *2003 IEEE International Conference on Robotics and Automation (ICRA)* (Vol. 2, pp. 1620–1626).

Kalakrishnan, M., Chitta, S., Theodorou, E., Pastor, P., & Schaal, S. (2011). STOMP: Stochastic trajectory optimization for motion planning. In *2011 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 4569–4574).

Kamikawa, Y., Kinoshita, M., Takasugi, N., Sugimoto, K., Kai, T., Kito, T., Sakamoto, A., Nagasaka, K., & Kawanami, Y. (2021). Tachyon: Design and control of high payload, robust, and dynamic quadruped robot with series-parallel elastic actuators. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 894–901).

Kaneko, K., Kaminaga, H., Sakaguchi, T., Kajita, S., Morisawa, M., Kumagai, I., & Kanehiro, F. (2019). Humanoid robot HRP-5P: An electrically actuated humanoid robot with high-power and wide-range joints. *IEEE Robotics and Automation Letters*, *4*(2), 1431–1438.

Karnchanachari, N., Valls, M. I., Hoeller, D., & Hutter, M. (2020). Practical reinforcement learning for MPC: Learning from sparse objectives in under an hour on a real robot. In *Learning for dynamics and control* (pp. 211–224).

Katayama, S. (2020-2022). *robotoc.* Retrieved from https://github.com/mayataka/robotoc

Katayama, S., Doi, M., & Ohtsuka, T. (2020). A moving switching sequence approach for nonlinear model predictive control of switched systems with state-dependent switches and state jumps. *International Journal of Robust and Nonlinear Control*, *30*(2), 719–740.

Katayama, S., & Ohtsuka, T. (2020). Efficient solution method based on inverse dynamics of optimal control problems for fixed-based rigid-body systems. In *the 21st IFAC World Congress* (pp. 363–368).

Katayama, S., & Ohtsuka, T. (2021). Efficient solution method based on inverse dynamics for optimal control problems of rigid body systems. In *2021 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 2070–2076).

Katayama, S., & Ohtsuka, T. (2021a). Lifted contact dynamics for efficient optimal control of rigid body systems with contacts. In *2022 IEEE International Conference on Robotics and Automation (IROS) (accepted).* Retrieved from arXiv:2108.01781

Katayama, S., & Ohtsuka, T. (2021b). Riccati recursion for optimal control problems of nonlinear switched systems. In *the 7th IFAC Conference on Nonlinear Model Predictive Control (NMPC 2021)* (Vol. 54, p. 172-178).

Katayama, S., & Ohtsuka, T. (2021c). *Structure-exploiting Newton-type method for optimal control of switched systems.* Retrieved from arXiv:2112.07232

Katayama, S., & Ohtsuka, T. (2022). Efficient Riccati recursion for optimal control problems with pure-state equality constraints. In *2022 American Control Conference (ACC)* (pp. 3579–3586).

Katz, B., Di Carlo, J., & Kim, S. (2019). Mini cheetah: A platform for pushing the limits of dynamic quadruped control. In *2019 International Conference on Robotics and Automation (ICRA)* (pp. 6295–6301).

Kau, N., Schultz, A., Ferrante, N., & Slade, P. (2019). Stanford doggo: An open-source, quasi-direct-drive quadruped. In *2019 International Conference on Robotics and Automation (ICRA)* (pp. 6309–6315).

Kavraki, L. E., Svestka, P., Latombe, J.-C., & Overmars, M. H. (1996). Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, *12*(4), 566–580.

Koenemann, J., Del Prete, A., Tassa, Y., Todorov, E., Stasse, O., Bennewitz, M., & Mansard, N. (2015). Whole-body model-predictive control applied to the HRP-2 humanoid. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 3346–3351).

Köhler, J., Soloperto, R., Müller, M. A., & Allgöwer, F. (2020). A computationally efficient robust model predictive control framework for uncertain nonlinear systems. *IEEE Transactions on Automatic Control*, *66*(2), 794–801.

Kouzoupis, D., Frison, G., Zanelli, A., & Diehl, M. (2018). Recent advances in quadratic programming algorithms for nonlinear model predictive control. *Vietnam Journal of Mathematics*, *46*(4), 863–882.

Kuffner, J. J., Kagami, S., Nishiwaki, K., Inaba, M., & Inoue, H. (2002). Dynamically-stable motion planning for humanoid robots. *Autonomous Robots*, *12*(1), 105–118.

Kuffner, J. J., & LaValle, S. M. (2000). RRT-connect: An efficient approach to single-query path planning. In *2000 IEEE International Conference on Robotics and Automation (ICRA)* (Vol. 2, pp. 995–1001).

Kuindersma, S., Deits, R., Fallon, M., Valenzuela, A., Dai, H., Permenter, F., Koolen, T., Marion, P., & Tedrake, R. (2016). Optimization-based locomotion planning, estimation, and control design for the Atlas humanoid robot. *Autonomous Robots*, *40*, 429–455.

LaValle, S. M., & Kuffner Jr, J. J. (2001). Randomized kinodynamic planning. *The International Journal of Robotics Research*, *20*(5), 378–400.

LaValle, S. M., et al. (1998). Rapidly-exploring random trees: A new tool for path planning. *The annual research report*.

Lengagne, S., Vaillant, J., Yoshida, E., & Kheddar, A. (2013). Generation of whole-body optimal dynamic multi-contact motions. *The International Journal of Robotics Research*, *32*(9-10), 1104–1119.

Li, H., Frei, R. J., & Wensing, P. M. (2021). Model hierarchy predictive control of robotic systems. *IEEE Robotics and Automation Letters*, *6*(2), 3373–3380.

Li, H., & Wensing, P. M. (2020). Hybrid systems differential dynamic programming for whole-body motion planning of legged robots. *IEEE Robotics and Automation Letters*, *5*(4), 5448–5455.

Lowrey, K., Rajeswaran, A., Kakade, S., Todorov, E., & Mordatch, I. (2019). Plan online, learn offline: Efficient learning and exploration via model-based control. In *International Conference on Learning Representations (ICLR).*

Lynch, K. M., & Park, F. C. (2017). *Modern Robotics*. Cambridge University Press.

Mastalli, C., Budhiraja, R., Merkt, W., Saurel, G., Hammoud, B., Naveau, M., Carpentier, J., Righetti, L., Vijayakumar, S., & Mansard, N. (2020). Crocoddyl: An efficient and versatile framework for multi-contact optimal control. In *2020 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 2536–2542).

Mastalli, C., Havoutis, I., Focchi, M., Caldwell, D. G., & Semini, C. (2020). Motion planning for quadrupedal locomotion: Coupled planning, terrain mapping, and whole-body control. *IEEE Transactions on Robotics*, *36*(6), 1635–1648.

Mastalli, C., Merkt, W., Xin, G., Shim, J., Mistry, M., Havoutis, I., & Vijayakumar, S. (2022). Agile maneuvers in legged robots: A predictive control approach. *arXiv preprint arXiv:2203.07554*.

Mesbah, A., Streif, S., Findeisen, R., & Braatz, R. D. (2014). Stochastic nonlinear model predictive control with probabilistic constraints. In *2014 American Control Conference (ACC)* (pp. 2413–2419).

Messerer, F., Baumgärtner, K., & Diehl, M. (2021). Survey of sequential convex programming and generalized Gauss-Newton methods. *Citation: ESAIM: Proceedings and Surveys*, *71*(1), 64–88.

Miki, T., Lee, J., Hwangbo, J., Wellhausen, L., Koltun, V., & Hutter, M. (2022). Learning robust perceptive locomotion for quadrupedal robots in the wild. *Science Robotics*, *7*(62), eabk2822.

Mur-Artal, R., Montiel, J. M. M., & Tardos, J. D. (2015). ORB-SLAM: a versatile and accurate monocular SLAM system. *IEEE Transactions on Robotics*, *31*(5), 1147–1163.

Muratore, F., Eilers, C., Gienger, M., & Peters, J. (2021). Data-efficient domain randomization with bayesian optimization. *IEEE Robotics and Automation*

Letters, *6*(2), 911–918.

Murray, R. M., Li, Z., & Sastry, S. S. (2017). *A Mathematical Introduction to Robotic Manipulation.* CRC press.

Nakanishi, J., Mistry, M., & Schaal, S. (2007). Inverse dynamics control with floating base and constraints. In *2007 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 1942–1947).

Natale, L., Bartolozzi, C., Pucci, D., Wykowska, A., & Metta, G. (2017). iCub: The not-yet-finished story of building a robot child. *Science Robotics*, *2*(13), eaaq1026.

Neunert, M., Giftthaler, M., Frigerio, M., Semini, C., & Buchli, J. (2016). Fast derivatives of rigid body dynamics for control, optimization and estimation. In *2016 IEEE International Conference on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR)* (pp. 91–97).

Neunert, M., Stäuble, M., Giftthaler, M., Bellicoso, C. D., Carius, J., Gehring, C., Hutter, M., & Buchli, J. (2018). Whole-body nonlinear model predictive control through contacts for quadrupeds. *IEEE Robotics and Automation Letters*, *3*(3), 1458–1465.

Nguyen, Q., Hereid, A., Grizzle, J. W., Ames, A. D., & Sreenath, K. (2016). 3D dynamic walking on stepping stones with control barrier functions. In *2016 IEEE 55th Conference on Decision and Control (CDC)* (pp. 827–834).

Nocedal, J., & Wright, S. J. (2006). *Numerical Optimization* (second ed.). Springer.

Nurkanović, A., Albrecht, S., & Diehl, M. (2020). Limits of MPCC formulations in direct optimal control with nonsmooth differential equations. In *2020 European Control Conference (ECC)* (pp. 2015–2020).

Ohtsuka, T. (2004). A continuation/GMRES method for fast computation of nonlinear receding horizon control. *Automatica*, *40*(4), 563–574.

Ohtsuka, T., & Fujii, H. A. (1997). Real-time optimization algorithm for nonlinear receding-horizon control. *Automatica*, *33*(6), 1147–1154.

Orin, D. E., Goswami, A., & Lee, S.-H. (2013). Centroidal dynamics of a humanoid robot. *Autonomous robots*, *35*(2), 161–176.

Ott, C., Roa, M. A., & Hirzinger, G. (2011). Posture and balance control for biped robots based on contact force optimization. In *2011 11th IEEE-RAS International Conference on Humanoid Robots (Humanoids)* (pp. 26–33).

Pandala, A. G., Ding, Y., & Park, H.-W. (2019). qpSWIFT: A real-time sparse quadratic program solver for robotic applications. *IEEE Robotics and Automation Letters*, *4*(4), 3355–3362.

Patterson, M. A., & Rao, A. V. (2014). GPOPS-II: A MATLAB software for solving multiple-phase optimal control problems using hp-adaptive Gaussian quadrature collocation methods and sparse nonlinear programming. *ACM Transactions on Mathematical Software*, *41*(1).

Posa, M., Cantu, C., & Tedrake, R. (2014). A direct method for trajectory optimization of rigid bodies through contact. *The International Journal of Robotics Research*, *33*(1), 69–81.

Posa, M., Kuindersma, S., & Tedrake, R. (2016). Optimization and stabilization of trajectories for constrained dynamical systems. In *2016 IEEE International*

Conference on Robotics and Automation (ICRA) (pp. 1366–1373).

Pratt, J., Carff, J., Drakunov, S., & Goswami, A. (2006). Capture point: A step toward humanoid push recovery. In *2006 6th IEEE-RAS International Conference on Humanoid Robots (Humanoids)* (pp. 200–207).

Quirynen, R., Houska, B., Vallerio, M., Telen, D., Logist, F., Van Impe, J., & Diehl, M. (2014). Symmetric algorithmic differentiation based exact Hessian SQP method and software for Economic MPC. In *53rd IEEE Conference on Decision and Control* (pp. 2752–2757).

Raimondo, D. M., Limon, D., Lazar, M., Magni, L., & ndez Camacho, E. F. (2009). Min-max model predictive control of nonlinear systems: A unifying overview on stability. *European Journal of Control*, *15*(1), 5–21.

Ramuzat, N., Boria, S., & Stasse, O. (2022). Passive inverse dynamics control using a global energy tank for torque-controlled humanoid robots in multi-contact. *IEEE Robotics and Automation Letters*, *7*(2), 2787-2794.

Rao, C., Wright, S. J., & Rawlings, J. B. (1998). Application of interior-point methods to model predictive control. *Journal of Optimization Theory and Applications*, *99*(3), 723–757.

Rathod, N., Bratta, A., Focchi, M., Zanon, M., Villarreal, O., Semini, C., & Bemporad, A. (2021). Model predictive control with environment adaptation for legged locomotion. *IEEE Access*, *9*, 145710–145727.

Rawlings, K., Mayne, D. Q., & Diehl, M. (2017). *Model Predictive Control: Theory, Computation, and Design.* Nob Hill Publishing, LCC.

Reher, J., & Ames, A. D. (2021). Inverse dynamics control of compliant hybrid zero dynamic walking. In *2021 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 2040–2047).

Robuschi, N., Zeile, C., Sager, S., & Braghin, F. (2021). Multiphase mixed-integer nonlinear optimal control of hybrid electric vehicles. *Automatica*, *123*, 109325.

Saab, L., Ramos, O. E., Keith, F., Mansard, N., Soueres, P., & Fourquet, J.-Y. (2013). Dynamic whole-body motion generation under rigid contacts and other unilateral constraints. *IEEE Transactions on Robotics*, *29*(2), 346–362.

Sager, S., Jung, M. N., & Kirches, C. (2011, 06). Combinatorial integral approximation. *Mathematical Methods of Operations Research*, *73*, 363–380.

Saputra, M. R. U., Markham, A., & Trigoni, N. (2018). Visual SLAM and structure from motion in dynamic environments: A survey. *ACM Computing Surveys (CSUR)*, *51*(2), 1–36.

Saut, J.-P., Sahbani, A., El-Khoury, S., & Perdereau, V. (2007). Dexterous manipulation planning using probabilistic roadmaps in continuous grasp subspaces. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 2907–2912).

Scheel, H., & Scholtes, S. (2000). Mathematical programs with complementarity constraints: Stationarity, optimality, and sensitivity. *Mathematics of Operations Research*, *25*(1), 1–22.

Schulman, J., Duan, Y., Ho, J., Lee, A., Awwal, I., Bradlow, H., Pan, J., Patil, S., Goldberg, K., & Abbeel, P. (2014). Motion planning with sequential convex optimization and convex collision checking. *The International Journal of Robotics*

Research, *33*(9), 1251–1270.

Schultz, G., & Mombaur, K. (2010). Modeling and optimal control of human-like running. *IEEE/ASME Transactions on Mechatronics*, *15*(5), 783–792.

Schwarz, M., Milan, A., Periyasamy, A. S., & Behnke, S. (2018). RGB-D object detection and semantic segmentation for autonomous manipulation in clutter. *The International Journal of Robotics Research*, *37*(4-5), 437–451.

Sideris, A., & Rodriguez, L. A. (2011). A Riccati approach for constrained linear quadratic optimal control. *International Journal of Control*, *84*(2), 370–380.

Sleiman, J.-P., Carius, J., Grandia, R., Wermelinger, M., & Hutter, M. (2019). Contact-implicit trajectory optimization for dynamic object manipulation. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 6814–6821).

Sleiman, J.-P., Farshidian, F., Minniti, M. V., & Hutter, M. (2021). A unified MPC framework for whole-body dynamic locomotion and manipulation. *IEEE Robotics and Automation Letters*, *6*(3), 4688–4695.

Smaldone, F. M., Scianca, N., Lanari, L., & Oriolo, G. (2021). Feasibility-driven step timing adaptation for robust MPC-based gait generation in humanoids. *IEEE Robotics and Automation Letters*, *6*(2), 1582–1589.

Solà, J., Deray, J., & Atchuthan, D. (2020). *A micro Lie theory for state estimation in robotics.* arXiv:1812.01537.

Stasse, O., Flayols, T., Budhiraja, R., Giraud-Esclasse, K., Carpentier, J., Mirabel, J., Del Prete, A., Souères, P., Mansard, N., Lamiraux, F., et al. (2017). TALOS: A new humanoid research platform targeted for industrial applications. In *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)* (pp. 689–695).

Stellato, B., Banjac, G., Goulart, P., Bemporad, A., & Boyd, S. (2020). OSQP: an operator splitting solver for quadratic programs. *Mathematical Programming Computation*, *12*(4), 637–672.

Stellato, B., Ober-Blöbaum, S., & Goulart, P. J. (2017). Second-order switching time optimization for switched dynamical systems. *IEEE Transactions on Automatic Control*, *62*(10), 5407–5414.

Stewart, D. E., & Trinkle, J. C. (1996). An implicit time-stepping scheme for rigid body dynamics with inelastic collisions and coulomb friction. *International Journal for Numerical Methods in Engineering*, *39*(15), 2673–2691.

Sugihara, T., Nakamura, Y., & Inoue, H. (2002). Real-time humanoid motion generation through ZMP manipulation based on inverted pendulum control. In *2002 IEEE International Conference on Robotics and Automation (ICRA)* (Vol. 2, pp. 1404–1409).

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction.* MIT press.

Takenaka, T., Matsumoto, T., & Yoshiike, T. (2009). Real time motion generation and control for biped robot -1st report: Walking gait pattern generation-. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 1084–1091).

Tassa, Y., Erez, T., & Todorov, E. (2012). Synthesis and stabilization of complex behaviors through online trajectory optimization. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 4906–4913).

Thrun, S. (2002). Probabilistic robotics. *Communications of the ACM*, *45*(3), 52–57.

Tobin, J., Fong, R., Ray, A., Schneider, J., Zaremba, W., & Abbeel, P. (2017). Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 23–30).

Todorov, E. (2014). Convex and analytically-invertible dynamics with contacts and constraints: Theory and implementation in mujoco. In *2014 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 6054–6061).

Todorov, E., & Li, W. (2005). A generalized iterative LQG method for locally-optimal feedback control of constrained nonlinear stochastic systems. In *2005 American Control Conference (ACC)* (pp. 300–306).

Tonneau, S., Del Prete, A., Pettré, J., Park, C., Manocha, D., & Mansard, N. (2018). An efficient acyclic contact planner for multiped robots. *IEEE Transactions on Robotics*, *34*(3), 586–601.

Unitree Robotics. (n.d.). *A1 website.* https://www.unitree.com/products/a1/.

Verschueren, R., Frison, G., Kouzoupis, D., Frey, J., Duijkeren, N. v., Zanelli, A., Novoselnik, B., Albin, T., Quirynen, R., & Diehl, M. (2021). acados—a modular open-source framework for fast embedded optimal control. *Mathematical Programming Computation*, 1–37.

Vukobratović, M., & Stepanenko, J. (1972). On the stability of anthropomorphic systems. *Mathematical Biosciences*, *15*(1-2), 1–37.

Wang, Y., & Boyd, S. (2010). Fast model predictive control using online optimization. *IEEE Transactions on Control Systems Technology*, *18*(2), 267–278.

Westervelt, E., Grizzle, J., & Koditschek, D. (2003). Hybrid zero dynamics of planar biped walkers. *IEEE Transactions on Automatic Control*, *48*(1), 42–56.

Wolfslag, W. J., McGreavy, C., Xin, G., Tiseo, C., Vijayakumar, S., & Li, Z. (2020). Optimisation of body-ground contact for augmenting the whole-body loco-manipulation of quadruped robots. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 3694–3701).

Wurm, K. M., Hornung, A., Bennewitz, M., Stachniss, C., & Burgard, W. (2010). Octomap: A probabilistic, flexible, and compact 3D map representation for robotic systems. In *2010 IEEE International Conference on Robotics and Automation (ICRA) Workshop on Best Practice in 3D Perception and Modeling for Mobile Manipulation* (Vol. 2).

Wächter, A., & Biegler, L. (2006). On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, *106*, 25–57.

Xu, X., & Antsaklis, P. J. (2002). Optimal control of switched systems via non-linear optimization based on direct differentiations of value functions. *International Journal of Control*, *75*(16-17), 1406–1426.

Xu, X., & Antsaklis, P. J. (2004). Optimal control of switched systems based on parameterization of the switching instants. *IEEE Transactions on Automatic*

Control, *49*(1), 2–16.

Yunt, K. (2011). An augmented Lagrangian based shooting method for the optimal trajectory generation of switching Lagrangian systems. *Dynamics of Continuous, Discrete and Impulsive Systems Series B: Applications and Algorithms*, *18*(5), 615–645.

Yunt, K., & Glocker, C. (2006). Trajectory optimization of mechanical hybrid systems using SUMT. In *9th IEEE International Workshop on Advanced Motion Control* (pp. 665–671).

Zanelli, A., Domahidi, A., Jerez, J., & Morari, M. (2020). FORCES NLP: An efficient implementation of interior-point methods for multistage nonlinear nonconvex programs. *International Journal of Control*, *93*(1), 13–29.

Zanelli, A., Frey, J., Messerer, F., & Diehl, M. (2021). Zero-order robust nonlinear model predictive control with ellipsoidal uncertainty sets. In *7th IFAC Conference on Nonlinear Model Predictive Control (NMPC 2021)* (Vol. 54, pp. 50–57). Elsevier.

Zanelli, A., Quirynen, R., Jerez, J., & Diehl, M. (2017). A homotopy-based nonlinear interior-point method for NMPC. *IFAC-PapersOnLine*, *50*(1), 13188–13193.

Zavala, V. M., & Biegler, L. T. (2009). The advanced-step NMPC controller: Optimality, stability and robustness. *Automatica*, *45*(1), 86–93.

Zhang, J., & Singh, S. (2014). LOAM: Lidar odometry and mapping in real-time. In *Robotics: Science and Systems (RSS)* (Vol. 2, pp. 1–9).

Zhao, W., Queralta, J. P., & Westerlund, T. (2020). Sim-to-real transfer in deep reinforcement learning for robotics: a survey. In *2020 IEEE Symposium Series on Computational Intelligence (SSCI)* (pp. 737–744).

Zhao, Z.-Q., Zheng, P., Xu, S.-t., & Wu, X. (2019). Object detection with deep learning: A review. *IEEE Transactions on Neural Networks and Learning Systems*, *30*(11), 3212–3232.

Zucker, M., Ratliff, N., Dragan, A. D., Pivtoraiko, M., Klingensmith, M., Dellin, C. M., Bagnell, J. A., & Srinivasa, S. S. (2013). Chomp: Covariant hamiltonian optimization for motion planning. *The International Journal of Robotics Research*, *32*(9-10), 1164–1193.

List of Publications

Peer Reviewed Journal Articles

1. S. Katayama and T. Ohtsuka, “Inverse dynamics-based formulation of finite horizon optimal control problems for rigid-body system,” *Optimal Control Applications and Methods*, Vol. 42, No. 6, pp. 1–19, 2021. **Chapter 3**

2. S. Katayama, M. Doi, and T. Ohtsuka, “A moving switching sequence approach for nonlinear model predictive control of switched systems with state-dependent switches and state jumps,” *International Journal of Robust and Nonlinear Control*, Vol. 30, No.2, pp. 719–740, 2020. **Chapter 7**

Peer Reviewed International Conference Proceedings

1. S. Katayama and T. Ohtsuka, “Whole-body model predictive control with rigid contacts via online switching time optimization,” 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2022) **Chapter 7**

2. S. Katayama and T. Ohtsuka, “Lifted contact dynamics for efficient optimal control of rigid body systems with contacts,” 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2022) **Chapter 4**

3. S. Katayama and T. Ohtsuka, “Efficient Riccati recursion for optimal control problems with pure-state equality constraints,” 2022 American Control Conference (ACC 2022), pp. 3579–3586, 2022. **Chapter 5**

4. S. Katayama and T. Ohtsuka, “Riccati recursion for optimal control problems of nonlinear switched systems,” The 7th IFAC Conference on Nonlinear Model Predictive Control (NMPC 2021), Vol. 54, No. 6, pp. 172–178, 2021. **Chapter 6**

5. S. Katayama and T. Ohtsuka, "Efficient solution method based on inverse dynamics for optimal control problems of rigid body systems," 2021 IEEE International Conference on Robotics and Automation (ICRA 2021), pp. 2070–2076, 2021. **Chapter 3**

6. S. Katayama and T. Ohtsuka, "Efficient solution method based on inverse dynamics of optimal control problems for fixed-based rigid-body systems," The 21st IFAC World Congress 2020, Vol. 53, No. 2, pp. 6483–6489, 2020. **Chapter 3**

7. S. Katayama and T. Ohtsuka, "Automatic code generation tool for nonlinear model predictive control with Jupyter," The 21st IFAC World Congress 2020, Vol. 53, No. 2, pp. 7033–7040, 2020.

8. S. Katayama and T. Ohtsuka, "Scenario-based nonlinear model predictive control for switched systems with externally forced switchings," The 57th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE), pp. 1098–1103, 2018.

9. S. Katayama, Y. Satoh, M. Doi and T. Ohtsuka, "Nonlinear model predictive control for systems with state-dependent switches and state jumps using a penalty function method," 2018 IEEE Conference on Control Technology and Applications (CCTA), pp. 312–317, 2018.

10. S. Katayama, Y. Satoh, M. Doi and T. Ohtsuka, "Nonlinear model predictive control for systems with autonomous state jumps using a penalty function method," The 11th Asian Control Conference (ASCC), pp. 2125–2130, 2017.

Domestic Conference Proceedings

1. S. Katayama and T. Ohtsuka, "Efficient numerical optimal control of switched systems and its application to whole-body optimal control of walking robots," The 64th Japan Joint Automatic Control Conference, 2021 (Outstanding Presentation Award)

2. S. Katayama and T. Ohtsuka, "Fast and robust numerical optimal control solver for multi-body robotic systems," The 39th annual conference of the Robotics Society of Japan (RSJ), 2021

Poster Presentation

1. S. Katayama and T. Ohtsuka, "Whole-body model predictive control (MPC) solver for robotics systems," The 16th ICT Innovation, Kyoto University, 2022 (Outstanding Research Award)