IMT School for Advanced Studies Lucca

Lucca, Italy

Model Predictive Control for Legged Robots

PhD Program in Systems Science - Track in Computer Science and Systems Engineering

XXXIII Cycle

By

Niraj Rathod

2023

The dissertation of Niraj Rathod is approved.

Program Coordinator: Prof. Alberto Bemporad,
IMT School for Advanced Studies Lucca, Italy

Supervisor: Prof. Alberto Bemporad,
IMT School for Advanced Studies Lucca, Italy

Co-supervisor: Dr. Michele Focchi,
DLS Lab, Istituto Italiano di Tecnologia, Genova, and
University of Trento, Italy

Co-supervisor: Assoc. Prof. Mario Zanon,
IMT School for Advanced Studies Lucca, Italy

The dissertation of Niraj Rathod has been reviewed by:

Assoc. Prof. Andrea Del Prete,
Industrial Engineering Department,
University of Trento, Italy

Assoc. Prof. Dimitrios Kanoulas,
Computer Science Department,
University College London, United Kingdom

IMT School for Advanced Studies Lucca

2023

To my parents,

Suman and Jogram

Contents

List of Figures

List of Tables

Acknowledgements

First and foremost, I express my deepest gratitude to my advisor Prof. Alberto Bemporad for such a rewarding thesis experience. I shall always be thankful for your guidance and support throughout the research work. I am also very grateful to my co-advisor Dr. Michele Focchi for your caring, patience, and unconditional support throughout my thesis. You have been a great help during the downfalls and an inspiration during my research experience. Your energy and excitement for research are unparallel that has boosted my confidence. I could have never hoped for a co-advisor better than you. I sincerely thank my co-advisor Dr. Mario Zanon for his help. Thanks for introducing me to the world of optimal control, without which this work would not have been possible to achieve.

Besides my advisors, I thank Dr. Claudio Semini for the collaboration opportunity with the DLS Lab at IIT Genova. He and the entire DLS have been very kind and supportive during my visits to the Lab. From DLS Lab, I would like to thank Amit, Abdelarhman, Chundri, Geoff, Gianluca, Luca, Romeo, Mattia, Matteo, Marco, Salvatore, Shamel, Octavio, Victor, and Ylenia for your generous support during lab experiments and brainstorming on the research topics. Thanks to DLS administrative staff for supporting the bureaucracy during my visits to DLS Lab. Last but not least, I would like to especially thank Angelo for collaborating with me on the PhD research. It has been a pleasure working with you.

I would like to thank my colleagues and friends from the DYSCO lab at IMT Lucca for their company. Special thanks to my friends Sampath, Laura, Jeniya, Manas, Surya, Pavan, Mengia, and Chinmay for being there for me during my PhD. You have motivated me during the most challenging periods and were the backbone of life here at IMT. I am deeply grateful to my dear friend Vihang for his support during the

PhD research. Thanks for pushing me to reach my best level and always encouraging me. I want to thank my friends Giulia, Laura, Marie, Amina, Divya, Kapil, and Akash for your pleasant company during my stay in Genova for the research period. My deepest gratitude goes to the IMT administrative staff for their continuous support and help to PhD students at IMT. I shall forever cherish my roommate Emiliano's support and company during PhD studies.

I also take this opportunity to thank my friends from my current workplace, Vito, Domenico, Leonardo, Tommaso, Francesco, and Andrea, for always keeping a positive and uplifting environment in the office while I was finalizing my thesis writing. Special thanks to my friends from Crema Daniele, Federica, Alessandro, and Massimo, for allowing me to blend in this new city with your warm company. I can never thank my friends Akash, Vignesh, Stefano, Harish, Nitin, Mauro, Giacomo, Katarina, Paolo, and Luca enough. You have all been a source of love, joy, and energy since I have known you.

From the bottom of my heart, I wish to thank my family members Sunita, Ashok, Arti, Vinod, Vidya, Santosh, Vanita, Tulsiram, Arjun, Vaishali, Shrikant, and Dipali for rooting for me during my PhD studies, without which this journey would have never happened. Kudos to my nieces Vaishnavi, Akanksha, Nivita, Sanchiti, Swara, and Mishti, and my nephews Himanshu, Aditya, Parth, Mayank, and Advait for keeping a smile on my face. You all are such a source of love and joy for me. Last, I would like to thank my mother and father for your unconditional love, belief in me, and support throughout my life. Words are never enough to express my gratitude for your sacrifices to get me where I am today.

Vita

Feb. 24, 1989	Born in Parasram Naik Tanda, India
2007 – 2011	B.Tech in Instrumentation & Control College Of Engineering Pune, India
2011 – 2014	System Engineer Emerson Export Engineering Centre (EEEC) Pune, India
2014 – 2017	M.Sc. in Automation and Control Engineering Politecnico di Milano, Italy
2015 – 2016	Research Intern Ricerca sul Sistema Energetico SpA Italy
2017 – 2022	PhD in Computer Science and Systems Engineering IMT School for Advanced Studies Lucca Italy
2018 – 2021	Affiliated Researcher Dynamic Legged Systems Lab, IIT Genova Italy
2022 – Present	Control Systems Research Engineer R&D, VHIT Spa Italy

Publications

Journal papers and Conference proceedings

a. A. Bratta, M. Focchi, N. Rathod, and C. Semini, “Optimization-Based Reference Generator for Nonlinear Model Predictive Control of Legged Robots”, *Robotics 2023*, 12(1), 6, 2023. (url)

b. N. Rathod, A. Bratta, M. Focchi, M. Zanon, O. Villarreal, C. Semini, A. Bemporad, “Model Predictive Control with Environment Adaptation for Legged Locomotion”, *IEEE Access*, vol. 9, pp. 145710-145727, 2021. (url)

c. A. Bratta, N. Rathod, M. Zanon, O. Villarreal, A. Bemporad, C. Semini, M. Focchi, “Towards a Nonlinear Model Predictive Control for Quadrupedal Locomotion on Rough Terrain”, *Italian Institute of Robotics and Intelligent Machines (I-RIM) Conference*, 3rd edition, Rome, Italy, 2021. (url)

d. N. Rathod, A. L. Bella, G. Puleo, R. Scattolini, A. Rossetti, C. Sandroni, “Modelling and predictive control of a solar cooling plant with flexible configuration”, *Journal of Process Control*, vol. 76, pp. 74-86, 2019.(url)

Workshops

a. N. Rathod, A. Bratta, M. Focchi, M. Zanon, O. Villarreal, C. Semini, A. Bemporad, “Real-time MPC with Mobility-enhanced Feature for Legged Locomotion”, *Towards Real-World Deployment of Legged Robots*, ICRA, Xi’an China, 2021. (url)

b. N. Rathod, M. Focchi, M. Zanon, A. Bemporad, “Optimal control based replanning for Quadruped robots”, *Numerical Optimization for Online Multi-Contact Planning and Control*, Robotics: Science and Systems, Freiburg, Germany, 2019.

Acronyms

Notation	**Description**
CD	Centroidal Dynamics.
CMM	Centroidal Momentum Matrix.
CNN	Convolutional Neural Network.
CoM	Center of Mass.
DDP	Differential Dynamic Programming.
DLS	Dynamic Legged System.
DoF	Degrees of Freedom.
GRF	Ground Reaction Force.
HAA	Hip Adduction-Abduction.
HFE	Hip Flexion-Extension.
HPU	Hydraulic Pressure Unit.
HyQ	Hydraulically actuated Quadruped.
IIT	Istituto Italiano Tecnologia.
IMU	Inertial Measurement Unit.
KFE	Knee Flexion-Extension.
LF	Left-Front.
LIP	Linear Inverted Pendulum.
LIPOpt	LIP model Optimization.
LP	Linear Programming.
LTI	Linear Time-Invariant.
LTV	Linear Time-Varying.
LTVOpt	LTV-Based Trajectory Optimization.

Notation	**Description**
MPC	Model Predictive Control.
NLP	Nonlinear Programming.
NMPC	Nonlinear Model Predictive Control.
OCP	Optimal Control Problem.
ODE	Ordinary Differential Equation.
ORG	Optimization-Based Reference Generator.
PD	Proportional-Derivative.
QP	Quadratic Programming.
QPMap	QP Mapping.
RBD	Rigid Body Dynamics.
RF	Right-Front.
RHP	Receding Horizon Principle.
ROS	Robot Operating System.
RTI	Real-Time Iteration.
SLQ	Sequential Linear Quadratic.
SQP	Sequential Quadratic Programming.
SQPOpt	SQP-Based Trajectory Optimization.
SRBD	Single Rigid Body Dynamics.
TO	Trajectory Optimization.
VFA	Vision-based Foothold Adaptation.
WBC	Whole-Body Control.
ZMP	Zero Moment Point.

Abstract

Optimal planning is essential when it comes to autonomy in legged locomotion. In the last few decades, different optimization techniques have been presented to design a legged locomotion framework, such as Trajectory Optimization (TO) and Model Predictive Control (MPC). The choice of a dynamic model utilized while synthesizing these planners plays a pivotal role because the chosen model defines the accuracy of the planning and also becomes a deciding factor for the computational cost of these techniques. In the first part of this thesis, we propose a closed-loop validation procedure for the Single Rigid Body Dynamics (SRBD) model and its variants used for optimal planning. Thereafter, we introduce a Linear Time-Varying (LTV) based TO for legged locomotion, followed by the simulation results and discussion on its limitations in re-planning.

Re-planning in legged locomotion is crucial to track the desired user velocity while adapting to the terrain and rejecting external disturbances. In the second part of this thesis, we propose and test in experiments a real-time Nonlinear Model Predictive Control (NMPC) tailored to a legged robot to achieve dynamic locomotion on various terrains. We introduce a novel mobility-based criterion to define an NMPC cost that enhances the locomotion of quadruped robots while maximizing leg mobility and improving adaptation to the terrain features. The NMPC is based on the Real-Time Iteration (RTI) scheme that allows us to re-plan online at 25 Hz with a prediction horizon of 2 seconds. In simulations, the NMPC is tested to traverse a set of pallets of different sizes, walk into a V-shaped chimney, and locomote over rough terrain. In real experiments, we demonstrate the effectiveness of our NMPC with the mobility feature that allowed IIT's 87 kg quadruped robot HyQ to achieve an omni-directional walk on flat terrain, traverse a static pallet, and adapt to a repositioned pallet during a walk.

In the final part of this thesis, we present the extension of the NMPC with other dynamic gaits, i.e., trot and pace. We also introduce an Optimization-Based Reference Generator (ORG) that computes dynamically feasible trajectories for the state and control input based on the Linear Inverted Pendulum (LIP) model-based optimization and Quadratic Programming (QP) based mapping. These feasible trajectories are passed to the NMPC to cope with the disturbances while following the user-defined trajectories with the dynamic gaits. We show the effectiveness of this two-stage optimization scheme in simulations and experiments performed on the AlienGo robot to trot in a straight line and to recover from the external disturbances while trotting. We also compare the performance of the two-stage scheme with respect to a traditional heuristic reference generator in an experiment.

Chapter 1

Introduction

Automating repetitive tasks has been an essential part of modern human life. In the last few decades, automation has made its way into our day-to-day life due to technological advancements. Decades of research and scientific discoveries have contributed to finding tailored solutions for automating mundane tasks in industries as well as our daily lives. Robots have gained popularity for being one of these mediums to perform automated tasks. Although robots are deployed in industries to perform meticulous and repetitive tasks, their other potential has also been realized. Be it a state-of-the-art humanoid, quadruped robot, cleaning robot, robotic lawn mower, or robot as a companion, robots have broken through what was once considered a repetitive task-performing machine in an industrial setup. These applications have opened up different possibilities for robots in a broader spectrum that could demand the agility to navigate through challenging environments. Unlike industrial robots, where the environment is usually predictable, robots in these scenarios may face rough terrains while performing any given task.

Of all the robot types, legged robots have gained immense popularity not only in the research and development domain but also in the public eye in the last two decades. One of the reasons to be welcomed by society is their resemblance to human beings and animals. However, these robots are of interest not merely because they resemble living beings but because of their agility to perform tasks in undefined and uneven terrain, unlike flying or wheeled robots. These robots resemble the fundamental anatomy of human beings, and animals, therefore, possess similar navigational advantages.

The focus of this dissertation is on the development and improvement of the navigational capability for legged robots using state-of-the-art control techniques. In the following sections, we motivate the thesis topic, list the contribution and outline the content of this dissertation.

1.1 Motivation

From a research and development point of view, robotics blends many engineering disciplines, such as mechanical, electrical, electronics, mathematical, computer, and control engineering. Building a sophisticated working robot requires a unique and extraordinary harmony of these engineering disciplines. The multidisciplinary approach required by the robotics domain has also given rise to state-of-the-art humanoids and quadrupeds.

Even though legged robots resemble living beings, these robots lack sensory mechanisms and intelligence identical to living beings that have undergone rigorous evolution over thousands of years. While designing animal-like robots, the focus has not been to replicate exact mechanics or behavior as living beings but to cherry-pick only those that allow a legged robot to navigate through the environment with the help of equipped actuators and sensors. Besides the mechanical design of legged robots, the development of their intelligence has seen exponential growth in the last few decades. The increased computational power of computers has allowed many complex algorithms to run on these robots with lightning speed and efficiency.

When it comes to the navigational capability of a legged robot, it is incomparable to living beings with similar anatomy since they do not share similar actuation and sensory elements. Even if some of these sensing elements are present in robots, how a robot processes information differs significantly from an animal during locomotion. Therefore, extensive research currently focuses on developing the intelligence for legged robots to improve their navigational capabilities. Improving the navigation capability of a legged robot requires efficient use of the information about the environment and rapidly processing that information to choose a locomotion strategy to navigate through challenging terrains optimally. In this dissertation, we focus on developing a real-time optimal motion planning scheme that improves the navigational capability of a legged robot to locomote through challenging environments.

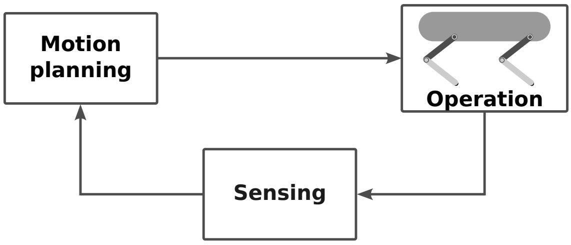


Figure 1: A schematic overview of the components involved in Legged locomotion.

1.2 Legged Locomotion in a Nutshell

There are broadly three essential aspects to legged locomotion, as depicted in Fig. 1. The sensing and motion planning modules are where the information about the robot and environment is collected and processed to operate the tasks defined for a legged robot. We will briefly discuss these aspects in the following.

Motion planning

This module defines a task for the robot at a higher level and can be generated heuristically or with the intelligent algorithms aware of the robot's dynamics and environment. This module is where also adaptive planning is performed to cope with the changes in the environment and improvise the robot's motion according to the disturbances acting on the robot, such as external pushes. Motion adaptation is achieved through re-planning by continuously processing the information about the environment and acting accordingly.

Sensing

This module makes the robot aware of its state and environment by reading through various sensors, such as a camera. The broad classification of these sensors is 1) Proprioceptive, which measures the robot's state (e.g., body position), and 2) Exteroceptive, which measures the environment state (e.g., temperature, pressure, mapping). This information is fed to the motion planning module to plan a desired locomotion.

Operation

In this step, the robot executes the planned motion provided by the motion planning module. After digesting the updated information from the sensing module, the motion planner sends a motion plan to the robot either to execute task-specific locomotion or to adapt to the changes if diverted from the original plan due to disturbance or changes in the environment.

The scope of this dissertation lies in the *motion planning* module to develop a real-time motion planning strategy for a legged robot. Next, we will introduce this strategy.

1.3 Optimal Planning for Legged Locomotion

Trajectory Optimization (TO) has been utilized in Legged robots for a long time and has proven one of the successful approaches to improve the locomotion of legged robots. Use of TO has shown outstanding results, such as achieving a back-flip with a biped Atlas™ BostonDynamics (2023). Recently, Cassie AgilityRobotics (2023) has set a Guinness world record for the fastest $100\,\mathrm{m}$ by a bipedal robot. However, there is still scope for improvement regarding the robot's posture while walking. A robot can maintain a flat base while climbing a staircase or adapt its pose while performing the same task. In the first case, the robot might lose leg mobility during navigation, eventually hindering its ability to navigate through a challenging environment. On the other hand, if the robot continuously adjusts its body posture to align with the plane of the terrain, then the leg mobility of the robot increases. Further, this also allows a robot to reach desired footholds while walking.

In this work, we address the problem of leg mobility of a legged robot in an Nonlinear Model Predictive Control (NMPC) setup. This work also addresses the fundamental issue of the computation time in real-time implementation of the NMPC tailored for legged locomotion. Further, we also study the efficacy of a two-stage optimization scheme in which a real-time Optimization-Based Reference Generator (ORG) provides physically feasible references to the NMPC. In the following sections, we describe the contribution of this dissertation.

1.4 Contribution

The contribution of this dissertation is divided into three major parts. The first part focuses on the Single Rigid Body Dynamics (SRBD) model validation and TO for legged locomotion. In the second part, we propose a real-time NMPC for environment adaptation and show its performance in simulation and experiments. In the last part of the thesis, we extend the application of our NMPC to cope with external disturbances with the help of an optimal reference generator.

1.4.1 Model Validation and Trajectory Optimization

In the first part of the dissertation, we propose a method for the closed-loop validation of the SRBD model for a legged robot. In particular, we begin with the continuous-time SRBD model and derive its discrete-time nonlinear and Linear Time-Varying (LTV) versions. Then, we perform validation with respect to the data obtained from the high-fidelity model of Hydraulically actuated Quadruped (HyQ). Next, we use the LTV model to first design the LTV-Based Trajectory Optimization (LTVOpt) and then SQP-Based Trajectory Optimization (SQPOpt). Finally, we present and discuss the result of these TO methods in simulation for HyQ robot.

1.4.2 NMPC with Mobility

In the second part of this dissertation which is a major contribution, we demonstrate in experiments with our $87\,\mathrm{kg}$ HyQ robot Semini et al. (2011) that a suitably formulated NMPC can tackle rough terrain locomotion, account for leg mobility, and provide optimal base orientation while being real-time feasible. Indeed, optimizing for leg mobility allows our NMPC to devise a robot base orientation and height that improves locomotion on rough terrain. This is particularly useful to achieve *environment adaptation* on rough terrains. Another advantage is that minimal heuristics is required from the user, i.e., no reference trajectory for the robot's height, and its base roll and pitch orientation is needed. This work is a system integration on the same line of our previous work Mastalli et al. (2020). However, while in Mastalli et al. (2020) only offline optimization was performed, here we achieve real-time feasible online replanning in an MPC fashion. To achieve this goal:

a. We consider a simplified SRBD model that describes the angular and translational dynamics of the robot base but neglects the dynamics of legs.

b. We employ the Real-Time Iteration (RTI) scheme Diehl et al. (2005b,a); Gros et al. (2020) that allows us to run our NMPC online with the prediction horizon of $2\,\mathrm{s}$ (50 nodes) as opposed to the $0.5\,\mathrm{s}$ (125 nodes) used by Neunert et al. (2018). Differently from Cebe et al. (2020) (that re-plans at each foot touchdown event), we continuously re-plan at the rate of $25\,\mathrm{Hz}$.

c. We run our NMPC on a single computer along with the rest of our locomotion framework[1] unlike in Cebe et al. (2020); Neunert et al. (2018) where they use dedicated computers for their TO and NMPC, respectively.

We show in simulation the robot traversing a set of pallets of different dimensions placed relatively at varying distances, walking into a V-shaped chimney and over randomly generated rough terrain. We present *experimental* results that demonstrate the capability of our NMPC to generate an omni-directional walk and to traverse a pallet for our quadruped robot HyQ. We tested the re-planning capability of our approach by pushing a pallet in front of the robot while walking, such that the control algorithm has to re-plan online in order to adapt to a dynamically changing environment.

1.4.3 Two-Stage Optimization

In the second part of the contributions, we present the NMPC with environment adaptation for crawl gait on HyQ robot. In this work, we extend the application of the NMPC with ORG for trot and pace gait on the AlienGo robot. We modify our framework to introduce the Optimization-Based Reference Generator (ORG) instead of the heuristic reference generator presented in the previous contribution. The ORG provides physically feasible references to the NMPC that allow the NMPC to trot and pace in simulation. We compare the performance of the ORG with the heuristic reference generator. Finally, the experiment shows that the two-stage optimization, i.e., NMPC with ORG, enables the robot to cope with external disturbances while trotting.

[1]Except the perception related modules that run on a dedicated computer

1.5 Outline

This dissertation is organized into different chapters to discuss the background and literature survey of the thesis topic, robot models, and validation of the chosen model for the TO and NMPC, TO based on two different techniques, NMPC for environment adaptation, and finally, NMPC with the optimization-based reference generator. We end the manuscript by concluding remarks and future works based on the topics discussed in this dissertation. In the following, we outline the content of each chapter.

Chapter 2 presents the literature survey on the related work to this dissertation. In this chapter, we analyze state-of-the-art related to 1) dynamic models used in TO and NMPC, 2) solution methods for MPC, 3) environment adaptation with the leg mobility inside the NMPC, and 4) two-stage planning with NMPC and optimal reference generator.

Chapter 3 starts by describing the quadruped robots used in this work. Further, it discusses the most common robot models for optimal planning and describes the SRBD model and its discrete-time and LTV forms. This chapter closes with the model validation of the SRBD model performed for the HyQ robot.

Chapter 4 briefly discusses MPC and OCP concepts. Then, it discusses direct methods for the solution of OCP and describes sequential and simultaneous optimal control. Ultimately, we propose the LTV-based and SQP-based trajectory optimization for legged robot and close the chapter with simulation results.

Chapter 5 starts with an overview of our locomotion planning pipeline and describes the NMPC setup. Then, we describe leg mobility and other features of our NMPC design. Thereafter, we describe the generation of references for the NMPC, followed by a section on the WBC. We then summarize the RTI scheme for our NMPC. Finally, we demonstrate several simulation and experimental results with HyQ robot.

Chapter 6 introduces our modified locomotion framework with the ORG. In this chapter, we discuss the setup of the NMPC with ORG to perform dynamic locomotion with trot and pace gait. Then, we discuss the elements involved in the design of the ORG. At the end of the chapter, we present the result of two-stage optimization with NMPC and ORG in simulation and experiment on the AlienGo robot.

Chapter 7 draws conclusions and discusses the final thoughts on the work presented in this dissertation. This chapter also highlights possible future directions to the work presented in this dissertation.

Chapter 2

Related Work

The main advantage of legged robots with respect to their wheeled counterpart is their ability to traverse complex and unstructured environment such as forests, obstacles, and debris. However, the control of legged robots poses complex problems related to underactuation (the body is controlled only indirectly through the legs), and to the hybrid nature of the forces required to generate motion, since the robot needs to establish and interrupt contact between its feet and the ground. The control design for legged robots was initially dealt with by using heuristic approaches which yielded successful results such as in the walking machines from Raibert (1986), the virtual model control of Pratt et al. (2001) and in the heuristic locomotion planning for quadrupedal robots by Focchi et al. (2020a). However, heuristic approaches have several limitations, for example: 1) they cannot be easily generalized to all kinds of terrain and motions; 2) they cannot account for the future state of the robot hence, they have no possibility to guarantee physical feasibility of the planned trajectories. The challenge of avoiding these undesirable *myopic* behaviors in heuristic planning approaches has motivated the research towards new optimization-based predictive locomotion planning.

Formulating the locomotion planning as an optimization problem allows one to represent high-level locomotion tasks as cost functions and system dynamics using constraints. Besides robot dynamics, the locomotion tasks should also respect the contact dynamics such as unilateral force and friction cone constraints, that are critical to stabilize the locomotion. The use of optimization techniques to design WBC has enabled legged robots to traverse soft terrains Fahmi et al. (2020) and to be versatile in terms of type of gait and motions that a legged robot can achieve Bellicoso et al. (2017). The aforementioned examples are based on the solution of a Quadratic Program that only considers the instantaneous Kuindersma et al. (2014) effects of the joint torques on the robot's base. Further, similar to heuristic approaches mentioned earlier, these approaches do not consider the information about the future states of the robot and hence cannot assure recursive feasibility.

In order to address this issue, several approaches make use of TO-based locomotion planning considering the full dynamics of the robot Neunert et al. (2017); Posa et al. (2016). However, these approaches usually suffer from high computational time hence they are often restricted to *offline* (open-loop) use. In general, offline planners Aceituno-Cabezas et al. (2018); Melon et al. (2020) neither adapt to quick terrain changes nor cope with state drifts and uncertainties. To address this issue the concept of *online* re-planning can be used. Online re-planning can intrinsically cope with the problem of error accumulation in planned motion that is common in real scenarios.

For online re-planning, MPC has gained broad interest in the robotics community for legged locomotion. Moreover, the intrinsic feedback mechanism offered by MPC can compensate for modeling errors and disturbances acting on the system provided that the MPC is executed at a sufficiently high rate in closed-loop.

2.1 Dynamic Robot Models for TO and MPC

A careful choice of the dynamic model by the MPC formulation is typically required to achieve a desired re-planning frequency in closed-loop, given the limited computational resources available for online computations. For example, using a full dynamics model of legged robots inside MPC Neunert et al. (2018); Koenemann et al. (2015) with long prediction horizon may result in an optimization problem which requires excessive computations for real-time deployment at high sampling rates. Using approximate models is a way to reduce the complexity of the optimization, trading the accuracy with computational efficiency. Following this line, Dai et al. (2014) used a Centroidal Dynamics (CD) plus a full-kinematic model to enforce the kinematic limits in TO to plan complex behaviors on the humanoid robot Atlas. The CD model considers contact forces as input and links the linear and angular momentum of the robot to the external wrench Orin et al. (2013).

A simplified version of the CD model is the SRBD model where the inertia of the legs is neglected (assumption of massless legs) and the robot's body and legs are lumped into a single rigid body. This model is well suited for quadrupeds, since they usually concentrate their mass and inertia in the robot base, unlike humanoids. The SRBD model was used for TO Winkler et al. (2018) and MPC Bledt et al. (2017) to jointly optimize for footholds, Center of Mass (CoM) trajectories and contact forces. By further linearizing the angular part of the dynamics of the SRBD, Di Carlo et al. (2018) was able to achieve a variety of quadrupedal gaits in experiments but their approach was not suitable for motions that involve large variations from the horizontal orientation. The simplest among all the approximate models mentioned earlier is the Linear Inverted Pendulum (LIP) and it has been used inside MPC for quadruped Horvat et al. (2017) and biped Herdt et al. (2010) locomotion. However, there are two main limitations in LIP model, namely it neglects angular dynamics and assumes constant robot height. Additionally, it does not account for friction cones, so that the contact stability on non-flat terrain can not be guaranteed. A comprehensive list of models and their characteristics has been reported in Table 1. In chapter 3, we will briefly review some of these models commonly used in the TO and MPC.

2.2 Solution Methods for MPC

While models play an important role in obtaining computationally light MPC formulations, the choice of solution method is also paramount to achieve fast online re-planning with MPC. A Differential Dynamic Programming (DDP) based approach demonstrated the real-time performance with whole-body MPC Koenemann et al. (2015) on HRP-2 humanoid. Recently, Grandia et al. (2019) proposed a DDP-based MPC using a kinodynamic model which re-plans at a frequency of $15\,\mathrm{Hz}$ with a prediction horizon of $1\,\mathrm{s}$ on a quadruped. The main drawback of DDP based approaches is the difficulty in implementing hard-inequality and switching constraints. Hard-inequality constraints need to be implemented as penalties (e.g., with relaxed-barriers) Hauser and Saccon (2006) while the switching constraints are formulated using Augmented Lagrangian methods Lantoine and Russell (2012); Li and Wensing (2020). Though not obvious at first sight, these methods are essentially equivalent to direct optimal control based on multiple-shooting Bock and Plitt (1984) in combination with some form of Nonlinear Programming (NLP)

Table 1: Types of dynamic model commonly appear in legged locomotion. Variable z, $\mathbf{f}_i$, F_c and $\boldsymbol{\tau}$ are the Zero Moment Point (ZMP), contact force, Ground Reaction Force (GRF) and torque, respectively. Parameter n is the number of joints. Symbols $\times$ and ✓ represents if the corresponding feature is present or absent in the model.

Model	DoF	Flight phase	Uneven terrain	Angular Dynamics	Friction constraint	Torque constraint	Input
LIP (walking)	2	$\times$	$\times$	$\times$	$\times$	$\times$	z
LIP + Flywheel	2	$\times$	$\times$	✓	$\times$	$\times$	z
SLIP (Running)	3	✓	✓	$\times$	$\times$	$\times$	F_c
Point Mass	3	$\times$	✓	$\times$	$\times$	$\times$	F_c
SRBD	6	✓	✓	✓	✓	$\times$	$\mathbf{f}_i$
CD	6+n	✓	✓	✓	✓	$\times$	$\mathbf{f}_i$
RBD	6+n	✓	✓	✓	✓	✓	$\boldsymbol{\tau}$, $\mathbf{f}_i$

solvers using barrier functions in the real-time iteration scheme Diehl et al. (2005b). One such framework is provided by `acados` Verschueren et al. (2019).

In addition to DDP-based MPC, there also exist a few implementations of NLP-based MPC for legged robots. One such NMPC implementation with CoM dynamics plus full kinematic model was demonstrated in Farshidian et al. (2017) using a Sequential Linear Quadratic (SLQ) algorithm for a trotting gait on flat terrain. Neunert et al. Neunert et al. (2018) achieved a fast re-planning frequency of 80-170 Hz for a small prediction horizon of 0.5 s (125 nodes) with their NMPC using the full dynamics of the robot, and optimizing foot locations, swing timing, and locomotion sequences along with full-body dynamics. However, in the real experiments they have only demonstrated slow trotting on flat terrain. Moreover, since their approach does not consider the map of the terrain, it has limited application on uneven terrain conditions. An interesting observation is that they did not see a noticeable degradation in the closed-loop performance of the NMPC when the re-planning frequency was dropped until 30 Hz, demonstrating that the predictive nature of the MPC empowers the robot to tolerate much lower re-planning frequency. A similar observation was made in Bledt and Kim (2019) with an MPC scheme which optimizes foot locations, but requires a heuristic conditioning of the cost function. In their experiments, the robot is stable if the re-planning occurs at 20 Hz, unstable for lower frequencies and the performance improvement is observed over 40 Hz.

2.3 Leg Mobility in Predictive Control

The aforementioned approaches have been successful in controlling legged robots, but neglected an important aspect of these robots, which is usually referred to as *mobility*. In this paper, we define the mobility as the attitude of the robot leg to arbitrarily change its foot position Sciavicco et al. (2000). We noticed that maximizing mobility improves terrain adaptation hence it is advantageous to account for it in the motion planning of legged robots. Furthermore, as discussed in Section 5.3.1, adding mobility in the NMPC cost eliminates the need to specify references for the roll, pitch and height of the robot.

To achieve kinematically suitable configurations for the legs, a common heuristic is to align the robot base with the terrain inclination (estimated in Focchi et al. (2020a) via fitting an averaging plane through the stance feet). This approach aims to bring the legs as close as possible to the middle of their workspaces in order to avoid the violation of the kinematic limits. Optimization of mobility allows to achieve a similar behaviour in an automatic way. Fankhauser et al. in Fankhauser et al. (2018) maximized mobility by encoding it in a cost function that penalizes the distance with respect to a default foot position. Recently, Cebe et al. (2020) implemented TO using an SRBD model and also incorporating the feet positions in the optimization. They re-plan only at the feet touchdowns due to the high computation demand of their TO algorithm and showed experimental results on uneven terrain. Since their planner does not plan during the swing phase of the legs, they do not run their planner in an MPC fashion. Apart from the previously mentioned contributions, to the best of our knowledge no prior work has addressed the mobility with MPC in legged locomotion.

2.4 Optimal Reference Generator for MPC

The importance of the MPC approach in legged locomotion is evident in the presence of disturbances since the high-frequency re-planning of MPC allows a legged robot to comply with external pushes, e.g., in Meduri et al. (2023). One of the shortcomings of a standard MPC implementation is related to the fact that the robot becomes *transparent* to external pushes, i.e., at each re-planning instance, the MPC delivers a trajectory starting from an actual robot position. Therefore, changes in the generation of Cartesian reference position are required to return to the original position after a push. A common practice is to exploit user-defined values as references in the MPC Di Carlo et al. (2018), but this approach requires a user to manually change the reference velocity in order to compensate for the deviation caused by a push.

Another possible solution is to add a simple Cartesian Proportional-Derivative (PD) controller parallel to the MPC that attracts the CoM position to the reference Grimminger et al. (2020). However, this control strategy has some notable limitations, in particular: 1) the PD wrench is unaware of the wrench provided by the MPC; therefore, these two controllers can conflict and violate the feasibility; 2) the PD is unaware of the hybrid dynamics of the legged robot, i.e., the intermittent contacts and the (possible) under-actuation. Moreover, footholds play an essential role in the robot's agility since their coherence with the CoM can improve the stability when the robot reacts to an external disturbance. 3)

Finally, change in the feet trajectories is not considered in this approach, which is crucial to deal with lateral pushes.

A third possibility is to track a fixed goal inside the MPC cost function. This addresses the first issue of the aforementioned PD + MPC architecture; because a constant position is included in the MPC cost, and the conflict is resolved by providing feasible Ground Reaction Forces (GRFs). However, this solution suffers from the fact that it is only able to apply a limited resistive force before the legs lose their control authority (e.g., when the CoM/Zero Moment Point (ZMP) goes out of the support polygon), and therefore is of restricted applicability. Also, in this case, footholds are not generally designed to be consistent with the resulting CoM trajectory since they are usually computed with simple heuristics.

In addition, the MPC typically uses references for GRFs that are computed with the crude heuristic (e.g., dividing the robot weight across the legs on the ground). These values are purely vertical GRF components, often dynamically infeasible for the robot base motion. Accurately tuning the weights for the different cost terms is a tedious task Bouyarmane and Kheddar (2018), and having more physically meaningful references is a preferable solution Bledt and Kim (2019). To address this problem, Bjelonic et al. (2022) use the result of an offline TO as cost terms for their MPC.

The contribution of this dissertation lies in developing a computationally efficient real-time NMPC that runs in closed-loop at a suitable frequency and supports environment adaptation. We utilize a validated SRBD model inside the NMPC and incorporate the mobility cost that allows a legged robot to improve leg mobility for environment adaptation. The performance of the NMPC is witnessed in various simulation and experimental setups while crawling. Lastly, we introduce the ORG that provides physically informed reference trajectory to the NMPC to cope with the disturbances while trotting and pacing.

In this chapter, we have conducted a literature survey on the relevant work related to MPC for legged locomotion, leg mobility inside the MPC, and the need for ORG in two-stage optimization with the MPC. In the following chapters, we will describe this dissertation's contribution (refer to section 1.4) in detail.

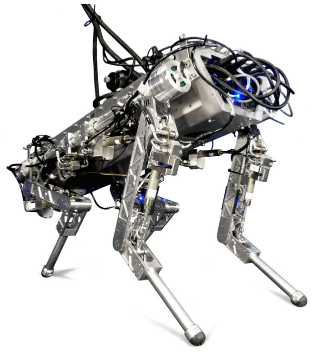
Chapter 3

Legged Robots

This chapter begins by describing the state-of-the-art robots developed by the Dynamic Legged System (DLS) lab of Istituto Italiano Tecnologia (IIT), Genova, Italy and Unitree Robotics that are utilized in this work. This chapter discusses some of the main dynamic models of the legged system. These models are briefly described in this chapter's first part, including their usage inside the optimal control for the legged system. Ultimately, we validate the Single Rigid Body Dynamics (SRBD) model in a closed-loop against the data collected from our robot's high-fidelity simulation environment.

3.1 Robot Description

In this dissertation, we have worked on the HyQ and AlienGo Unitree (2022) robots for simulation and experimentation purposes. HyQ is a fully torque-controlled Hydraulically actuated Quadruped robot developed in the Advanced Robotics lab since 2011. On the other hand, AlienGo is an electric robot developed by Unitree Robotics and launched in 2019. These legged robots are equipped with four legs, each of which has 3-DoFs, namely a hip joint for Hip Adduction-Abduction (HAA), a hip joint for Hip Flexion-Extension (HFE), and a knee joint for Knee Flexion-Extension (KFE). We will briefly describe both of these robots in the following.


(a) HyQ

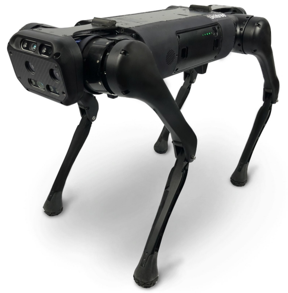

(b) AlienGo

Figure 2: Quadruped robots used in this dissertation

3.1.1 HyQ

HyQ, as shown in Fig. 2a, weighs around $90\,\mathrm{kg}$ and runs with an off-board HPU. It is roughly $1\,\mathrm{m}$ long and $0.5\,\mathrm{m}$ wide, with each leg measuring around $0.78\,\mathrm{m}$ in a fully-extended configuration Semini et al. (2011). This robot carries three perception sensors: an Inertial Measurement Unit (IMU), a stereo camera and a LIDAR. All these sensors, along with leg odometry, are utilized by the state estimator Nobili et al. (2017) to provide the HyQ's body pose and velocity estimates with respect to the world.

Figure 3 shows the anatomy of HyQ's leg, which weighs around $6.5\,\mathrm{kg}$. In HyQ, all four HAA joints are driven by rotatory hydraulic motors and equipped with strain-gauge-based torque sensors. Linear hydraulic cylinders actuate the remaining eight joints of HFE and KFE in the sagittal plane. Torques on these eight joints are computed with load cells by measuring the force applied by the cylinders. All the twelve joints of HyQ are furnished with relative optical and absolute magnetic encoders for position measurements.

Various locomotion capabilities of HyQ with different gaits have been demonstrated in experiments such as crawl Focchi et al. (2020a), trot Barasuol et al. (2013) and bound Orsolino et al. (2017). In this dissertation we mainly emphasize crawl gait that relies on WBC and is supervised by NMPC.

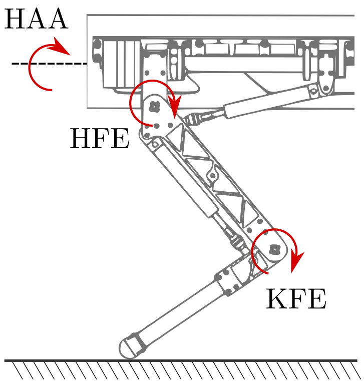


Figure 3: A leg of HyQ robot representing HAA, HFE and KFE joints.

3.1.2 AlienGo

AlieGo Unitree (2022) robot weighs around $22\ \mathrm{kg}$ and has a dimension in stand: length = $0.65\,\mathrm{m}$, width $0.31\,\mathrm{m}$, and height = $0.6\,\mathrm{m}$. It is equipped with a 2D LIDAR, a 3D LIDAR, an RGB-D Camera, an IMU, a GPS, and Ultrasonic sensors. The robot, as depicted in Fig. 2b, has twelve joints each actuated by high-torque brushless DC motors. It is powered by $14.4\,\mathrm{V}$, $6\,\mathrm{Ah}$ LiPo battery that provides up to 2 hours of continuous operation. Intel core i7/i5 CPU performs the main motion control, whereas the perception master runs on Nvidia TX2 GPU. This robot can reach up to $1.5\,\mathrm{m/s}$.

3.2 Mathematical Models for Legged Robots

While the simulation environment of a robot desires high-fidelity models, in predictive control, the complexity of a model utilized depends on the accuracy required in a given application. Some models might provide more accurate predictions than others at the cost of significant computation time in the MPC setting. Depending on the requirements and application, one could trade-off between accuracy and computation time while choosing a model for MPC. In what follows, we will introduce some of the most relevant robot models from the literature which have been extensively used either for simulation or predictive control.

Convention

A variable with the left subscript of a variable denotes its frame of reference. For example, ${}_{\mathcal{C}}\boldsymbol{\omega}$ represents the angular velocity of the robot base expressed in the Center of Mass (CoM) frame $\mathcal{C}$. Note that, unless explicitly specified, all the relevant quantities in this dissertation are defined in the inertial frame $\mathcal{W}$. Throughout this dissertation we define $(a, \ldots, b)$ as the column vector stacking any generic column vectors $a, \ldots, b$.

3.2.1 Rigid Body Dynamics

Legged robots are categorized as *floating base* systems in which the base is free to move instead of being fixed in space. Since the base can freely move, it can be connected to a fixed inertial frame through a fictitious 6-Degrees of Freedom (DoF) joint Featherstone (2008). Figure 4 represents a floating base system with the 6-DoF joint at the CoM and connected to the Inertial frame. Therefore, the floating base systems can be modeled similarly to the generalized manipulator equation of motions by considering the kinematic tree with a specific number of joints, i.e., n. Considering $\mathbf{q} = (\mathbf{q}_{\mathrm{b}}, \mathbf{q}_{\mathrm{j}}) \in \mathbb{SE}(3) \times \mathbb{R}^n$, which holds the position and orientation of the base and angular position of each joint, we can represent the Rigid Body Dynamics (RBD) model with,

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{h}(\mathbf{q}, \dot{\mathbf{q}}) = \mathbf{S}^{\top}\boldsymbol{\tau} + \mathbf{J}_{\mathrm{c}}(\mathbf{q})^{\top}\mathbf{F} \tag{3.1}$$

where $\mathbf{M} \in \mathbb{R}^{(6+n)\times(6+n)}$ is the joint space inertia matrix, vector $\mathbf{h} \in \mathbb{R}^{(6+n)}$ incorporates the effect of centrifugal, Coriolis and gravity terms, Selection matrix $\mathbf{S} \in \mathbb{R}^{(6+n)\times n}$ applies torque $\boldsymbol{\tau} \in \mathbb{R}^n$ only to the n actuated joint rows. Finally, the contact Jacobian $\mathbf{J}_{\mathrm{c}}$ maps forces $\mathbf{F} = (\mathbf{f}_1, ..., \mathbf{f}_{n_c}) \in \mathbb{R}^{3n_c}$ at the n_c end-effectors to $6 + n$ dimensional generalized forces.

Equation (3.1) can further be split into 6 unactuated and n actuated rows as

$$\begin{bmatrix} \mathbf{M}_{\mathrm{b}}(\mathbf{q}) & \mathbf{M}_{\mathrm{bj}}(\mathbf{q}) \\ \mathbf{M}_{\mathrm{jb}}(\mathbf{q}) & \mathbf{M}_{\mathrm{j}}(\mathbf{q}) \end{bmatrix} \begin{bmatrix} \ddot{\mathbf{q}}_{\mathrm{b}} \\ \ddot{\mathbf{q}}_{\mathrm{j}} \end{bmatrix} + \begin{bmatrix} \mathbf{h}_{\mathrm{b}}(\mathbf{q}, \dot{\mathbf{q}}) \\ \mathbf{h}_{\mathrm{j}}(\mathbf{q}, \dot{\mathbf{q}}) \end{bmatrix} = \begin{bmatrix} 0_{6\times n} \\ \mathbf{I}_{n\times n} \end{bmatrix} \boldsymbol{\tau} + \begin{bmatrix} \mathbf{J}_{\mathrm{cb}}^{\top}(\mathbf{q}) \\ \mathbf{J}_{\mathrm{cj}}^{\top}(\mathbf{q}) \end{bmatrix} \mathbf{F} \tag{3.2}$$

where $\mathbf{M}_{\mathrm{b}} \in \mathbb{R}^{6\times 6}$ is the composite rigid body inertia matrix related to the unactuated part, and $\mathbf{M}_{\mathrm{j}} \in \mathbb{R}^{n\times n}$ is the joint mass matrix representing the actuated part in the joint space inertia matrix $\mathbf{M}$. The coupling terms between actuated and unactuated are captured by matrices $\mathbf{M}_{\mathrm{bj}} \in \mathbb{R}^{6\times n}$

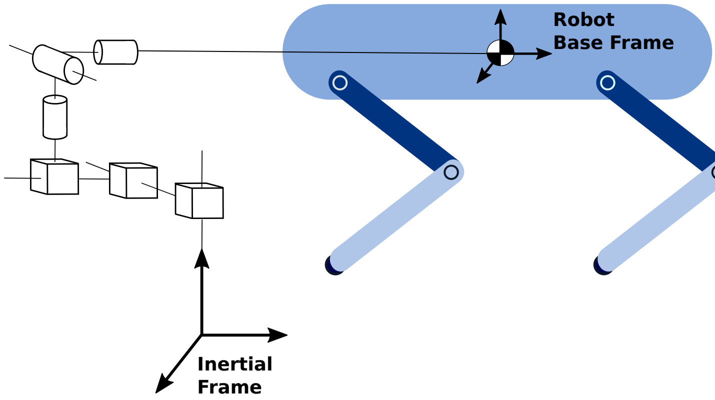


Figure 4: Representation of a quadruped robot as a floating base system. The floating base is connected to the Inertial frame through a fictitious 6-DoFs joint.

and $\mathbf{M}_{\mathrm{jb}} \in \mathbb{R}^{n \times 6}$. Vectors $\mathbf{h}_{\mathrm{b}} \in \mathbb{R}^{6}$ and $\mathbf{h}_{\mathrm{j}} \in \mathbb{R}^{n}$ correspond to the effect of centrifugal, Coriolis and gravity terms for the actuated and unactuated dynamics, respectively. The actuated and unactuated contact Jacobians are denoted by $\mathbf{J}_{\mathrm{cb}}^{\top} \in \mathbb{R}^{3n_c \times 6}$ and $\mathbf{J}_{\mathrm{cj}}^{\top} \in \mathbb{R}^{3n_c \times n}$.

The RBD model captures the dynamics of the whole body of the legged system hence commonly referred to as *whole-body dynamics* or *full-body dynamics* model. Notice that the joint-space inertial matrix $\mathbf{M}$, vector $\mathbf{h}$, and contact Jacobian $\mathbf{J}_{\mathrm{c}}$ are nonlinearly dependent on joint configuration $\mathbf{q}$ and generalized velocity $\dot{\mathbf{q}}$. Furthermore, the coupling between actuated and unactuated dynamics in (3.2) also contributes to the nonlinearity, increasing the complexity of the overall model.

A large number of variables and nonlinear complexity of this model makes it challenging to utilize in predictive control. However, the RBD model has been employed in some applications for Trajectory Optimization (TO) Posa et al. (2016); Neunert et al. (2017) and MPC Koenemann et al. (2015); Neunert et al. (2018); Mastalli et al. (2022) in legged locomotion. Incorporating the RBD model in predictive control strategies often results in a large optimization problem formulation. Therefore, it might impact the real-time implementation on real hardware due to considerable computation time required to solve the resulting optimization problem. Next, we will discuss a modified and simpler version of the RBD model.

3.2.2 Centroidal Dynamics

The CoM is a location where the total mass of a robot is concentrated, and it is of utmost importance while expressing the robot's dynamics. It is a point where a robot's overall linear and angular momentum is defined if the total momentum is conserved. Thus, one of the ways to simplify the RBD model is to consider the dynamics of a robot projected at the CoM. This can be achieved by selecting the first row of (3.2) related to the unactuated dynamics, i.e., CoM dynamics. If the change of momentum is expressed in a frame attached at the CoM of a robot, then the first row of (3.1) can be rewritten as

$$\mathbf{A}(\mathbf{q})\ddot{\mathbf{q}} + \dot{\mathbf{A}}(\mathbf{q})\dot{\mathbf{q}} = \begin{bmatrix} m\mathbf{g} + \sum_{i=1}^{n_c} \mathbf{f}_i \\ \sum_{i=1}^{n_c} \mathbf{p}_{\text{cf}_i} \times \mathbf{f}_i \end{bmatrix} \tag{3.3}$$

where the CMM $\mathbf{A} \in \mathbb{R}^{6\times(6+n)}$ maps the generalized velocities of the robot into its spatial momentum in a common reference frame attached to its CoM Orin et al. (2013). Parameter m represents the mass of a robot, and $\mathbf{g}$ is the gravitational acceleration. The distance between CoM position $\mathbf{p}_\text{c} \in \mathbb{R}^3$ and the position of i^{th} foot $\mathbf{p}_{\text{f},i} \in \mathbb{R}^3$ is denoted by $\mathbf{p}_{\text{cf},i} \in \mathbb{R}^3$. The Ground Reaction Force (GRF) at foot i is given by $\mathbf{f}_i \in \mathbb{R}^3$.

Since this formulation concerns the dynamics of a robot projected at CoM, also known as *centroid*, it is named Centroidal Dynamics (CD). Intuitively, the CD model deals with the change of linear and angular momentum of the robot projected at the CoM under external forces. For this model, we make a point feet assumption for the legged robot, which is reasonable for quadrupeds. Henceforth, we continue with this assumption throughout this dissertation while discussing the simplified models.

We must emphasize that in (3.3), matrix $\mathbf{A}$ depends on the joint state, which is a source of nonlinearity in the robot dynamics. On a positive note, the number of variables decreases in the CD model with respect to the RBD model because the joint torques are absent in this formulation. Also, the number of dynamic equations is reduced from $6 + n$ to 6, with respect to the RBD model. Therefore, from a predictive control perspective, this model is easier to deal with when compared to the RBD model. This model, along with a kinematic model, has been used by Dai et al. (2014) in TO on the Atlas robot. However, dependency on the joint state can still be problematic for predictive control because of extra n joint variables and the nonlinear dependency of matrix $\mathbf{A}$ on them. This joint state dependency can be removed with some assumptions, which we will explain in the next section.

3.2.3 Single Rigid Body Dynamics

A simpler version of the CD is derived by making the following two main assumptions.

Assumption 1. *The influence of the inertia of the legs is neglected. Therefore, they are considered massless, so the contribution of momentum from joint velocity is eliminated.*

Assumption 2. *The full-body inertia of a robot remains equal to the one in a nominal joint position.*

This assumption is valid for robots with the high base-to-leg mass ratio. To name a few, in HyQ Semini et al. (2011), Cassie AgilityRobotics (2017), PADWQ Kim et al. (2021) and ANYMal Hutter et al. (2016), the base majorly contributes to the total mass of the robot. In these robots, their small leg masses do not contribute significantly to the momentum of a robot, even for a quick leg motion. Further, the leg position of these robots has a negligible contribution to the total inertia of the robot. With assumptions 1 and 2 into consideration, the robot dynamics can be expressed with Newton-Euler equations of motion for a single rigid body,

$$m\dot{\mathbf{v}}_{\mathrm{c}} = m\mathbf{g} + \sum_{i=1}^{n_c} \mathbf{f}_i \tag{3.4a}$$

$$\mathbf{I}_{\mathrm{c}}(\boldsymbol{\Phi})\,\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times \mathbf{I}_{\mathrm{c}}(\boldsymbol{\Phi})\boldsymbol{\omega} = \sum_{i=1}^{n_c} \mathbf{p}_{\mathrm{cf}_i} \times \mathbf{f}_i \tag{3.4b}$$

where m is the robot mass, $\dot{\mathbf{v}}_{\mathrm{c}} \in \mathbb{R}^3$ is the CoM acceleration, $\mathbf{I}_{\mathrm{c}}(\boldsymbol{\Phi}) \in \mathbb{R}^{3\times 3}$ is the inertia tensor, $\dot{\boldsymbol{\omega}} \in \mathbb{R}^3$ is the angular acceleration of the robot's base. The robot base orientation is represented by $\boldsymbol{\Phi}$ and its angular velocity by $\boldsymbol{\omega}$. This model deals with the linear (3.4a) and angular dynamics (3.4b) of a robot assuming that it is a single rigid body, thus referred to as a Single Rigid Body Dynamics (SRBD). Although (3.4a) is linear, (3.4b) represents nonlinear angular dynamics, bringing nonlinearity to the overall dynamic model. The derivation of this model, starting from the CD model, has been explained in this thesis Winkler (2018).

This model has been employed by TO Winkler et al. (2018) and MPC Bledt et al. (2017) to jointly optimize for footholds, CoM trajectories and GRFs. To tackle the nonlinearity, Di Carlo et al. (2018) linearized the angular part of the SRBD model demonstrated a variety of quadrupedal gaits in experiments with the MPC setup for moderate terrain inclinations.

3.3 Model Validation

The process of determining if the model accurately represents the system's behavior is called model validation Aumann (2007). Model validation verifies both the conceptual and operational aspects Kerr and Goethel (2014) of the model. Conceptually, it checks if the underlying theory or assumptions are justifiable. Operationally, it checks if the model output agrees with the validation data collected from the system. The model validation is performed by collecting simulated data from the model against the data obtained from the actual system.

Model validation plays an important role in model identification techniques to validate an identified model against the validation data. Similarly, it is vital to validate models derived from physical laws often composed of parameters (for example, mass, inertial, and heat coefficient) along with the physical variables. In real-world examples, these model parameters are either heuristically calculated through experiments or estimated. Therefore, the accuracy of the physics-based model relies on the parameters used within these models. Apart from conceptual and operational verification, model validation also helps to verify the accuracy of the parameters used by the model.

Generally, approximate models are derived from the complex/detailed model of a system. The approximate models are usually less complex and have fewer variables than the complex model. Model validation also checks the accuracy of the approximate model with respect to the detailed model.

The approximations made to derive an approximate model may involve discretizing or linearizing the original continuous-time system or combinations of both of these techniques. One may study the impact of chosen discretization method and sampling time on the accuracy of the model based on the application. In the case of linearization, one could inspect how the system behaves at the linearization points and its accuracy when perturbed from these linearization points. Model linearization is also helpful to access the local stability at an equilibrium point of a nonlinear dynamical system.

For a stable system, validating the model by providing the inputs and cross-checking the output against the validation data is relatively straightforward. The inputs given to the model are exactly the ones provided to the physical system while collecting output validation data. On the other hand, for an unstable system, a closed-loop system is required to carry out the validation process. A closed-loop system means that a stabilizing controller must be added to stabilize the unstable dynamics of the system. This closed-loop controller stabilizes the system while operating at a given nominal condition to avoid the divergence of the state trajectories. The references passed to the closed-loop system specify these nominal operating conditions. A schematic of such a closed-loop system is shown in Fig. 5.

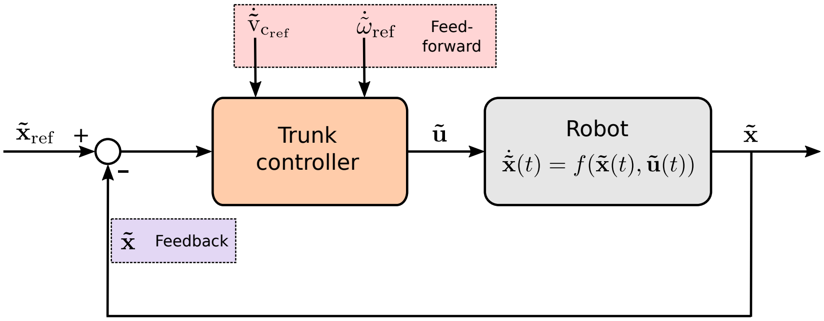
3.3.1 Closed-Loop Model

The nonlinear dynamics governed by any of the RBD, CD, or SRBD models are unstable and require a stabilizing controller for proper functioning at the nominal conditions. One of the popular stabilizing controllers used for nonlinear mechanical systems is Proportional-Derivative (PD). Intuitively, when a mass-spring-damper system is considered, the proportional term of the PD controller is analogous to the spring and the derivative term to the damper. In a PD controller, the higher the proportional gain, the smaller the state error, whereas increasing the derivative gain damps the system response. For legged robots, the PD controller is synthesized in the Cartesian space. Therefore, it aims to control the robot's position and orientation, ensuring locomotion stability. This controller can also be coupled with the feed-forward control and gravity compensation terms. In the context of legged robots, we name the combination of PD, gravity compensation, and feed-forward controller as *trunk controller*. Refer to Fig. 7 for the graphical structure of such a trunk controller in the closed-loop setup.

Consider a closed-loop system depicted in Fig. 5 with a high-fidelity robot model governed by a system of nonlinear equations,

$$\dot{\tilde{\mathbf{x}}}(t) = f(\tilde{\mathbf{x}}(t), \tilde{\mathbf{u}}(t)) \tag{3.5}$$

In our case, this high-fidelity model mathematically represents the dynamics of a real legged robot. Let $\tilde{\mathbf{x}}(t) = (\mathbf{x}_{\text{sel}}(t), \hat{\mathbf{x}}(t)) \in \mathbb{R}^{n_{\text{xf}}}$ and $\tilde{\mathbf{u}}(t) = (\mathbf{u}_{\text{sel}}(t), \hat{\mathbf{u}}(t)) \in \mathbb{R}^{n_{\text{uf}}}$ so that the entire model can be divided into two parts. The selected state $\mathbf{x}_{\text{sel}}(t)$ and control input $\mathbf{u}_{\text{sel}}(t)$ represent part of the model that we would like to approximate with a reduced order model. On the flip side, state $\hat{\mathbf{x}}(t)$ and control input $\hat{\mathbf{u}}(t)$ represent



$\dot{\tilde{\mathrm{v}}}_{\mathrm{c}_{\mathrm{ref}}}$
$\dot{\tilde{\omega}}_{\mathrm{ref}}$
Feed-forward
$\tilde{\mathbf{x}}_{\mathrm{ref}}$
+
-
Trunk controller
$\tilde{\mathbf{u}}$
Robot
$\dot{\tilde{\mathbf{x}}}(t) = f(\tilde{\mathbf{x}}(t), \tilde{\mathbf{u}}(t))$
$\tilde{\mathbf{x}}$
$\tilde{\mathbf{x}}$ Feedback

Figure 5: A closed-loop controller scheme where the state reference $\tilde{\mathbf{x}}_{\mathrm{ref}}$, feed-forward CoM acceleration $\dot{\tilde{\mathbf{v}}}_{\mathrm{c}_{\mathrm{ref}}}$ and feed-forward angular acceleration $\dot{\tilde{\boldsymbol{\omega}}}_{\mathrm{ref}}$ are the exogenous inputs to the closed-loop system. In this scheme a real robot is represented by a high-fidelity model and it is controlled by controller composed of feedback and feed-forward control actions.

the part of the high-fidelity model that will not be considered inside the approximate model.

We also assume that the control law for the separated control inputs can be synthesized with

$$\mathbf{u}_{\mathrm{sel}}(t) = \kappa(\mathbf{x}_{\mathrm{sel}}(t), \mathbf{x}_{\mathrm{sel}}^{\mathrm{ref}}(t)) \tag{3.6a}$$

$$\hat{\mathbf{u}}(t) = \hat{\kappa}(\hat{\mathbf{x}}(t), \hat{\mathbf{x}}^{\mathrm{ref}}(t)) \tag{3.6b}$$

The separated closed-loop control scheme can be visualized in Fig. 6, where two different controllers control the robot.

Further, let us say that the robot dynamics are approximated by another nonlinear system,

$$\dot{\mathbf{x}}(t) = g(\mathbf{x}(t), \mathbf{u}(t)), \tag{3.7}$$

for example, the SRBD model. Where $\mathbf{x}(t) \in \mathbb{R}^{n_{\mathrm{xa}}}$ and $\mathbf{u}(t) \in \mathbb{R}^{n_{\mathrm{ua}}}$ are the state and control inputs of this approximate model. We assume that the state and control input of the approximate model is contained in the state set of the high-fidelity model, i.e., $\mathbf{x}(t) \subseteq \tilde{\mathbf{x}}(t)$ and $\mathbf{u}(t) \subseteq \tilde{\mathbf{u}}(t)$ or they can be represented as a function of the state and input of the high-fidelity model. For validation, we also select the state $\mathbf{x}_{\mathrm{sel}}(t) \in \mathbb{R}^{n_{\mathrm{xa}}}$ and control input $\mathbf{u}_{\mathrm{sel}}(t) \in \mathbb{R}^{n_{\mathrm{ua}}}$ from the high-fidelity model to compare with the state and control input of approximate model during validation.

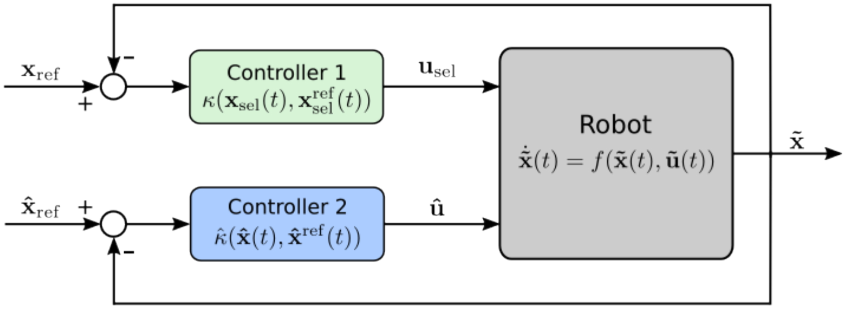

$\mathbf{x}_{\mathrm{ref}}$
+
-
Controller 1
$\kappa(\mathbf{x}_{\mathrm{sel}}(t), \mathbf{x}_{\mathrm{sel}}^{\mathrm{ref}}(t))$
$\mathbf{u}_{\mathrm{sel}}$
Robot
$\dot{\tilde{\mathbf{x}}}(t) = f(\tilde{\mathbf{x}}(t), \tilde{\mathbf{u}}(t))$
$\tilde{\mathbf{x}}$
$\hat{\mathbf{x}}_{\mathrm{ref}}$
+
-
Controller 2
$\hat{\kappa}(\hat{\mathbf{x}}(t), \hat{\mathbf{x}}^{\mathrm{ref}}(t))$
$\hat{\mathbf{u}}$

Figure 6: A closed-loop controller representation of Fig. 5 with two separate controllers. Controller 1 represents the closed-loop controller related to the selected part of the model under validation. On the other hand, Controller 2 represents the closed-loop controller related to the part of the high-fidelity model not considered in the validation.

If the control law for the approximate model is given by $\mathbf{u}(t) = \kappa(\mathbf{x}(t), \mathbf{x}^{\mathrm{ref}}(t))$ similar to the one for the (3.6a), then for an equivalent control law $\kappa(\cdot)$ for both the closed-loop systems, for a given reference $\mathbf{x}^{\mathrm{ref}}(t)$ and for any chosen control law $\hat{\kappa}$ for the excluded dynamics, from the validation we may find,

$$\mathbf{x}(t) \simeq \mathbf{x}_{\mathrm{sel}}(t) \tag{3.8a}$$

$$\mathbf{u}(t) \simeq \mathbf{u}_{\mathrm{sel}}(t) \tag{3.8b}$$

Then, we can say that

$$g(\mathbf{x}(t), \mathbf{u}(t)) \approx f(\tilde{\mathbf{x}}(t), \tilde{\mathbf{u}}(t)) \tag{3.9}$$

Equation (3.9) establishes the equivalence relationship between the approximate model and the real robot or high-fidelity model only if (3.8) holds true. We would like to stress that the approximate model only captures the state dynamics of the selected state $\mathbf{x}_{\mathrm{sel}}(t)$ from the high-fidelity model. Moreover, the selected state analogous to the state vector of the approximate model will be the variables of interest while designing the NMPC in this dissertation. The control law for the excluded state $\hat{\kappa}$ can be synthesized with any of the available techniques, and the validation process described in this section can be performed without a loss of generality.

In the following sections, we will select an approximate model for our validation procedure and describe the architecture of the closed-loop controller.

3.3.2 Approximate Model Selection

One of the elements involved in our model validation process is an approximate model. The approximate model is chosen with a goal in mind to utilize it for optimal planning. This dissertation aims to conduct simulations and experiments involving robot position and orientation dynamics with the optimal planner. Thus, a chosen model should include both linear and angular dynamics. At the same time, we require this model to be less complex with respect to the RBD and CD models. Lastly, the model should use minimum state variables to define linear and angular robot dynamics. Of all the models described in Section 3.2, the SRBD model fulfills our requirements. Therefore, we choose the SRBD model to design the optimal planning algorithms.

The robot dynamics in SRBD model is governed by (3.4), and it can be represented as the continuous-time state-space model:

$$\begin{bmatrix} \dot{\mathbf{p}}_{\mathrm{c}} \\ \dot{\mathbf{v}}_{\mathrm{c}} \\ \dot{\mathbf{\Phi}} \\ \dot{\boldsymbol{\omega}} \end{bmatrix} = \begin{bmatrix} \mathbf{v}_{\mathrm{c}} \\ 1/m \sum_{i=1}^{4} \mathbf{f}_i + \mathbf{g} \\ \mathbf{E}^{-1}(\mathbf{\Phi})\boldsymbol{\omega} \\ -\mathbf{I}_{\mathrm{c}}^{-1}(\mathbf{\Phi})(\boldsymbol{\omega} \times \mathbf{I}_{\mathrm{c}}(\mathbf{\Phi})\boldsymbol{\omega}) + \sum_{i=1}^{4} \mathbf{I}_{\mathrm{c}}^{-1}(\mathbf{\Phi})(\mathbf{p}_{\mathrm{f}_i} - \mathbf{p}_c) \times \mathbf{f}_i \end{bmatrix} \tag{3.10}$$

where the robot CoM velocity is denoted by $\mathbf{v}_{\mathrm{c}}$. The sequence of Z-Y-X Euler angles Diebel (2006) $\mathbf{\Phi} = (\phi,\ \theta,\ \psi)$ i.e., roll (ϕ), pitch (θ) and yaw (ψ) represent the robot base orientation. The *Euler angle rates matrix* $\mathbf{E}(\mathbf{\Phi})$ establishes a relationship between the Euler Angles rates $\dot{\mathbf{\Phi}}$ and angular velocity $\boldsymbol{\omega}$ as discussed in Appendix A. Defining the state and control vectors as $\mathbf{x} = (\mathbf{p}_{\mathrm{c}},\ \mathbf{v}_{\mathrm{c}},\ \mathbf{\Phi},\ \boldsymbol{\omega})$, and $\mathbf{u} = (\mathbf{f}_1, \ldots, \mathbf{f}_4)$, (5.4) can be concisely written as:

$$\dot{\mathbf{x}}(t) = g_{\mathrm{c}}(\mathbf{x}(t), \mathbf{u}(t), \mathbf{p}_{\mathrm{f}}(t)), \tag{3.11}$$

where vector $\mathbf{p}_{\mathrm{f}} = (\mathbf{p}_{\mathrm{f}_1}, \ldots, \mathbf{p}_{\mathrm{f}_4})$ collects the foot positions of all the robot legs and it is a parameter inside the state space model (3.11). The discrete-time representation of this model can be given by,

$$\mathbf{x}_{k+1} = g_{\mathrm{d}}\left(\mathbf{x}_k, \mathbf{u}_k, \mathbf{p}_k\right) \tag{3.12}$$

This discrete-time model can be obtained by exploiting any of the state-of-the-art numerical integrations schemes Quirynen (2017); Butcher (2003); Hairer et al. (1993, 1996).

Linearization and Discretization

Apart from the validation of the SRBD model, we would like to validate the Linear Time-Varying (LTV) representation of this model obtained by linearizing over the state and control input trajectories. The main reason for validating the LTV is that it is used by 1) LTVOpt and SQP-Opt algorithms, 2) the solution method used for NMPC, i.e., RTI scheme uses LTV model obtained by the linearizing on initial guess Diehl et al. (2005b). Hence, it is essential to carry the validation of the LTV model to check the accuracy with respect to the continuous-time SRBD model. The usage of the LTV model in optimal planning is detailed in sections 4.4.1, 4.4.2 and 5.6 for various planner designs.

If we consider the nonlinear dynamical model (3.11) and apply the first-order Taylor series approximation around linearization points $\bar{\mathbf{x}}$ and $\bar{\mathbf{u}}$ with a fixed value of parameter $\bar{\mathbf{p}}_\mathrm{f}$, then we obtain,

$$\begin{aligned}\dot{\mathbf{x}}(t) = \quad & \frac{\partial g(\mathbf{x}(t), \mathbf{u}(t), \bar{\mathbf{p}}_\mathrm{f})}{\partial \mathbf{x}(t)}\bigg|_{\mathbf{x}=\bar{\mathbf{x}},\mathbf{u}=\bar{\mathbf{u}}} (\mathbf{x}(t) - \bar{\mathbf{x}}) \\ & + \frac{\partial g(\mathbf{x}(t), \mathbf{u}(t), \bar{\mathbf{p}}_\mathrm{f})}{\partial \mathbf{u}(t)}\bigg|_{\mathbf{x}=\bar{\mathbf{x}},\mathbf{u}=\bar{\mathbf{u}}} (\mathbf{u}(t) - \bar{\mathbf{u}}) \\ & + g(\bar{\mathbf{x}}, \bar{\mathbf{u}}, \bar{\mathbf{p}}_\mathrm{f})\end{aligned} \tag{3.13}$$

Defining the Jacobian matrix resulting from the partial derivative with respect to $\mathbf{x}$ and $\mathbf{u}$ as $\mathbf{A}_\mathrm{c}$ and $\mathbf{B}_\mathrm{c}$ we have,

$$\dot{\mathbf{x}}(t) = \mathbf{A}_\mathrm{c}(\mathbf{x}(t) - \bar{\mathbf{x}}) + \mathbf{B}_\mathrm{c}(\mathbf{u}(t) - \bar{\mathbf{u}}) + g(\bar{\mathbf{x}}, \bar{\mathbf{u}}, \bar{\mathbf{p}}_\mathrm{f}) \tag{3.14}$$

Rearranging the terms leads to

$$\dot{\mathbf{x}}(t) = \mathbf{A}_\mathrm{c}\mathbf{x}(t) + \mathbf{B}_\mathrm{c}\mathbf{u}(t) + g(\bar{\mathbf{x}}, \bar{\mathbf{u}}, \bar{\mathbf{p}}_\mathrm{f}) - \mathbf{A}_\mathrm{c}\bar{\mathbf{x}} - \mathbf{B}_\mathrm{c}\bar{\mathbf{u}} \tag{3.15}$$

Setting $\mathbf{r}_\mathrm{c} = g(\bar{\mathbf{x}}, \bar{\mathbf{u}}, \bar{\mathbf{p}}_\mathrm{f}) - \mathbf{A}_\mathrm{c}\bar{\mathbf{x}} - \mathbf{B}_\mathrm{c}\bar{\mathbf{u}}$ and after discretization by the explicit Euler with sampling time T_s gives,

$$\begin{aligned}\mathbf{x}_{k+1} &= \mathbf{x}_k + T_\mathrm{s}\left[\mathbf{A}_c x(t) + \mathbf{B}_c\mathbf{u}(t) + \mathbf{r}_\mathrm{c}\right] \\ &= (\mathbf{I} + T_\mathrm{s}\,\mathbf{A}_c)\mathbf{x}_k + (T_\mathrm{s}\mathbf{B}_c)\mathbf{u}_k + T_\mathrm{s}\,\mathbf{r}_\mathrm{c} \\ &= \mathbf{A}\mathbf{x}_k + \mathbf{B}\mathbf{u}_k + \mathbf{r}\end{aligned} \tag{3.16}$$

where, $\mathbf{A} = \mathbf{I} + T_\mathrm{s}\,\mathbf{A}_c$, $\mathbf{B} = T_\mathrm{s}\mathbf{B}_c$ and $\mathbf{r} = T_\mathrm{s}\,\mathbf{r}_\mathrm{c}$. Equation (3.16) is a linearized model of the nonlinear system (3.11) at the linearization points $\bar{\mathbf{x}}$ and $\bar{\mathbf{u}}$. This model can be a good approximation when the system operates around these linearization points. However, if the linearization points are time-varying, this model can be generalized over the linearization trajectory $\mathbf{x}_k^\mathrm{L} \in \mathbb{R}^{(n_\mathrm{x} \times N_\ell)}$ and $\mathbf{u}_k^\mathrm{L} \in \mathbb{R}^{(n_\mathrm{u} \times N_\ell)}$,

$$\mathbf{x}_{k+1} = \mathbf{A}_k\mathbf{x}_k + \mathbf{B}_k\mathbf{u}_k + \mathbf{r}_k \tag{3.17}$$

where N_ℓ is a length of linearization trajectory. Equation (3.17) is a discrete-time representation of the LTV system.

3.3.3 Trunk Controller

The objective of the trunk controller is to compensate for the tracking error on the robot state $\mathbf{x}$ while stabilizing the robot at nominal conditions specified through the state reference $\mathbf{x}_\mathrm{ref}$. The output of the trunk controller is a *wrench* [1]composed of feed-forward wrench $\mathbf{W}_\mathrm{ff}$, feedback wrench $\mathbf{W}_\mathrm{fb}$ and the wrench resulting from the gravity compensation $\mathbf{W}_\mathbf{g} = (m\mathbf{g}, 0_{3\times 1})$. This wrench is later mapped into the GRFs before sending to the robot. Figure 7 shows the schematic of the elements of the trunk controller in the closed-loop.

$$\mathbf{W}_\mathrm{t} = \mathbf{W}_\mathrm{ff} + \mathbf{W}_\mathrm{fb} - \mathbf{W}_\mathrm{g} \tag{3.18}$$

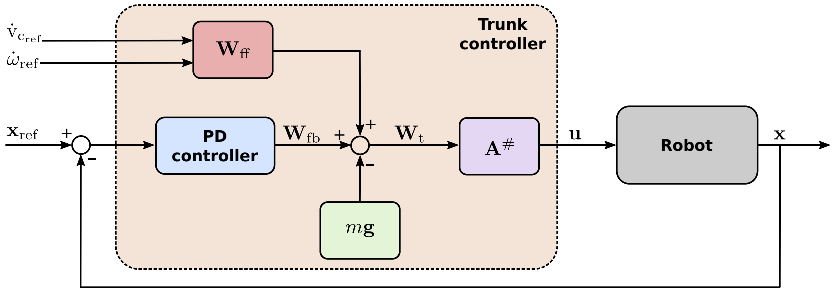


Figure 7: Elements of the trunk controller in the closed-loop

[1]A wrench is a vector of forces and torque that arise by applying Newton's laws on the rigid body.

Feedforward Controller

The feed-forward wrench is calculated assuming that the Coriolis and centrifugal forces are negligible, which is valid for the slow motions of a robot.

$$\mathbf{W}_{\mathrm{ff}} = \begin{bmatrix} m\dot{\mathbf{v}}_{\mathrm{c_{ref}}} \\ \mathbf{I}_{\mathrm{c}}\dot{\boldsymbol{\omega}}_{\mathrm{ref}} \end{bmatrix} \tag{3.19}$$

where $\dot{\mathbf{v}}_{\mathrm{c_{ref}}}$ and $\dot{\omega}_{\mathrm{ref}}$ are references for the CoM acceleration and angular acceleration of a robot.

Feedback Controller

To define the desired feedback wrench obtained from a Cartesian impedance, we use the approach of Focchi et al. (2016) given by:

$$\mathbf{W}_{\mathrm{fb}} = \mathbf{K}\begin{bmatrix} \mathbf{p}_{\mathrm{c_{ref}}} - \mathbf{p}_{\mathrm{c}} \\ \mathbf{e}({}_{\mathrm{w}}\mathbf{R}_{\mathrm{b}}^{\top}\,{}_{\mathrm{w}}\mathbf{R}_{\mathrm{ref}}) \end{bmatrix} + \mathbf{D}\begin{bmatrix} \mathbf{v}_{\mathrm{c_{ref}}} - \mathbf{v}_{\mathrm{c}} \\ \boldsymbol{\omega}_{\mathrm{ref}} - \boldsymbol{\omega} \end{bmatrix} \tag{3.20}$$

where ${}_{\mathrm{w}}\mathbf{R}_{\mathrm{b}}$ and ${}_{\mathrm{w}}\mathbf{R}_{\mathrm{ref}} \in \mathbb{R}^{3\times3}$ are the rotation matrices representing the actual and reference orientation of the base with respect to the inertial frame, respectively. Function $\mathbf{e}(\cdot) : \mathbb{R}^{3\times3} \to \mathbb{R}^{3}$ is a mapping from a rotation matrix to the associated rotation vector. References for the CoM position, CoM velocity, and angular velocity of a robot are represented with $\mathbf{p}_{\mathrm{c_{ref}}}$, $\mathbf{v}_{\mathrm{c_{ref}}}$, and $\boldsymbol{\omega}_{\mathrm{ref}}$, respectively. Diagonal matrices $\mathbf{K}$ and $\mathbf{D}$ contain the proportional and derivative gains and carry the meaning of impedances.

Ground Reaction Forces

The total wrench $\mathbf{W}_{\mathrm{t}}$ (3.18) computed by the trunk controller must be mapped into the GRFs before sending it to the robot. Inspired by Focchi et al. (2016), we set the right hand-side of (3.4) equal to the total wrench,

$$\underbrace{\begin{bmatrix} \mathbf{I} & \dots & \mathbf{I} \\ [\mathbf{p}_{\mathrm{cf},1}\times] & \dots & [\mathbf{p}_{\mathrm{cf},4}\times] \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} \mathbf{f}_1 \\ \vdots \\ \mathbf{f}_4 \end{bmatrix}}_{\mathbf{u}} = \mathbf{W}_{\mathrm{t}} \tag{3.21}$$

Therefore, the GRFs can be obtained by the following transformation,

$$\mathbf{u} = \mathbf{A}_{\mathrm{s}}^{\#}\mathbf{W}_{\mathrm{t}} \tag{3.22}$$

where $\mathbf{A}_s^{\#}$ is a Moore-Penrosei pseudo inverse of matrix $\mathbf{A}_s = \mathbf{AS} \in \mathbb{R}^{6\times12}$. Selection matrix $\mathbf{S} \in \mathbb{R}^{12\times12}$ is diagonal and picks only the GRFs generated from the stance legs during this computation. In other words, the GRFs corresponding to the swinging legs are set to zero.

Remark: This closed-loop validation procedure applies to any other controller different from the trunk controller mentioned in this section. For example, a popular alternative is the Whole-Body Control (WBC), which solves the optimization problem and provides the instantaneous control input to the robot. Refer Section 5.5.1 for more details on WBC.

3.4 Model Validation Results

The SRBD model incorporates both the linear and angular dynamics of the robot. We are particularly interested in validating the nonlinear dynamics involved in this model. Because the objective is to use this model inside an optimal planner and perform locomotion that involves significant variations in pitch and yaw orientations while crawling. Further, we validate the linear dynamics given by the SRBD model without any additional changes in the validation process. For validation, we collect the data from the HyQ simulator while crawling at varying forward velocity. The robot crawls with a large body pitch or yaw variation during this locomotion. The user commands the robot with a joystick to crawl forward while changing the body pitch or yaw. In the following sections, we will describe the validation setup and results.

3.4.1 Validation Setup

The locomotion software framework for HyQ is developed in Robot Operating System (ROS) OpenRobotics (2022). We use Gazebo Koenig and Howard (2004); OSRF (2014) simulator for our locomotion framework that provides a robust physics engine, high-quality graphics, and convenient programmatic and graphical interfaces. A robot-specific library called RobCoGen Frigerio et al. (2016) that generates optimized code for kinematics and dynamics routines commonly encountered in robotics is utilized for the kinematic and dynamic engines of the HyQ robot.

The validation of the computer simulation of HyQ developed in Gazebo with respect to the real data obtained during locomotion is reported in Frigerio et al. (2017). Therefore, we rely on the high-fidelity simulator of HyQ to obtain the validation data for this work. The validation data are collected from the closed-loop system discussed in section 3.3.1 and depicted in Fig. 5.

To validate the approximate model, i.e., the SRBD model, we use the trunk controller (refer to section 3.3.3) in closed-loop with the SRBD model. The same PD gains are used while collecting the validation data from the HyQ simulator and for the PD controller inside the closed-loop system with the SRBD model under validation. The values for these gains are $\mathbf{K} = \mathrm{diag}(1500, 1500, 1500, 300, 300, 300)$ and $\mathbf{D} = \mathrm{diag}(1500, 1500, 1500, 200, 200, 200)$.

We validate three different versions of the SRBD model: 1) continuous-time nonlinear model (3.10), 2) discrete-time nonlinear model (3.12), and 3) discrete-time LTV model (3.17). To simulate the continuous-time nonlinear model, we choose the variable order `ode15s` MATLAB (2022a) integrator. On the other hand, for both the discrete-time nonlinear and discrete-time LTV model, we select the explicit Euler integrator to discretize with a sampling time of $40\,\mathrm{ms}$. We want to stress that the choice of sampling time is based on the frequency at which we aim to run our optimal planner online, i.e., $\approx 25\,\mathrm{Hz}$. Furthermore, the validation results obtained utilizing this sampling time are acceptable for the simulations and experiments performed in this dissertation.

Linearization Trajectory for LTV Model

Obtaining the LTV form of the SRBD requires linearization trajectories for the robot's state and control inputs. In our case, these trajectories are not known a priori. Therefore, we choose the initial condition of the state $\mathbf{x}_0$, compute the control inputs $\mathbf{u}_0$ as a function of the $\mathbf{x}_0$ and $\mathbf{x}_0^{\mathrm{ref}}$ with the trunk controller, as explained in section 3.3.3. We utilize $\mathbf{x}_0$ and computed $\mathbf{u}_0$ to linearize the SRBD model at time instance $k = 0$ as detailed in section 3.3.2. Then, we simulate this linear model to predict the state $\mathbf{x}_1$ starting from initial condition $\mathbf{x}_0$ and applying control input $\mathbf{u}_0$. Again, using state $\mathbf{x}_1$, we compute $\mathbf{u}_1$ with the trunk controller. Then, we obtain a linearized model of the SRBD using $\mathbf{x}_1$ and $\mathbf{u}_1$, and the process repeats for the duration of the simulation. With this method, we obtain a linearized model of the system based on the available state feedback and computed control inputs using this state feedback. The obtained model is therefore time-varying over the duration of the simulation.

3.4.2 Results

In this section, we show the validation results of the varying pitch with the forward walk of the HyQ robot. A similar validation is performed for varying yaw while crawling, and the results are reported in Appendix B.

The validation related to angular dynamics are demonstrated in Fig. 8 and 9. The pitch angle θ shows very good fit to the validation data for all the models. For roll ϕ and yaw ψ angles, the PD controller tracks small references (in the order of 10^{-6}); therefore, the difference between the validation data and model simulation is observed. It is worth noticing that the variation from the validation data about these references is due to the model approximation, and it is acceptable for our work. A good tracking of the angular velocities specifically for ω_y can be seen in Fig. 9.

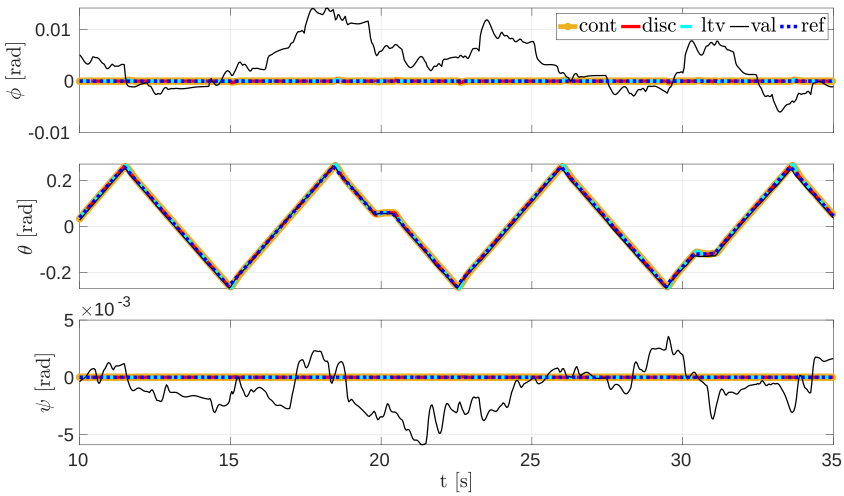


Figure 8: Body orientation from the pitch validation results. Yellow, red, and cyan lines represent the continuous-time nonlinear, discrete-time nonlinear, and discrete-time LTV models, respectively. Black and dotted blue lines correspond to the validation data and references to the PD controller

Figure 10 shows the plots of the CoM position where all three models closely fit the X-Y positions and show a similar trend for the Z position with respect to the validation data. The discontinuity in the X-Y position for the references is due to heuristic planning Focchi et al. (2020a)

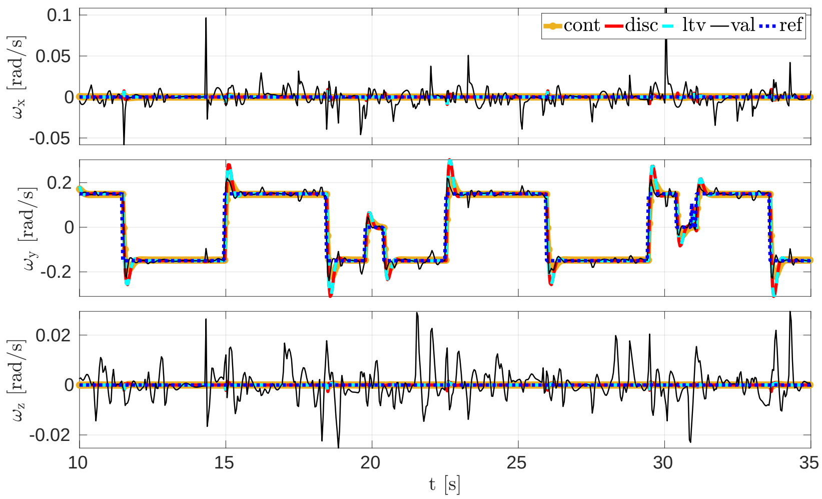


Figure 9: Body angular velocity from the pitch validation results. Yellow, red, and cyan lines represent the continuous-time nonlinear, discrete-time nonlinear, and discrete-time LTV models, respectively. Black and dotted blue lines correspond to the validation data and references to the PD controller

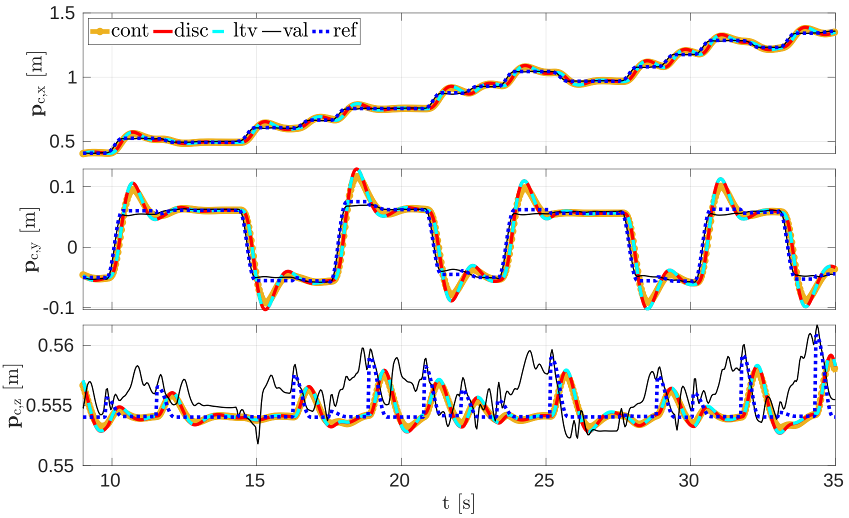


Figure 10: CoM position from the pitch validation results. Yellow, red, and cyan lines represent the continuous-time nonlinear, discrete-time nonlinear, and discrete-time LTV models, respectively. Black and dotted blue lines correspond to the validation data and references to the PD controller.

developed for crawl gait. The validation results for CoM velocities are shown in Fig. 11, where all the models follow the trends of validation data.

While performing a closed-loop validation of the SRBD model, the PD controller, in addition to tracking the reference trajectories, also compensates for the model discrepancy. Hence it is not sufficient to validate only the state of this model but also the control inputs. As explained in section 3.3.1, the approximate model's control inputs and states must match the validation data with a certain accuracy for successful validation.

The validation result of GRFs are reported in Fig. 12, 13, and 14. The X, Y, and Z components of the GRFs follow the validation data for all the models. We would like to stress that the adjustments made in GRFs by the PD controller in closed-loop with these three versions of the SRBD model to track the state references. Further, these three models show similar validation results even after the additional approximations introduced by discretization to obtain the discrete-time model and approxim-

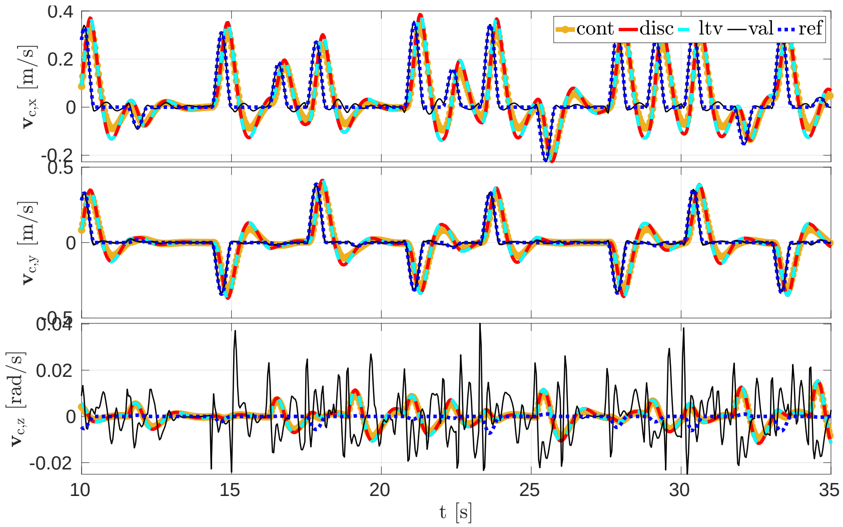


Figure 11: CoM velocity from the pitch validation results. Yellow, red, and cyan lines represent the continuous-time nonlinear, discrete-time nonlinear, and discrete-time LTV models, respectively. Black and dotted blue lines correspond to the validation data and references to the PD controller

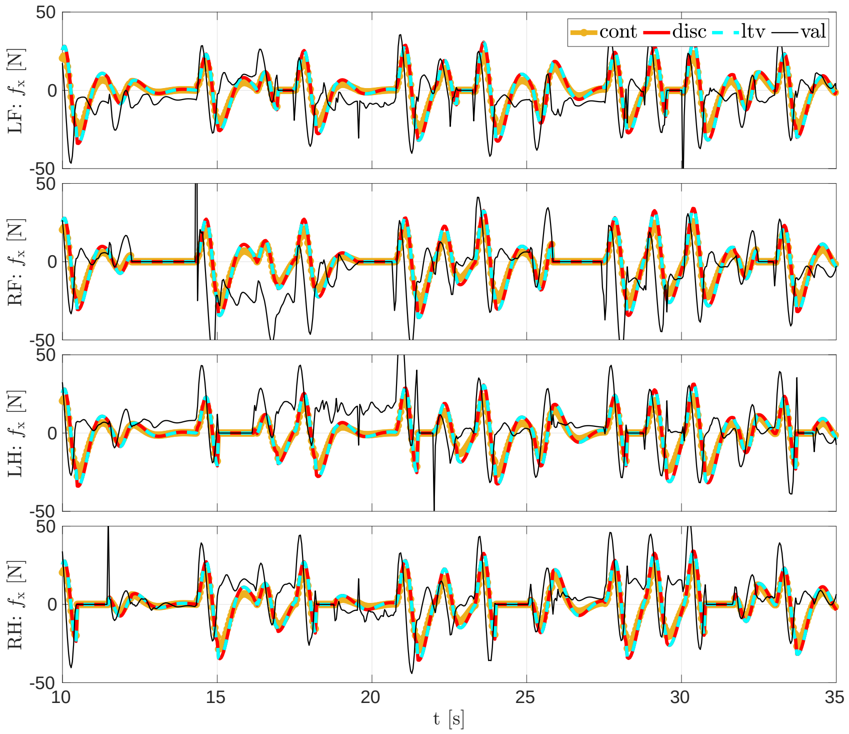


Figure 12: The X component plots of GRFs at respective legs from the pitch validation results. Yellow, red, and cyan lines represent the continuous-time nonlinear, discrete-time nonlinear, and discrete-time LTV models, respectively. The black line corresponds to the validation data.

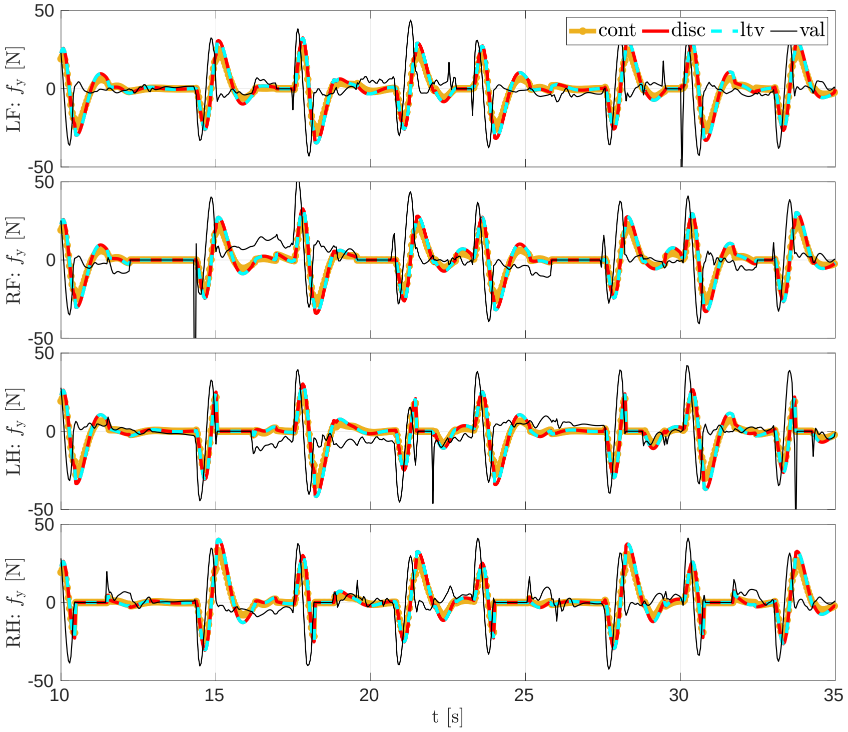


Figure 13: The Y component plots of GRFs at respective legs from the pitch validation results. Yellow, red, and cyan lines represent the continuous-time nonlinear, discrete-time nonlinear, and discrete-time LTV models, respectively. The black corresponds to the validation data.

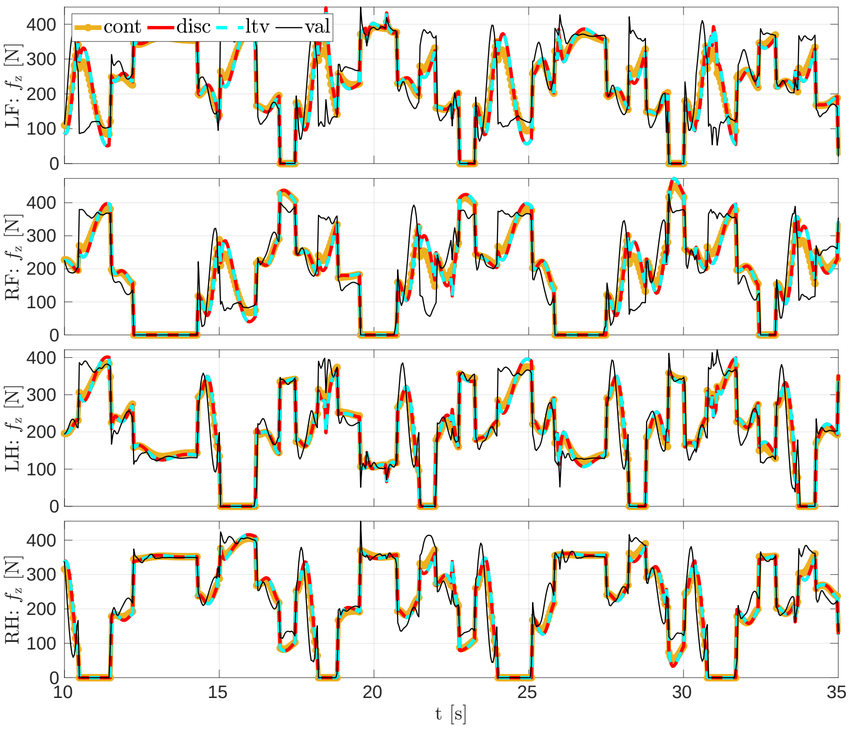


Figure 14: The Z component plots of GRFs at respective legs from the pitch validation results. Yellow, red, and cyan lines represent the continuous-time nonlinear, discrete-time nonlinear, and discrete-time LTV models, respectively. The black line corresponds to the validation data.

ation by discretization and linearization to obtain the LTV model. Note that there is no reference for GRFs in these plots since the trunk controller computes them without a need for GRFs references (refer to section 3.3.3).

Since this model validation is performed on the SRBD model that will be used inside the NMPC formulation, it is acceptable if the model approximately replicates the behavior of the validation data. This means that the SRBD model is only an approximation of the real robot and the control decision made by NMPC will be based on this approximation. The output of the NMPC is usually provided to the downstream controller that runs at a higher frequency than NMPC and takes care of the faster dynamics as well as compensates for the model discrepancy. Therefore, providing approximate but feasible state and input references to the downstream controller is sufficient. We will discuss the structure of the locomotion framework with NMPC and downstream controller in Chapter 5.

3.5 Summary

In this chapter, we have introduced the HyQ and AlienGo robots, discussed various mathematical models for legged robots and presented the model validation results of the SRBD model for the HyQ robot. In the next chapter, we will use this validated LTV model to develop two optimal planners.

Chapter 4

Trajectory Optimization for Legged Locomotion

An exhaustive list of Trajectory Optimization (TO) applied in robotics is documented in Almasri and Uyguroğlu (2021). Since TO allows formulating a particular robot task using optimization problem formulation, it has provided remarkable results in legged locomotion Posa et al. (2016); Neunert et al. (2017); Aceituno-Cabezas et al. (2018); Melon et al. (2020). However, mitigating tracking errors while following the optimized trajectories from TO can be challenging if online re-planning occurs at a low frequency. Therefore, the current robot state used by TO is updated less frequently. This problem can be addressed using Receding Horizon Principle (RHP) principle in the Model Predictive Control (MPC) setting, where the current robot state is fed back to the optimization problem each time the MPC algorithm is called. This solution allows incorporating the latest information on the robot state in MPC, thus mitigating tracking errors by computing the optimized trajectories with the current robot state.

In the following sections, we will briefly introduce MPC and discuss the Optimal Control Problem (OCP) formulation that arises when solving an MPC or a TO problem. Then, we present two TO-based planners using the LTV model presented in the previous chapter. These TO-based planners will form the basis of the NMPC formulation presented in the next chapter.

4.1 Model Predictive Control

MPC has been famous in the process industry over the past few decades. Moreover, some of its first usage dates back to the 1980's Qin and Badgwell (2003); Bauer and Craig (2008). The advancement in the computational power of the new-age computing devices has allowed MPC to break through in the realm of embedded applications. Since the past decade, MPC has gained immense popularity in legged locomotion and has become a preferred technology for research as well as commercial applications in this domain Neunert et al. (2018); Koenemann et al. (2015); Di Carlo et al. (2018); Bledt et al. (2017). The prominent characteristics of MPC listed below make it a compelling and suitable control strategy in various applications.

- Different control goals can be expressed in terms of the cost function inside an optimization problem
- It allows one to embed state and control input constraints directly
- Incorporates system model while evaluating the optimal control law

Particularly, MPC involves an optimization problem formulation that is composed of several ingredients listed as follows:

- Mathematical model of the system
- Constraints on the state and control inputs
- Cost function
- Optimization algorithm to solve an OCP that uses the preceding three ingredients
- The Receding Horizon Principle (RHP) principle: Given the system state $\mathbf{x}_k$ at any sampling instance k, solve an optimization problem to find the future control input sequence $\mathbf{u} = (\mathbf{u}_k, \ldots, \mathbf{u}_{k+N-1})$ and apply only the first control input $\mathbf{u}_k$ to the system. At the next time instance $k+1$, a new optimization problem is solved by reading the state of the system at $k+1$, along the prediction horizon $[k+1, k+N]$ and the process repeats. In short, MPC involves repetitively solving an OCP at each sampling instance in the RHP fashion. Figure 15 depicts the RHP principle.

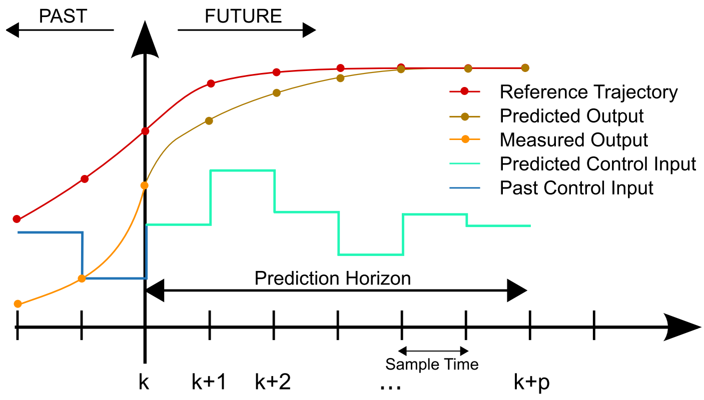


Figure 15: Illustration of the RHP utilized in the MPC scheme (Source: Behrendt (2009)).

Numerical optimization methods sit at the core of solving an MPC problem, and choosing the suitable solution algorithm can drastically affect the accuracy, reliability, and performance of the MPC solution. The solution algorithm, generally called the optimization algorithm, solves an OCP derived from an MPC formulation. These optimization algorithms rely on numerical methods Rawlings et al. (2019). The choice of these numerical methods for MPC is typically based on the trade-off between the accuracy and computation time. Some methods can find the solution of the OCP with the same accuracy as others with significantly lower computation time. Moreover, there is a reliability aspect, i.e., some methods succeed finding approximate solution of the OCP while others might fail. Since the optimization problem formulation arising from MPC falls under the paradigm of OCP, in the next section, we will introduce a generic formulation of continuous-time OCP. Further, we proceed to define the discrete-time version of the OCP that is of prime interest to this work.

4.2 Optimal Control Problem

Often, the system dynamics are available in continuous-time ODE, similar to (3.11). In this scenario, the state $\mathbf{x}(t)$ and control inputs $\mathbf{u}(t)$ are defined in continuous-time. To take care of these dynamics in MPC or TO setting, a solution to a continuous-time OCP might be required[1]. Consider the continuous-time trajectories of state $\mathbf{x}^{\mathrm{c}}(.)$ and $\mathbf{u}^{\mathrm{c}}(.)$ are defined in the time interval of interest $t \in [0, 0+T]$, where T is a prediction horizon. Then, a generic form of such an OCP formulation can be given by

$$\min_{\mathbf{x}^{\mathrm{c}}(.),\mathbf{u}^{\mathrm{c}}(.)} \quad \int_0^T \ell_{\mathrm{c}}\left(\mathbf{x}(t), \mathbf{u}(t)\right) + \ell_{c_{\mathrm{T}}}\left(\mathbf{x}(T)\right) \tag{4.1a}$$

$$\text{s.t.} \quad \mathbf{x}(0) = \mathbf{x}_0, \tag{4.1b}$$

$$\mathbf{x}(t) = f_c\left(\mathbf{x}(t), \mathbf{u}(t)\right), \qquad t \in [0, T], \tag{4.1c}$$

$$h_c\left(\mathbf{x}(t), \mathbf{u}(t)\right) \leq 0, \qquad t \in [0, T], \tag{4.1d}$$

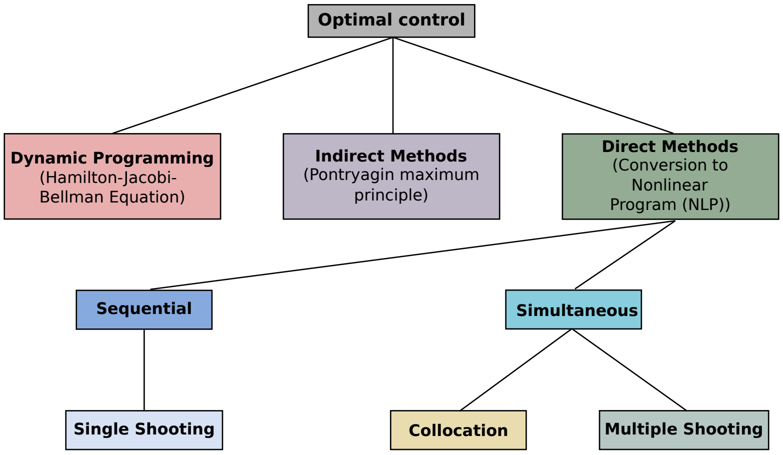
$$\mathbf{x}(T) \in \mathbb{X}_{\mathrm{T}} \tag{4.1e}$$

where ℓ_{c} and $\ell_{c_{\mathrm{T}}}$ are the stage and terminal costs, respectively. Equation (4.1b) uniquely defines the value for the initial state $\mathbf{x}(0)$ that marks the starting point of the state trajectory $\mathbf{x}^{\mathrm{c}}(.)$ in the solution of the OCP (4.1). Equation (4.1c) introduces the continuous-time system dynamics in the OCP, and (4.1d) represents the algebraic path constraints on the state and control input. Finally, (4.1e) enforces constraint on the terminal state $\mathbf{x}(T)$ to be contained in a chosen terminal state $\mathbb{X}_{\mathrm{T}}$.

In continuous-time OCP, time t runs through infinitely many points in the interval $t \in [0, T]$. Hence, continuous-time OCP is an infinite-dimensional optimization problem with infinite-dimensional optimization variables and constraints.

The OCP (4.1) problem can be solved with several existing numerical methods, and their pictorial classification is laid out in Fig. 16. A common factor these methods share is that, at some point, the infinite-dimensional problem requires discretization. The first category of these methods are based on the Hamilton-Jacobi-Bellman (HJB) equation called Dynamic Programming Bertsekas (2005), which suffers from the curse of dimensionality and is thus limited to small scale applications. The second category of methods is based on Pontryagin maximum principle Gamkrelidze (1999) and is known as *Indirect methods*. They are characterized by first optimize and then discretize approach. The third category of methods is called *Direct methods*, where the optimization problem is first discretized and then optimized. The indirect methods

[1] This is depends on the synthesis method chosen for an MPC or a TO problem formulation. One might opt for the discrete-time OCP to solve an MPC or a TO problem which will be explained in the following sections of this chapter.


Optimal control
Dynamic Programming (Hamilton-Jacobi-Bellman Equation)
Indirect Methods (Pontryagin maximum principle)
Direct Methods (Conversion to Nonlinear Program (NLP))
Sequential
Simultaneous
Single Shooting
Collocation
Multiple Shooting
Figure 16: Classification chart of an OCP (Inspired by Diehl et al. (2005a)).

provide accurate, reliable and fast solutions, but they are challenging to initialize Wensing et al. (2022) and do not allow one to incorporate state inequality constraints directly. On the other hand, direct methods are widely used in the robotics community for motion generation Wensing et al. (2022) and, in general, for MPC Rawlings et al. (2019) applications. These methods allow one to incorporate the state inequality constraints directly. A solution method called RTI Diehl et al. (2005b,a) utilized in this dissertation for the NMPC falls in the third category. Therefore, we will briefly discuss direct methods in what follows.

4.3 Discrete-time Optimal Control Problem

The conversion of continuous-time OCP to discrete-time leads to what is commonly known as NLP Rawlings et al. (2019). We start with the continuous-time OCP (4.1) to express the NLP formulation. First, the continuous time t is divided into N points of equal length h. The continuous time cost (4.1a) is replaced with the finite sum of the cost evaluated at each sampling instance k. Defining the predicted state and control input with $\mathbf{x}^{\mathrm{s}} := \{\mathbf{x}_0, \dots, \mathbf{x}_N\}$ and $\mathbf{u}^{\mathrm{s}} := \{\mathbf{u}_0, \dots, \mathbf{u}_{N-1}\}$, respectively, an

NLP can be expressed as

$$
\begin{aligned}
\min_{\mathbf{x}^{\mathrm{s}},\mathbf{u}^{\mathrm{s}}} \quad & \sum_{k=0}^{N-1} \ell\left(\mathbf{x}_k, \mathbf{u}_k\right) + \ell_{\mathrm{T}}\left(\mathbf{x}_N\right) && && \text{(4.2a)}\\
\text{s.t.} \quad & \mathbf{x}_0 = \hat{\mathbf{x}}_0, && && \text{(4.2b)}\\
& \mathbf{x}_{k+1} = f_{\mathrm{d}}\left(\mathbf{x}_k, \mathbf{u}_k\right), && k \in \mathbb{I}_0^{N-1}, && \text{(4.2c)}\\
& h\left(\mathbf{x}_k, \mathbf{u}_k\right) \leq 0, && k \in \mathbb{I}_0^{N-1}, && \text{(4.2d)}\\
& \mathbf{x}_{\mathrm{N}} \in \mathbb{X}_{\mathrm{T}} && && \text{(4.2e)}
\end{aligned}
$$

Similar to continuous-time OCP, the initial condition is introduced by (4.2b). The system dynamics (4.2c) is presented in discrete-time that can be obtained through any state-of-the-art integration schemes, for example, it could be a simple explicit Euler method. Again, the path constraints on the decision variables are defined with (4.2d) only at the sampling instance k. Ultimately, the terminal constraint is introduced with (4.2e).

The are multiple ways to convert the (4.2) problem into NLP. First, based on how the optimization algorithm handles the simulation and optimization problems, we will categorize them into sequential and simultaneous optimal control. Then, we will go through their respective subcategories.

4.3.1 Sequential Optimal Control

In sequential optimal control, first, a forward simulation determines $\mathbf{x}^{\mathrm{s}}$ for a given initial condition $\mathbf{x}_0$ and initial values of control input trajectory $\mathbf{u}^{\mathrm{s}}$. This can be achieved by replacing all the values $\mathbf{x}_k \,\forall\, k \in (0, N]$ as a function of the initial state and the corresponding control input $\mathbf{u}_k$. For instance,

$$
\begin{aligned}
\mathbf{x}_1 &= f_{\mathrm{d}}(\mathbf{x}_0, \mathbf{u}_0), && \text{(4.3a)}\\
\mathbf{x}_2 &= f_{\mathrm{d}}(f_{\mathrm{d}}(\mathbf{x}_0, \mathbf{u}_0), \mathbf{u}_1), && \text{(4.3b)}\\
&\vdots && \text{(4.3c)}\\
\mathbf{x}_N &= f_{\mathrm{d}}(f_{\mathrm{d}}(\mathbf{x}_0, \mathbf{u}_0), \ldots, f(\mathbf{x}_0, \mathbf{u}_{N-2}), \mathbf{u}_{N-1}). && \text{(4.3d)}
\end{aligned}
$$

Second, the control input $\mathbf{u}_{\mathrm{s}}$ is updated in each iteration to move closer to an optimal solution. Due to its sequential nature of first performing a forward simulation followed by finding optimal control inputs, it is called *sequential optimal control*.

A famously known *Single shooting method* falls under this category of optimal control. This approach exploits a reduced number of decision variables because the state trajectory is completely removed as a function of the initial state and control input. Further, this approach results in a dense structure of hessian and gradients, and it can be preferable for a stable system and with an optimization solver that can not exploit the sparsity Rawlings et al. (2019). This method demands the initialization of the control input trajectory $\mathbf{u}^{\mathrm{s}}$. Moreover, in this approach, the initial guess for state trajectory $\mathbf{x}^{\mathrm{s}}$ can not be provided to the optimization solver.

4.3.2 Simultaneous Optimal Control

The NLP (4.2) considers the state $\mathbf{x}^{\mathrm{s}}$ and control input $\mathbf{u}^{\mathrm{s}}$ trajectories as decision variables. One could directly feed this problem to a suitable optimization algorithm to determine the solution. While finding the solution, if (4.2b) and (4.2c) are respected, then $\mathbf{x}^{\mathrm{s}}$ and $\mathbf{u}^{\mathrm{s}}$ trajectories are consistent with the initial condition and simulation model. This is true for any *feasible* solution. Since the optimization algorithm solves simulation and optimization problems simultaneously, it is named the *simultaneous optimal control*. In this approach, at the intermediate steps of computing the solution, the state trajectory $\mathbf{x}^{\mathrm{s}}$ might not be a valid simulation corresponding to the control input $\mathbf{u}^{\mathrm{s}}$ Rawlings et al. (2019).

Although the simultaneous method involves state and control input as decision variables, leading to increased problem size, it results in a sparse structure. Many optimization solvers can exploit this sparsity Frison and Diehl (2020); Banjac et al. (2018) that potentially increase the computational performance while finding the solution. This method is preferable for nonlinear unstable systems and a problem with state constraints since they can be directly imposed inside the formulation.

The simultaneous approach can further be subclassified into *Direct collocation* and *Direct multiple shooting* methods. In the direct collocation method, all the continuous functions in the problem definition of a continuous-time OCP are approximated by a polynomial spline. The advantages of using polynomials are: i) They require a small number of coefficients to define, ii) They are easy to integrate or derivative in terms of these coefficients. On the other hand, in direct multiple shooting, the state trajectory is divided into multiple segments, and each segment is introduced as a separate simulation problem. This is in contrast with the single shooting method where an entire trajectory is formulated as a single simulation problem. Both in single and multiple shooting, the simulation methods can be chosen from any of the state-of-the-art integration schemes Quirynen (2017); Butcher (2003); Hairer et al. (1993, 1996) such as explicit/implicit Euler and Runge-Kutta methods.

In this section, we have discussed a general OCP formulation and briefly laid out a background for the subcategories of direct solution methods. In the next section, we will dive into applying some of these methods for the optimal control of legged locomotion.

4.4 Optimal Planning for Legged Robot

This section aims to design a TO-based planner to provide feasible trajectories for a legged robot for dynamic locomotion. These locomotion involve capturing the information about the angular dynamics of the robot that are described by nonlinear equations as detailed in section 3.2. Moreover, physical constraints exist on the state and control inputs involved during optimal planning. These constraints can also be nonlinear in nature, such as the robot feet contact model. In our case, we opt for a nonlinear SRBD model of a quadruped robot for OCP formulations. The SRBD model gives a reasonable approximation of the dynamics for a quadruped because most of the robot mass is concentrated in the trunk. Hence, the legs can be considered mass-less, and the joint dynamics can be neglected. However, the OCP formulation can easily be extended to any other nonlinear robot model. Further, these OCP formulations usually include a linearized form of these robot models while transcribing into discrete-time form Andersson et al. (2019). Indeed, one can linearize these models and incorporate them inside Linear Programming (LP) or Quadratic Programming (QP) to synthesize a TO problem. Additionally, linear constraints or linearized forms of nonlinear constraints can also be imposed in TO formulations, such as bounds on state and control inputs. These constraints can be in the form of equalities or inequalities. The advantage is that the LP and QP problems can be solved using corresponding off-the-shelf solvers.

Next, we will explain our LTV-Based Trajectory Optimization (LTV-Opt) formulation and its merits. Then, we discuss our SQP algorithm for optimal planning. We also give insight into the performance of these two techniques and compare their results. After that, we motivate our path for selecting RTI algorithm for our NMPC that will be discussed thoroughly in the next chapter.

4.4.1 LTV-Based Trajectory Optimization

The objective of this work is to enable a legged robot to perform dynamic locomotion autonomously with optimal planning. Therefore, capturing its dynamics with certain accuracy during optimal planning is essential. This is done by using a suitable robot model inside an optimization formulation. One could utilize a nonlinear model of the robot for the optimal planning to solve the resulting OCP using solvers such as `Ipopt` Wächter and Biegler (2006), `SNOPT` Gill et al. (2002), `KNITRO` Byrd et al. (2006), and `fmincon` MATLAB (2022b). However, using these nonlinear models inside an OCP can be computationally demanding due to their complexity. Hence, a remedy can be to incorporate a linearized version of a nonlinear model in the optimization formulation.

The first candidate for the linearized model is an LTI model. This model is obtained by linearizing a nonlinear model at a given linearization point. Typically, a legged robot performs a dynamic task that might be difficult to capture with a single operating point. Thus, it is challenging to determine a single linearization point ($\bar{\mathbf{x}}$, $\bar{\mathbf{u}}$) to obtain a linear model that can represent its dynamics with a certain degree of accuracy while performing dynamic tasks, such as walking on a pile of rubble. Further, the LTI model is accurate only in the neighborhood of a linearization point and can become unusable for the dynamics that deviate from the linearization point by a certain amount. Hence, a more accurate model than LTI is desirable. A suitable candidate for our TO formulation is the LTV model, obtained by linearizing over a set of linearization trajectories ($\mathbf{x}_k^{\mathrm{L}}$, $\mathbf{u}_k^{\mathrm{L}}$) instead of a single point. The advantage of an LTV model is that it allows one to locally approximate the dynamics of a system at multiple linearization points over a trajectory known a priori. Moreover, an LTV model combined with either a linear or quadratic cost function can also be solved with an LP or a QP solver, respectively.

LTVOpt Formulation

The proposed LTV-based TO formulation consists of a cost function (4.4a) to track references of the state $\mathbf{x}^{\mathrm{p}} \in \mathbb{R}^{n_x \times N}$ and control input $\mathbf{u}^{\mathrm{p}} \in \mathbb{R}^{n_u \times N}$ variables, where n_x and n_u denote the number of state and control inputs, respectively. The discrete time control intervals N

are obtained by dividing the prediction horizon T by the sampling time T_s. Further, the initial condition (4.4b) and LTV dynamic model (4.4c) are imposed with the equality constraints in this formulation, whereas the path constraints on the state and control inputs are introduced through inequalities (4.4d).

$$
\begin{aligned}
\min_{\mathbf{x}^\mathrm{p},\mathbf{u}^\mathrm{p}} \quad & \sum_{k=0}^{N-1} \| \mathbf{x}_k - \mathbf{x}_k^\mathrm{ref} \|^2_{\mathbf{W}_\mathrm{x}} + \| \mathbf{u}_k - \mathbf{u}_k^\mathrm{ref} \|^2_{\mathbf{W}_\mathrm{u}} \\
& + \| \mathbf{x}_N - \mathbf{x}_N^\mathrm{ref} \|^2_{\mathbf{W}_{\mathrm{x_N}}} && \text{(4.4a)} \\
\text{s.t.} \quad & \mathbf{x}_0 = \hat{\mathbf{x}}_0, && \text{(4.4b)} \\
& \mathbf{x}_{k+1} = \mathbf{A}_k \mathbf{x}_k + \mathbf{B}_k \mathbf{u}_k + \mathbf{r}_k, \qquad k \in \mathbb{I}_0^{N-1}, && \text{(4.4c)} \\
& \mathbf{C}_k \mathbf{x}_k + \mathbf{D}_k \mathbf{u}_k + \mathbf{h}_k \geq 0, \qquad k \in \mathbb{I}_0^{N-1} && \text{(4.4d)}
\end{aligned}
$$

Vector $\hat{\mathbf{x}}_0$ is the current system state. The weight matrices $\mathbf{W}_\mathrm{x}$, $\mathbf{W}_\mathrm{u}$ and $\mathbf{W}_{\mathrm{x_N}}$ are used to define the relative importance of the respective cost terms inside the formulation. Matrices $\mathbf{A}_k$, $\mathbf{B}_k$, $\mathbf{C}_k$, $\mathbf{D}_k$, and vectors $\mathbf{r}_k$ and $\mathbf{h}_k$ are obtained by linearizing the nonlinear dynamics and constraints. Next, we will specify the dynamic model and constraints introduced in this formulation.

Robot Model

In our case, to obtain the LTV model (3.17), we rely on linearizing the SRBD model (3.11) over a set of state $\mathbf{x} \in \mathbb{R}^{12}$ and control input $\mathbf{u} \in \mathbb{R}^{12}$ trajectories. We choose the reference trajectory ($\mathbf{x}^\mathrm{ref}$, $\mathbf{u}^\mathrm{ref}$) as a linearization trajectory since the robot needs to track some of these references depending on a user-defined goal.

Friction Cone and Unilateral Constraints

Friction cone constraints are encoded with their square pyramid approximation:

$$
\begin{aligned}
-\mu_i \mathbf{f}_{\mathrm{z},i} \leq \mathbf{f}_{\mathrm{x},i} \leq \mu_i \mathbf{f}_{\mathrm{z},i} && \text{(4.5a)} \\
-\mu_i \mathbf{f}_{\mathrm{z},i} \leq \mathbf{f}_{\mathrm{y},i} \leq \mu_i \mathbf{f}_{\mathrm{z},i} && \text{(4.5b)} \\
\underline{\mathbf{f}}_\mathrm{z} \leq \mathbf{f}_{\mathrm{z},i} \leq \overline{\mathbf{f}}_\mathrm{z} && \text{(4.5c)}
\end{aligned}
$$

where, $\underline{\mathbf{f}}_{\mathrm{z}}$ and $\overline{\mathbf{f}}_{\mathrm{z}}$ are upper and lower bounds on GRFs Z component, respectively, and μ_i is the friction coefficient of the contact surface. Choosing $\underline{\mathbf{f}}_{z}$ greater than or equal to zero enforces unilateral constraints on the normal forces $\mathbf{f}_{\mathrm{z}}$. The friction cone and unilateral constraint are represented by (4.4d) in the LTVOpt formulation.

Reference Generation

The references to obtain the LTV form of the SRBD model and required in the tracking cost (4.4a) are generated using heuristics. The CoM X-Y positions are computed by integrating the user X-Y velocities with the explicit Euler method. The CoM Z position is set equal to the height of the robot. The references for base orientation are chosen to be zero, except if the user defines a heading velocity, then the yaw is computed by integrating this velocity. The reference angular velocities are also set to zero. The foot location is generated similarly to the one reported by Focchi et al. (2020b). Finally, the normal GRFs are computed by dividing the robot mass by the number of feet on the ground. More details on the reference generation are given in section 5.4 of the next chapter.

The LTVOpt formulation uses reference trajectory to linearize the model of the system. Often, the reference trajectories are not dynamically feasible, meaning that these trajectories do not obey the robot's dynamics. Therefore, the LTV model linearized over this trajectory can be inaccurate and lead to the infeasible solution of the LTVOpt (4.4). Many tackle this problem with a widely known solution method called SQP. In SQP, several iterations of QP problems are performed to achieve a dynamically feasible solution while respecting the constraints. In the following sections, we will describe the SQP formulation and present the results obtained from both techniques.

4.4.2 Sequential Quadratic Programming

SQP is a popular algorithm that solves an NLP by iteratively solving local quadratic approximations (QPs) of the problem Nocedal and Wright (2006). At each SQP iteration, the solution from the previous step is recycled to define an initial guess $(\mathbf{x}_k^{\mathrm{L}}, \mathbf{u}_k^{\mathrm{L}})$, which is then used to construct a QP approximation of the NLP (4.2), given by

$$\min_{\Delta\mathbf{x},\Delta\mathbf{u}} \quad \sum_{k=0}^{N-1} \frac{1}{2} \begin{bmatrix} \Delta\mathbf{x}_k \\ \Delta\mathbf{u}_k \end{bmatrix}^\top \mathbf{H}_k \begin{bmatrix} \Delta\mathbf{x}_k \\ \Delta\mathbf{u}_k \end{bmatrix} + \mathbf{J}_k^\top \begin{bmatrix} \Delta\mathbf{x}_k \\ \Delta\mathbf{u}_k \end{bmatrix} \tag{4.6a}$$

$$\text{s.t.}\quad \Delta\mathbf{x}_0 = \hat{\mathbf{x}}_0 - \mathbf{x}_0^{\mathrm{L}}, \tag{4.6b}$$
$$\Delta\mathbf{x}_{k+1} = \mathbf{A}_k\Delta\mathbf{x}_k + \mathbf{B}_k\Delta\mathbf{u}_k + \mathbf{r}_k, \tag{4.6c}$$
$$\mathbf{C}_k\Delta\mathbf{x}_k + \mathbf{D}_k\Delta\mathbf{u}_k + \mathbf{h}_k \geq 0, \tag{4.6d}$$

where, $\Delta\mathbf{x}_k = \mathbf{x}_k - \mathbf{x}_k^{\mathrm{L}}$, $\Delta\mathbf{u}_k = \mathbf{u}_k - \mathbf{u}_k^{\mathrm{L}}$, $\hat{\mathbf{x}}_0$ is the current system state, $\mathbf{a}_k$ is the parameter, and

$$\begin{aligned}
\mathbf{A}_k &= \left.\frac{\partial f(\mathbf{x},\mathbf{u},\mathbf{a}_k)}{\partial \mathbf{x}}\right|_{\mathbf{x}_k^{\mathrm{L}},\mathbf{u}_k^{\mathrm{L}}}, & \mathbf{B}_k &= \left.\frac{\partial f(\mathbf{x},\mathbf{u},\mathbf{a}_k)}{\partial \mathbf{u}}\right|_{\mathbf{x}_k^{\mathrm{L}},\mathbf{u}_k^{\mathrm{L}}}, \\
\mathbf{C}_k &= \left.\frac{\partial h(\mathbf{x},\mathbf{u},\mathbf{a}_k)}{\partial \mathbf{x}}\right|_{\mathbf{x}_k^{\mathrm{L}},\mathbf{u}_k^{\mathrm{L}}}, & \mathbf{D}_k &= \left.\frac{\partial h(\mathbf{x},\mathbf{u},\mathbf{a}_k)}{\partial \mathbf{u}}\right|_{\mathbf{x}_k^{\mathrm{L}},\mathbf{u}_k^{\mathrm{L}}}, \\
\mathbf{r}_k &= g\left(\mathbf{x}_k^{\mathrm{L}},\mathbf{u}_k^{\mathrm{L}},\mathbf{a}_k\right) - \mathbf{x}_{k+1}^{\mathrm{L}}, & \mathbf{h}_k &= h\left(\mathbf{x}_k^{\mathrm{L}},\mathbf{u}_k^{\mathrm{L}},\mathbf{a}_k\right) \\
\mathbf{J}_k &= \mathbf{W}_k \begin{bmatrix} \mathbf{x}_k^{\mathrm{L}} - \mathbf{x}_k^{\mathrm{ref}} \\ \mathbf{u}_k^{\mathrm{L}} - \mathbf{u}_k^{\mathrm{ref}} \end{bmatrix}
\end{aligned} \tag{4.7}$$

Matrix $\mathbf{H}_k$ is the diagonal blocks of a suitable approximation of the Lagrangian Hessian. Since our problem relies on a least-squares cost, we adopt the popular Gauss-Newton Hessian approximation Diehl et al. (2005b); Gros et al. (2020) that gives $\mathbf{H}_k = \mathbf{W}_k$.

The pseudo-code of the SQP is laid out in Algorithm 1. This algorithm takes an initial condition $\hat{\mathbf{x}}_0$ and reference trajectories $(\mathbf{x}^{\mathrm{ref}}, \mathbf{u}^{\mathrm{ref}})$ as inputs and returns the optimal trajectories ($\mathbf{x}^{\mathrm{sqp}}$, $\mathbf{u}^{\mathrm{sqp}}$) when the algorithm is terminated based on the tolerances ζ_{l} and ζ_{e} (usually set to machine precision) on the Lagrangian $\Delta\mathcal{L}(\Delta\mathbf{x}, \Delta\mathbf{u})$ Nocedal and Wright (2006) and 1-norm of the equality constraints $f_{\mathrm{d}}(\mathbf{x}^{\mathrm{L}}, \mathbf{u}^{\mathrm{L}})$, respectively. In the first iteration of this algorithm, the linearization trajectories $(\mathbf{x}^{\mathrm{L}}, \mathbf{u}^{\mathrm{L}})$ are set equal to the input references. In the following iterations, these trajectories are updated with the previous QP solution trajectories.. The algorithm also relies on computing the sensitivities to formulate the QP (4.6).

One of the essential aspects of the SQP algorithm is to find the Newton step α that can be calculated with a globalization method. One of these methods is called backtracking line search Nocedal and Wright (2006) as described in Algorithm 2. This algorithm takes as inputs linearization trajectories $(\mathbf{x}^{\mathrm{L}}, \mathbf{u}^{\mathrm{L}})$ and Lagrange multiplier λ from the SQP algorithm and computes α by evaluating the merit function M. The merit function comprises the NLP (4.2a) cost function and 1-norm of equality constraints as expressed in line 2 of Algorithm 2. The algorithm termin-

Algorithm 1 SQP with line search

```
    Input: initial state $\hat{x}_0$, reference trajectory $\mathbf{x}^{\mathrm{ref}}, \mathbf{u}^{\mathrm{ref}}$
 1: while $\| [\Delta\mathcal{L}(\Delta\mathbf{x}, \Delta\mathbf{u})] \|_1 \geq \zeta_{\mathrm{l}}$ or $\| f_{\mathrm{d}}(\mathbf{x}^{\mathrm{L}}, \mathbf{u}^{\mathrm{L}}) \|_1 \geq \zeta_{\mathrm{e}}$ do
 2:     if firstIteration then
 3:         Set initial guess to reference trajectory
            $(\mathbf{x}^{\mathrm{L}}, \mathbf{u}^{\mathrm{L}}) \leftarrow (\mathbf{x}^{\mathrm{ref}}, \mathbf{u}^{\mathrm{ref}})$
 4:     else
 5:         Set initial guess to shifted $(\mathbf{x}^{\mathrm{L}}, \mathbf{u}^{\mathrm{L}})$
            $\mathbf{x}^{\mathrm{L}}_k \leftarrow \mathbf{x}_{k+1} \quad \forall \quad k = 0, 1, \cdots, N-1$
            $\mathbf{x}^{\mathrm{L}}_N \leftarrow \mathbf{x}_{N-1}$
            $\mathbf{u}^{\mathrm{L}}_k \leftarrow \mathbf{u}_{k+1} \quad \forall \quad k = 0, 1, \cdots, N-2$
            $\mathbf{u}^{\mathrm{L}}_{N-1} \leftarrow \mathbf{u}_{N-2}$
 6:     end if
 7:     Evaluate $\mathbf{r}_k, \mathbf{h}_k$, and the sensitivities $\mathbf{A}_k, \mathbf{B}_k, \mathbf{C}_k, \mathbf{D}_k, \mathbf{H}_k$, $\mathbf{J}_k$ at
        $(\mathbf{x}^{\mathrm{L}}, \mathbf{u}^{\mathrm{L}}, \mathbf{a})$ according to (4.7)
 8:     Formulate and solve QP (4.6) to obtain Newton direction $\Delta\mathbf{x}$, $\Delta\mathbf{u}$
        and Lagrange multiplier $\lambda$
 9:     Perform a line-search with Algorithm 2 to find a Newton step
        $\alpha \in (0, 1]$ that guarantees descent
10:     Update initial guess with the newton step:
         $\mathbf{x}^{\mathrm{L}} \leftarrow \mathbf{x}^{\mathrm{L}} + \alpha\Delta\mathbf{x}$
         $\mathbf{u}^{\mathrm{L}} \leftarrow \mathbf{u}^{\mathrm{L}} + \alpha\Delta\mathbf{u}$
11: end while
    return $(\mathbf{x}^{\mathrm{sqp}}, \mathbf{u}^{\mathrm{sqp}}) = (\mathbf{x}^{\mathrm{L}}, \mathbf{u}^{\mathrm{L}})$
```

Algorithm 2 Backtracking line search

```
    **Input:** $\mathbf{x}^{\mathrm{L}}$, $\mathbf{u}^{\mathrm{L}}$, $\lambda$
 1: Initialization: $\alpha \leftarrow 1$, $\beta \leftarrow (0,1]$, $\sigma \leftarrow \| \lambda \|_\infty$
 2: Compute merit function:
        $M \leftarrow \ell(\mathbf{x}^{\mathrm{L}}, \mathbf{u}^{\mathrm{L}}) + \sigma \parallel f_{\mathrm{d}}(\mathbf{x}^{\mathrm{L}}, \mathbf{u}^{\mathrm{L}}) \parallel_1$
 3: while conv == false do
 4:     Take a full step:
        $\mathbf{x}^{\mathrm{L}}_{\mathrm{next}} \leftarrow \mathbf{x}^{\mathrm{L}} + \alpha \Delta x$
        $\mathbf{u}^{\mathrm{L}}_{\mathrm{next}} \leftarrow \mathbf{u}^{\mathrm{L}} + \alpha \Delta u$
 5:     Compute merit at new linearization points:
        $M_{\mathrm{next}} \leftarrow \ell(\mathbf{x}^{\mathrm{L}}_{\mathrm{next}}, \mathbf{u}^{\mathrm{L}}_{\mathrm{next}}) + \sigma \parallel f_{\mathrm{d}}(\mathbf{x}^{\mathrm{L}}_{\mathrm{next}}, \mathbf{u}^{\mathrm{L}}_{\mathrm{next}}) \parallel_1$
 6:     if $M_{\mathrm{next}} <= M$ or $\alpha == 0$ then
 7:         conv $\leftarrow$ true
 8:     else
 9:         $\alpha \leftarrow \beta * \alpha$
10:     end if
11: end while
    return: $\alpha$
```

ates when the merit function M_{next} value with an updated value of α is smaller or equal to the original value M or when α is 0.

Henceforth, we refer to the optimal planning based on the SQP algorithm as SQP-Based Trajectory Optimization (SQPOpt). For SQPOpt, we keep the same setup as LTVOpt for what regards the robot model, inequality constraints and reference generation as described in section 4.4.1.

4.4.3 Results

The parameters and weights used for the LTVOpt and SQPOpt algorithms are listed in Tables 2 and 3, respectively. Since both of these formulations involve linearization/sensitivity computation, we rely on `CasADi` Andersson et al. (2019) tool, which has the symbolic framework to compute derivatives efficiently using algorithmic differentiation. The linearization and discretization using the explicit Euler method are performed as described in section 3.3.2. The LTVOpt (4.4) and a QP (4.6) subproblem of SQP can be reformulated as QP as described in Appendix C. Therefore, it allows accessible interfaces to many off-the-shelf QP solvers. We solve the resulting QP for LTVOpt and SQPOpt using Gurobi Gurobi Optimization, LLC (2023).

The results from the LTVOpt and SQPOpt are shown in Fig. 17-21 for HyQ robot with crawl gait. The robot crawls with the user-commanded forward velocity of $0.05\,\mathrm{m/s}$, and the references are generated based on this goal. A tracking cost on CoM velocity ensures that the robot follows the target forward velocity. We do not track the CoM X-Y position because we let the planning algorithm to optimize them based on the linear velocities. The CoM Z position is tracked to maintain the robot height of $0.55\,\mathrm{m}$. The base orientation and angular velocities are penalized minimizing the deviation from zero references. The GRFs are penalized in the tracking cost to optimize them around the reference values while respecting the dynamic feasibility and friction and cone constraints.

The CoM position and velocity output in Fig. 17 and 18 show similar trends for both methods. Note that the references are computed from the heuristic. Therefore, both algorithms calculate the feasible state $\mathbf{x}$ and control input $\mathbf{u}$ trajectory for the robot, respecting the dynamics given the SRBD model and friction and unilateral constraints. The feasibility

[1]Note that some of these parameters not required for LTVOpt, therefore not applicable to it.

Table 2: LTVOpt[1]and SQP parameters

Parameter	**Symbol**	**Value**	**Unit**
Number of state	n_x	12	-
Number of control inputs	n_u	12	-
Prediction horizon	T	4	s
Sampling time	T_s	0.04	s
Number of control intervals	N	100	-
Friction coefficient	μ	0.7	-
GRFs lower bound	$\underline{\mathbf{f}}_\mathrm{z}$	0	N
GRFs upper bound	$\overline{\mathbf{f}}_\mathrm{z}$	500	N
Initial Newton step	α	1	m
Scaling factor	β	0.9	-
Lagrangian tolerance	ζ_l	10^{-6}	-
Equality tolerance	ζ_e	10^{-6}	-

Table 3: Weights used in the LTVOpt and SQP

Cost	**Weight**	**Value**
State	$\mathbf{W}_{\mathbf{p}_\mathrm{c}}$	diag(0, 0, 1000)
	$\mathbf{W}_{\mathbf{v}}$	diag(100, 100, 100)
	$\mathbf{W}_{\boldsymbol{\Phi}}$	diag(100, 100, 100)
	$\mathbf{W}_{\boldsymbol{\omega}}$	diag(100, 100, 100)
Force	$\mathbf{W}_{\mathrm{u}_\mathrm{x}}$	1×10^{-3}
	$\mathbf{W}_{\mathrm{u}_\mathrm{y}}$	1×10^{-3}
	$\mathbf{W}_{\mathrm{u}_\mathrm{z}}$	1×10^{-5}

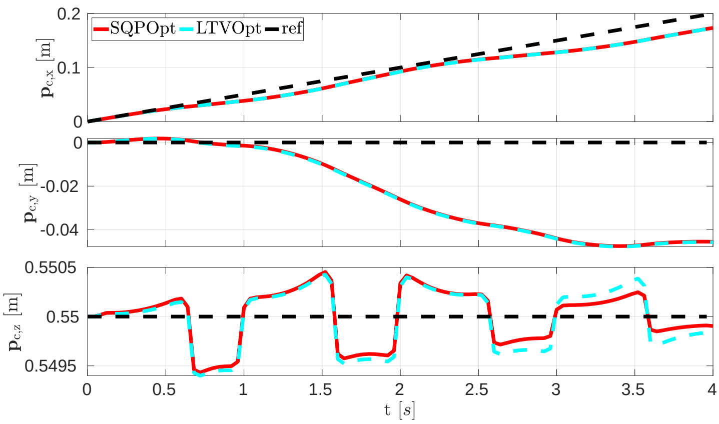


Figure 17: CoM position plots from LTVOpt and SQPOpt results. Red line is the output of SQPOpt, dashed cyan line is the result of LTVOpt and black dashed line is CoM reference position.

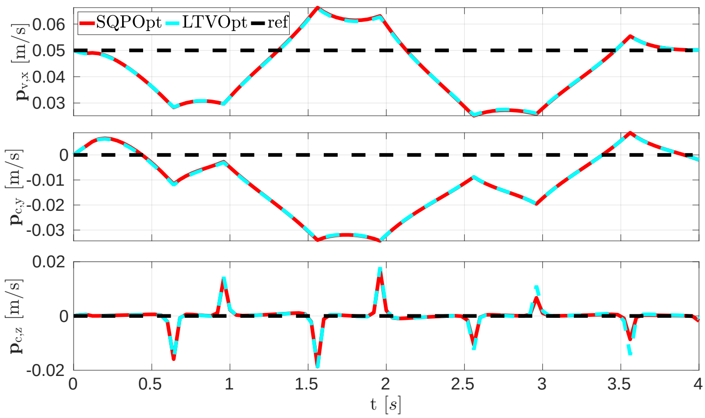


Figure 18: CoM velocity plots from LTVOpt and SQPOpt results. Red line is the output of SQPOpt, dashed cyan line is the result of LTVOpt and black dashed line is CoM reference velocity.

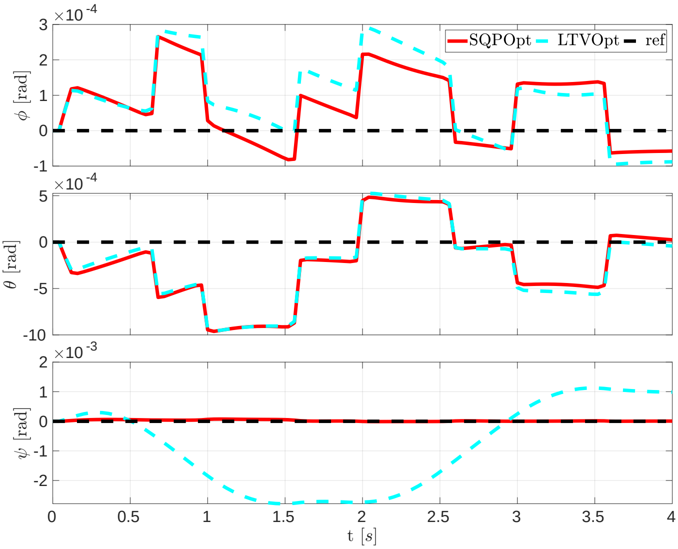


Figure 19: Orientation plots from LTVOpt and SQPOpt results. Red line is the output of SQPOpt, dashed cyan line is the result of LTVOpt and black dashed line is CoM reference orientation.

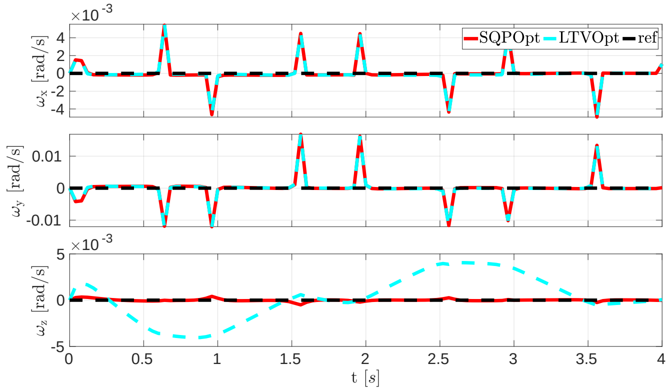


Figure 20: Angular velocity plots from LTVOpt and SQPOpt results. Red line is the output of SQPOpt, dashed cyan line is the result of LTVOpt and black dashed line is CoM reference angular velocity.

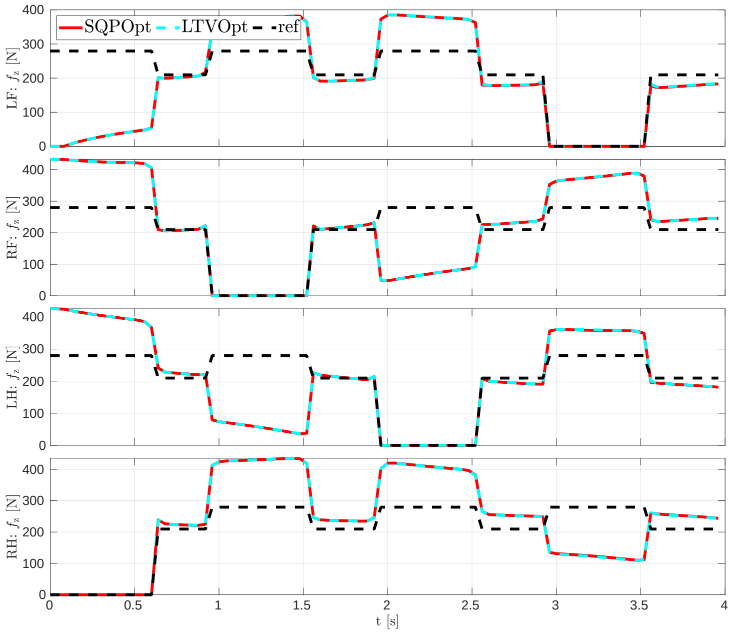


Figure 21: GRFs plots from LTVOpt and SQPOpt results. Red line is the output of SQPOpt, dashed cyan line is the result of LTVOpt and black dashed line is reference GRFs.

of the solution is higher in the SQPOpt compared to the LTVOpt, since the SQP algorithm performs several iterations to respect the Lagrangian ζ_l and equality constraint tolerances ζ_e.

As demonstrated in Fig. 19 and 20, the base orientations and angular velocities are similar for both methods except for yaw ψ and angular velocity ω_z. However, the variance of ψ and ω_z is small (order of 10^{-3}).

The normal components of control inputs, i.e., GRFs, have similar trends for both these methods. The variance of the output from LTVOpt and SQPOpt with respect to the reference GRFs clearly show that the reference GRFs are not dynamically feasible. Indeed, these components are computed by dividing the total robot mass among the feet on the ground for a crawl gait. Therefore, these methods compute dynamically feasible GRFs based on the LTV model and respecting the friction and cone constraints.

The choice of weights in these algorithms plays a crucial role while finding the solution. Increasing the weights on the tracking cost (4.4a) and (4.6a) decides which optimization variable is prioritized the most while optimizing. For example, increasing relatively the weights on GRFs tracking cost with respect to the other cost terms prioritizes tracking the GRFs references while finding the optimal solution. Therefore, achieving the right balance between the different cost terms inside these optimization problems to achieve a particular goal can be a cumbersome task Bouyarmane and Kheddar (2018). In our case, we choose the weight with the trial and error method. We tune the weights of the different cost terms relative to each other to obtain the desired outcome.

Solution Time

The average solution time by LTVOpt is $200\,\mathrm{ms}$, whereas the SQPOpt takes roughly 1-$1.2\,\mathrm{s}$ to solve the optimization problem until convergence. Since one of the goals of this dissertation is to re-plan in the RHP fashion at a frequency higher than $20\,\mathrm{Hz}$ for locomotion stability and performance Bledt and Kim (2019), the solution time required by both of these algorithms is limiting to reaching this goal.

Dynamic Feasibility

LTVOpt is a promising solution to achieve optimal motion planning, but since it is linearized over a reference trajectory, its solution might be dynamically infeasible. However, from the results presented before, it can be verified that the output of the LTVOpt is very similar to the converged SQPOpt. Further, in the context of the SQP algorithm, solving the LTVOpt is equivalent to solving the first iteration of the SQP. Therefore in some cases, it might not be necessary to solve the problem until convergence if the first iteration of SQPOpt, i.e., the solution of LTVOpt, is similar to the converged SQPOpt solution. Moreover, one could use the solution of the previous LTVOpt to warm-start the following LTVOpt formulation. By warm-start, we mean to utilize the solution of the previous LTVOpt as linearization trajectory for the following LTVOpt formulation instead of using the reference trajectory each time the re-planning is performed. This is a valid approach when the robot follows a planned trajectory and does not diverge significantly from the planned trajectory. Also, with the warm-start strategy, the computation time of LTVOpt can improve.

Reasoning on this line, we propose a formulation based on a well-known solution method called RTI in the next chapter to achieve performance in terms of dynamic feasibility and solution time for our NMPC. The RTI is specifically tailored for real-time implementation and also inherits the convergence properties SQP under certain assumptions Diehl et al. (2005b).

4.5 Summary

This chapter briefly described a standard MPC scheme and OCP formulation. We also discussed the discrete-time OCP formulation, followed by a short discussion on direct methods. Then, we proposed two solution methods for optimal planning for a legged robot, i.e., LTVOpt and SQPOpt. We presented their formulations, algorithms, and implementation details, followed by the results obtained from them in simulation. Later, we discussed that the LTVOpt has an advantage in the computational aspect, whereas SQPOpt gives a converged and dynamically feasible solution to the OCP. Considering these two aspects, in the next chapter, we will introduce our NMPC planner based on the RTI solution scheme that inherits the advantages of LTVOpt and SQPOpt.

Chapter 5

Nonlinear MPC for Legged Locomotion

The use of MPC Koenemann et al. (2015); Farshidian et al. (2017); Neunert et al. (2018); Bledt and Kim (2019); Grandia et al. (2019) in legged locomotion has picked up a pace in the last two decades. Linear MPC for the legged locomotion is utilized to achieve a variety of gaits. However, some of these approaches either neglect Herdt et al. (2010); Horvat et al. (2017) angular dynamics or require the linearization Di Carlo et al. (2018) of angular dynamics that is not suitable for the motions that involve substantial variation from the horizontal position. To address this issue, NMPC can be utilized where the nonlinear dynamics of the robot can be included as equality constraints. The NMPC also allows one to include nonlinear inequality constraints in contrast to the linear inequalities required by the linear MPC. Lastly, in the NMPC, the cost function can be non-quadratic, i.e., linear or nonlinear. This dissertation aims to achieve environment adaptation by legged robots that involve considerable angular (nonlinear) dynamics. Therefore, we select the NMPC for optimal re-planning.

This work includes a simplified SRBD model with stance status parameters inside the NMPC formulation to eliminate the complementarity constraints contributing to the computational benefits. Further, we include leg mobility in the NMPC cost to improve the base orientation, enabling the robot to adapt to rough terrains. To the best of our knowledge, apart from the work mentioned in Section 2.3, this is the first implementation of mobility inside the NMPC for legged locomotion. The proposed NMPC runs at 25 Hz with a prediction horizon of 2 s (50 nodes) on the a single computer along with the rest of the locomotion framework. The efficacy of the NMPC proposed in this dissertation is demonstrated through several simulations and experimental scenarios on challenging terrains.

This chapter reprints the content of the following article Rathod et al. (2021a) that is under a Creative Commons License (CC BY-NC-ND 4.0).

N. Rathod, A. Bratta, M. Focchi, M. Zanon, O. Villarreal, C. Semini, and A. Bemporad. Model predictive control with environment adaptation for legged locomotion, *IEEE Access*, vol. 9, pp. 145710-145727, 2021.

In this chapter, we describe the setup and ingredients required to develop our NMPC for legged locomotion. The chapter begin with the locomotion framework overview spotting different elements involved along with the NMPC. Then we present formulation of our NMPC, followed by the explanation regarding different enhancement feature that empower our NMPC. We also discuss about the reference generator, WBC and a solution method RTI in the last three sections of this chapter. Finally, we present simulation and experimental results with the NMPC on the HyQ robot.

5.1 Locomotion Framework

Fig. 22 illustrates the planning pipeline of our locomotion framework. The *reference generator*, as discussed in Section 5.4, takes the user input (longitudinal, lateral and angular velocity), schedule of the gait (e.g., a crawl) timing, the initial state of the robot, and a map of the terrain to generate reference trajectories for the state $\mathbf{x}^{\mathrm{ref}}$ and control input $\mathbf{u}^{\mathrm{ref}}$ required by the NMPC. The reference generator also provides a vector of parameters $\mathbf{a}$ to the NMPC, that includes foot locations and sequences of contact status. The NMPC running at 25 Hz delivers the optimal trajectories of the state $\mathbf{x}^{\mathrm{p}}$ and control input $\mathbf{u}^{\mathrm{p}}$, as detailed in Section 5.2. All the components of the Whole-Body controller (highlighted with dashed box in Fig. 22) are discussed in Section 5.5. The *WBC interface* interpolates the optimal state $\mathbf{x}^{\mathrm{p}}$ at a rate of 250 Hz to generate a desired signal $\mathbf{x}^{\mathrm{d}}$ for a Cartesian virtual impedance controller Focchi et al. (2016). The WBC

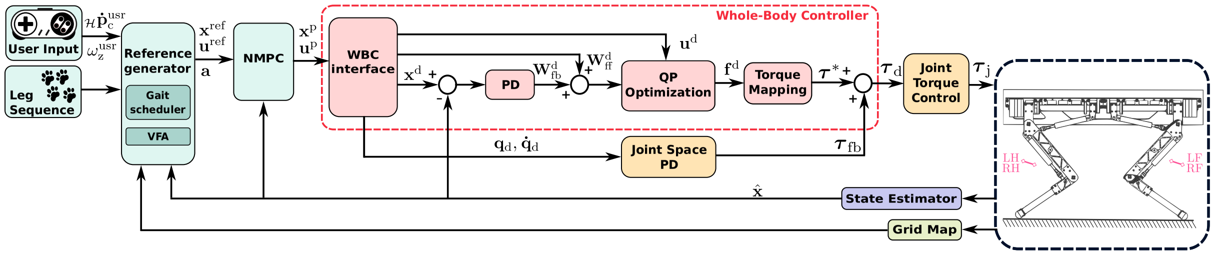


Figure 22: Block diagram of the planning pipeline with the NMPC in our locomotion framework. The reference generator provides the references ($\mathbf{x}^{ref}$, $\mathbf{u}^{ref}$) to NMPC after receiving the user inputs. Then the NMPC passes optimal state $\mathbf{x}^{P}$ and control $\mathbf{x}^{P}$ trajectories to the Whole-Body Controller. The torque $\boldsymbol{\tau}_d$ is given as reference to low level joint torque controller. The state estimator provides the state estimation $\hat{\mathbf{x}}$ to the required blocks. Finally, the heightmap is generated by Grid Map and given to the reference generator.

interface also computes the feedforward wrench $\mathbf{W}^{\mathrm{d}}_{\mathrm{ff}}$ that is added to a feedback wrench $\mathbf{W}^{\mathrm{d}}_{\mathrm{fb}}$ that renders the Cartesian impedance. Moreover, the WBC interface provides the joint position $\mathbf{q}_{\mathrm{d}}$ and velocities $\dot{\mathbf{q}}_{\mathrm{d}}$ to a Joint Space PD controller running at 1 kHz. After acquiring the feedback and feed-forward wrenches, a QP optimization computes the vector of desired GRFs $\mathbf{f}^{\mathrm{d}}$ accounting for the friction cone constraints and penalizing the the difference between $\mathbf{f}^{\mathrm{d}}$ and $\mathbf{u}^{\mathrm{p}}$ coming from the NMPC solution. Then, $\mathbf{f}^{\mathrm{d}}$ is mapped to the torque vector $\boldsymbol{\tau}^{*}$ that is added to the Joint Space PD torques $\boldsymbol{\tau}_{\mathrm{fb}}$ resulting into the total desired torque $\boldsymbol{\tau}_{\mathrm{d}}$. Ultimately, $\boldsymbol{\tau}_{\mathrm{d}}$ is passed to a low-level joint torque controller as reference Boaventura et al. (2015).

An online state estimator Nobili et al. (2017) that runs at 500 Hz provides the estimation of the robot state $\hat{\mathbf{x}}$ to all the components inside our locomotion framework that require it. A dedicated on-board computer takes inputs from an RGB-D camera (RealSense) mounted in front of the robot and generates a 2.5D heightmap at the rate of 30 Hz using the *Grid Map* library from Fankhauser and Hutter (2016). This heightmap is later sent to the reference generator.

5.2 NMPC Formulation

In our planning algorithm, we choose a real-time NMPC formulation because it has the ability to handle both the nonlinear system dynamics and the constraints, explicitly. NMPC is based on solving an OCP given the current state $\hat{\mathbf{x}}_0$ of the system. Only the first element of the optimized input trajectory is applied to the system, then the state is measured and the OCP is solved again based on the new state measurement to close the loop.

We define the decision variables as the predicted state and control input with $\mathbf{x}^{\mathrm{p}} := \{\mathbf{x}_0, \ldots, \mathbf{x}_N\}$ and $\mathbf{u}^{\mathrm{p}} := \{\mathbf{u}_0, \ldots, \mathbf{u}_{N-1}\}$, respectively, such that an NLP formulation can be stated as:

$$\min_{\mathbf{x}^{\mathrm{p}},\mathbf{u}^{\mathrm{p}}} \quad \sum_{k=0}^{N-1} \ell\left(\mathbf{x}_k, \mathbf{u}_k, \mathbf{a}_k\right) + \ell_{\mathrm{T}}\left(\mathbf{x}_N\right) \tag{5.1a}$$

$$\text{s.t.} \quad \mathbf{x}_0 = \hat{\mathbf{x}}_{\mathbf{0}}, \tag{5.1b}$$

$$\mathbf{x}_{k+1} = f\left(\mathbf{x}_k, \mathbf{u}_k, \mathbf{a}_k\right), \qquad k \in \mathbb{I}_0^{N-1}, \tag{5.1c}$$

$$h\left(\mathbf{x}_k, \mathbf{u}_k, \mathbf{a}_k\right) \leq 0, \qquad k \in \mathbb{I}_0^{N-1}, \tag{5.1d}$$

where, $\ell : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_a} \rightarrow \mathbb{R}$ is the stage cost function; $\ell_{\mathrm{T}} : \mathbb{R}^{n_x} \rightarrow \mathbb{R}$ is the terminal cost function. The initial condition (5.1b) is expressed by setting $\mathbf{x}_0$ equal to the state estimate $\hat{\mathbf{x}}_\mathbf{0}$ received from the state estimator. The vector of model parameters $\mathbf{a}_k$ is not optimized but it is computed externally by the reference generator and provided to the optimization problem formulation. The nonlinear system dynamics are introduced by the equality constraints (5.1c). Finally, the path constraints are included with (5.1d) which, for example, can be bounds on the decision variables. The NLP (5.1) is defined for a *prediction horizon* T that is divided into N discrete time *control intervals* of lengths $T_\mathrm{s} = \frac{T}{N}$. Hereafter, we will refer to T_s as the *sampling time*.

5.2.1 Cost

In our NMPC formulation we use a cost function of the form:

$$\ell\left(\mathbf{x}_k, \mathbf{u}_k, \mathbf{a}_k\right) = \ell_\mathrm{t} + \ell_\mathrm{m} + \ell_\mathrm{r}, \tag{5.2a}$$

$$\ell_\mathrm{t} = \| \mathbf{x}_k - \mathbf{x}_k^{\mathrm{ref}} \|_\mathbf{Q}^2 + \| \mathbf{u}_k - \mathbf{u}_k^{\mathrm{ref}} \|_\mathbf{R}^2, \tag{5.2b}$$

$$\ell_\mathrm{m} = \| {}_{\mathcal{C}}\mathbf{p}_{\mathrm{hf}_k} - {}_{\mathcal{C}}\mathbf{p}_{\mathrm{hf}_k}^{\mathrm{ref}} \|_\mathbf{M}^2, \tag{5.2c}$$

$$\ell_\mathrm{r} = \rho \, \| {}_{\mathcal{K}}\mathbf{u}_k \|_\mathbf{P}^2 \tag{5.2d}$$

- The tracking cost (5.2b) is associated to state and control input and the references trajectories $\mathbf{x}_k^{\mathrm{ref}}, \mathbf{u}_k^{\mathrm{ref}}$ are provided by the reference generator for each sampling instance k (refer to Section 5.4).

- The mobility cost (5.2c) is one of the contributions of this work that accounts for improving the leg mobility by penalizing the difference between the hip-to-foot distance ${}_{\mathcal{C}}\mathbf{p}_{\mathrm{hf}}$ and the reference value ${}_{\mathcal{C}}\mathbf{p}_{\mathrm{hf}}^{\mathrm{ref}}$ of maximum mobility. This cost allows the NMPC to optimize the robot base orientation (e.g. align it to the terrain shape) in order to increase the leg mobility which has as a desirable consequence to stay far from kinematic limits during locomotion. The derivation of ${}_{\mathcal{C}}\mathbf{p}_{\mathrm{hf}}^{\mathrm{ref}}$ is detailed separately in Section 5.3.1.

- In some locomotion scenarios Focchi et al. (2016), to cope with uncertainties in the contact normal estimation and increase robustness to external disturbances, it is desirable to have the GRFs $\mathbf{f}_i$ as close as possible to the center of the friction cone. This can be achieved with by penalizing X-Y components of $\mathbf{u}$ in a frame $\mathcal{K}$ (see Fig. 23) that is aligned to the normal of the contact and it is included in our cost function by a control input regularization term (5.2d), refer to Section 5.3.3 for the details.

The positive definite weight matrices $\mathbf{Q} \in \mathbb{S}_+^{n_x}, \mathbf{R} \in \mathbb{S}_+^{n_u}$, $\mathbf{M} \in \mathbb{S}_+^{12}$, $\mathbf{P} \in \mathbb{S}_+^{n_u}$ act as important tuning parameters in the NMPC formulation. The regularization factor ρ decides the trade-off between force robustness cost (5.2d) and both the tracking (5.2b) and mobility (5.2c) cost. Finally, we define the terminal cost $\ell_{\mathrm{T}} = \| \mathbf{x}_N - \mathbf{x}_N^{\mathrm{ref}} \|_{\mathbf{Q}_N}$ and use the weight matrix $\mathbf{Q}_N = \mathbf{Q}$ for this cost.

5.2.2 Robot Model

The inertial frame $\mathcal{W}$ and the CoM frame $\mathcal{C}$ are shown in Fig. 23. The CoM frame is aligned with the base of the robot and its origin is located at the CoM.

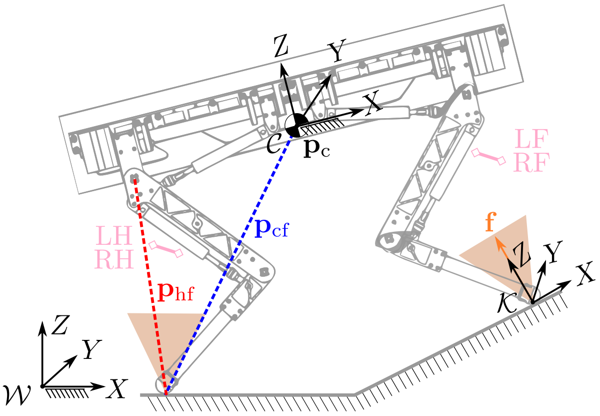

Z
Y
X
$\mathcal{C}$
$\mathbf{p}_{\mathrm{c}}$
LF
RF
LH
RH
$\mathbf{p}_{\mathrm{cf}}$
$\mathbf{f}$
Z
Y
X
$\mathcal{K}$
$\mathbf{p}_{\mathrm{hf}}$
Z
Y
X
$\mathcal{W}$

Figure 23: HyQ schematic showing the inertial frame ($\mathcal{W}$), the CoM frame ($\mathcal{C}$) attached to the CoM of the robot, and the contact frame ($\mathcal{K}$). The robot legs are shown in the *default* configuration.

We use a simplified reduced-order SRBD model Winkler et al. (2018) defined in a 6D space that describes the translational and angular dynamics of the robot while neglecting the dynamics of its swinging legs. This is a valid approximation for the HyQ robot because most of its mass is concentrated in the base, as mentioned in Semini et al. (2011) (the mass of the base is $61\,\mathrm{kg}$ and the mass of each leg is $6.5\,\mathrm{kg}$). The robot is approximated as a rigid body with the inertia computed considering the robot in a default leg configuration as shown in Fig. 23. We choose to define the SRBD model in the CoM frame (specifically the angular dynamics) because this choice yields a constant inertia tensor. Thus, the angular dynamic equations are much simpler i.e., less non-linear because the inertia tensor is not time varying. In the SRBD model, GRFs are applied as inputs to control the position and orientation of the robot base. The SRBD model is:

$$m\dot{\mathbf{v}}_{\mathrm{c}} = m\mathbf{g} + \sum_{i=1}^{4} \delta_i \mathbf{f}_i \tag{5.3a}$$

$$_{\mathcal{C}}\mathbf{I}_{\mathrm{c}}\, _{\mathcal{C}}\dot{\boldsymbol{\omega}} + {}_{\mathcal{C}}\boldsymbol{\omega} \times {}_{\mathcal{C}}\mathbf{I}_{\mathrm{c}} {}_{\mathcal{C}}\boldsymbol{\omega} = \sum_{i=1}^{4} \delta_i {}_{\mathcal{C}}\mathbf{p}_{\mathrm{cf},i} \times {}_{\mathcal{C}}\mathbf{f}_i \tag{5.3b}$$

where m is the robot mass, $\dot{\mathbf{v}}_{\mathrm{c}} \in \mathbb{R}^3$ is the CoM acceleration, $\mathbf{g}$ is the gravitational acceleration, $\mathbf{f}_i \in \mathbb{R}^3$ is the ground reaction force at foot i, $_{\mathcal{C}}\mathbf{I}_{\mathrm{c}} \in \mathbb{R}^{3\times3}$ is the inertia tensor computed at the CoM frame origin, $_{\mathcal{C}}\dot{\boldsymbol{\omega}} \in \mathbb{R}^3$ is the angular acceleration of the robot's base, $\mathbf{p}_{\mathrm{cf},i} \in \mathbb{R}^3$ is the distance between the CoM position $\mathbf{p}_{\mathrm{c}} \in \mathbb{R}^3$ and the position $\mathbf{p}_{\mathrm{f},i} \in \mathbb{R}^3$ of foot i. We introduce binary parameters $\delta_i = \{0, 1\}$ to define whether foot i is in contact with the ground and can therefore generate contact forces or not.

The robot dynamics governed by (5.3) can be expressed as the continuous-time state-space model:

$$\begin{bmatrix} \dot{\mathbf{p}}_{\mathrm{c}} \\ \dot{\mathbf{v}}_{\mathrm{c}} \\ \dot{\mathbf{\Phi}} \\ _{\mathcal{C}}\dot{\boldsymbol{\omega}} \end{bmatrix} = \begin{bmatrix} \mathbf{v}_{\mathrm{c}} \\ 1/m \sum_{i=1}^{4} \delta_i \mathbf{f}_i + \mathbf{g} \\ \mathbf{E}'^{-1}(\mathbf{\Phi}) {}_{\mathcal{C}}\boldsymbol{\omega} \\ -{}_{\mathcal{C}}\mathbf{I}_{\mathrm{c}}^{-1}({}_{\mathcal{C}}\boldsymbol{\omega} \times {}_{\mathcal{C}}\mathbf{I}_{\mathrm{c}} {}_{\mathcal{C}}\boldsymbol{\omega}) + \sum_{i=1}^{4} \delta_i {}_{\mathcal{C}}\mathbf{I}_{\mathrm{c}}^{-1} {}_{\mathcal{C}}\mathbf{p}_{\mathrm{cf},i} \times {}_{\mathcal{C}}\mathbf{f}_i \end{bmatrix} \tag{5.4}$$

where $\mathbf{v}_{\mathrm{c}}$ is the CoM velocity of the robot. The robot base orientation is represented by the sequence of Z-Y-X Euler angles [1] Diebel (2006) $\mathbf{\Phi} = (\phi,\, \theta,\, \psi)$ i.e., roll (ϕ), pitch (θ) and yaw (ψ), respectively. The relation between the Euler Angles rates $\dot{\mathbf{\Phi}}$ and angular velocity $_{\mathcal{C}}\boldsymbol{\omega}$ is well-known and discussed in Appendix A for the sake of completeness. We define the state and control vectors as $\mathbf{x} = (\mathbf{p}_{\mathrm{c}},\, \mathbf{v}_{\mathrm{c}},\, \mathbf{\Phi},\, {}_{\mathcal{C}}\boldsymbol{\omega})$, and $\mathbf{u} = (\mathbf{f}_1, \ldots, \mathbf{f}_4)$.

[1]Note that Euler angles can suffer from singularities that occur in certain configurations Younes et al. (2012). Because in this work we do not consider motions that involve such configurations, using Euler angles does not pose any issue. A singularity-free implementation is out of the scope of this work and is left for future research.

Equation (5.4) can be concisely written as:

$$\dot{\mathbf{x}}(t) = g(\mathbf{x}(t), \mathbf{u}(t), \mathbf{a}(t)), \tag{5.5}$$

where $\mathbf{a} = (\mathbf{p}_\mathrm{f},\, \boldsymbol{\delta})$ is a vector of parameters that includes the feet positions $\mathbf{p}_\mathrm{f}$ and the contact status $\boldsymbol{\delta} \in \mathbb{R}^4$. One of the things required by the NLP formulation 5.1 is a model of the robot.

The rigid-body dynamics (5.5) are discretized using numerical integration Quirynen (2017); Butcher (2003); Hairer et al. (1993, 1996) to obtain the discrete-time model:

$$\mathbf{x}_{k+1} = f\left(\mathbf{x}_k, \mathbf{u}_k, \mathbf{a}_k\right), \tag{5.6}$$

which defines equality constraints (5.1c) imposed at every stage k in MPC to ensure that the state trajectory satisfies the system dynamics for the given control inputs.

One specific feature of legged robots is the need to ensure that the values of the GRFs equal to zero for a swinging leg. This is typically done by introducing complementarity constraints Cebe et al. (2020); Villarreal et al. (2020). These constraints, however, pose several difficulties in the solution of the optimization problem, since the vast majority of the NLP algorithms cannot handle them and tailored solvers are required. Ultimately, this results in a significant increase in computation time. An alternative to complementarity constraints consists in providing the sequence of contact status $\boldsymbol{\delta}$ as input parameters in the state space model (5.5). In this manner, a contact mode δ_i is multiplied with the terms involving force $\mathbf{f}_i$ in (5.4) and the contribution of that force is nullified during the swing phase of the corresponding leg i. Hence, there is no more need to include complementarity constraints separately in (5.1) which results in fewer constraints and, consequently, in a relatively smaller NMPC formulation.

5.2.3 Friction Cone and Unilateral constraints

Friction cone and unilateral constraints used by the NMPC formulation are detailed in section 4.4.1. These constraints are represented by $h\left(\mathbf{x}_k, \mathbf{u}_k, \mathbf{a}_k\right) \leq 0$ in the NMPC formulation.

5.3 Locomotion-Enhancing Features

In this section we discuss the main distinctive features of our approach, which we found relevant to improve locomotion ability of our quadruped robot. These features are *mobility*, *force robustness* and *ZMP margin*.

5.3.1 Mobility and Mobility Factor

Terrain adaptability is vital when it comes to locomotion of the legged robots. Adjusting the posture of the robot depending on the environment is important for safe locomotion. A way to enable our NMPC to choose robot orientation adaptively to any terrain is to employ the concept of *mobility* Focchi et al. (2017). In order to rigorously discuss mobility in mathematical terms, we first define it in words as the attitude of a manipulator (leg) to arbitrarily change end-effector position/orientation Sciavicco et al. (2000).

In order to penalize low leg mobility in the cost function (5.2c) we need to compute the reference value of hip-to-foot distance ${}_{\mathcal{C}}\mathbf{p}^{\mathrm{ref}}_{\mathrm{hf}_k}$. Our goal in this section is to define a convenient metric to represent mobility and compute ${}_{\mathcal{C}}\mathbf{p}^{\mathrm{ref}}_{\mathrm{hf}_k}$ corresponding to the maximum value of such a metric. Among several ways to compute mobility Focchi et al. (2017), the velocity transformation ratio Yoshikawa (1984) allows one to evaluate mobility in a particular direction. However, the velocity transformation ratio cannot be used in our setting because it requires prior knowledge of the evolution of the relative foot position with respect to CoM. In our case it is not available in advance because it is an output of the NMPC.

As an alternative approach, we consider the volume of the *manipulability ellipsoid* $\big(\mathbf{v}(\mathbf{J}\mathbf{J}^\top)^{-1}\mathbf{v} = 1\big)$ Sciavicco et al. (2000) as a metric to evaluate mobility. A change in the volume of the manipulability ellipsoid with different leg configurations is visualized in Fig. 24 (left). Inspecting Fig. 24 (top right), it can be seen that the maximum volume V is in the vicinity of the most extended leg configuration because the mobility becomes very big in the X and Y direction, even if it is still very limited in the Z direction. However, because it is desirable to achieve a good mobility in all the directions of the leg configuration, a better metric is the one that also accounts for the *isotropy* of the manipulability ellipsoid. A measure of the isotropy of an ellipsoid can be expressed as the inverse of its *eccentricity* E. Hence, a new manipulability index that we call *mobility factor* (5.7) can be defined in terms of both the eccentricity and the volume of manipulability ellipsoid. Again from Fig. 24, it can be visualized that

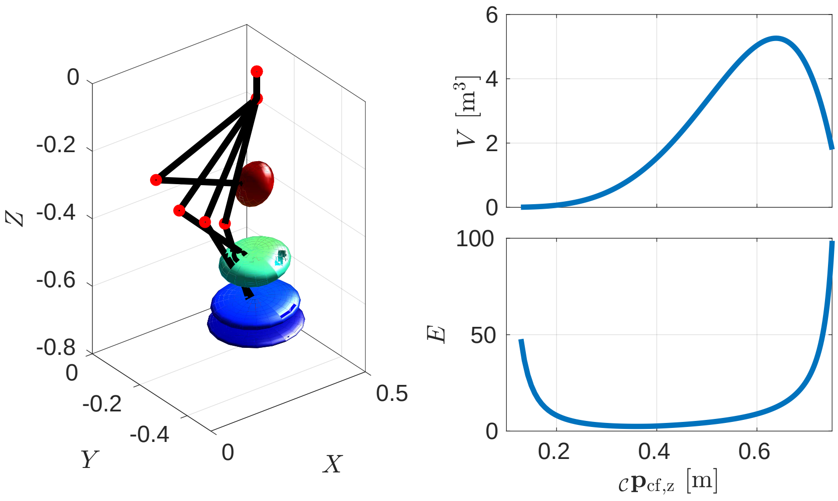


Figure 24: Manipulability ellipsoid changing with leg configuration (left) of the right front leg. Volume of the ellipsoid (top right) and Eccentricity of the ellipsoid (bottom right).

to keep a good mobility (left plot) in all directions, the volume (top right plot) should be maximized while the eccentricity (bottom right plot) as small as possible. Defining a foot Jacobian $\mathbf{J}(\mathbf{q}) \in \mathbb{R}^{3\times 3}$ computed at a particular joint configuration $\mathbf{q}$, the volume V of a manipulability ellipsoid is evaluated as a product of the eigenvalues of $\left(\mathbf{J}(\mathbf{q})\mathbf{J}(\mathbf{q})^{\top}\right)^{-1}$ while the eccentricity is the ratio between its maximum and minimum eigenvalue Focchi et al. (2017). First, the volume and eccentricity of manipulability ellipsoid are normalized by their ranges $\bar{V}$ and $\bar{E}$. Then we define the mobility factor as:

$$m_{\mathrm{f}} = \beta\frac{V}{\bar{V}} - \gamma\frac{E}{\bar{E}} \tag{5.7}$$

The minus sign in (5.7) represents conflicting contributions of the V and E in the definition of the mobility factor (i.e. the goal is to achieve high volume and low eccentricity). Parameters β and γ are introduced to find a best trade-off between volume and eccentricity while deciding a mobility factor.

The mobility factor is a convex nonlinear function $m_{\mathrm{f}} : \mathbb{R}^3 \rightarrow \mathbb{R}$ that can be numerically evaluated inside the workspace of each leg. By se-

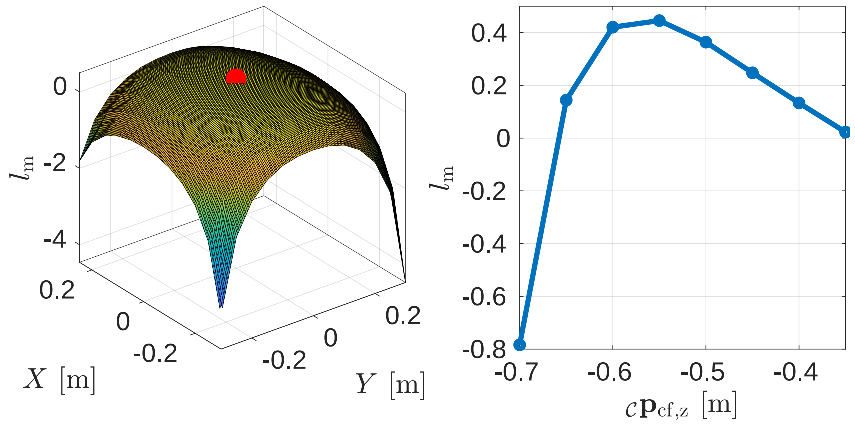


Figure 25: Slices of the mobility factor function for the RF leg: the left figure plots it against the X-Y components keeping Z constant. The red dot in the left plot represents the maxima. The right figure plots it against the Z component for a constant X-Y foot position.

lecting $\beta = 1$ and $\gamma = 4$, and after conducting a numerical analysis for all the feet positions in the workspace of a leg of the HyQ robot we found that hip-to-foot distance ${}_{\mathcal{C}}\mathbf{p}_{\mathrm{hf}} = (0, 0, -0.55)\,\mathrm{m}$ maximizes m_{f}. In Fig. 25 (left) we show a slice of the scalar function m_{f} in the X-Y plane for ${}_{\mathcal{C}}\mathbf{p}_{\mathrm{hf,z}} = -0.55\,\mathrm{m}$ obtained for the RF leg. Instead, in Fig. 25 (right) we plot m_{f} against the change of foot position in the Z direction considering the hip under the foot ($X = 0$, $Y = 0$) which clearly highlights ${}_{\mathcal{C}}\mathbf{p}_{\mathrm{hf,z}} = -0.55\,\mathrm{m}$ corresponding to the maximum value of the mobility factor m_{f} (i.e., around 0.41). We use the output of this analysis as a reference for the hip-to-foot distance ${}_{\mathcal{C}}\mathbf{p}_{\mathrm{hf}}^{\mathrm{ref}}$ in the mobility cost (5.2c) for all the legs.

In the mobility cost, we multiply δ_i to the term corresponding to the i^{th} leg. Thus, the mobility cost solely accounts for stance legs because the robot can only use them to control its base orientation. Since, including the mobility cost enables NMPC to provide the optimal base orientation for a particular locomotion that retains mobility, there is no need to separately specify tracking cost for roll and pitch in the NMPC. This relieves a user from the burden of implementing a customized heuristic (e.g., to align the robot base to the terrain), as was necessary in, e.g., Focchi et al. (2020a); Gehring et al. (2015); Villarreal et al. (2020). The relative tracking task for the CoM Z position is no longer required either, because maximizing the mobility in the Z direction automatically takes care of keeping an average distance of hips from the terrain to ${}_{\mathcal{C}}\mathbf{p}_{\mathrm{hf,z}}$, consequently keeping the robot base at a certain height.

Moreover, the yaw motion results by penalizing the mobility cost along the X-Y directions. This has the effect of driving the hips of the robot base over the feet, naturally aligning the base to the feet, similar to what was done in Raiola et al. (2020). However, a tracking cost on yaw was still necessary in the NMPC to track the heading velocity $\omega_{\mathrm{z}}^{\mathrm{usr}}$ commanded by the user and to avoid oscillations.

Remark: The concept of mobility is model independent hence, it can also be used with other models such as full body dynamics in the MPC setting.

5.3.2 ZMP Margin

In legged locomotion, the robot is often operated close to unstable configurations which require a controller to continuously compensate for model inaccuracies and external disturbances while maintaining locomotion stability. However, a configuration in which the ZMP Wieber (2006) is close to the boundary of the support polygon Bretl and Lall (2008) could cause instability even with small perturbations due to the loss of control authority.

In our case, the reference generator computes references for the GRFs by dividing the robot mass with the number of legs as explained in Section 5.4. Penalizing GRFs Z component heavily in the tracking cost (5.2b) ensures that they stay close to the reference, consequently maintaining a higher loading on the diagonally opposite leg to the swinging one, and therefore maintaining some margin for the locomotion stability. To evaluate the locomotion stability, we define the *ZMP margin* which is computed as the minimum of the distance of the ZMP from each support polygon edge, i.e.,

$$m_{\mathrm{c}} = \min(\mathbf{d}) \tag{5.8}$$

where $\mathbf{d}$ is a vector of the distances of ZMP projection (on a horizontal plane) from the support polygon edges.

5.3.3 Force Robustness

Similar to the considerations on mobility, in order to effectively compensate for disturbances acting on the system, robustness in the GRFs is required. The closer the GRF is to the friction cone boundary, the less lateral force is available to compensate for perturbations. An approach penalizing GRFs that are in the vicinity of the cone boundaries has been proposed in Focchi et al. (2016); Fahmi et al. (2020) inside the WBC. These WBC based approaches instantaneously generate GRFs that are as close as possible to the normals of the cones while yielding the prescribed resultant wrench on the robot base. However, WBC does not account for the future state of the robot and hence, it leaves some room for the NMPC to compensate for the contact normal estimation error and recover from external disturbances. Introducing these margins on GRFs from the cone boundaries is especially important in some scenarios, such as the one reported in simulation in Section 5.7.2.

In this paper, we adopt a similar idea to Focchi et al. (2016); Fahmi et al. (2020) and introduce the additional cost term (5.2d) in the NMPC, which penalizes the tangential components of GRFs in the contact frame $\mathcal{K}$ (see Fig. 23) to obtain the resultant GRFs as close as possible to the contact normals. The weight matrix $\mathbf{P}$ used in this cost is defined in Table 5 (Chapter 5.7). Note that it is required to penalize the X-Y components higher than Z component of GRFs in the contact frame to achieve this behaviour.

5.4 Reference Generator

In our approach, the NMPC requires a reference trajectory of the state and control input along with the model parameters i.e., foot positions and contact status. For the very first run of the NMPC, this reference trajectory also serves as an initial guess. The references are generated for the length of control intervals N, since the reference generator is called before every iteration of the NMPC in order to obtain prompt adaptation to terrain changes and user set-point. Our reference generator is based on heuristics and it takes as inputs:

- the user commanded longitudinal and lateral CoM velocity ${}_{\mathcal{H}}\mathbf{v}_{\mathrm{c}}^{\mathrm{usr}} \in \mathbb{R}^2$ in the horizontal frame $\mathcal{H}$,[1]

[1]The horizontal frame is placed like the CoM frame but with the Z-axis aligned with the gravity

- user commanded heading velocity $\omega_{\mathrm{z}}^{\mathrm{usr}} \in \mathbb{R}$,
- current pose of the robot $(\mathbf{p}_{\mathrm{c}}, \mathbf{\Phi})$,
- current feet positions $\mathbf{p}_{\mathrm{f}} \in \mathbb{R}^{12}$,
- heightmap of the terrain

The reference generator outputs:

- the references for the NMPC cost: states $\mathbf{x}^{\mathrm{ref}} \in \mathbb{R}^{n_x \times (N+1)}$, control $\mathbf{u}^{\mathrm{ref}} \in \mathbb{R}^{n_u \times N}$,
- parameters $\mathbf{a}$ of the model: sequence of the contact status $\boldsymbol{\delta}$ ($\in \mathbb{R}^{4 \times N}$) and sequence of the foot locations $\mathbf{p}_{\mathrm{f}}$ ($\in \mathbb{R}^{12 \times N}$),
- normals of the terrain at the foothold locations, which are provided as inputs to the NMPC for the cone constraints.

First we compute the X-Y components of the total velocity $\mathbf{v}_{\mathrm{c}}^{\mathrm{ref}} \in \mathbb{R}^3$, which depend on both $\mathbf{v}_{\mathrm{c}}^{\mathrm{usr}}$ and the X-Y components of the tangential velocity due to the heading velocity[1] $\boldsymbol{\omega}^{\mathrm{usr}} = (0, 0, \omega_{\mathrm{z}}^{\mathrm{usr}})$.

$$\mathbf{v}_{\mathrm{c},(\mathrm{x},\mathrm{y})}^{\mathrm{ref}} = \mathbf{v}_{\mathrm{c}}^{\mathrm{usr}} + (\boldsymbol{\omega}^{\mathrm{usr}} \times \mathbf{p}_{\mathrm{c}}^{\mathrm{ref}})_{(\mathrm{x},\mathrm{y})} \tag{5.9}$$

The X-Y CoM position $\mathbf{p}_{\mathrm{c},(\mathrm{x},\mathrm{y})}^{\mathrm{ref}}$ is obtained by integrating the $\mathbf{v}_{\mathrm{c},(\mathrm{x},\mathrm{y})}^{\mathrm{ref}}$ (in the world frame) with the explicit Euler scheme. The references for CoM Z, roll and pitch are set to 0 because we do not track them in the NMPC cost (5.2b). Instead, the reference for the yaw ψ is obtained by integrating the user defined yaw rate $\dot{\psi}^{\mathrm{usr}}$ with $\dot{\mathbf{\Phi}}^{\mathrm{usr}} = \mathbf{E}^{-1}(\mathbf{\Phi}^{\mathrm{ref}})\boldsymbol{\omega}^{\mathrm{usr}}$ (see Appendix A for the transformation between angular velocity and Euler rates). The reference for angular velocity, instead, coincides with $\boldsymbol{\omega}^{\mathrm{usr}}$.

The references for GRFs $\mathbf{u}^{\mathrm{ref}}$ are calculated by simply dividing the total mass of the robot by the number of legs in stance. Dividing the forces equally onto the legs is correct only if the robot is static, but, in case of dynamic conditions, it is a better approximation than passing no references to the NMPC.

The sequence of contact status $\boldsymbol{\delta}$ and of footholds are computed by the *gait scheduler* and *robocentric stepping* strategy, respectively. It is important to mention that the reference generator does not compute the swing trajectories and they are obtained from the WBC interface discussed in Section 5.5.1.

[1]Note that if the $\mathbf{v}_{\mathrm{c},(\mathrm{x},\mathrm{y})}^{\mathrm{ref}}$ is computed in the CoM frame, then the contribution of heading velocity is null.

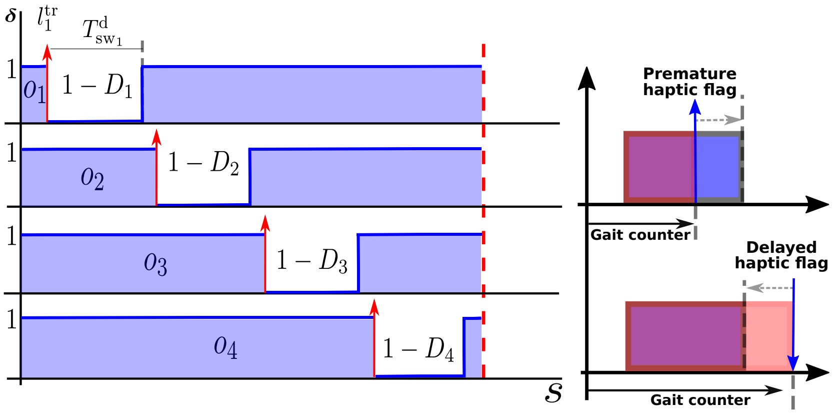


Figure 26: Gait schedule for a walk. Offsets $\mathbf{o} = [0.05, 0.3, 0.55, 0.8]$, duty-factors $\mathbf{D}_{\mathrm{f}} = [0.85, 0.85, 0.85, 0.85]$. The red arrows represent the trigger l_i^{tr} for a swing leg i. Right part shows the fast-forwarding (top) or re-winding (bottom) of the gait counter to recover synchronization between actual (haptic) and planned touchdown.

5.4.1 Gait Scheduler

The gait scheduler is logically decoupled from the reference trajectory generation and determines if a leg is either in swing or in stance (δ_i) at each time instance for the entire gait cycle as shown in Fig. 26 (left).

The leg duty factor D_i and offsets o_i can be used to encode different gaits such as crawl, trot and pace. The gait scheduler implements a time parametrization $s \in [0, 1]$ (*stride phase*) which is normalized about the cycle time duration T_{c} such that the leg duty factor D_i and offsets o_i are independent from the cycle time. Each trigger l_i^{tr} (red arrow in Fig. 26 (left)) corresponds to a new lift-off event. We can express the value of δ for leg i as:

$$\delta_i = \begin{cases} 1, & s < o_i \vee s > ((o_i + (1 - D_i)) \bmod 1) \\ 0, & \text{otherwise} \end{cases} \tag{5.10}$$

Every time the reference generator is called, it extracts N points from the gait schedule starting from an index called *gait counter*. It keeps memory of the index of the gait schedule achieved by the previous call of the reference generator. The synchronization between the first point of a contact sequence δ_{i_k} computed by the reference generator and the actual contact state of the robot avoids the reference generator to compute a zero reference force while the leg is in stance and vice-versa. In case of premature or delayed touchdown events, the synchronization is lost and the gait counter is shifted backwards or forward to re-conciliate the planned touchdown with the actual touchdown as shown in Fig. 26 (right). This is a crucial feature when dealing with rough terrains.

5.4.2 Robocentric Stepping

The choice of foothold is a key element in locomotion, since it deals with the kinematic limits of the robot. Inspired by Raibert (1986), we use an approach that continuously computes footholds consistent with the current position of the robot. To compute a foothold for a swinging leg i, we consider its hip position $\mathbf{h}_i$ instead of using the foot position at the moment of lift-off. In this way a disturbance acting on the robot or a tracking error occurred during a swing can be recovered in the following swing. For leg i, dropping the index to simplify the notation and defining the lift-off trigger as $l_k^{\mathrm{tr}} = \delta_k \wedge \overline{\delta}_{k+1}$, the foot position is computed as:

$$\mathbf{p}_{\mathrm{f}_{k+1}} = \begin{cases} \mathbf{p}_{\mathrm{f}_k}^{\mathrm{td}} & l_k^{\mathrm{tr}} = 1 \\ \mathbf{p}_{\mathrm{f}_k} & l_k^{\mathrm{tr}} = 0 \end{cases} \tag{5.11}$$

Notice that at the lift-off condition $l^{\mathrm{tr}} = 1$ at instance k, $\mathbf{p}_{\mathrm{f}}$ is set equal to the touchdown point $\mathbf{p}_{\mathrm{f}}^{\mathrm{td}}$ and it is kept constant until the next lift-off event occurs. The X-Y component of the touchdown point is given by:

$$\mathbf{p}_{\mathrm{f}_k,(\mathrm{x},\mathrm{y})}^{\mathrm{td}} = \mathbf{h}_k + \alpha T_{\mathrm{sw}}^{\mathrm{d}}(\mathbf{v}_{\mathrm{c}}^{\mathrm{usr}} + (\boldsymbol{\omega}^{\mathrm{usr}} \times \mathbf{p}_{\mathrm{bh}})_{(\mathrm{x},\mathrm{y})}) \tag{5.12}$$

The second term in (5.12) represents the step length (red arrow in Fig. 27) which is computed with respect to the hip instead of the previous foot location. Parameter α is an empirically chosen scaling factor. Parameter $T_{\mathrm{sw}}^{\mathrm{d}}$ is the default swing duration computed starting from user-defined offsets $\mathbf{o}$ and duty-factors $\mathbf{D}_{\mathrm{f}}$. The distance between hip and center of the base is denoted by $\mathbf{p}_{\mathrm{bh}} \in \mathbb{R}^3$. A 2.5D heightmap of the terrain is evaluated in correspondence of the touchdown point $\mathbf{p}_{\mathrm{f}_k,(\mathrm{x},\mathrm{y})}^{\mathrm{td}}$ to obtain $\mathbf{p}_{\mathrm{f}_k,\mathrm{z}}^{\mathrm{td}}$ that does not penetrate the terrain. If $\mathbf{p}_{\mathrm{f}}^{\mathrm{td}}$ is located near to an edge or leads to collisions (e.g., of the foot or the shin) during the step cycle, this can be harmful for the robot's balance. To prevent this from happening, the robot acquires a local heightmap in the vicinity of the touchdown

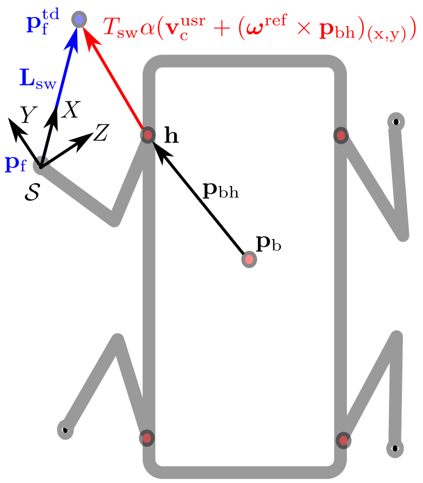

$\mathbf{p}_{\mathrm{f}}^{\mathrm{td}}$
$T_{\mathrm{sw}}\alpha(\mathbf{v}_{\mathrm{c}}^{\mathrm{usr}} + (\boldsymbol{\omega}^{\mathrm{ref}} \times \mathbf{p}_{\mathrm{bh}})_{(\mathrm{x,y})})$
$\mathbf{L}_{\mathrm{sw}}$
Y
X
Z
$\mathbf{h}$
$\mathbf{p}_{\mathrm{f}}$
$\mathcal{S}$
$\mathbf{p}_{\mathrm{bh}}$
$\mathbf{p}_{\mathrm{b}}$

Figure 27: Representation of the robocentric stepping strategy and of the Swing Frame, located at the lift-off point. The red arrow shows the distance between the touchdown point $\mathbf{p}_{\mathrm{f}}^{\mathrm{td}}$ and the hip $\mathbf{h}$. The blue vector $\mathbf{L}_{\mathrm{sw}}$ connects lift-off and touchdown point.

point $\mathbf{p}_{\mathrm{f}}^{\mathrm{td}}$ and adjusts the foot landing location using the Vision-based Foothold Adaptation (VFA) module presented in Villarreal et al. (2019), which works as follows.

5.4.3 Vision-based Foothold Adaptation

The robot acquires a local heightmap in the vicinity of the touchdown location $\mathbf{p}_{\mathrm{f}}^{\mathrm{td}}$ extracting it from the full 2.5D heightmap surrounding the robot. The local heightmap corresponds to a 15×15 grid centered at $\mathbf{p}_{\mathrm{f}}^{\mathrm{td}}$ with a resolution of $16\,\mathrm{cm}^2$. The VFA adjusts the foot landing location to avoid collisions, sharp edges, narrow gaps/holes, and footholds that are outside the workspace of the swinging leg, using a pre-trained CNN. The CNN has low computational time ($\approx 0.2\,\mathrm{ms}$ per adjustment), which allows the robot to adapt the landing location of the feet without compromising the computation of the control action provided by the NMPC. Details on how the CNN is trained can be found in Villarreal et al. (2019).

5.5 Whole-Body Controller

In this section, we describe the WBC that tracks planned trajectories $\mathbf{x}^{\mathrm{p}}$ and $\mathbf{u}^{\mathrm{p}}$ provided by the NMPC. The WBC first computes feed-forward $\mathbf{W}_{\mathrm{ff}}^{\mathrm{d}}$ and feedback $\mathbf{W}_{\mathrm{fb}}^{\mathrm{d}}$ wrenches from the planned trajectories and then the sum of these wrenches are mapped into GRFs through the QP optimization (5.16). The WBC also maps the GRFs into the joint torques $\boldsymbol{\tau}^*$. This joint torque along with low-impedance feedback torque $\boldsymbol{\tau}_{\mathrm{fb}}$ results in the total torque $\boldsymbol{\tau}_{\mathrm{d}}$ required by the low-level joint torque control block. Refer to Fig. 22 for the block representation of WBC inside our locomotion framework.

5.5.1 WBC Interface

In our planning framework, the NMPC runs at re-planning frequency of 25 Hz whereas the WBC requires state and control inputs at 250 Hz (we will call this the *WBC frequency*). Hence, we introduce a WBC interface block that re-samples state and control inputs at the WBC frequency. In particular, in order to obtain the desired $\mathbf{u}^{\mathrm{d}}$ we use a zero-order hold filter of $\mathbf{u}^{\mathrm{p}}$. The planned states $\mathbf{x}^{\mathrm{p}}$ from the NMPC, instead, are re-sampled with a linear interpolation to obtain $\mathbf{x}^{\mathrm{d}}$. [1]

Finally, the feed-forward wrench $\mathbf{W}_{\mathrm{ff}} \in \mathbb{R}^6$ is computed from the desired GRFs $\mathbf{u}^{\mathrm{d}}$ as:

$$\mathbf{W}_{\mathrm{ff}}^{\mathrm{d}} = \begin{bmatrix} \sum_{i=1}^{4} \mathbf{u}_i^{\mathrm{d}} & \sum_{i=1}^{4} \mathbf{p}_{\mathrm{cf},i}^{\mathrm{d}} \times \mathbf{u}_i^{\mathrm{d}} \end{bmatrix}^\top \tag{5.13}$$

5.5.2 Feedback Wrench

We use the approach of Focchi et al. (2016) to define desired feedback wrench obtained from a Cartesian impedance and briefly recall it next for completeness:

$$\mathbf{W}_{\mathrm{fb}}^{\mathrm{d}} = \mathbf{K} \begin{bmatrix} \mathbf{p}_{\mathrm{c}}^{\mathrm{d}} - \mathbf{p}_{\mathrm{c}} \\ \mathbf{e}({}_{\mathrm{w}}\mathbf{R}_{\mathrm{b}}^\top {}_{\mathrm{w}}\mathbf{R}_{\mathrm{d}}) \end{bmatrix} + \mathbf{D} \begin{bmatrix} \mathbf{v}_{\mathrm{c}}^{\mathrm{d}} - \mathbf{v}_{\mathrm{c}} \\ \boldsymbol{\omega}_{\mathrm{b}}^{\mathrm{d}} - \boldsymbol{\omega}_{\mathrm{b}} \end{bmatrix} \tag{5.14}$$

where ${}_{\mathrm{w}}\mathbf{R}_{\mathrm{b}}$ and ${}_{\mathrm{w}}\mathbf{R}_{\mathrm{d}} \in \mathbb{R}^{3\times 3}$ are the rotation matrices representing actual and desired orientation of the base with respect to the inertial frame,

[1]The rigorous approach is to use the model (5.3) to predict the evolution of the system in the T_{s} time interval, considering the $\mathbf{u}^{\mathrm{p}}$ coming from the NMPC, but for the motions considered in this paper the result is very similar, so a linear interpolation is a fair approximation.

respectively, $\mathbf{e}(\cdot) : \mathbb{R}^{3\times 3} \to \mathbb{R}^3$ is a mapping from a rotation matrix to the associated rotation vector. Matrices $\mathbf{K}$ and $\mathbf{D}$ are diagonal matrices containing the proportional and derivative gains and they can be interpreted as impedances.

Remark: At each re-planning instance of NMPC, the state reference is computed from the current state of the robot $\hat{\mathbf{x}}_\mathbf{0}$. Thus, at each re-planning instance the feedback term is nullified.

5.5.3 Projection of the GRFs

While the feedforward wrenches $\mathbf{W}^{\mathrm{d}}_{\mathrm{ff}}$ provided by MPC satisfy the friction cone and unilateral constraints by construction, this guarantee is lost with the addition of the feedback term $\mathbf{W}^{\mathrm{d}}_{\mathrm{fb}}$ to the wrenches. Therefore, one needs to project the total wrenches $\mathbf{W}^{\mathrm{d}}_{\mathrm{ff}} + \mathbf{W}^{\mathrm{d}}_{\mathrm{fb}}$ onto the set of wrenches that satisfy the constraints. The matrix representation

$$\underbrace{\begin{bmatrix} \delta_1 \mathbf{I} & \dots & \delta_4 \mathbf{I} \\ \delta_1 \left[\mathbf{p}_{\mathrm{cf},1} \times\right] & \dots & \delta_4 \left[\mathbf{p}_{\mathrm{cf},4} \times\right] \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} \mathbf{f}_1 \\ \vdots \\ \mathbf{f}_4 \end{bmatrix}}_{\mathbf{f}} = \underbrace{\mathbf{W}^{\mathrm{d}}_{\mathrm{ff}} + \mathbf{W}^{\mathrm{d}}_{\mathrm{fb}}}_{\mathbf{b}} \tag{5.15}$$

is derived from a simplified SRBD model Focchi et al. (2016) and allows us to map the desired wrenches into GRFs . To compute the desired GRFs $\mathbf{f}^{\mathrm{d}}$ we solve the following QP:

$$\mathbf{f}^{\mathrm{d}} = \underset{\mathbf{f}}{\operatorname{argmin}} \;\; \| \mathbf{A}\mathbf{f} - \mathbf{b} \|^2_{\mathbf{S}_{\mathrm{w}}} + \| \mathbf{f} - \mathbf{u}^{\mathrm{d}} \|^2_{\mathbf{T}_{\mathrm{w}}} \tag{5.16a}$$

$$\text{s.t.} \quad \underline{\mathbf{d}} \leq \mathbf{C}\mathbf{f} \leq \overline{\mathbf{d}} \tag{5.16b}$$

The term $\| \mathbf{f} - \mathbf{u}^{\mathrm{d}} \|^2_{\mathbf{T}_{\mathrm{w}}}$ in the cost (5.16) allows the tracking of the desired forces $\mathbf{u}^{\mathrm{d}}$ received from the NMPC. Matrices $\mathbf{S}_{\mathrm{w}} \in \mathbb{S}^6_+$ and $\mathbf{T}_{\mathrm{w}} \in \mathbb{S}^{12}_+$ are positive-definite weight matrices. Inequality (5.16b) encodes the friction cone and unilateral constraints similar to (4.5) for which further details can be found in Focchi et al. (2016). It is important to note that gravity compensation is already incorporated in the NMPC formulation through the SRBD model.

5.5.4 Mapping GRFs to Joint Torques

The GRFs $\mathbf{f}^{\mathrm{d}}$ must be mapped into joint torques $\boldsymbol{\tau}^*$. We do so by exploiting the joint dynamics:

$$\boldsymbol{\tau}^* = -\mathbf{J}(\mathbf{q})^\top \mathbf{f}^{\mathrm{d}} + \mathbf{h}(\mathbf{q}, \dot{\mathbf{q}}) \tag{5.17}$$

where $\mathbf{J}(\mathbf{q}) \in \mathbb{R}^{n_u \times n}$ is the contact Jacobian and $\mathbf{h}(\mathbf{q}, \dot{\mathbf{q}})$ the vector of gravity/Coriolis terms in the leg joint dynamics. The number of joints is denoted by n. We neglect the joint acceleration contribution, because it is very small with respect to the other terms.

5.5.5 Joint-Space PD

A $1\,\mathrm{kHz}$ Joint-Space PD is put in cascade with the WBC before sending torques to the low-level controller. In this way, we track the desired trajectories of the swinging legs and we increase the robustness in case a foot loses contact with the ground. The WBC interface provides the joint trajectories $\mathbf{q}^{\mathrm{d}}$ and $\dot{\mathbf{q}}^{\mathrm{d}}$ required by the Joint-Space PD. To compute the joint trajectories, inverse kinematics is required which in turn needs the swing trajectory $\mathbf{p}_{\mathrm{f}}^{\mathrm{sw}}$. We define the swing frame $\mathcal{S}$ Raiola et al. (2020) (Fig. 27), whose X-axis is aligned with the vector that links lift-off and touchdown point ($\mathbf{L}_{\mathrm{sw}}$), Y-axis is perpendicular to the X-axis of the swing frame and to the Z-axis of the world frame. Finally the Z-axis is such that $\mathcal{S}$ is a counter-clockwise coordinate system. The origin of the swing frame $\mathcal{S}$ coincides with the lift-off point. In this way the swing trajectory lies on the X-Z plane and we shape it as a semi-ellipse with $\mathbf{L}_{\mathrm{sw}}$ and H_{sw} as lengths of the axes:

$$_{\mathcal{S}}\mathbf{p}_f^{\mathrm{sw}} = \begin{bmatrix} \frac{\mathbf{L}_{\mathrm{sw}}}{2}(1-\cos(\pi f_{\mathrm{sw}} t_{\mathrm{sw}})) \\ 0.0 \\ H_{\mathrm{sw}} \sin(\pi f_{\mathrm{sw}} t_{\mathrm{sw}}) \end{bmatrix} \tag{5.18}$$

where t_{sw} is the time elapsed from the beginning of a swing and $f_{\mathrm{s}} = 1/T_{\mathrm{sw}}^{\mathrm{d}}$ is the swing frequency. We map $_{\mathcal{S}}\mathbf{p}_{\mathrm{f}}^{\mathrm{sw}}$and its derivative in the inertial frame $\mathcal{W}$ to obtain $\mathbf{p}_{\mathrm{f}}^{\mathrm{sw}}$ and $\dot{\mathbf{p}}_{\mathrm{f}}^{\mathrm{sw}}$, respectively. Finally, after evaluating the relative foot position $_{\mathcal{C}}\mathbf{p}_{\mathrm{cf}}$ and velocity $_{\mathcal{C}}\dot{\mathbf{p}}_{\mathrm{cf}}$ we can obtain $\mathbf{q}^{\mathrm{d}}$ and $\dot{\mathbf{q}}^{\mathrm{d}}$ via inverse kinematics.

5.6 Real-Time Iteration for NMPC

One of the main drawbacks of NMPC is its computational burden, thus efficient tailored algorithms are necessary in order to achieve fast sampling rates for complex systems with fast dynamics. While many approaches have been developed for optimal control, a complete discussion about all possible approaches is beyond the scope of this paper. We focus on direct multiple shooting methods derived from SQP that have been specifically developed for real-time NMPC Diehl et al. (2005a).

In multiple shooting methods both state $\mathbf{x}$ and control input $\mathbf{u}$ are decision variables unlike in single shooting where the decision vector only includes the control input. We ought to stress that this does not increase the computational complexity with respect to single shooting (where computations are moved from linear algebra to the evaluation of derivatives). Furthermore, multiple-shooting allows one to provide an initial guess also for the state trajectory, which is typically beneficial for unstable systems in an NMPC context Diehl et al. (2005a).

While in SQP one solves several QPs until convergence is reached, the RTI scheme consists in solving a single QP per sampling time. This is motivated by the observation that in NMPC two subsequent problems have very similar solutions. Therefore, by reusing the solution of the previous NMPC problem, one obtains a very good initial guess for the next problem, which essentially only needs to correct for external perturbations and model mismatch. For all details on the RTI scheme, we refer to Diehl et al. (2005b); Gros et al. (2020) and references therein. We limit ourselves to observe that, since the linearization trajectory $(\mathbf{x}_k^{\mathrm{L}}, \mathbf{u}_k^{\mathrm{L}})$ is known *before* the next state measurement is available, one can already evaluate the functions and their derivatives (4.7) before the initial state $\hat{\mathbf{x}}_0$ is available. Consequently, the QP can be constructed and prepared beforehand; note that this also includes the first factorization of the QP Hessian. Once $\hat{\mathbf{x}}_0$ is available, one only has to finish solving the QP. Therefore, while the overall sampling time must still be long enough to prepare the next QP, the latency between the time at which $\hat{\mathbf{x}}_0$ is available and the time at which the control input can be applied to the system is very small. Algorithm 3 covers the steps involved in the RTI scheme.

Note that in the RTI scheme proposed above, the functions and their derivatives (4.7) are evaluated along a guess obtained from the previous solution, rather than along the reference trajectory. Another important aspect to highlight is the fact that there exist several approaches to compute (4.7). One choice consists of first linearizing the continuous-

Algorithm 3 RTI for NMPC

```
   Input: reference trajectory $\mathbf{x}^{\text{ref}}, \mathbf{u}^{\text{ref}}$ and parameter $\mathbf{a}$
   Preparation phase:
1: if FirstIteration then
2:     Set initial guess to reference trajectory
         $(\mathbf{x}^{\text{L}}, \mathbf{u}^{\text{L}}) \leftarrow (\mathbf{x}^{\text{ref}}, \mathbf{u}^{\text{ref}})$
3: else
4:     Set initial guess to shifted $(\mathbf{x}^{\text{L}}, \mathbf{u}^{\text{L}})$
         $\mathbf{x}^{\text{L}}(k) \leftarrow \mathbf{x}(k+1)\ \forall k = 0, 1, \cdots, N-1$
         $\mathbf{x}^{\text{L}}(N) \leftarrow \mathbf{x}(N-1)$
         $\mathbf{u}^{\text{L}}(k) \leftarrow \mathbf{u}(k+1)\ \forall k = 0, 1, \cdots, N-2$
         $\mathbf{u}^{\text{L}}(N-1) \leftarrow \mathbf{u}(N-2)$
5: end if
6: Evaluate $\mathbf{r}_k, \mathbf{h}_k$ and the sensitivities $\mathbf{A}_k, \mathbf{B}_k, \mathbf{C}_k, \mathbf{D}_k, \mathbf{H}_k$, $\mathbf{J}_k$ at
   $(\mathbf{x}^{\text{L}}, \mathbf{u}^{\text{L}}, \mathbf{a})$ according to (4.7)
7: Formulate QP (4.6) omitting $\hat{\mathbf{x}}_0$ and prepare all possible computa-
   tions
   Feedback phase:
8: Read and pass initial state $\hat{\mathbf{x}}_0$ to QP (4.6) to compute $(\Delta\mathbf{x}, \Delta\mathbf{u})$
9: Take the full Newton step
         $(\mathbf{x}^{\text{L}}, \mathbf{u}^{\text{L}}) \leftarrow (\mathbf{x}^{\text{L}}, \mathbf{u}^{\text{L}}) + (\Delta\mathbf{x}, \Delta\mathbf{u})$
   return NMPC solution $(\mathbf{x}^{\text{P}}, \mathbf{u}^{\text{P}}) = (\mathbf{x}^{\text{L}}, \mathbf{u}^{\text{L}})$
```

time system dynamics and then using the matrix exponential to obtain a discrete-time linear system. This approach presents some advantages, but can be computationally demanding. For the time-varying and infeasible references, however, it is preferred to *first discretize and then linearize* Gros et al. (2020). In this work we deal with time-varying and infeasible references, hence we opt for first discretize and then linearize approach. An advantage of this apporach is that after numerically approximating the discrete-time dynamics, the linearization can be obtained at a desired accuracy.

A very popular way to obtain discret-time dynamics is with the explicit Euler integrator, which is computationally inexpensive, but can be inaccurate and unstable. Therefore, it is usually more efficient to resort to higher-order integration schemes, such as, e.g., the popular Runge-Kutta methods. Finally, we should further stress that there also exist implicit integration schemes, which require more computations per step, but they are typically much more stable and accurate than explicit schemes for some classes of systems. Unfortunately, the selection of the least computationally demanding integrator which delivers sufficient accuracy depends on the problem setting and typically requires some trial-and error approach, which can be educated using some guidelines based on the theoretical properties of each integrator Quirynen (2017); Quirynen et al. (2014); Hairer et al. (1993, 1996); Butcher (2003).

In this work, we relied on the RTI implementation provided by `acados` Verschueren et al. (2019), which consists of tailored efficient implementations of QP solvers, numerical integration schemes, and all other components of the RTI scheme.

5.7 Simulation and Experimental Results

In this section we discuss the implementation details and results obtained from the simulations and experiments with the NMPC scheme proposed in Chapter 5.

5.7.1 Implementation Details

To check the efficacy of our RTI based NMPC algorithm with the proposed features mentioned in Section 5.3, we performed several simulations and experiments in challenging scenarios. The simulation and experiments were performed on the HyQ robot of mass $m = 87\,\mathrm{kg}$. The

CoM is computed considering the mass of the individual link of the robot and the actual position of the links' CoM. The position of the links' CoM in their local frame is obtained from their CAD models. The feedback gains used in the WBC are $\mathbf{K} = \mathrm{diag}(1500, 1500, 1500, 100, 100, 100)$ and $\mathbf{D} = \mathrm{diag}(1000, 1000, 1000, 50, 50, 50)$. We chose the weights $\mathbf{S}_{\mathrm{w}} = \mathrm{diag}(5, 5, 10, 10, 10, 10)$ and $\mathbf{T}_{\mathrm{w}} = \mathrm{diag}(1000, \cdots, 1000)$ for the QP (5.16). The parameters and weights used by the NMPC are reported in Table 4 and 5, respectively. In all of our simulations and experiments, we do not set any weights on the CoM position ($\mathbf{p}_{\mathrm{c}}$), roll (ϕ) and pitch (θ) tasks because we wanted the NMPC to sort out these CoM quantities autonomously.

Table 4: NMPC parameters

Parameter	**Symbol**	**Value**	**Unit**
Number of state	n_x	12	-
Number of control inputs	n_u	12	-
Number of model parameters	n_a	16	-
Prediction horizon	T	2	s
Sampling time	T_{s}	0.04	s
Number of control intervals	N	50	-
Friction coefficient	μ	0.7	-
GRFs lower bound	$\underline{\mathbf{f}}_{\mathrm{z}}$	0	N
GRFs upper bound	$\overline{\mathbf{f}}_{\mathrm{z}}$	500	N
Hip-to-foot distance reference	${}_{\mathcal{C}}\mathbf{p}_{\mathrm{hf},i}^{\mathrm{ref}}$	$(0.0, 0.0, -0.55)$	m
Regularization parameter	ρ	3×10^{-5}	-

Discretization

For discretization of the dynamic constraints, we mainly investigated two integration schemes, i.e, a single step of the explicit Euler of order 1 and implicit midpoint method of order 2 due to their low computational complexity that favors our real-time implementation needs. We chose the implicit midpoint method of order 2 because of its stability and accuracy properties. The sampling time $T_{\mathrm{s}} = 40\,\mathrm{ms}$ was chosen and it was sufficient to conduct NMPC computation *online* along with the other necessary computations for the re-planning.

Table 5: Weights used in the NMPC

Cost	Weight	Value
State	$\mathbf{Q}_{\mathbf{p}_c}$	diag(0, 0, 0)
	$\mathbf{Q}_{\mathbf{v}}$	diag(100, 100, 100)
	$\mathbf{Q}_{\mathbf{\Phi}}$	diag(0, 0, 100)
	$\mathbf{Q}_{\boldsymbol{\omega}}$	diag(100, 100, 1000)
Force	$\mathbf{R}_{\mathrm{x}}$	1×10^{-3}
	$\mathbf{R}_{\mathrm{y}}$	1×10^{-3}
	$\mathbf{R}_{\mathrm{z}}$	8×10^{-4}
Mobility	$\mathbf{M}_{\mathrm{x}}$	1×10^{-4}
	$\mathbf{M}_{\mathrm{y}}$	2×10^{-3}
	$\mathbf{M}_{\mathrm{z}}$	1000
Force robustness	$\mathbf{P}_{\mathrm{x}}$	100
	$\mathbf{P}_{\mathrm{y}}$	100
	$\mathbf{P}_{\mathrm{z}}$	1

NMPC Software

We use the `acados` software package Verschueren et al. (2019) to implement the RTI scheme described in Section 5.6. Since `acados` comes with a Python interface allowing rapid prototyping, we first tuned the algorithm in simulation and then used the generated `C`-code to perform real experiments. We employ the QP solver High-Performance Interior Point Method (HPIPM) Frison and Diehl (2020), which exploits the sparsity structure of the MPC QP sub-problem (4.6), and supports inequality constraints.

The computation time required by the NMPC was in the range of 5-7 ms with the prediction horizon of 2 s and control intervals N equal to 50 on the on-board computer (a Quad Core Intel Pentium PC104 @ 1GHz) of HyQ for all the experiments. This computation time corresponds to the feedback phase of the RTI scheme where the QP (4.6) is solved after receiving the current state of the robot. The preparation phase of the RTI takes about 2-3 ms which is a fraction of the sampling time we chose. Refer to Section 5.6 for more details on these phases of the RTI scheme. Even though the computation time of NMPC is mostly consistent, we observed some outliers. Hence we opted for a conservative approach to run the NMPC at $25\,\mathrm{Hz}$ to guarantee that the computation time stays always less than $40\,\mathrm{ms}$. Besides the computation time of the NMPC, we also account for the time required by other blocks such as reference generator so that the total computation time does not exceed $40\,\mathrm{ms}$.

Integration with the Locomotion Framework

The NMPC is integrated in a ROS node that publishes $\mathbf{x}^{\mathrm{p}}$ and $\mathbf{u}^{\mathrm{p}}$ at a frequency of 25 Hz. The on-board computer along with our locomotion framework (WBC Interface, WBC, etc., illustrated in Fig. 22) runs a real-time node that subscribes to the topic of the NMPC ROS node. ROS is not a real-time operating system, so it can introduce quite a significant and unpredictable communication delay if the NMPC is run on an external (e.g. more powerful) computer. These delays are difficult to compensate for and they can cause a loss of synchronization between the NMPC ROS node and the WBC interface. Therefore, we decided to launch the NMPC node natively on the on-board computer to avoid communication delays between two different computers. Even though we chose not to use a more powerful dedicated off-board computer for the NMPC ROS node, we obtained a better performance in the overall implementation by avoiding the communication delays of ROS.

5.7.2 Simulations

We show our NMPC planner in action on challenging terrain starting with simulations. The main simulations are pallet crossing, walking over unstructured rough terrain and walk into a V-shaped Chimney.

Pallet Crossing

In this simulation, HyQ traverses pallets of different heights, placed at varying distances form each other. This simulation highlights the importance of including mobility in the NMPC formulation (5.2c). In particular, the simulation scenario includes a set of pallets, each one of $1\,\mathrm{m}$ length, with variable heights between 0.13 and $0.17\,\mathrm{m}$ and placed at unequal gap lengths ranging from 0.2 and $0.7\,\mathrm{m}$. We performed multiple trials commanding the robot to move forward at different velocities i.e., $0.05\,\mathrm{m/s}$ and $0.1\,\mathrm{m/s}$ to show the repeatability of our approach. To avoid stepping on undesired locations such as pallet edges and to prevent foot or shin collisions, the nominal footholds are adjusted by using the VFA (refer to Section 5.4.2). Fig. 28 shows the results of five different trials for each of the commanded velocities. The top plot shows the pitch angle θ of the robot as it traverses the scenario. Since the robot is only commanded to move along its X direction with a constant forward velocity and the foot locations are provided as known quantities to the NMPC, the adjustment in pitch is the result of minimizing the deviation from the hip-to-foot distance configuration corresponding to high mobility for all four legs. Without this feature, the robot would maintain a constant horizontal orientation (see Fig. 29) eventually reaching low mobility in some legs as shown in the attached video Rathod et al. (2021b) for a single pallet simulation.

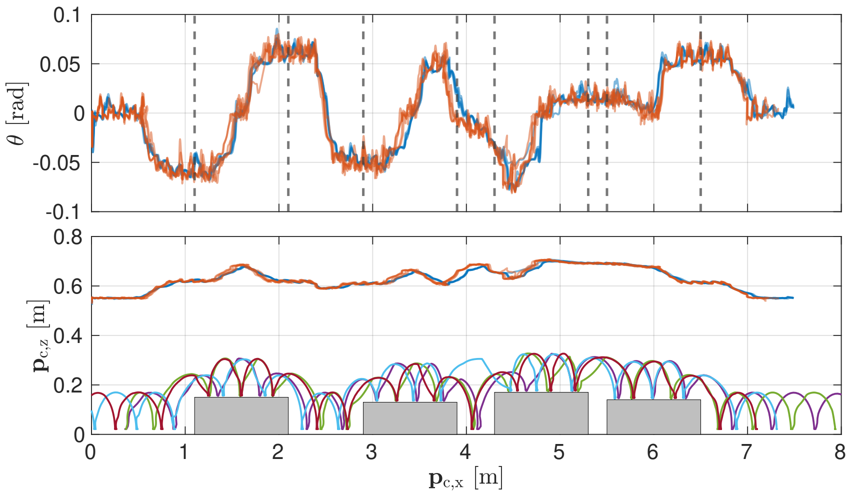


Figure 28: Simulation of pallet crossing scenario for five different trials commanding the robot to cross at $0.1\ \mathrm{m/s}$ (blue) and $0.05\ \mathrm{m/s}$ (red). The top graph shows the pitch angle and the dashed vertical lines indicate the edges of the pallets for that specific location in the plot. The bottom graph shows the CoM Z position for all the trials and the feet trajectories for one of the trials performed at $0.1\ \mathrm{m/s}$. The color of swings are related to the different legs.

We have also showed in the accompanying video Rathod et al. (2021b) the simulation of a walk on randomly generated rough terrain (using terrain generation tool by Abbyasov et al. (2020)) with the forward velocity of $0.3\ \mathrm{m/s}$ further stressing the advantages of mobility cost mentioned earlier.

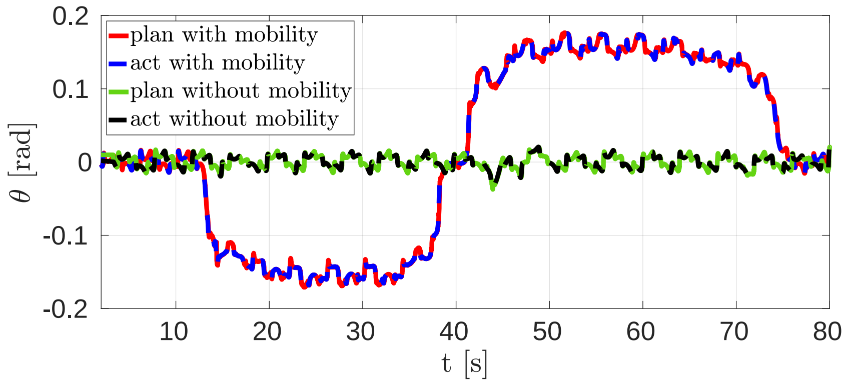


Figure 29: Comparison of the robot pitch θ in simulation with and without mobility cost in the NMPC. The red and green lines represent the planned pitch values delivered by the NMPC. The blue and black dashed lines are actual pitch of the robot. Without mobility cost, the robot maintains a horizontal orientation, whereas the robot pitches to improve leg mobility when it is included.

Walk into a V-shaped Chimney

In this simulation, we show HyQ walking at $0.03\,\mathrm{m/s}$ commanded velocity in the X direction into a V-shaped chimney with friction coefficient $\mu = 0.7$ and walls inclined at 35° to the ground. This simulation exploits the cone constraints and force robustness cost defined inside the NMPC formulation that is vital for the success of this task. The robot receives an online update of the map of the environment through an on-board camera to get the information about the normals at the location of the contact. These normals are used to formulate the force robustness cost (5.2d) in the contact frame $\mathcal{K}$. With this cost, the NMPC provides optimal GRFs to stay close to the normals of the friction cones at the contacts.

As shown in the accompanying video Rathod et al. (2021b), without the cone constraints the robot slips while climbing the chimney and ultimately falls. When the force robustness cost is enabled, the forces are regularized to stay in the middle of the cones, thanks to the robustness feature described in Section 5.3.3. In this case the robot walks successfully into the chimney. In Fig. 30, it can be seen that the longitudinal and lateral components of the GRF at the LF foot stay within the bound μf_z (in red) imposed by the cone constraints. Moreover, Fig. 31 plots the normal versus the tangential force of the GRF together with the cone bound μf_z (red line). The picture shows that the GRF stays well within the bound without any violation. Therefore, including the force regularization term enables the NMPC to account for the estimation error in the orientation of contact normals and increase robustness to the external disturbances.

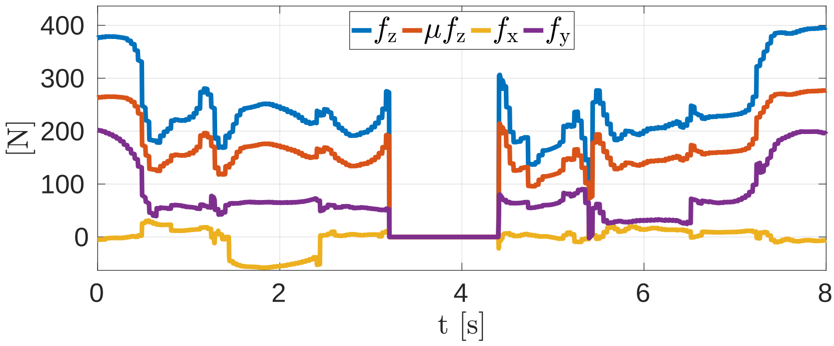


Figure 30: Walk into a V-shaped chimney simulation: GRFs of LF leg for a single gait cycle with cone constraints and regularization cost. Both the longitudinal f_x and lateral f_y lie conservatively within the bound μf_z imposed by cone constraints.

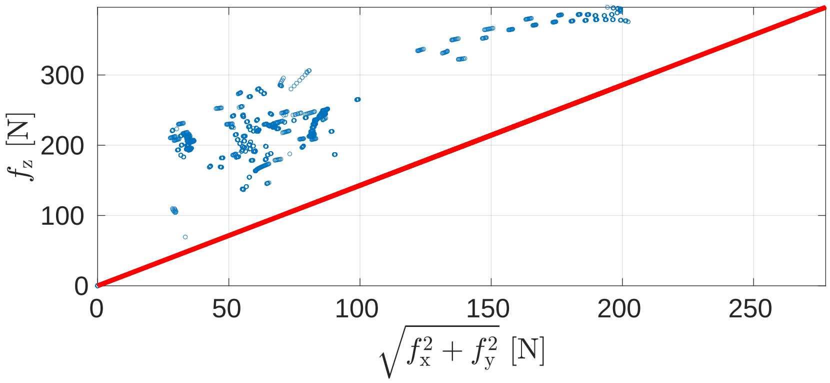


Figure 31: Walk into a V-shaped chimney simulation: Normal force f_z versus tangential force $\sqrt{f_x^2 + f_y^2}$ of the LF leg for a single gait cycle expressed in the contact frame. The red line is the cone bound μf_z.

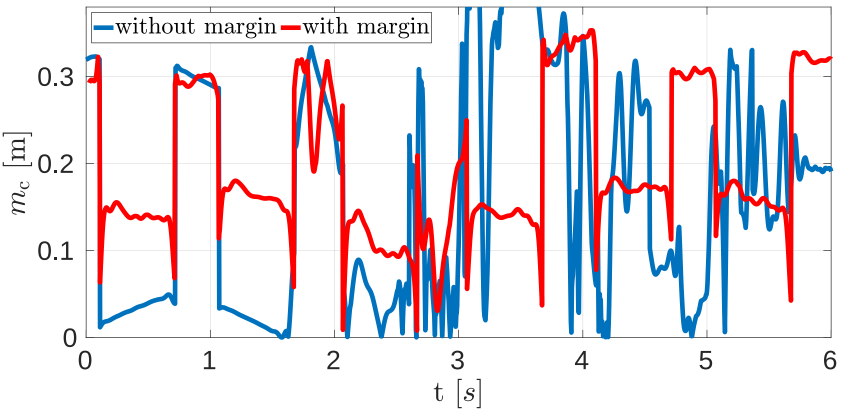


Figure 32: Plot of the ZMP margin m_c used to measure locomotion stability. The robot is pushed immediately after $2\,\mathrm{s}$ with lateral force of $200\,\mathrm{N}$ while walking on a flat terrain at $0.1\,\mathrm{m/s}$ CoM X velocity. The discontinuities are due to the switching between $3/4$ stance legs in a crawl gait.

ZMP Margin Simulation

Apart from the simulation mentioned above, we also have added in the attached video Rathod et al. (2021b), the simulations regarding the ZMP margin (refer to Section 5.3.2) and the importance of the re-planning at a higher rate. For the ZMP margin simulation, the robot is pushed with $200\,\mathrm{N}$ of lateral force for $1\,\mathrm{s}$ both in case of sufficient (higher weight on GRFs Z) and no (lower weight on GRFs Z) ZMP margin. The ZMP margin plots for this simulation can be seen in Fig. 32 where the ZMP margin is improved in case of the red line compared to the blue one because the GRFs Z components are penalized relatively more $(100$ times) for the red line. Because of the improved margin the robot walks stably, whereas it falls while walking when there is no margin.

Re-planning Simulation

In the second simulation, the robot is commanded with a constant CoM X velocity and heading velocity simultaneously. In case of re-planning at lower rate of $0.8\,\mathrm{Hz}$, the robot becomes unstable and falls due to increase in the model uncertainties and tracking errors. We would like to stress that when the re-planning is done at a lower frequency than $25\,\mathrm{Hz}$, the robot is in open-loop for the time interval between two consecutive re-planning instances, hence, it is no more NMPC but an online open-loop trajectory optimization. On the other hand, at a higher re-planning frequency of $25\,\mathrm{Hz}$ the robot walks successfully because the NMPC compensates for the model uncertainties and tracking errors.

5.7.3 Experiments

We performed three different experiments to demonstrate the real-time implementation of our NMPC running on the on-board computer of the robot as follows.

Omni-directional Walk

With this experiment, we show the omni-directional walk performed by HyQ with the NMPC on a flat terrain. This experiment validates that the NMPC computes feasible trajectories after receiving different velocity commands from the user while walking. In this experiment, the robot is commanded with a longitudinal velocity ${}_{\mathcal{H}}\mathbf{v}^{\mathrm{usr}}_{\mathrm{c,x}}$ by the user to walk forward/backward and then a lateral velocity ${}_{\mathcal{H}}\mathbf{v}^{\mathrm{usr}}_{\mathrm{c,y}}$. Finally, a heading velocity $\omega^{\mathrm{usr}}_{\mathrm{z}}$ is commanded to turn in the left/right direction. Fig. 33 shows the CoM X-Y position and yaw angle of the robot base and it can be noticed that the actual values track very closely the planned trajectories provided by the NMPC. Fig. 34 depicts the deviation of the actual velocities from the reference values while following the planned trajectories form NMPC. It can be seen in Fig. 35 that the GRFs generated by the WBC are compliant with the planned values $\mathbf{u}^{\mathrm{p}}$ and again the actual values of GRFs track closely the planned values. From these plots, it can be observed that the continuous re-planning with NMPC plays an important role to achieve good tracking of the planned trajectory.

Traversing a Static Pallet

The purpose of this experiment is to demonstrate that the mobility cost (5.2c) incorporated in the NMPC formulation provides the necessary body pitch for the robot to traverse over a static pallet (refer Fig. 37) while maintaining good leg mobility. The pallet used in this experiment is $0.13\,\mathrm{m}$ in height and $0.8\,\mathrm{m}$ in length. Fig. 37 shows that the robot pitches up while climbing up the pallet and pitches down consequently while climbing down from the pallet. As shown in the attached video Rathod

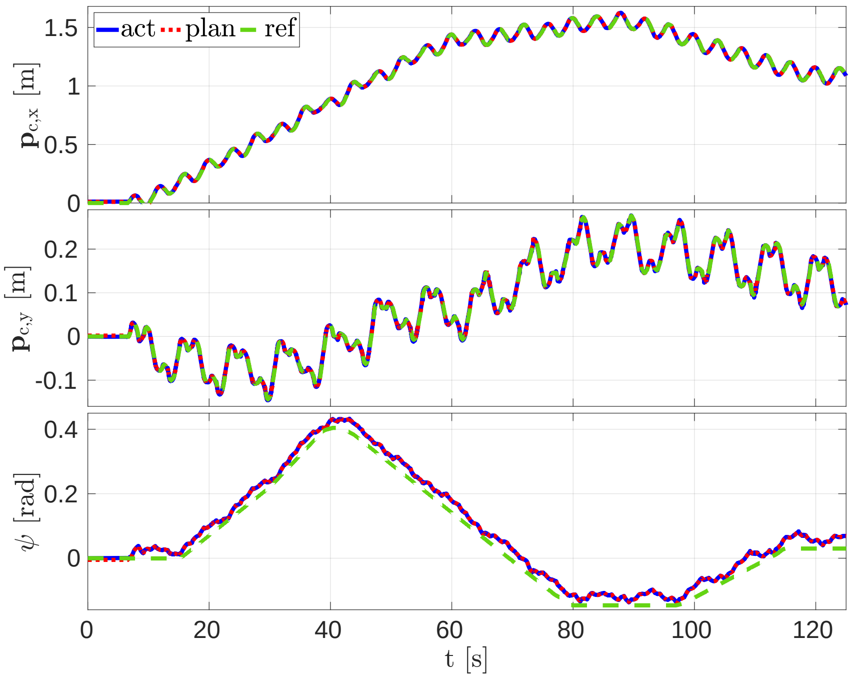


Figure 33: CoM X-Y position and yaw ψ in omni-directional walk experiment. The blue, dotted red and dashed green line represent the actual, planned and reference values, respectively.

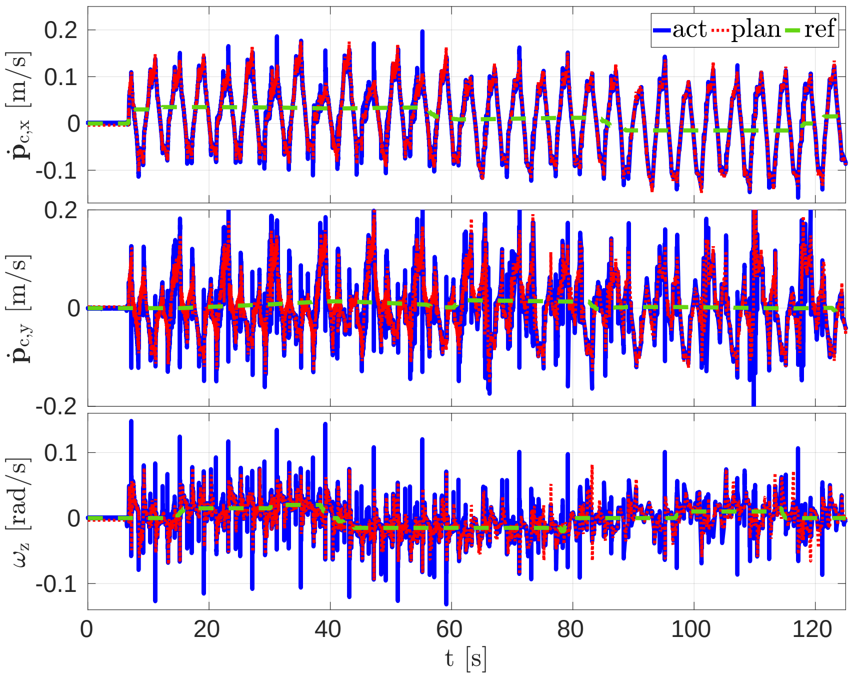


Figure 34: The longitudinal $\dot{\mathbf{p}}_{c,x}$, lateral $\dot{\mathbf{p}}_{c,y}$ and angular $\boldsymbol{\omega}_z$ velocity of the robot in omni-directional walk experiment. The blue, dotted red and the green line represent the actual, planned and reference values, respectively.

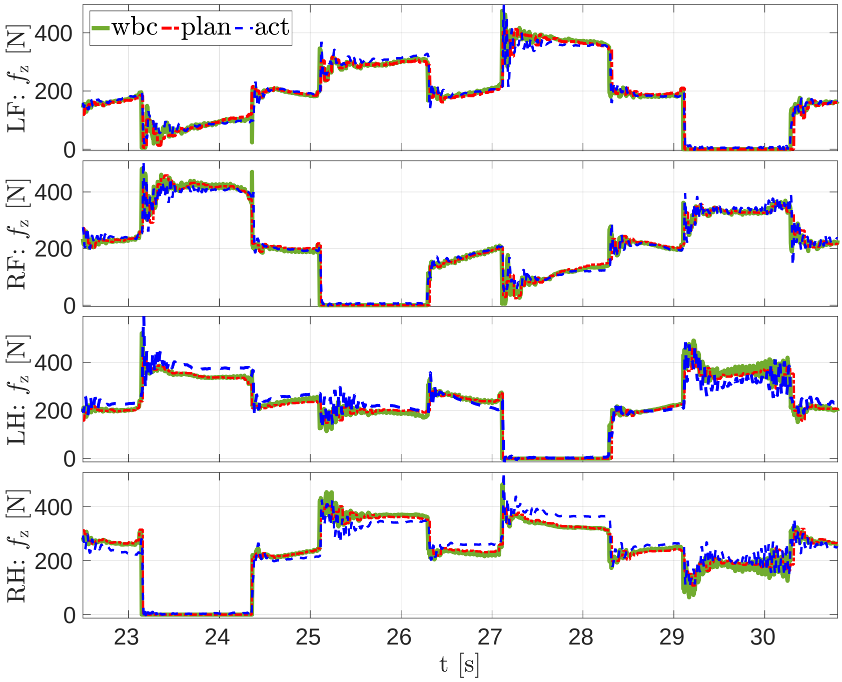


Figure 35: GRFs from one gait cycle in the omni-directional walk experiment (We show only one cycle for better visibility of the data). The green, dotted red and dashed blue line represent the output from WBC i.e., $f^{\mathrm{d}}_{\mathrm{z},i}$, planned and reference values, respectively.

et al. (2021b) in the simulation, the NMPC maintains the horizontal base orientation when the mobility cost is deactivated. This causes a reduced hip-to-foot distance while stepping up/down on the pallet ultimately resulting in low leg mobility. When mobility cost is activated, it directs the NMPC solution to achieve the necessary pitch that allows to maintain the hip-to-foot distance at the reference value and hence the leg mobility is improved. Moreover, the VFA provides the corrected foot position (i.e., to avoid shin or feet collisions with the edges of the pallet) to the NMPC and this further enhances the overall locomotion.

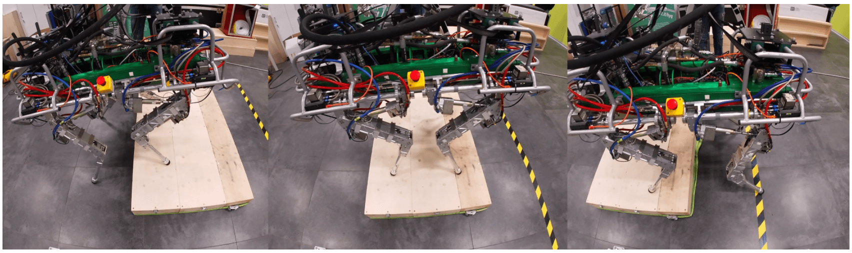


Figure 36: IIT's quadruped robot HyQ traversing a static pallet with the mobility enhanced real-time NMPC.

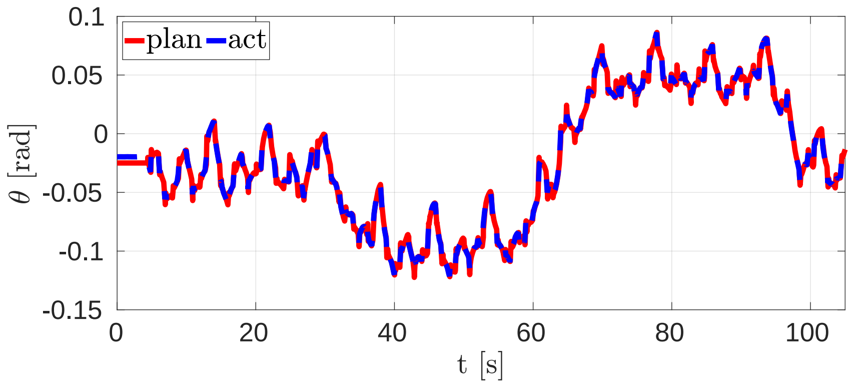


Figure 37: Planned (red) and Actual (dashed blue) pitch of the robot base while traversing a static pallet in the experiment at a commanded CoM X velocity of $0.03\,\mathrm{m/s}$.

Traversing a Repositioned Pallet

In this experiment we test our NMPC to plan the robot motion in real-time by adapting the changes in the environment with the help of VFA. As it can be seen from the attached video Rathod et al. (2021b), when the pallet ($0.13\,\mathrm{m}$ in height and $0.8\,\mathrm{m}$ in length) is pushed in front of HyQ while walking, the heightmap detects the pallet and the VFA provides updated foot locations to the NMPC. The NMPC after receiving these updated foot locations delivers a solution by pitching up the robot base in order to adapt to the change in the environment while maintaining the mobility. Even though the mobility cost is defined for the stance legs, it is interesting to notice that the NMPC decides to adjust the base pitch while swinging the RF leg onto the pallet (see Fig. 38) by forecasting the change in hip-to-foot distance at the touchdown. This experiment highlights the advantage of the predictive control over the traditional control apporaches for its ability to incorporate the knowledge of the future states. It also validates the effectiveness of our mobility cost in the NMPC coupled with the VFA to adapt to the changes in the locomotion environment.

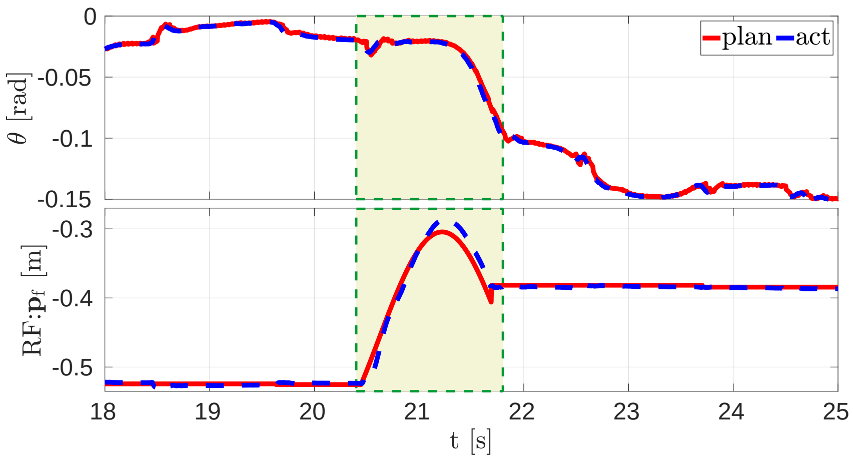


Figure 38: Robot base pitch achieved during the swing of RF leg while traversing a repositioned pallet in the experiment a commanded CoM X velocity of $0.05\,\mathrm{m/s}$. The red and dashed blue line are planned and actual values.

5.8 Summary

In this chapter, we have introduced our locomotion framework with the NMPC. Then, we discussed in detail our NMPC formulation and its ingredients. We proposed SRBD model with contact status as parameter that allowed us to eliminate unnecessary complementarity constraints in the NMPC formulation. Further, we described our features inside NMPC formulation namely, 1) mobility and mobility factor 2) ZMP margin 3) force robustness, to enhance environment adaptation during a locomotion in challenging scenarios. After that, we presented a tailored reference generator for the NMPC, and described the WBC and RTI scheme for the NMPC. Finally, we have demonstrated the effectiveness our NMPC in several simulations and experiments for a crawl gait on HyQ robot.

Chapter 6

Two-Stage Optimization

This chapter discusses the two-stage optimization scheme of NMPC with a novel Optimization-Based Reference Generator (ORG) to perform trot and pace gait in simulation and experiments. First, we motivate a need for feasible references for the NMPC formulation. Then, we describe the ORG formulation that provides the physically feasible trajectory to the NMPC based on optimization techniques. Finally, we present the simulation and experimental result on the AlienGo robot with our two-stage optimization framework.

This chapter briefly explains the author's contribution to the following article. For more details, readers are encouraged to refer to the article.

A. Bratta, M. Focchi, N. Rathod, and C. Semini. Optimization-Based Reference Generator for Nonlinear Model Predictive Control of Legged Robots. *Robotics 2023*, 12(1), 6, 2023. doi: 10.3390/robotics12010006

6.1 NMPC with Optimal Reference Generator

In the previous chapter, we demonstrated the performance of our NMPC in simulations and experiments with a crawl gait. Crawl is a statically stable gait where the robot follows a regime of switching between three and four legs in stance alternatively. Because this gait is statically stable, it does not excite underactuation dynamics to a great extent at lower speeds as compared to the other gaits where a robot has two feet on the ground during a swing phase, e.g., trot and pace. The underactuation dynamics the robot faces while performing these gaits can be challenging to control, especially during locomotion involving large angular motions. Therefore, in this section, we investigate the performance of our NMPC for these other gaits. As we will discuss later in the results, the NMPC moves laterally without feasible references while trotting in a straight line. Therefore, we propose an approach where the ORG computes feasible references considering the underactuation to drive the robot to a desired target. These feasible references are passed to the NMPC, enabling the robot to reach a desired goal while trotting or pacing.

Generally, in an MPC formulation, a tracking cost is involved that requires references for the optimization variables being tracked. This is applicable in our NMPC as formulated in (5.2). These references are usually generated with heuristics, for example, as described in Section 5.4 and Di Carlo et al. (2018). These heuristically generated references might not be dynamically feasible for particular locomotion. Furthermore, the performance of NMPC relies on the references used in the cost and might be hampered when subjected to disturbances such as external push to the robot. In our case, the references also act as the linearization trajectory in the first iteration of the RTI scheme used inside our NMPC. Therefore, it is essential to generate feasible references to increase the robustness of the NMPC inside our locomotion framework.

In the first part of this chapter, we present the extension of our locomotion framework with the ORG. Further, we discuss how the optimization-based reference generator provides feasible references to the NMPC, improving its performance to compensate for the disturbances acting on the robot while executing a user-defined task. In addition, we present the results of our modified locomotion framework on an electric robot AlienGo Unitree (2022).

6.2 Locomotion Framework with the Optimal Reference Generator

The locomotion framework with our novel ORG is depicted in Fig. 39. In this framework, in place of a heuristic reference generator, the ORG is included that provides feasible/physically informed reference trajectories to the NMPC to realize a specific goal defined by the user. These references are utilized by the NMPC to provide feasible trajectories to the downstream controllers, i.e., WBC and PD joint controller. The NMPC uses a nonlinear SRBD model in its formulation. Therefore, it optimizes and outputs feasible state trajectories for the linear and angular dynamics while tracking the references provided by the ORG. In this work, the footholds are computed with the robocentric stepping (refer to 5.4.2) using the optimized velocity by the LIPOpt to obtain the consistency between the CoM position and footholds.

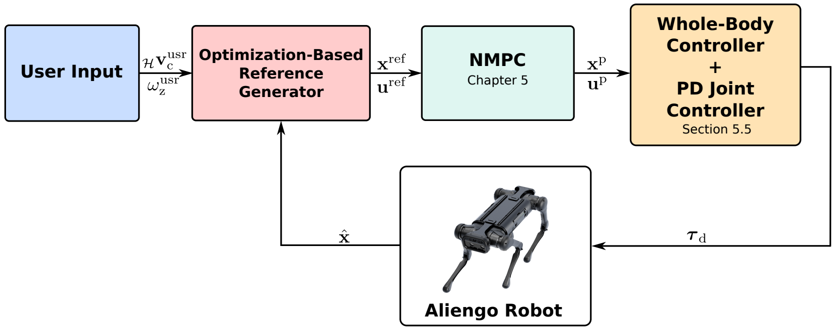

User Input
${}_{\mathcal{H}}\mathbf{v}_{\mathrm{c}}^{\mathrm{usr}}$
$\omega_{\mathrm{z}}^{\mathrm{usr}}$
Optimization-Based Reference Generator
$\mathbf{x}^{\mathrm{ref}}$
$\mathbf{u}^{\mathrm{ref}}$
NMPC
Chapter 5
$\mathbf{x}^{\mathrm{p}}$
$\mathbf{u}^{\mathrm{p}}$
Whole-Body Controller + PD Joint Controller
Section 5.5
$\boldsymbol{\tau}_{\mathrm{d}}$
$\hat{\mathbf{x}}$
Aliengo Robot

Figure 39: Overview of the locomotion framework. The optimization-based reference generator presented in this work is integrated with the NMPC presented in Chapter 5. The WBC + PD controllers described in Section 5.5 allows a robot to track the trajectories computed by the NMPC. We have used the quadruped robot AlienGo for the simulations and experiments.

The proposed ORG consists of two main components, i.e., the LIP model Optimization (LIPOpt) and QP Mapping (QPMap). The LIPOpt finds feasible values of CoM position, CoM velocity, and ZMP position by utilizing the LIP model inside an optimization problem. Using these values, the QPMap finds feasible GRFs by solving a QP problem. Finally, these values are used by the NMPC inside a tracking cost. These ORG components are briefly discussed in the following two sections.

6.2.1 LIP Model Optimization

During locomotion, a quadruped robot experiences underactuation, and the dynamics caused by the underactuation are prominent in the gaits such as trot. The ORG is designed to consider these underactuation by utilizing a LIP-based optimization to generate the references. In this formulation, the CoM trajectories are optimized and subjected to the Zero Moment Point (ZMP) constraints for a user-defined task. The ZMP is a point where the moment generated by the tangential components of the GRFs becomes zero. Imposing ZMP inside a supporting polygon while finding GRFs, satisfies unilateral constraints. This optimization problem takes the actual state of the robot $\mathbf{x}_{\mathrm{c},0}^{\mathrm{act}}$, footholds $\mathbf{p}_{\mathrm{f}}$, and contact status $\boldsymbol{\delta}$ as inputs. It is represented by,

$$\mathbf{x}_{\mathrm{c}}^{\mathrm{g}}, \mathbf{w}^{\mathrm{g}} = \operatorname{argmin} \mathrm{LIPOpt}(\mathbf{x}_{\mathrm{c},0}^{\mathrm{act}}, \mathbf{p}_{\mathrm{f}}, \boldsymbol{\delta}) \tag{6.1}$$

where $\mathbf{x}_{\mathrm{c}}^{\mathrm{g}}$ includes the X-Y components of the CoM position and linear velocity. Variable $\mathbf{w}^{\mathrm{g}}$ denotes the ZMP position, which is an input to the LIP model. The formulation is detailed in Bratta et al. (2023a). After obtaining the feasible trajectories from LIPOpt, we will explain how they are used to find the feasible GRFs in the following section.

6.2.2 QP Mapping

Apart from the robot state $\mathbf{x}$, the NMPC tracking cost also includes control inputs GRFs as optimization variables. Thus, this term inside the cost requires GRFs references. Furthermore, along with the state references, the control inputs references also act as linearization trajectories for the first iteration of the RTI scheme (refer to section 5.6). Therefore the quality of these references plays a vital role in the performance of the NMPC algorithm.

Since in the previous section, the CoM linear quantities and ZMP position are computed using the LIP model inside an optimization problem, these optimized values can further be exploited to find the feasible trajectories of the GRFs by solving the following QP,

$$\mathbf{u}^{\mathrm{qp}} = \operatorname{argmin} \mathrm{QPMap}(\mathbf{x}_{\mathrm{c}}^{\mathrm{g}}, \mathbf{w}^{\mathrm{g}}) \tag{6.2}$$

This formulation considers GRFs only for the stance legs depending on the type of gait. The advantage is that the number of optimization variables are reduced since the optimization is performed only for the legs with non-zero GRFs. This translates into the reduction of the solution time for this QP problem. The cost term in (6.2) includes a regularization term on GRFs and also a term to minimize the angular momentum rate produced by the GRFs (refer to Bratta et al. (2023a) for more details). The constraints utilized in this formulation are:

- An equation relating GRFs to the ZMP position
- Gravity compensation
- An equation to guarantee coherency between X-Y CoM acceleration computed with the LIP model and SRBD model inside NMPC

In summary, out of the state reference $\mathbf{x}^{\mathrm{ref}} = (\mathbf{p}_{\mathrm{c}}^{\mathrm{ref}}, \mathbf{v}_{\mathrm{c}}^{\mathrm{ref}}, \mathbf{\Phi}^{\mathrm{ref}}, {}_{\mathcal{C}}\boldsymbol{\omega}^{\mathrm{ref}})$, $\mathbf{p}_{\mathrm{x,y}}^{\mathrm{ref}}$ and $\mathbf{v}_{\mathrm{x,y}}^{\mathrm{ref}}$ are the output of LIPOpt, i.e., $\mathbf{x}_{\mathrm{x}}^{\mathrm{g}}$. The CoM height $\mathbf{p}_{\mathrm{c,z}}^{\mathrm{ref}}$ equals the desired robot height and $\mathbf{v}_{\mathrm{z}}^{\mathrm{ref}}$ is set to zero. References for roll ϕ^{ref} and pitch θ^{ref} are also set to zero, whereas the yaw ψ^{ref} is computed by integrating the user-defined yaw rate as mentioned in Section 5.4. The reference GRFs $\mathbf{u}^{\mathrm{ref}} = \mathbf{u}^{\mathrm{qp}}$ are obtained by solving the QPMap (6.2). After briefly discussing the setup of the ORG, next, we will present the results of the new locomotion framework in simulation and experiments.

6.3 Simulation and Experimental Result

The ORG empowers the NMPC by providing feasible references to recover from disturbances and avoid drifting when faced with uncertainties. In this section, we discuss some scenarios in simulation and experiments to evaluate the potentiality of our work in practice. We perform the simulations and experiments on the $22\,\mathrm{kg}$ quadruped robot AlienGo. First, we show trot and pace forward walk in simulations. Then, we consider the trot as a template gait for our experiments due to its inherently unstable nature. To show the effectiveness of ORG and NMPC together, we tailored two template scenarios: (a) motion along a straight line and (b) recovery from external pushes.

6.3.1 Implementation Details

The sampling time T_{s} is $40\,\mathrm{ms}$, and the prediction horizon $2\,\mathrm{s}$ is defined for ORG and NMPC. Therefore, the control horizon for both of them, N_{g} and N, are equal to 50. The loop frequency of $25\,\mathrm{Hz}$ is maintained for the ORG similar to the NMPC. This guarantees that references are always available to the NMPC at each call. Table 5 in Bratta et al. (2023a) reports the values of the weighting matrices for the ORG, which have been tuned with a trial-and-error procedure. The NMPC weights are kept equal to those reported in Table 5, and the WBC parameters are mentioned in Section 5.7.1. Cycle time $T_{\mathrm{c}} = 1\,\mathrm{s}$ and duty factor $D = 0.65$ are used for trot parameters. Instead, for pace gait, these parameters are configured to be $T_c = 1.3\,\mathrm{s}$ and $D = 0.7$.

We use the HPIPM Frison and Diehl (2020) solver integrated into `acados` Verschueren et al. (2019) library to solve the problem (6.1). The problem (6.1) is solved using `eiquadprog` Guennebaud et al. (2017) since it offers a more straightforward interface for the QPs. Both reference generator and NMPC run on an Intel Core i7-10750H CPU @2.60 GHz. The solution time by LIPOpt is in the range of 2-4 ms with a prediction horizon of $2\,\mathrm{s}$ and thus $N_g = 50$, while the time required by each QPMap is negligible, around 0.01 ms.

In this section, we will call the output of the reference generator as *reference*, the output of the NMPC as *plan*, and the robot state estimated by the state estimator as *actual*.

6.3.2 Simulations

In the first simulation, the robot is commanded to walk forward with a velocity of $0.08\,\mathrm{m/s}$. The robot follows the reference X position and velocity computed by the ORG, as shown in Fig. 40 . Since the NMPC receives feasible trajectories from the ORG, the tracking of both the plan and actual signals is good.

Similarly, in the second simulation, the robot paces at a commanded forward velocity of $0.08\,\mathrm{m/s}$. Similar to the case of trot and as shown in Fig. 41, the robot shows a good tracking of the references provided by the ORG, further optimized by NMPC and WBC to follow the user-defined task. The footage of both of these simulations are recorded in an accompanying video Bratta et al. (2023b).

6.3.3 Experiments

In this section, we present experimental results for scenarios (a) and (b) performed on a real robot. In the first experiment, we perform scenario (a), where the robot is commanded to walk in a straight line, as illustrated in Fig. 42. This figure demonstrates that when the NMPC and heuristic reference generator (dashed lines) are used together, the robot is unable to walk in a straight line while trotting. Indeed, the robot suffers lateral and backward (less visible) drifts for two reasons: (1) Trot is an unstable gait. Thus, the CoM always diverges between two four-legged-stance events in opposite directions. Furthermore, any minute mechanical asymmetry in the robot can result in a cumulative drift in one

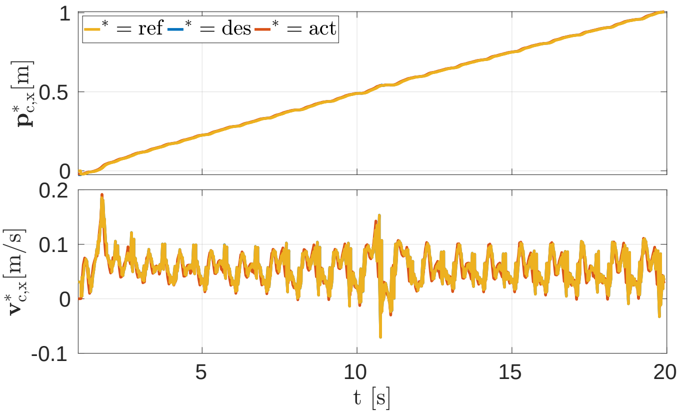


Figure 40: Simulation result of scenario (a) with AlienGo performing a forward trot. In this simulation, the user commands the robot to trot with a forward velocity of $0.08\,\mathrm{m/s}$. The yellow, blue, and red lines represent the reference, output of the NMPC and actual value of the CoM position and velocity, respectively.

direction; and (2) Since AlienGo has c-shaped legs, they create nonzero moments about the pitch axis during a leg swing.

On the other hand, when the ORG is activated to provide references to the NMPC, the robot succeeds in reaching the goal and prevents major drifts in the CoM Y position from the original trajectory (see the continuous line in Fig. 42). The GRFs Z components for a short period of the experiment for better visibility of the data are plotted in Fig. 43. This figure shows a satisfactory tracking of the references by the plan and actual signals. Since the NMPC uses a more detailed model in its formulation, the values computed by the NMPC diverge slightly from the references while maintaining the reference signal trend.

Figure 44 shows the change in the reference lateral velocity (yellow line) computed by the ORG. When the average Y position $\bar{\mathbf{p}}_{\mathrm{c,y}}$ (red line) exceeds the bounds $\bar{\mathbf{p}}_{\mathrm{y}}^{\mathrm{bound}}$ (green and violet dashed lines), the reference generator mode is switched from heuristics to ORG, and the ORG provides lateral velocity references to the NMPC to bring back the CoM Y position close to the goal. A threshold of $1\,\mathrm{cm}$ around the constant goal is chosen to activate the ORG. Once the goal is reached, the reference generator automatically resets to heuristic, and the reference velocity equals the user-defined value, i.e., 0. The continuous change of the reference

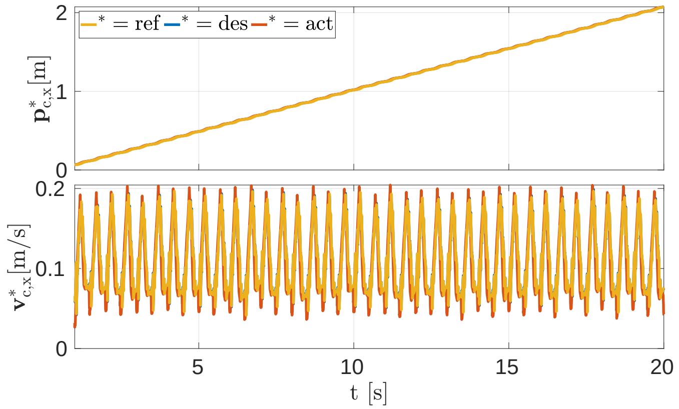


Figure 41: Simulation result of scenario (a) with AlienGo performing a forward pace. In this simulation, the user commands the robot to pace with a forward velocity of $0.08\,\mathrm{m/s}$. The yellow, blue and red lines represent the reference, output of the NMPC and actual value of the CoM position and velocity, respectively.

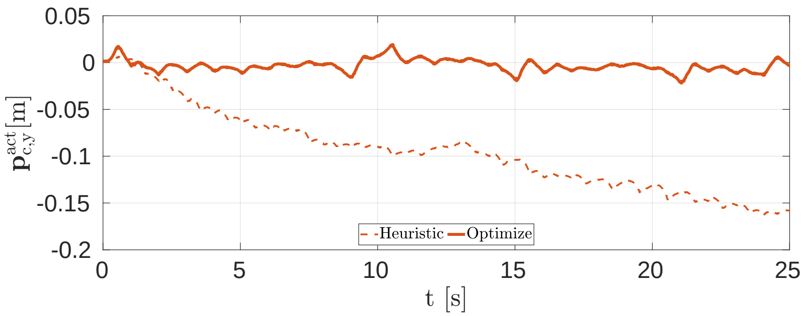


Figure 42: Experiment, scenario (a): Aliengo moving forward with the user set zero lateral velocity. The dashed line represents the actual CoM Y position when the reference generator is set to heuristic mode. In this case, the robot diverges from the goal of $\mathbf{p}^{\text{act}}_{c,y} = 0$, since no correction is made in the reference trajectory in this mode of operation. On the flip side, a continuous line shows the actual CoM Y position when the reference generator automatically switches between heuristic to ORG according to the error on CoM Y position. Due to the corrections provided by the ORG, AlienGo stays close to the goal.

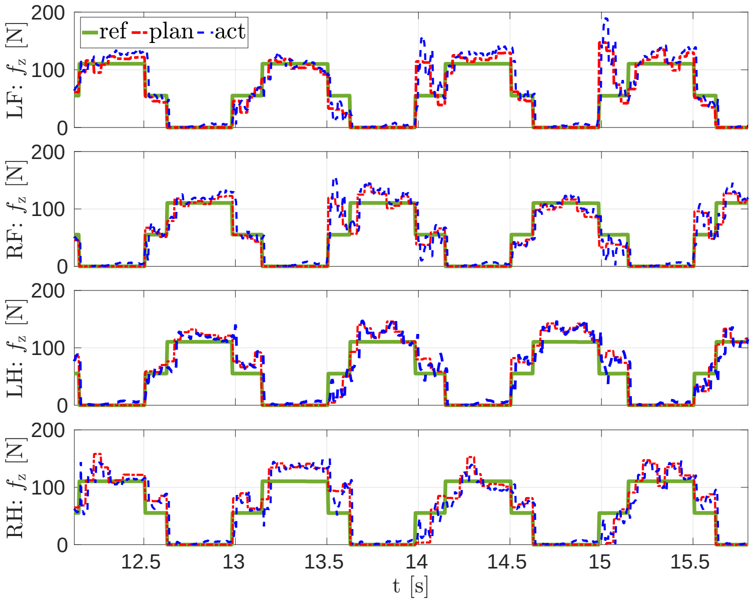


Figure 43: The GRFs in the experiment for scenario (a): Aliengo moving forward with the zero commanded lateral velocity. The plan from NMPC (dashed red line) follows the references provided by ORG (green line). The actual GRFs (dotted blue line) also follows the plan with some modifications to compensate for the disturbances.

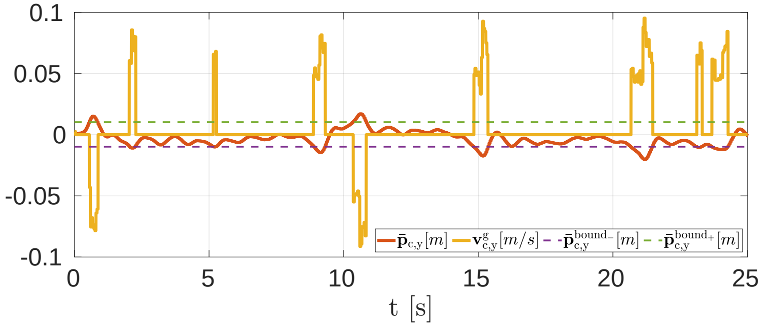


Figure 44: Experiment scenario (a): AlienGo moving forward with zero commanded lateral velocity. The peaks in the reference velocity (yellow line) represent the moment in which the average Y position (red line) crosses the threshold (dashed lines) around the initial position, and during those moments, the ORG is activated.

generator mode from heuristic to ORG justifies the need for the higher level module that provides corrected reference trajectories to the NMPC during a trot.

In the last experiment, we show the efficacy of our ORG in dealing with external disturbances (see Figure 45). We would like to clarify that the task is not to reject the disturbance, but to cope with it in order to recover from its effect. An analysis of techniques to reject disturbances goes beyond the scope of this work. Figure 46 shows the CoM Y position of the robot during the experiment for scenario (b), in which the robot receives two manually applied pushes. The threshold on the error is set to $1\,\mathrm{cm}$ (dashed purple line). During a push, the robot resists this disturbance. Due to the high-frequency re-planning of the NMPC and corrective references provided by ORG during these pushes, it maintains stability and thus avoids falling. Once the pushing force is removed, the reference generator drives the robot back toward the initial position. Similar to the previous cases, the reference generator activates the ORG mode when the robot diverges from the goal.

Readers are encouraged to check the footage of these experiments in the accompanying video Bratta et al. (2023b).

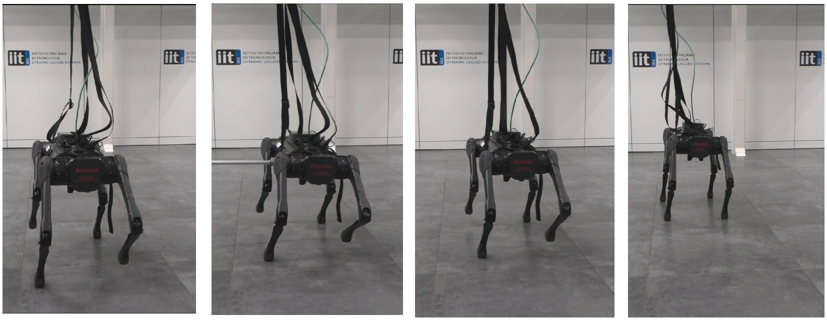


Figure 45: Experiments, scenario (b), sequence of screenshots. The robot moves forward (picture 1) and is suddenly pushed with a stick (picture 2). Once the push is removed (picture 3), the optimized reference generator automatically drives the robot back to its initial position. Finally, the robot follows the user-defined velocity (picture 4).

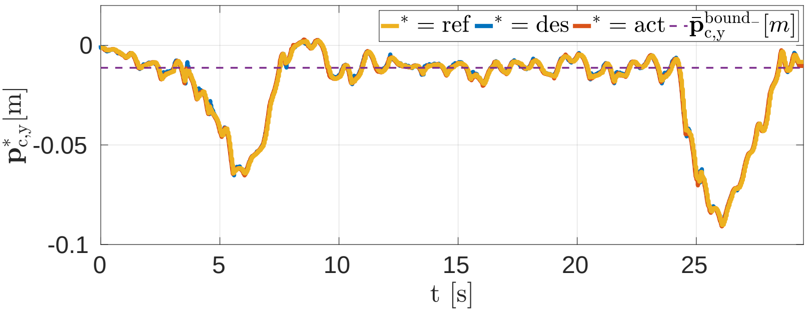


Figure 46: Experiment scenario (b): CoM Y position of the robot. The robot is pushed twice during the motion. It automatically comes back to the initial position when the push is removed.

6.4 Summary

In this chapter, we started by motivating the need for a two-stage optimization scheme, i.e., NMPC with the ORG. Then, we presented the locomotion framework of this two-stage optimization architecture. After that, we introduced the components of the ORG, LIPOpt, and QPMap. The LIPOpt optimizes the CoM quantities based on the LIP model, and the QP finds the optimal GRFs . These quantities are then used by the NMPC as linearization trajectory for the first RTI iteration and as references inside the tracking cost. Finally, we show the effectiveness of the two-stage optimization scheme on the AlienGo robot in simulations and experiments. We also compare the performance of the ORG with respect to the heuristic reference generator from Chapter 5 in experiments.

Chapter 7

Conclusions

The goal of this dissertation has been to design a real-time optimal planner for legged locomotion that enhances the locomotion based on the features included in the planner formulation. Important aspects while designing such an optimal planner are 1) the choice of a suitable robot model, 2) a tailored planning algorithm, and 3) the computational efficiency and reliability of the solution.

In the second chapter, we laid out the background for existing methods. After choosing a suitable model in chapter 3, which establishes the basis of this dissertation, we have proposed methods for optimal planning in the following chapters.

7.1 Summary

In this section, we summarize the contributions of this dissertation and draw some conclusions based on these contributions. In the following, we cover them for each of the contributions.

Model Validation

In the first part of this dissertation, we discussed different models used in legged locomotion for TO and MPC purposes. After selecting the SRBD model for our work, we derived its state-space form in continuous-time, discrete-time, and also LTV form. We also proposed a method to perform a closed-loop validation of these models. We show in the validation results that all these models fit the validation data for the varying pitch and yaw motions while crawling. Based on the result obtained from model validation, we selected the LTV form of the SRBD model for the development of the TO and NMPC algorithms.

Trajectory Optimization for Legged Locomotion

In the second part of this dissertation, we have briefly introduced the idea of MPC and OCP. We then described in brevity the direct methods for OCP solution followed by the description of the single shooting and multiple shooting methods. Next, we proposed two TO methods based on the LTV model derived from the continuous-time SRBD model. The first method, called LTVOpt is solved using an off-the-shelf QP solver that has an advantage over the second method in terms of computation time. However, this method uses reference trajectories for linearization to obtain LTV model that might lead to dynamic infeasibility. The second method, called SQPOpt addresses this drawback by iteratively QP approximation to the original OCP formulation until convergence. However, the second method is computationally expensive. We also showed in simulation results that the solution of LTVOpt is similar to the converged SQPOpt. Therefore, we set requirements for a method that combines the advantages of both methods, i.e., low computational cost and dynamic feasibility for our NMPC formulation. The chosen method is called RTI.

Nonlinear MPC for Legged Locomotion

In this work, we have demonstrated in experiments a real-time NMPC which leverages optimization of leg mobility to achieve terrain adaptation. The contact sequence parameters embedded inside the SRBD model allows us to encode the complementarity constraints directly and without enforcing these constraints separately in the NMPC. We exploited the RTI scheme for our NMPC that enabled us to close the loop at $25\,\mathrm{Hz}$ on the NMPC with a prediction horizon of $2\,\mathrm{s}$. Closing the loop on NMPC at $25\,\mathrm{Hz}$ allows us to compensate for the state drifts due to model uncertainties and tracking errors, and also adapt to the changes in the environment while following user velocity commands both in the simulations and experiments.

In our NMPC, the mobility cost penalizes the hip-to-foot distance from a reference value corresponding to a high mobility factor, and hence it directs the NMPC to compute essential robot orientation to maintain high mobility while respecting the kinematic limits. This is evident from the pallet experiments where we included VFA to correct undesired foot positions defined by the heuristics and avoid possible foot and shin collision. Accounting for the ZMP margin in our NMPC improved the locomotion stability of the robot in all of our experiments and simulation by keeping a sufficiently large ZMP margin from support polygon boundaries. Incorporating a force robustness term in the NMPC ensures that the GRFs stay close to the contact normals, enabling the robot to cope with the estimation error of the orientation of the contact normals.

Two-Stage Optimization

As the last contribution of this dissertation, we have introduced a two-stage optimization scheme with the NMPC and ORG. We have motivated the requirement for the ORG: 1) to tackle the underactuation dynamics, 2) to satisfy the need for physically informed trajectories as references for the NMPC, and 3) to cope with the external disturbances. The novel ORG is composed of two components, the LIPOpt and QPMap. The LIPOpt and QPMap optimize the CoM quantities and GRFs, respectively, that are fed to NMPC as the references for the tracking cost and linearization trajectory for the first RTI iteration. The effectiveness of the proposed two-stage architecture is demonstrated on the AlienGo robot for trot and pace gaits in simulations. With the same robot, we have also reported the results for trot gait in experiments to cope with external disturbances. Lastly, in an experiment, we have compared the performance of the ORG with the heuristic reference generator presented in chapter 5.

7.2 Future Directions

With our NMPC, we have performed successful dynamic locomotion in simulation and experiments on different rough terrains. With two-stage optimization, we have also demonstrated the robustness of our approach when subjected to external disturbances. After taking a bird's-eye view of the presented work in this dissertation, we enlist some of the possible future directions related to our work.

- The performance of our NMPC is evaluated with the SRBD model in simulation and experiments. In the future, we would like to compare its performance with respect to the detailed model such as CD in terms of the accuracy of the optimization variables and computational cost.

- We currently rely on VFA to correct the undesired foothold computed by the heuristic. In our future work, we would like to extend our NMPC formulation to optimize the step timing and foot locations.

- A comparative study of LTI-MPC, LTV-MPC, and NMPC for legged locomotion to investigate the relative performances, limitations, and applicability is a promising future direction to our current work.

- Tuning the weights in the optimization problem for both the NMPC and ORG is tedious and time-consuming. Developing an auto-tuning algorithm to tune these weights automatically is a desirable future direction for our work.

- The NMPC formulated in this dissertation is based on the uniform grid to obtain the discrete-time form of the SRBD model. In the future, we would like to study the impact of the nonuniform grid discretization method (fine at the beginning of the prediction horizon and coarse at the end) on the computational performance and accuracy of the solution.

Appendix A

Angular Velocity

We employ the Z-Y-X convention Diebel (2006) for the Euler angles sequence $\mathbf{\Phi} = (\phi, \theta, \psi)^\top$ to represent the orientation of the robot base where, ϕ, θ and ψ are the roll, pitch and yaw, respectively. The angular velocity in inertial and CoM frame is related to the Euler angle rates with the following relations:

$$\boldsymbol{\omega} = \mathbf{E}(\mathbf{\Phi})\,\dot{\mathbf{\Phi}} \tag{A.1}$$

$$_{\mathcal{C}}\boldsymbol{\omega} = \mathbf{E}'(\mathbf{\Phi})\,\dot{\mathbf{\Phi}} \tag{A.2}$$

$\mathbf{E}(\mathbf{\Phi})$ and $\mathbf{E}'(\mathbf{\Phi})$ are the *Euler angle rates matrix* and *conjugate Euler angle rates matrix* respectively given by,

$$\mathbf{E}(\mathbf{\Phi}) = \begin{bmatrix} \cos(\theta)\cos(\psi) & -\sin(\psi) & 0 \\ \cos(\theta)\sin(\psi) & \cos(\psi) & 0 \\ -\sin(\theta) & 0 & 1 \end{bmatrix} \tag{A.3}$$

$$\mathbf{E}'(\mathbf{\Phi}) = \begin{bmatrix} 1 & 0 & -\sin(\theta) \\ 0 & \cos(\phi) & \cos(\theta)\sin(\phi) \\ 0 & -\sin(\phi) & \cos(\theta)\cos(\phi) \end{bmatrix} \tag{A.4}$$

Remark: $\mathbf{E}$ depends on pitch and yaw, whereas $\mathbf{E}'$ on roll and pitch. Thus, the Euler angle rates $\dot{\mathbf{\Phi}}$ is

$$\dot{\mathbf{\Phi}} = \mathbf{E}^{-1}(\mathbf{\Phi})\,\boldsymbol{\omega} \tag{A.5}$$

$$\dot{\mathbf{\Phi}} = \mathbf{E}'^{-1}(\mathbf{\Phi})\,_{\mathcal{C}}\boldsymbol{\omega} \tag{A.6}$$

Appendix B

Model Validation

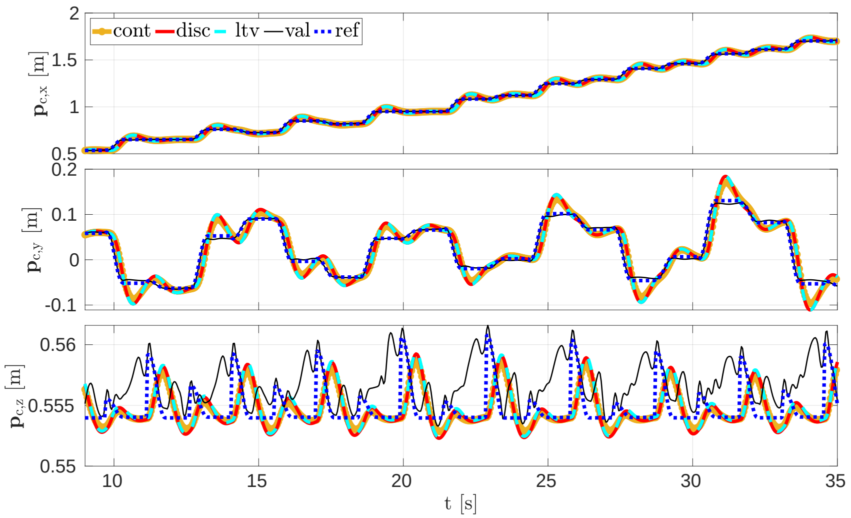


Figure 47: CoM position from the yaw validation results. Yellow, red, and cyan lines represent the continuous-time nonlinear, discrete-time nonlinear, and discrete-time LTV models, respectively. Black and dotted blue lines correspond to the validation data and references to the PD controller.

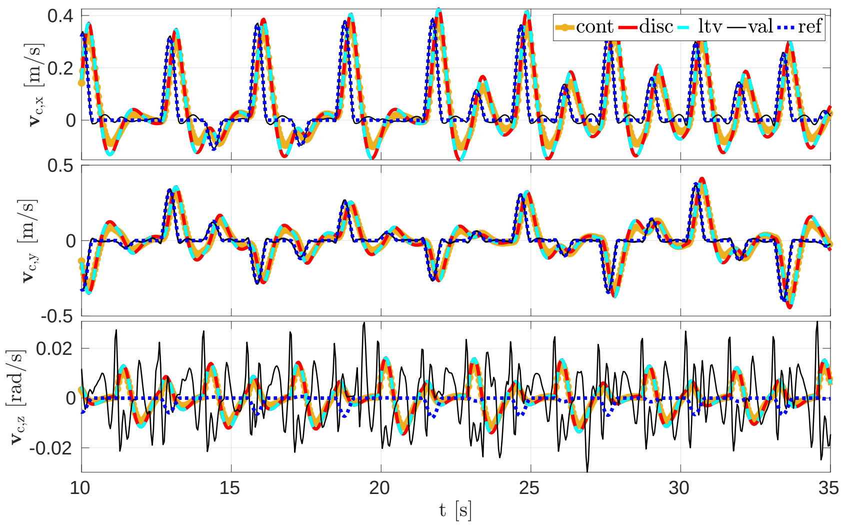


Figure 48: CoM velocity from the yaw validation results. Yellow, red, and cyan lines represent the continuous-time nonlinear, discrete-time nonlinear, and discrete-time LTV models, respectively. Black and dotted blue lines correspond to the validation data and references to the PD controller

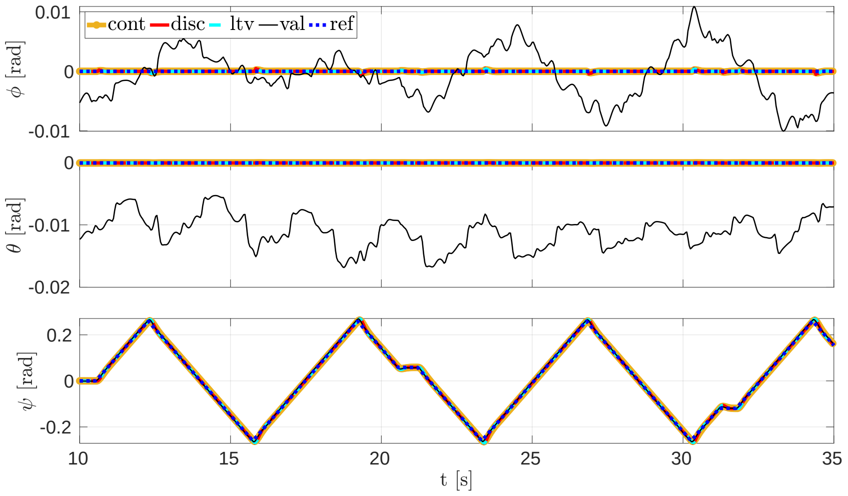


Figure 49: Body orientation from the yaw validation results. Yellow, red, and cyan lines represent the continuous-time nonlinear, discrete-time nonlinear, and discrete-time LTV models, respectively. Black and dotted blue lines correspond to the validation data and references to the PD controller

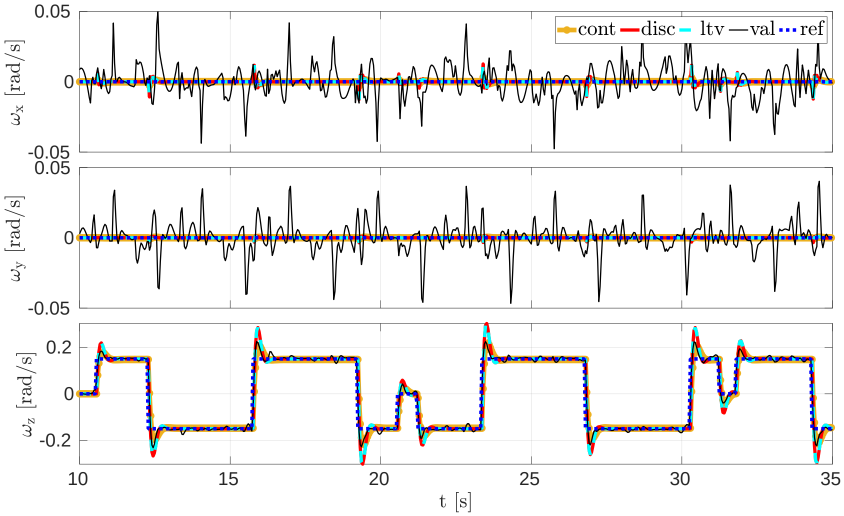


Figure 50: Body angular velocity from the yaw validation results. Yellow, red, and cyan lines represent the continuous-time nonlinear, discrete-time nonlinear, and discrete-time LTV models, respectively. Black and dotted blue lines correspond to the validation data and references to the PD controller

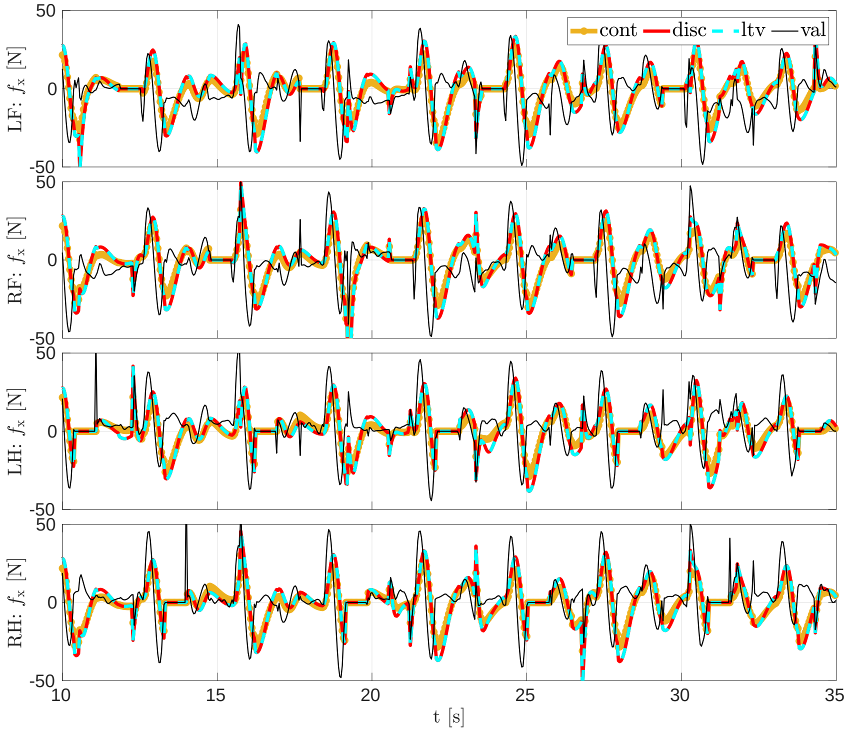


Figure 51: The X component plots of GRFs at respective legs from the yaw validation results. Yellow, red, and cyan lines represent the continuous-time nonlinear, discrete-time nonlinear, and discrete-time LTV models, respectively. The black line corresponds to the validation data.

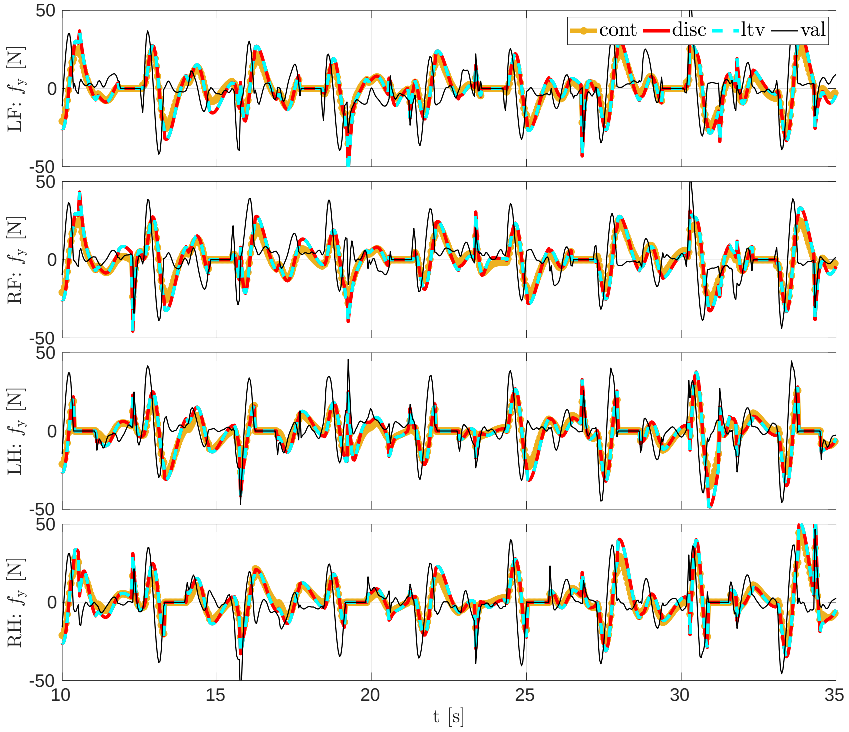


Figure 52: The Y component plots of GRFs at respective legs from the yaw validation results. Yellow, red, and cyan lines represent the continuous-time nonlinear, discrete-time nonlinear, and discrete-time LTV models, respectively. The black line corresponds to the validation data.

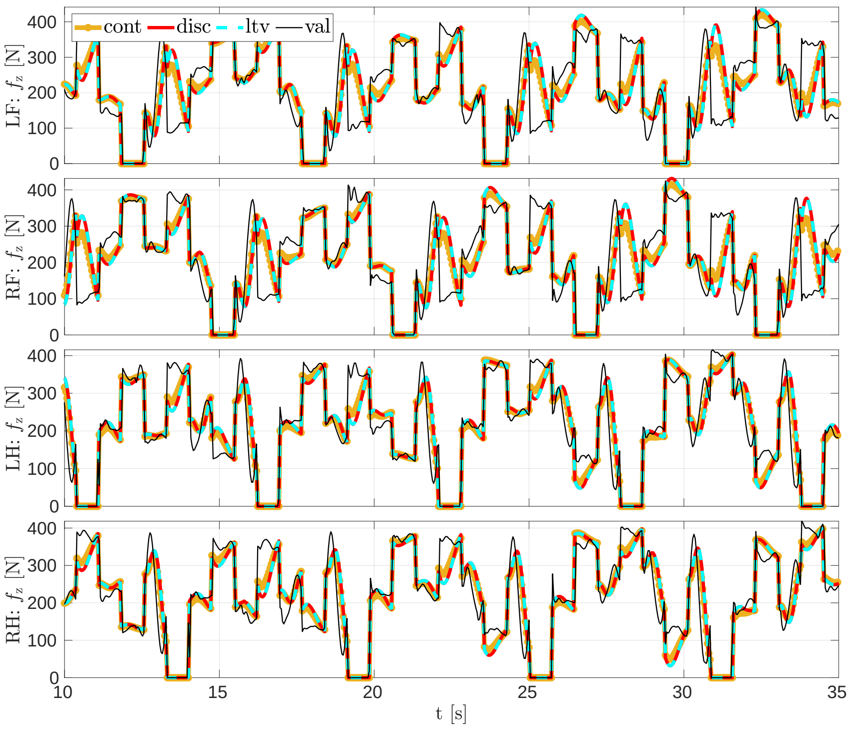


Figure 53: The Z component plots of GRFs at respective legs from the yaw validation results. Yellow, red, and cyan lines represent the continuous-time nonlinear, discrete-time nonlinear, and discrete-time LTV models, respectively. The black line corresponds to the validation data.

Appendix C

QP Formulation

The standard QP from can be written as

$$\min_{\mathbf{w}} \quad \frac{1}{2}\mathbf{w}^\top \mathbf{P}\mathbf{w} + \mathbf{q}^\top \mathbf{w} \tag{C.1a}$$

$$\text{s. t.} \quad \mathbf{w}_{\min} \leq \mathbf{A}_{\mathrm{c}}\mathbf{w} \leq \mathbf{w}_{\max} \tag{C.1b}$$

where $\mathbf{w} \in \mathbb{R}^n$ is the optimization variable. The objective function is defined by a positive semidefinite matrix $\mathbf{P} \in \mathbb{S}^n_+$

$$\begin{aligned} \mathbf{P} &= \operatorname{diag}\left(\mathbf{P_x}, \mathbf{P_u}\right) \\ \mathbf{P_x} &= \operatorname{diag}\left(\mathbf{Q}, \mathbf{Q}, \cdots, \mathbf{Q}_N\right) \\ \mathbf{P_u} &= \operatorname{diag}\left(\mathbf{R}, \cdots, \mathbf{R}\right) \end{aligned} \tag{C.2}$$

and vector $\mathbf{q} \in \mathbb{R}^n$

$$\mathbf{q} = \begin{bmatrix} -\mathbf{Q}\,\mathbf{x}_{\mathrm{r}} \\ -\mathbf{Q}\,\mathbf{x}_{\mathrm{r}} \\ \vdots \\ -\mathbf{Q}_N\,\mathbf{x}_{\mathrm{r}} \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{C.3}$$

The linear constraints are defined by matrix $\mathbf{A}_\mathrm{c} \in \mathbb{R}^{m \times n}$

$$\mathbf{A}_\mathrm{c} = \left[\begin{array}{ccccc|cccc}
-\mathbf{I} & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\
\mathbf{A} & -\mathbf{I} & 0 & \cdots & 0 & \mathbf{B} & 0 & \cdots & 0 \\
0 & \mathbf{A} & -\mathbf{I} & \cdots & 0 & 0 & \mathbf{B} & \cdots & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & 0 & \cdots & -\mathbf{I} & 0 & 0 & \cdots & \mathbf{B} \\
\hline
\mathbf{I} & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\
0 & \mathbf{I} & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\
0 & 0 & \mathbf{I} & \cdots & 0 & 0 & 0 & \cdots & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & 0 & \cdots & \mathbf{I} & 0 & 0 & \cdots & 0 \\
0 & 0 & 0 & \cdots & 0 & \mathbf{I} & 0 & \cdots & 0 \\
0 & 0 & 0 & \cdots & 0 & 0 & \mathbf{I} & \cdots & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & \mathbf{I}
\end{array}\right] \tag{C.4}$$

Vectors $\mathbf{w}_\mathrm{min} \in \mathbb{R}^m$ and $\mathbf{w}_\mathrm{max} \in \mathbb{R}^m$ are given by

$$\mathbf{w}_\mathrm{min} = \begin{bmatrix} -\mathbf{x}_0 \\ \mathbf{r} \\ \vdots \\ \mathbf{r} \\ \mathbf{x}_\mathrm{min} \\ \vdots \\ \mathbf{x}_\mathrm{min} \\ \mathbf{u}_\mathrm{min} \\ \vdots \\ \mathbf{u}_\mathrm{min} \end{bmatrix} \qquad \mathbf{w}_\mathrm{max} = \begin{bmatrix} -\mathbf{x}_0 \\ \mathbf{r} \\ \vdots \\ \mathbf{r} \\ \mathbf{x}_\mathrm{max} \\ \vdots \\ \mathbf{x}_\mathrm{max} \\ \mathbf{u}_\mathrm{max} \\ \vdots \\ \mathbf{u}_\mathrm{max} \end{bmatrix} \tag{C.5}$$

References

B. Abbyasov, R. Lavrenov, A. Zakiev, K. Yakovlev, M. Svinin, and E. Magid. Automatic tool for gazebo world construction: from a grayscale image to a 3d solid model. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7226–7232, 2020. doi: 10.1109/ICRA40945.2020.9196621. 88

B. Aceituno-Cabezas, C. Mastalli, D. Hongkai, M. Focchi, A. Radulescu, D. G. Caldwell, J. Cappelletto, J. C. Grieco, G. Fernando-Lopez, and C. Semini. Simultaneous Contact, Gait and Motion Planning for Robust Multi-Legged Locomotion via Mixed-Integer Convex Optimization. In *IEEE Robot. Autom. Lett.*, 2018. 10, 41

AgilityRobotics. Cassie, 2017. URL https://youtu.be/Is4JZqhAy-M. 22

AgilityRobotics. Cassie robot, 2023. URL https://agilityrobotics.com/news/2022/cassie-sets-a-guinness-world-record. 4

E. Almasri and M. K. Uyguroğlu. Trajectory optimization in robotic applications, survey of recent developments. *Preprints*, May 2021. 41

J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl. CasADi – A software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, 11(1):1–36, 2019. doi: 10.1007/s12532-018-0139-4. 48, 55

C. A. Aumann. A methodology for developing simulation models of complex systems. *Ecological Modelling*, 202(3):385–396, 2007. doi: 10.1016/j.ecolmodel.2006. 23

G. Banjac, B. Stellato, N. Moehle, P. Goulart, A. Bemporad, and S. Boyd. Embedded code generation using the OSQP solver. *2017 IEEE 56th Annual Conference on Decision and Control, CDC 2017*, 2018-January(Cdc):1906–1911, 2018. doi: 10.1109/CDC.2017.8263928. 47

V. Barasuol, J. Buchli, C. Semini, M. Frigerio, E. R. De Pieri, and D. G. Caldwell. A reactive controller framework for quadrupedal locomotion on challenging terrain. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2554–2561, 5 2013. doi: 10.1109/ICRA.2013.6630926. 17

M. Bauer and I. K. Craig. Economic assessment of advanced process control - A survey and framework. *Journal of Process Control*, 18(1):2–18, 2008. ISSN 09591524. doi: 10.1016/j.jprocont.2007.05.007. 42

M. Behrendt. Mpc scheme basic, 2009. URL `https://creativecommons.org/licenses/by-sa/3.0`. via Wikimedia Commons. 43

C. D. Bellicoso, F. Jenelten, P. Fankhauser, C. Gehring, J. Hwangbo, and M. Hutter. Dynamic locomotion and whole-body control for quadrupedal robots. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3359–3365, 2017. doi: 10.1109/IROS.2017.8206174. 10

D. P. Bertsekas. *Dynamic Programming and Optimal Control*, volume I. Athena Scientific, Belmont, MA, USA, 3rd edition, 2005. 44

M. Bjelonic, R. Grandia, M. Geilinger, O. Harley, V. S. Medeiros, V. Pajovic, E. Jelavic, S. Coros, and M. Hutter. Offline motion libraries and online mpc for advanced mobility skills. *The International Journal of Robotics Research*, 41 (9-10):903–924, 2022. doi: 10.1177/02783649221102473. 15

G. Bledt and S. Kim. 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)*, pages 6316–6323, 2019. doi: 10.1109/IROS40897.2019.8968031. 13, 60, 62

G. Bledt and S. Kim. 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)*, pages 6316–6323, 2019. doi: 10.1109/IROS40897.2019.8968031. 15

G. Bledt, P. M. Wensing, and S. Kim. 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)*, pages 4102–4109, 2017. doi: 10.1109/IROS.2017.8206268. 11, 22, 42

T. Boaventura, J. Buchli, C. Semini, and D. Caldwell. Model-based hydraulic impedance control for dynamic robots. *Robotics, IEEE Transactions on*, 31(6): 1324–1336, Dec 2015. ISSN 1552-3098. doi: 10.1109/TRO.2015.2482061. 65

H. Bock and K. Plitt. A multiple shooting algorithm for direct solution of optimal control problems*. In *1984 IFAC Proceedings Volumes*, volume 17, pages 1603–1608, 1984. doi: https://doi.org/10.1016/S1474-6670(17)61205-9. 11

BostonDynamics. Atlas™, 2023. URL https://www.bostondynamics.com/atlas. 4

K. Bouyarmane and A. Kheddar. On weight-prioritized multitask control of humanoid robots. *IEEE Transactions on Automatic Control*, 63(6):1632–1647, 2018. doi: 10.1109/TAC.2017.2752085. 15, 60

A. Bratta, M. Focchi, N. Rathod, and C. Semini. Optimization-based reference generator for nonlinear model predictive control of legged robots. *Robotics*, 12 (6), 2023a. doi: 10.3390/robotics12010006. URL https://www.mdpi.com/2218-6581/12/1/6. 102, 103

A. Bratta, M. Focchi, N. Rathod, and C. Semini. Video of the optimization-based reference generator simulations and experiments, 2023b. URL https://youtu.be/Jp0D8_AKiIY. 104, 108

T. Bretl and S. Lall. Testing static equilibrium for legged robots. *IEEE Transactions on Robotics*, 24(4):794–807, 2008. doi: 10.1109/TRO.2008.2001360. 73

J. C. Butcher. *Numerical Methods for Ordinary Differential Equations*. J. Wiley, 2003. 28, 48, 69, 84

R. H. Byrd, J. Nocedal, and R. A. Waltz. *Knitro: An Integrated Package for Nonlinear Optimization*. Springer US, Boston, MA, 2006. ISBN 978-0-387-30065-8. doi: 10.1007/0-387-30065-1_4. 49

O. Cebe, C. Tiseo, G. Xin, H.-c. Lin, J. Smith, and M. Mistry. Online Dynamic Trajectory Optimization and Control for a Quadruped Robot. *arXiv*, 2020. 6, 14, 69

H. Dai, A. Valenzuela, and R. Tedrake. Whole-body Motion Planning with Simple Dynamics and Full Kinematics. *IEEE-RAS Int. Conf. Humanoid Robot.*, 2014. 10, 21

J. Di Carlo, P. M. Wensing, B. Katz, G. Bledt, and S. Kim. Dynamic locomotion in the mit cheetah 3 through convex model-predictive control. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–9, 2018. doi: 10.1109/IROS.2018.8594448. 11, 14, 22, 42, 62, 100

J. Diebel. Representing Attitude: Euler Angles, Unit Quaternions, and Rotation Vectors. *Matrix*, 58:1–35, 2006. 27, 68, 115

M. Diehl, H. G. Bock, H. Diedam, and P.-B. Wieber. Fast Direct Multiple Shooting Algorithms for Optimal Robot Control. In *Fast Motions in Biomechanics and Robotics*, Heidelberg, Germany, 2005a. 6, 45, 82

M. Diehl, H. G. Bock, and J. P. Schlöder. A real-time iteration scheme for nonlinear optimization in optimal feedback control. *SIAM J. Control Optim.*, 43(5): 1714–1736, 2005b. ISSN 03630129. doi: 10.1137/S0363012902400713. 6, 13, 28, 45, 52, 61, 82

S. Fahmi, M. Focchi, A. Radulescu, G. Fink, V. Barasuol, and C. Semini. STANCE: Locomotion adaptation over soft terrain. *IEEE Trans. Robot. (T-RO)*, 36(2):443–457, Apr. 2020. doi: 10.1109/TRO.2019.2954670. 9, 74

P. Fankhauser and M. Hutter. A Universal Grid Map Library: Implementation and Use Case for Rough Terrain Navigation. In A. Koubaa, editor, *Robot Operating System, Vol. 1*, chapter 5. Springer, Cham, Switzerland, 2016. ISBN 978-3-319-26052-5. doi: 10.1007/978-3-319-26054-9{_}5. 65

P. Fankhauser, M. Bjelonic, C. Dario Bellicoso, T. Miki, and M. Hutter. Robust rough-terrain locomotion with a quadrupedal robot. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5761–5768, 2018. doi: 10.1109/ICRA.2018.8460731. 14

F. Farshidian, E. Jelavic, A. Satapathy, M. Giftthaler, and J. Buchli. Real-time motion planning of legged robots: A model predictive control approach. In *Humanoid Robot. (Humanoids), 2017 IEEE-RAS 17th Int. Conf.*, pages 577–584. IEEE, 2017. 13, 62

R. Featherstone. *Rigid Body Dynamics Algorithms*. Springer-Verlag, Berlin, Heidelberg, 2008. ISBN 0387743146. 19

M. Focchi, A. del Prete, I. Havoutis, R. Featherstone, D. G. Caldwell, and C. Semini. High-slope terrain locomotion for torque-controlled quadruped robots. *Auton. Robots*, pages 1–14, 2016. ISSN 1573-7527. doi: 10.1007/s10514-016-9573-1. 30, 63, 66, 74, 79, 80

M. Focchi, R. Featherstone, R. Orsolino, D. G. Caldwell, and C. Semini. Viscosity-based height reflex for workspace augmentation for quadrupedal locomotion on rough terrain. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5353–5360, 2017. doi: 10.1109/IROS.2017.8206430. 70, 71

M. Focchi, R. Orsolino, M. Camurri, V. Barasuol, C. Mastalli, D. G. Caldwell, and C. Semini. *Heuristic Planning for Rough Terrain Locomotion in Presence of External Disturbances and Variable Perception Quality*, pages 165–209. Springer, Cham, Switzerland, 2020a. ISBN 978-3-030-22327-4. doi: 10.1007/978-3-030-22327-4_9. 9, 13, 17, 33, 72

M. Focchi, R. Orsolino, M. Camurri, V. Barasuol, C. Mastalli, D. G. Caldwell, and C. Semini. Heuristic Planning for Rough Terrain Locomotion in Presence of External Disturbances and Variable Perception Quality. In A. Grau,

Y. Morel, A. Puig-Pey, and F. Cecchi, editors, *Adv. Robot. Res. From Lab to Mark. ECHORD++ Robot. Sci. Support. Innov.*, pages 165–209. Cham, Switzerland: Springer International Publishing, Cham, Switzerland:, 2020b. ISBN 978-3-030-22327-4. doi: 10.1007/978-3-030-22327-4_9. URL https://doi.org/10.1007/978-3-030-22327-4_9. 51

M. Frigerio, J. Buchli, D. G. Caldwell, and C. Semini. 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, 2016. 31

M. Frigerio, V. Barasuol, M. Focchi, D. G. Caldwell, and C. Semini. Validation of computer simulations of the hyq robot. In *International Conference on Climbing and Walking Robots (CLAWAR)*, 2017. 31

G. Frison and M. Diehl. HPIPM: a high-performance quadratic programming framework for model predictive control. *arXiv*, 2020. 47, 86, 104

R. V. Gamkrelidze. Discovery of the maximum principle. *Journal of Dynamical and Control Systems*, 5(4):437–451, Oct 1999. ISSN 1573-8698. doi: 10.1023/A:1021783020548. URL https://doi.org/10.1023/A:1021783020548. 44

C. Gehring, C. D. Bellicoso, S. Coros, M. Bloesch, P. Fankhauser, M. Hutter, and R. Siegwart. Dynamic trotting on slopes for quadrupedal robots. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5129–5135, 2015. doi: 10.1109/IROS.2015.7354099. 72

P. E. Gill, W. Murray, and M. A. Saunders. Snopt: An sqp algorithm for large-scale constrained optimization. *SIAM J. Optimization*, 12(4):979–1006, 2002. doi: https://doi.org/10.1137/S1052623499350013. 49

R. Grandia, F. Farshidian, R. Ranftl, and M. Hutter. Feedback mpc for torque-controlled legged robots. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4730–4737, 2019. doi: 10.1109/IROS40897.2019.8968251. 11, 62

F. Grimminger, A. Meduri, M. Khadiv, J. Viereck, M. Wüthrich, M. Naveau, V. Berenz, S. Heim, F. Widmaier, T. Flayols, J. Fiene, A. Badri-Spröwitz, and L. Righetti. An open torque-controlled modular robot architecture for legged locomotion research. *IEEE Robotics and Automation Letters*, 5(2):3650–3657, 2020. doi: 10.1109/LRA.2020.2976639. 14

S. Gros, M. Zanon, R. Quirynen, A. Bemporad, and M. Diehl. From linear to nonlinear MPC: bridging the gap via the real-time iteration. *Int. J. Control*, 93 (1):62–80, 2020. ISSN 13665820. doi: 10.1080/00207179.2016.1222553. 6, 52, 82, 84

G. Guennebaud, A. Furfaro, L. Di Gaspero, and S. Benjamin. eiquadprog, 2017. URL https://github.com/fx74/uQuadProg. 104

Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2023. URL https://www.gurobi.com. 55

E. Hairer, S. Nørsett, and G. Wanner. *Solving Ordinary Differential Equations I*. Springer Series in Computational Mathematics. Springer, Berlin, 2nd edition, 1993. 28, 48, 69, 84

E. Hairer, S. Norsett, and G. Wanner. *Solving Ordinary Differential Equations II – Stiff and Differential-Algebraic Problems*. Springer Series in Computational Mathematics. Springer, Berlin, 2nd edition, 1996. 28, 48, 69, 84

J. Hauser and A. Saccon. A barrier function method for the optimization of trajectory functionals with constraints. In *Proceedings of the 45th IEEE Conference on Decision and Control*, pages 864–869, 2006. doi: 10.1109/CDC.2006.377331. 11

A. Herdt, H. Diedam, P. B. Wieber, D. Dimitrov, K. Mombaur, and M. Diehl. Online walking motion generation with automatic footstep placement. *Advanced Robotics*, 24(5-6):719–737, 2010. ISSN 01691864. doi: 10.1163/016918610X493552. 11, 62

T. Horvat, K. Melo, and A. J. Ijspeert. Model predictive control based framework for com control of a quadruped robot. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3372–3378, 2017. doi: 10.1109/IROS.2017.8206176. 11, 62

M. Hutter, C. Gehring, D. Jud, A. Lauber, C. D. Bellicoso, V. Tsounis, J. Hwangbo, K. Bodie, P. Fankhauser, M. Bloesch, R. Diethelm, S. Bachmann, A. Melzer, and M. Hoepflinger. Anymal - a highly mobile and dynamic quadrupedal robot. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 38–44, 2016. doi: 10.1109/IROS.2016.7758092. 22

L. A. Kerr and D. R. Goethel. Chapter twenty one - simulation modeling as a tool for synthesis of stock identification information. In S. X. Cadrin, L. A. Kerr, and S. Mariani, editors, *Stock Identification Methods (Second Edition)*, pages 501–533. Academic Press, San Diego, second edition edition, 2014. ISBN 978-0-12-397003-9. doi: https://doi.org/10.1016/B978-0-12-397003-9.00021-7. URL https://www.sciencedirect.com/science/article/pii/B9780123970039000217. 23

J. Kim, T. Kang, D. Song, and S.-J. Yi. Design and control of a open-source, low cost, 3d printed dynamic quadruped robot. *Applied Sciences*, 11(9), 2021. ISSN 2076-3417. doi: 10.3390/app11093762. URL https://www.mdpi.com/2076-3417/11/9/3762. 22

J. Koenemann, A. D. Prete, Y. Tassa, E. Todorov, O. Stasse, M. Bennewitz, and N. Mansard. Whole-body model-predictive control applied to the HRP-2 humanoid. In *IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, pages 3346–3351, 2015. doi: 10.1109/IROS.2015.7353843. 10, 11, 20, 42, 62

N. Koenig and A. Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, volume 3, pages 2149–2154 vol.3, 2004. doi: 10.1109/IROS.2004.1389727. 31

S. Kuindersma, F. Permenter, and R. Tedrake. An efficiently solvable quadratic program for stabilizing dynamic locomotion. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2589–2594, 2014. doi: 10.1109/ICRA.2014.6907230. 10

G. Lantoine and R. P. Russell. A hybrid differential dynamic programming algorithm for constrained optimal control problems. part 1: Theory. *Journal of Optimization Theory and Applications*, 154(2):382–417, Aug 2012. ISSN 1573-2878. doi: 10.1007/s10957-012-0039-0. 11

H. Li and P. M. Wensing. Hybrid systems differential dynamic programming for whole-body motion planning of legged robots. *IEEE Robotics and Automation Letters*, 5(4):5448–5455, 2020. ISSN 23773766. doi: 10.1109/LRA.2020.3007475. 11

C. Mastalli, I. Havoutis, M. Focchi, D. G. Caldwell, and C. Semini. Motion planning for quadrupedal locomotion: Coupled planning, terrain mapping and whole-body control. *IEEE Trans. Robot. (T-RO)*, pages 1–14, 2020. 5

C. Mastalli, W. Merkt, G. Xin, J. Shim, M. Mistry, I. Havoutis, and S. Vijayakumar. Agile maneuvers in legged robots: a predictive control approach. 2022. 20

MATLAB. *ode15s*. Natick, Massachusetts, United State, 2022a. URL `https://www.mathworks.com/help/matlab/ref/ode15s.html`. 32

MATLAB. *fmincon*. Natick, Massachusetts, United State, 2022b. URL `https://www.mathworks.com/help/optim/ug/fmincon.html`. 49

A. Meduri, P. Shah, J. Viereck, M. Khadiv, I. Havoutis, and L. Righetti. Biconmp: A nonlinear model predictive control framework for whole body motion planning. *IEEE Transactions on Robotics*, pages 1–18, 2023. doi: 10.1109/TRO.2022.3228390. 14

O. Melon, M. Geisert, D. Surovik, I. Havoutis, and M. Fallon. Reliable trajectories for dynamic quadrupeds using analytical costs and learned initializations. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1410–1416, 2020. doi: 10.1109/ICRA40945.2020.9196562. 10, 41

M. Neunert, F. Farshidian, A. W. Winkler, and J. Buchli. Trajectory Optimization Through Contacts and Automatic Gait Discovery for Quadrupeds. *IEEE Robot. Autom. Lett.*, 2(3):1502–1509, 2017. ISSN 23773766. doi: 10.1109/LRA.2017.2665685. 10, 20, 41

M. Neunert, M. Stauble, M. Giftthaler, C. D. Bellicoso, J. Carius, C. Gehring, M. Hutter, and J. Buchli. Whole-Body Nonlinear Model Predictive Control Through Contacts for Quadrupeds. *IEEE Robot. Autom. Lett.*, 3(3):1458–1465, 2018. ISSN 23773766. doi: 10.1109/LRA.2018.2800124. 6, 10, 13, 20, 42, 62

S. Nobili, M. Camurri, V. Barasuol, M. Focchi, D. G. Caldwell, C. Semini, and M. Fallon. Heterogeneous sensor fusion for accurate state estimation of dynamic legged robots. In *Proceedings of Robotics: Science and Systems*, Boston, USA, July 2017. 17, 65

J. Nocedal and S. Wright. *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering. Springer New York, 2006. ISBN 9780387400655. URL https://link.springer.com/book/10.1007/978-0-387-40065-5. 51, 52

OpenRobotics. Ros, 2022. URL https://www.ros.org/. 31

D. E. Orin, A. Goswami, and S.-H. Lee. Centroidal dynamics of a humanoid robot. *Auton. Robots*, 35(2-3):161–176, jun 2013. ISSN 0929-5593. doi: 10.1007/s10514-013-9341-4. 10, 21

R. Orsolino, M. Focchi, D. G. Caldwell, and C. Semini. A combined limit cycle - zero moment point based approach for omni-directional quadrupedal bounding. In *International Conference on Climbing and Walking Robots (CLAWAR)*, 2017. 17

OSRF. Gazebo, 2014. URL http://gazebosim.org/. 31

M. Posa, S. Kuindersma, and R. Tedrake. Optimization and stabilization of trajectories for constrained dynamical systems. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1366–1373, 2016. doi: 10.1109/ICRA.2016.7487270. 10, 20, 41

J. Pratt, C.-M. Chew, A. Torres, P. Dilworth, and G. Pratt. Virtual model control: An intuitive approach for bipedal locomotion. *The International Journal of Robotics Research*, 20(2):129–143, 2001. doi: 10.1177/02783640122067309. 9

S. J. Qin and T. A. Badgwell. Process Control Dynamic. *Control Engineering Practice*, 11:733–764, 2003. ISSN 09670661. 42

R. Quirynen. *Numerical Simulation Methods for Embedded Optimization*. PhD thesis, Arenberg Doctoral School, Dept. Eng. Sci., KU Leuven, Leuven, Belgium, Dept. Math. Phys., Univ. Freiburg, Freiburg im Breisgau, Germany, Jan. 2017. p. 327. 28, 48, 69, 84

R. Quirynen, M. Vukov, M. Zanon, and M. Diehl. Autogenerating Microsecond Solvers for Nonlinear MPC: a Tutorial Using ACADO Integrators. *Optimal Control Applications and Methods*, 36:685–704, 2014. 84

M. H. Raibert. *Legged robots that balance*. MIT press, 1986. 9, 77

G. Raiola, E. Mingo Hoffman, M. Focchi, N. Tsagarakis, and C. Semini. A simple yet effective whole-body locomotion framework for quadruped robots. *Frontiers in Robotics and AI*, 7:159, 2020. ISSN 2296-9144. doi: 10.3389/frobt.2020.528473. 73, 81

N. Rathod, A. Bratta, M. Focchi, M. Zanon, O. Villarreal, C. Semini, and A. Bemporad. Model predictive control with environment adaptation for legged locomotion. *IEEE Access*, 9:145710–145727, 2021a. doi: 10.1109/ACCESS.2021.3118957. 63

N. Rathod, A. Bratta, M. Focchi, M. Zanon, O. Villarreal, C. Semini, and A. Bemporad. Video of the model predictive control with environment adaptation for legged locomotion, 2021b. URL https://youtu.be/r0-KIiw0eWM. 88, 89, 91, 92, 97

J. Rawlings, D. Mayne, and M. Diehl. *Model Predictive Control: Theory, Computation, and Design*. Nob Hill Publishing, LLC, Santa Barbara, CA 93117, 2019. ISBN 9780975937730. URL https://sites.engineering.ucsb.edu/~jbraw/pubs_books.html. 43, 45, 47

L. Sciavicco, B. Siciliano, and B. Sciavicco. *Modelling and Control of Robot Manipulators*. Springer-Verlag, Berlin, Heidelberg, 2nd edition, 2000. ISBN 1852332212. 13, 70

C. Semini, N. G. Tsagarakis, E. Guglielmino, M. Focchi, F. Cannella, and D. G. Caldwell. Design of hyq - a hydraulically and electrically actuated quadruped robot. *IMechE Part I: Journal of Systems and Control Engineering*, 225(6):831–849, 2011. doi: 10.1177/0959651811402275. 5, 17, 22, 67

R. Unitree. Aliengo, 2022. URL https://www.unitree.com/en/aliengo/. 16, 18, 100

R. Verschueren, G. Frison, D. Kouzoupis, N. van Duijkeren, A. Zanelli, B. Novoselnik, J. Frey, T. Albin, R. Quirynen, and M. Diehl. Acados - A modular open-source framework for fast embedded optimal control. *arXiv*, 2019. ISSN 23318422. 13, 84, 86, 104

O. Villarreal, V. Barasuol, M. Camurri, L. Franceschi, M. Focchi, M. Pontil, D. G. Caldwell, and C. Semini. Fast and continuous foothold adaptation for dynamic locomotion through cnns. *IEEE Robotics and Automation Letters*, 4(2): 2140–2147, 2019. doi: 10.1109/LRA.2019.2899434. 78

O. Villarreal, V. Barasuol, P. M. Wensing, D. G. Caldwell, and C. Semini. MPC-based controller with terrain insight for dynamic legged locomotion. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2436–2442, 2020. doi: 10.1109/ICRA40945.2020.9197312. 69, 72

A. Wächter and L. T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57, Mar 2006. ISSN 1436-4646. doi: 10.1007/s10107-004-0559-y. URL https://doi.org/10.1007/s10107-004-0559-y. 49

P. M. Wensing, M. Posa, Y. Hu, A. Escande, N. Mansard, and A. D. Prete. Optimization-based control for dynamic legged robots, 2022. 45

P.-B. Wieber. Trajectory free linear model predictive control for stable walking in the presence of strong perturbations. In *2006 6th IEEE-RAS International Conference on Humanoid Robots*, pages 137–142, 2006. doi: 10.1109/ICHR.2006.321375. 73

A. W. Winkler. Optimization-based motion planning for legged robots. page 100, 2018. URL https://doi.org/10.3929/ethz-b-000272432. 22

A. W. Winkler, C. D. Bellicoso, M. Hutter, and J. Buchli. Gait and Trajectory Optimization for Legged Systems Through Phase-Based End-Effector Parameterization. *IEEE Robot. Autom. Lett.*, 3(3):1560–1567, 2018. ISSN 23773766. doi: 10.1109/LRA.2018.2798285. 11, 22, 67

T. Yoshikawa. Analysis and Control of Robot Manipulators with Redundancy. In M. Brady and R. Paul, editors, *Robot. Res. First Int. Symp.*, pages 735–747, Cambridge, MA, USA, 1984. MIT Press. doi: 10.1.1.18.7268. 70

A. B. Younes, J. D. Turner, D. Mortari, and J. L. Junkins. A Survey of attitude error representations, 2012. 68