D I S S E R T A T I O N

submitted

to the

Combined Faculty for the Natural Sciences and Mathematics

of

Heidelberg University, Germany

for the degree of

Doctor of Natural Sciences

put forward by

Diplom–Mathematiker

Manuel Kudruss

born in Eberbach in Baden–Württemberg

Date of the oral examination

January 28, 2020

Nonlinear Model Predictive Control for Motion Generation of Humanoids

Referees

Professor Dr. Katja Mombaur

Professor Dr. Christian Kirches

Zusammenfassung

Das Ziel dieser Arbeit ist die Untersuchung und Entwicklung numerischer Methoden zur Bewegungserzeugung von humanoiden Robotern basierend auf nichtlinearer modell-prädikativer Regelung. Ausgehend von der Modellierung der Humanoiden als komplexe Mehrkörpermodelle, die sowohl durch unilaterale Kontaktbedingungen beschränkt als auch durch die Formulierung unteraktuiert sind, wird die Bewegungserzeugung als Optimalsteuerungsproblem formuliert.

In dieser Arbeit werden numerische Erweiterungen basierend auf den Prinzipien der *Automatischen Differentiation* für rekursive Algorithmen, die eine effiziente Auswertung der dynamischen Größen der oben genannten Mehrkörperformulierung erlauben, hergeleitet, sodass sowohl die nominellen Größen als auch deren ersten Ableitungen effizient ausgewertet werden können. Basierend auf diesen Ideen werden Erweiterungen für die Auswertung der Kontaktdynamik und der Berechnung des Kontaktimpulses vorgeschlagen.

Die Echtzeitfähigkeit der Berechnung von Regelantworten hängt stark von der Komplexität der für die Bewegungerzeugung gewählten Mehrkörperformulierung und der zur Verfügung stehenden Rechenleistung ab. Um einen optimalen Trade-Off zu ermöglichen, untersucht diese Arbeit einerseits die mögliche Reduktion der Mehrkörperdynamik und andererseits werden maßgeschneiderte numerische Methoden entwickelt, um die Echtzeitfähigkeit der Regelung zu realisieren.

Im Rahmen dieser Arbeit werden hierfür zwei reduzierte Modelle hergeleitet: eine nichtlineare Erweiterung des linearen inversen Pendelmodells sowie eine reduzierte Modellvariante basierend auf der *centroidalen* Mehrkörperdynamik. Ferner wird ein Regelaufbau zur Ganzkörper-Bewegungserzeugung vorgestellt, deren Hauptbestandteil jeweils aus einem speziell diskretisierten Problem der nichtlinearen modell-prädikativen Regelung sowie einer maßgeschneiderter Optimierungsmethode besteht. Die Echtzeitfähigkeit des Ansatzes wird durch Experimente mit den Robotern *HRP-2* und *HeiCub* verifiziert.

Diese Arbeit schlägt eine Methode der nichtlinear modell-prädikativen Regelung vor, die trotz der Komplexität der vollen Mehrkörperformulierung eine Berechnung der Regelungsantwort in Echtzeit ermöglicht. Dies wird durch die geschickte Kombination von linearer und nichtlinearer modell-prädikativer Regelung auf der aktuellen beziehungsweise der letzten Linearisierung des Problems in einer parallelen Regelstrategie realisiert. Experimente mit dem humanoiden Roboter *Leo* zeigen, dass, im Vergleich zur nominellen Strategie, erst durch den Einsatz dieser Methode eine Bewegungserzeugung auf dem Roboter möglich ist.

Neben Methoden der modell-basierten Optimalsteuerung werden auch modell-freie Methoden des *verstärkenden Lernens* (Reinforcement Learning) für die Bewegungserzeugung untersucht, mit dem Fokus auf den schwierig zu modellierenden Modellunsicherheiten der Roboter.

Im Rahmen dieser Arbeit werden eine allgemeine vergleichende Studie sowie Leistungskennzahlen entwickelt, die es erlauben, modell-basierte und -freie Methoden quantitativ bezüglich ihres Lösungsverhaltens zu vergleichen. Die Anwendung der Studie auf ein akademisches Beispiel zeigt Unterschiede und Kompromisse sowie Break-Even-Punkte zwischen den Problemformulierungen.

Diese Arbeit schlägt basierend auf dieser Grundlage zwei mögliche Kombinationen vor, deren Eigenschaften bewiesen und in Simulation untersucht werden. Außerdem wird die besser abschneidende Variante auf dem humanoiden Roboter *Leo* implementiert und mit einem nominellen modell-basierten Regler verglichen.

Abstract

The aims of this thesis are the investigation and development of numerical methods for the whole-body motion generation of humanoid robots based on nonlinear model predictive control. Proceeding from modeling the humanoids as complex multi-body systems subject to kinematic constraints that are underactuated through their mathematical formulation, we formulate the general motion generation task by means of an optimal control problem.

Within the scope of this thesis, we propose the numerical extension in the sense of automatic differentiation for the recursive algorithms commonly applied for the evaluation of dynamic quantities of above mentioned multi-body formulation, such that they efficiently evaluate both the nominal quantity as well as the first-order forward derivative. Based on these ideas, we further propose extensions to the linear algebra required for the evaluation of the contact dynamics and impulses.

The real-time feasibility of the computation of the feedback response is strongly dependent on the chosen multi-body formulation and the available on-board computing power of the humanoid robot. In order to allow an optimal trade-off, in this thesis, we derive reduced models of the multi-body dynamics on the one hand and, on the other hand, develop tailored algorithms to guarantee real-time feasibility.

In the context of this thesis, we propose two reduced models: one nonlinear extension to the linear inverted pendulum model and a reduced version based on the centroidal dynamics of the multi-body system. Based on this, we propose a control framework for whole-body motion generation, whose main part is build up by a specially discretized NMPC problem together with a tailored optimization method. The real-time feasibility of this approach is verified by experiments with the humanoid robots *HRP-2* and *HeiCub*.

In this thesis, we propose a method of nonlinear model predictive control that can compute feedback controls in real-time despite the complexity introduced by the full multi-body model. This is achieved by a combination of linear and nonlinear model predictive control in a parallel control strategy, which utilizes the last linearization for fast feedback while the parallel thread is still busy computing the current linearization according to the last feedback time. Experiments on the humanoid robot show that motion generation on the robot, compared to a nominal controller, can only be enabled by this strategy.

Alongside model-based methods of optimal control we also investigate model-free methods of machine learning in the form of reinforcement learning as well as their fruitful combination, especially for the treatment of model uncertainties of robots that are difficult to model.

This thesis carries out a general comparative study as well as proposes key performance indicators in order to quantitatively compare model-based and model-free methods with respect to their solution performance. The application of this study on an benchmark example reveals the differences, trade-offs and break-even points between the chosen methods.

In this thesis, we propose based on this knowledge two possible combinations, whose properties are proven and investigated in simulation. The superior method is then implemented on the robot *Leo* and compared to a nominal model-based controller.

Danksagung

An dieser Stelle möchte ich gerne allen Verantwortlichen, Unterstützern, Freunden und meiner Famile danken, die mich alle in meinem Dissertationsvorhaben begleitet haben.

Meinen Lehrern und Mentoren, *Christian Kirches*, *Katja Mombaur*, *Hans Georg Bock* und *Johannes Schlöder*, gilt mein aufrichtiger Dank für ihre hervorragende Unterstützung und Betreuung dieser Arbeit. Die offene akademische Atmosphäre am interdisziplinären Zentrum für wissenschaftliches Rechnen und an der Fakultät für Mathematik und Informatik der Ruprecht-Karls-Universität Heidelberg half enorm beim Gelingen und Fertigstellen meiner Dissertation und machte die letzten Jahre zu einem fruchtbaren Vergnügen.

Ferner haben zahlreiche Mitglieder der Arbeitsgruppen *Optimization of Uncertain Systems*, *Optimization, Robotics & Biomechanics*, *Simulation und Optimierung* und der *BASF Juniorgruppe – Optimum Experimental Design* ihren Beitrag zum Erfolg dieser Arbeit durch fachliche Diskussionen, Gedanktenaustausch oder einfach nur nette Gespräche auf dem Flur geleistet. Großer Dank gebührt deshalb meinen Kollegen *Christoph Weiler* und *Felix Lenders*, *Kai Henning Koch*, *Alexander Schubert*, *Eike Fokken* und *Benjamin Reh*. *Matthew Millard* rückte stets meine Realität zurecht und *Paul Manns* spornte mich bis zuletzt an.

Meinen *KoroiBot*-Kollegen *Yue Hu*, *Malin Kopitzsch* und *Debora Clever* bin ich dankbar für die gemeinsame Zeit in einem anspruchsvollen und erfolgreichen EU-Projekt. Außerdem möchte ich meinen internationalen Projektpartnern *Maximilien Naveau*, *Nicolas Mansard*, *Olivier Stasse* und *Phillipe Souères* am *CNRS-LAAS*, Toulouse, Frankreich, und in besonderer Weise meinen Kollegen *Ivan Koryakovskiy*, *Heike Vallery*, *Wouter Caarls* sowie *Robert Babuška*, der *Technischen Universität Delft*, Delft, Niederlande, für die tolle interdisziplinäre und internationale Zusammenarbeit im *KoroiBot*-Projekt danken.

Besonderer Dank gilt zudem meinen beiden Mentoren der *HGS MathComp*, *Sebastian Walter* und *Martin Felis*, die mir stets bei meinen Problemen helfen konnten und mir ansonsten auch über die Wissenschaft hinaus als Freunde mit Rat und Tat zur Seite stehen.

Für die finanzielle Unterstützung dieser Arbeit gebührt mein Dank der *Ruprecht-Karls-Universität Heidelberg*, der *Technischen Universität Carolo-Wilhelmina zu Braunschweig*, der *Heidelberg Graduate School of Mathematical and Computational Methods for the Sciences* sowie im Besonderen dem 7. Forschungsrahmenprogramm der Europäische Union unter *Grant Agreement No. 611909* im Rahmen des Projektes *KoroiBot*.

Während der Arbeit an dieser Dissertation hatte ich zudem das Glück, herausragende Studenten zu betreuen. *Felix Jost*, *Jan Lammel*, *Kevin Stein*, *Raphael Michel* und *Gaurav Gupta* gehört mein Dank für die gemeinsame Zeit und die von ihnen geleistete Arbeit, die einen wertvollen Beitrag zu dieser Arbeit darstellt.

Zuletzt möchte ich meiner Familie und meinen Freunden danken, die mich stets darin bestärkt haben, meinen eigenen Weg zu gehen und diesen erfolgreich zu meistern. Der größte Anteil gebührt dabei meiner Frau *Ivonne*, die in dieser Zeit oft für uns beide stark war und es immer wieder geschafft hat, auch an schlechten Tagen für mich die Sonne scheinen zu lassen. Zu guter Letzt noch vielen Dank an meine Tochter *Luisa* für die Motivation beim Endspurt. Du hast das Rennen dann schließlich doch noch gewonnen.

Contents

0 Introduction

Preface

Nature's ability to optimize design, behavior as well as locomotion of living beings, for example, humans, animals, and plants, is omnipresent. Human mankind has a burning desire to analyze and derive the principles behind this optimality and finally realize the underlying principles in technical form [131]. One example for this approach can be seen in the research field of humanoid robotics, which nowadays is a truly interdisciplinary research field in which different expertise in engineering, mathematics, biomechanics as well cognitive sciences are combined in order to understand what *human likeness* means in locomotion, behavior, and intelligence. Following the ambitious goal to enable humanoid robots to work with or even replace humans for certain tasks, for example in households, hospitals, factories, at disaster sites or even space missions, different research questions arise. Within this thesis, the focus is on the locomotion aspect of humanoids, i.e., bipedal locomotion in the form of walking in different situations and environments.

From the viewpoint of an applied mathematician, methods of model-based dynamic optimization or optimal control emerge quite naturally as the tools of choice to analyze, synthesize or reverse-engineer motions of humans and humanoids alike. Following the idea of optimal control, the process of interest, here a human or humanoid robot, is analyzed and a mathematical dynamic model of the process is derived, usually from first principles, with the ability to describe the timely evolution of the system, most commonly in the form of ordinary differential equations (ODEs) or differential algebraic equations (DAEs). When a reasonable dynamic model is available and is validated in simulation, the wish to find an optimal behavior of the model under consideration naturally arises next. This directly leads to the formulation of optimal control problems (OCPs) and the search for suitable methods of optimization to discretize and numerically solve them.

In the recent decades, the robotics community showed a growing interest in using methods of optimal control for behavior and motion generation as well as control as recent publications and workshops show. Following the assessment of the *IEEE Robotics and Automation Society* technical committee of *Model-based Optimization for Robotics*, c.f. [124], a consortium of experts who try to promote the application of model-based optimization in robotics, that "*in a wide range the methods currently applied are behind the current state of the art within the optimization community and therefore only very simple optimization problems can be solved. While some articles present advanced optimization approaches for even complex robotic systems as well as tasks, which covers impacts, varying constraints as well as interaction with the environment, a major challenge is still the actual realization of these approaches robustly on the real robotic platform. In this way, full exploitation of the optimization potential of existing robot platforms is not possible.*"

In the spirit of the *KoroiBot*[1] project, the way to fully utilize and/or additionally extend the humanoid robot's ability to dynamically walk, best possibly in human-like quality, lies in the development of novel motion generation and control methods for existing

[1] Visit the project's web page at http://orb.iwr.uni-heidelberg.de/koroibot/.

hardware. The human gait is at the same time efficient, robust, and versatile. In contrast to the human, the gait presented by today's bipedal robots are at best good in one of these criteria. In addition, they lack the ease and elegance of their human counterpart. While this problem is on the one hand linked to the current hardware, on the other hand it results from the applied control principles and the used software in an even larger extent. There are several reasons why the realization of bipedal locomotion on the actual hardware is still an open research questions, including redundancy, underactuation, stability of locomotion, hierarchy of tasks, predictability, perception and cognition to name only a few.

This doctoral thesis in applied mathematics is to be understood as one new step towards the actual realization of real-time optimal feedback control or nonlinear model predictive control (NMPC) for whole-body motion generation in both simulation as well as on the robotic hardware itself.

Related Work

The broad scope of this thesis requires to state the references of the state of the art in their respective context. Therefore, we embed the contributions of this thesis in the respective field by giving a brief overview of the state of the art in the following sections.

Efficient Derivative Evaluation for Rigid Body Dynamics

The application of the above mentioned rigid-body dynamics (RBD) of complex multi-body system (MBS), the mathematical representation of the humanoids, as dynamic model in the context of NMPC demands algorithms to efficiently evaluate the kinematics and dynamics quantities. State-of-the-art algorithms that enable these evaluations are based on efficient recursive algorithms that are tailored for the kinematic tree topology of the humanoids' MBS , c.f. [48]. There are several open- and closed-source codes available that implement a fast evaluation of RBD of MBS with tree topology, e.g., *Simbody*[2], *drake*[3], *Metapod*[4], and *RBDL*[5]. However, the application of RBD in the context of direct optimal control, as in the scope of this thesis, requires not only the efficient evaluation of the nominal dynamic quantities but also requires the evaluation of at least first order derivative information. While numerical differentiation (FD) can always be applied on those algorithms by treating them as *black boxes* the application of algorithmic differentiation (AD) requires the algorithms to be available as source code that is then augmented to also compute derivative information. However, the investigation of the mathematical properties and the underlying mathematical basis of the algorithms allows to derive tailored code for fast and efficient evaluation of the derivatives of RBD of MBS.

Literature presents approaches heading into the direction of full support of the derivative information of the basic quantities of RBD. One of the first approaches investigating derivative evaluation employs a *Lie* group formulation for rigid-body dynamics as well as the evaluation of the dynamics with respect to a scalar parameter. This approach is implemented in the *C++* library *GEAR*[6], which is based on the technical report [94]. In [57] a derivation based on *Lagrangian* formulation of the derivatives of common RBD quantities is presented without numerical results. While *drake* is the only library already

[2] https://github.com/simbody/simbody [3] https://github.com/RobotLocomotion/drake
[4] https://github.com/laas/metapod [5] https://rbdl.bitbucket.org [6] https://github.com/junggon/gear

offering an AD interface using template programming, the others do not yet enable an efficient derivative evaluation besides FD.

Recent approaches on the evaluation of derivatives for rigid-body dynamics are presented in [23, 58, 134, 184]. Regarding [184], the respective implementation is realized in the *C++* library *MBSlib*[7], which uses an off-the-shelf AD tool *ADOL-C*[8] based on operator overloading to evaluate derivatives of the dynamic computations. Following the same idea of applying AD to the RBD evaluation code, [58, 134] applies methods of source code transformation to the code generated from their RBD dynamic control toolbox *ct*[9].

In this way, code for the nominal as well as the derivative evaluation is generated, which is superior compared to its FD counterpart. [23] presents efficient derivative computations by differentiating the recursive algorithms within the RBD framework *Pinnochio*[10] [25] and using insights in the structure of the derivative arising for rigid-body systems to obtain an implementation of the recursive *Newton-Euler* algorithm (RNEA), computing the inverse dynamics, that is able to efficiently evaluate derivative information. The derivatives of the forward dynamics are evaluated by means of the implicit function theorem.

None of the above mentioned approaches proposes and implements the evaluation of derivatives of the contact dynamics or contact impulses occurring from the RBD of MBS subject to unilateral constraints and collisions, which requires tailored linear algebra to solve the respective descriptor form of the equation of motion for the evaluation of the dynamics of the system.

Nonlinear Model Predictive Control for Whole-body Motion Generation

The state of the art for whole-body motion generation based on optimal control or NMPC employs different reduced models to achieve motion generation in real-time. In Figure 0.1 an overview of the most frequently used versions is schematically depicted.

The most used variant of model reduction is the projection of the dynamics of the humanoid to its center of mass (CoM) or, more precisely, to reduce its dynamics to the motion of an inverted pendulum with its mass located at the CoM and the pivot located at the zero-moment point (ZMP). Subject to strong simplifications, this model is known to fulfill the real-time constraints dictated by the on-board computational power of today's robots.

Recent approaches use the centroidal dynamics of the humanoid, i.e., the projection of the whole-body dynamics onto the CoM such that the contact forces determine the motion of the CoM as well as the total change of momentum. While including all degrees of freedom (DoFs) of the dynamic model, the inherent sparsity of the resulting dynamics allows to derive real-time feasible control strategies.

Employing the whole-body dynamics of the humanoid including the selection of possible contact points for motion generation is a demanding task and it is not yet possible to solve the resulting problem in real-time.

Center of Mass Dynamics

The idea of interpreting the motion of a humanoid by means of an inverted pendulum is related to the fact that this is a good approximation of the CoM movement of a humanoid during the swing phase, where the body rolls over the support leg for the next step. The

[7] https://github.com/SIM-TU-Darmstadt/mbslib [8] https://projects.coin-or.org/ADOL-C
[9] https://bitbucket.org/adrlab/ct [10] https://stack-of-tasks.github.io/pinocchio

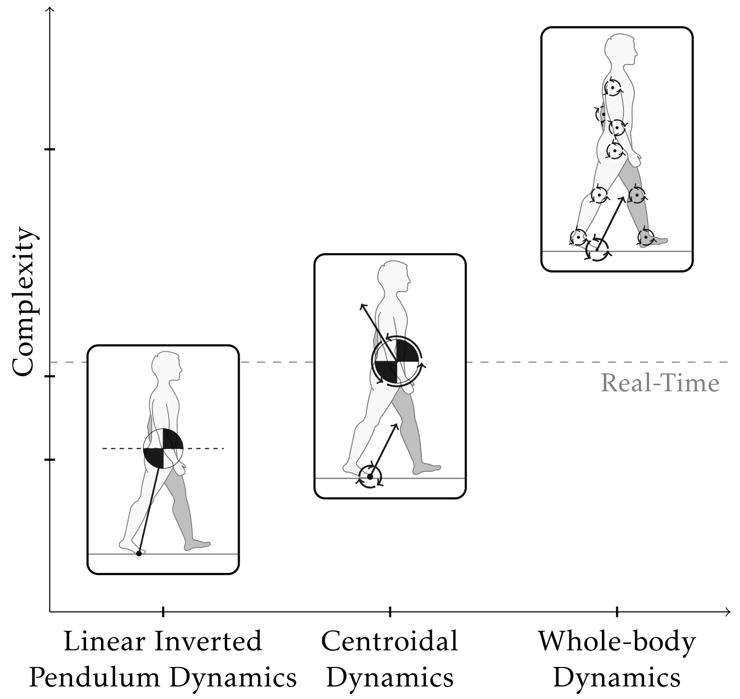


Figure 0.1: Schematic overview of common reduced model variants as well as whole-body dynamics for motion generation and their computational complexity. The scheme shows the two common reduced model variants commonly employed for motion generation and their hierarchy in terms of computational complexity. The least demanding reduced models are based on the motion of an inverted pendulum. A more demanding alternative are models based on the centroidal dynamics of the humanoid. Most demanding is the application of the whole-body dynamics of the humanoid. While pendulum and centroidal dynamics are known to perform in real-time (dashed line), employing the whole-body dynamics for motion generation is still open research.

nonlinearity of the dynamics of the inverted pendulum still imposes problems on the formulation of an online motion generation algorithm.

In [83] *Kajita* proposed a further reduction of the dynamics by limiting the evolution of the CoM motion to a plane in parallel to the ground and named it linear inverted pendulum model (LIPM). Furthermore a scheme is presented to generate a CoM motion from a given reference ZMP trajectory. This CoM trajectory can then be applied to generate a stable whole-body motion of the humanoid. When a chosen model allows an analytical solution, the motion planning can be carried out quickly [69, 129]. However, these formulation are based on specific assumptions that render their extension to other walking tasks difficult and tedious.

The issue with proper ZMP reference generation was solved by *Herdt et al.* in [71] by additionally including the foot step placement into the convex optimization and solving for both the CoM as well as the ZMP trajectory using the LIPM. In this way, the range of possible tasks could be extended. This approach could then be applied to implement visual servoing in [39, 40]. The extensions proposed in [137] can handle steep slopes of up to 10° and demonstrate the humanoid robot *HRP-2* can recover from obstacle collisions.

Another approach that employs the extensive simplifications based on the LIPM but with extensions improving the robustness of the resulting motion, e.g. step timing optimization and divergent component of motion offset, can be found in [92].

In [78], *Ibanez* proposes based on the same reduction principles a mixed-integer formulation of the problem that, for the first time, does not prescribe the support foot sequence. In this way, the flexibility and reactivity of the formulation could be improved by sacrificing the real-time feasibility due to the integrality constraints. A pure planning approach based on mixed-integer quadratic programming is presented by *Deits* in [33], who employs a template model to plan foot steps on cluttered terrain.

First, above mentioned approaches employing the LIPM enable robots to walk in different scenarios, freely optimize the foot steps and avoid obstacles. However, none of them combines the advantages of all approaches, i.e., combining the optimization of position and orientation in a single problem formulation that also enables collision avoidance, and provide a dedicated solver that allows its solution in real-time on the robotic platform.

Second, the above mentioned approach employing mixed-integer optimization allow the planning of whole body motions but are either limited to that (second approach) due to not considering the dynamics of the humanoid or restrict the range of motions due to a lack of generalization (first approach). To this end, no approach combines both aspects in order to have the simultaneous planning while guaranteeing a dynamically feasible motion based on the LIPM.

Centroidal Dynamics

In their survey [72], *Herr* and *Popovic* investigate the behavior of the angular momentum during human gait and show that active generation of angular momentum is a key strategy for bipedal maneuverability and stability. Furthermore, they discuss the limitations of inverted pendulum models, for example the LIPM used in the previous chapter, and conclude that while certain assumptions hold, i.e., that "*a realistic prediction of the dynamics is only possible by including further effects like the mechanical behavior of the legs and their interaction with the respective contact surfaces*".

While their analysis of the angular momentum showed an important effect on the walking capabilities, for motion generation these effects are mostly neglected to simplify the resulting models and speed up the computations. However, more recent approaches try to include these effects into the dynamics used for the motion generation.

In literature, dynamic models that include effects of angular momentum on the motion of the CoM of a MBS are named *centroidal dynamics*, see [140] for a comprehensive article. This usually involves considering the ground reaction forces and moments as well as their effect on the CoM movement. In [140], the authors derive efficient algorithms for the computation for centroidal quantities by means of spatial algebra. From this, they derive a balancing controller able to keep a humanoid subject to external perturbations balancing on a narrow beam.

One of the first approaches considering the centroidal dynamics can be found in *Hirukawa et al.* [74]. Here, the authors consider a set of contact points, the velocity of the feet, the hands, the free-flyer, an initial guess on the CoM, and a heuristic to distribute the amount of forces on the contact points. From this knowledge, they proposed a pattern generator that satisfies the friction cones using so-called resolved momentum control in order to track reference CoM accelerations and its angular momentum to compute a dynamically feasible motion for the CoM. In [141], a stabilization process based on the work of *Cheng* [77] is proposed, which assumes a quasi-static motion, i.e., accelerations are set to zero. However, this condition imposes strong restrictions on the possible motions.

The generation of whole-body motions utilizing multiple contacts between the robot and its environment extends the form of bipedal locomotion with a potential high impact on the functional range of humanoid robots. It enables a robot to climb ladders, perform crawling, evolve in cluttered environments and, less impressive, but yet very useful, to climb stairs. In [6], a multi-contact control scheme is presented that relies on a simplification of the centroidal dynamics and uses a pre-computed set of contacts. In contrast to this, in [31] an approach is presented that considers the full centroidal dynamics for motion generation together with both pre-computed contact configurations as well as a trial to resolve the force-contact complementarity in the optimization.

A similar approach is presented in [145], where the authors propose an approximation of the nonlinear constraints due to the centroidal dynamics. This is based on their previous work [73, 144] that applies a mixed-integer formulation related to [33] to plan contact stances together with contact forces based on an approximation of the centroidal dynamics in the form of a convex formulation.

Thus far, none of the approaches addresses a simplification to the core of the centroidal dynamics for planning of CoM trajectories and force distributions that, together with whole-body stabilizing control, enables the whole-body motion generation close to real-time on the embedded hardware on the robot. Furthermore, the verification of the approach on the robotic platform is mostly missing.

Whole-body Dynamics

Approaches that use the whole-body dynamics in order to generate motions for humanoids online and are validated on the respective hardware are not yet found in literature. However, some approaches that come close to this goal are mentioned below.

The locomotion problems described in [128], that include multiple contacts and consider the whole robot model over a time horizon, are not yet solvable in real-time and strongly depend on the models used to represent the physics. As presented in [127], a realization of the results required then tailored methods of model-robust optimization, as the nominal solution could not be applied otherwise.

As already mentioned above, the group of *Todorov* presented examples of optimal control applied to full-scale models of humanoids for example [167, 168]. This work is based on differential dynamic programming (DDP) techniques and a dedicated physics simulation [171]. In [42], they present a system that is supposed to achieve real-time model predictive control for humanoid robots, but their claim is not validated directly on the robotic hardware but in simulation only.

The principle approach was then ported to the humanoid robot *HRP-2* in [168], where the methodology of DDP was extended to include box constraints on the controls. While the proposed control architecture is now closer to real-time feasibility, as it was recently applied to *HRP-2* [101], it still requires powerful multi-core CPUs, which limits its integration on humanoid robots due to heat and power consumption. Despite numerous efforts to address this large scale nonlinear problem with roughly ten thousand variables [31, 101], no solution yet exists to generate physically consistent controls in real-time using humanoid robot embedded computers.

In [27], *Chrétien* presents a motion planning approach leveraging massive parallelization on graphic cards to make the resulting optimal control problem tractable in terms of computational time. They employ an inverse dynamics formulation together with a collocation approach to formulate a motion as nonlinear program (NLP) that is solved using an off-the-shelf solver. In this way, for motions in a fixed contact configuration on a horizon of up to 5 s, the evaluation time of nominal and derivative information of the dynamics required for a single iteration of the solver could be lowered below 1 s in simulation.

While the above approaches show the state of the art in whole-body motion generation using methods of NMPC, none of them allows yet an execution of dynamic whole-body motions on the robot in real-time.

Combining Nonlinear Model Predictive Control & Reinforcement Learning

In principle, with the goal to solve optimal control problems to synthesize humanoid motions online, there exist two common approaches to control nonlinear dynamic systems while coping with uncertainties in the form of model-plant mismatch, NMPC and reinforcement learning (RL). RL has been proven suitable as a real-time closed-loop control concept in robotics [97], while NMPC already starts to become the standard in industry [147]. As mentioned above, NMPC in robotic applications, especially humanoid robotics and bipedal walking, is still an open research field.

In robotics, the model-plant mismatch is of special interest, because the realization of a task by applying motions to the robot in open-loop requires special care in the form of robustification [127], self-stabilizing motions [125] or very explicit knowledge of the mechanical and control architecture of the robot [100]. While there are examples of the application of NMPC as well as RL for walking tasks, an analysis of the strength and weaknesses or an investigation of a possible combination for systems subject to structural uncertainties has rarely been done, especially not in a systematic and quantitative way.

A study of the influence of the RL reward function on steady-state error was performed in [41]. It was shown that direct translation of a quadratic objective function used in standard linear-quadratic regulator (LQR) resulted in a consistent, though not acceptable steady-state error. In contrast, using the absolute-value reward function yielded a response with negligible error.

A comparative study for ideal systems, i.e., not subject to any uncertainties, can be found in [43]. The authors highlight similarities of NMPC and RL, including optimality of methods, truncation of a time horizon, and continuous vs. discrete actions. They show that, for an electrical power oscillations damping problem, NMPC slightly outperforms RL, yet both policies were essentially similar. Furthermore, the authors propose the idea of combining RL and NMPC. In an on-line, (local) optimal mode, NMPC could start optimization from the initial guess of the optimal trajectory pre-computed by RL in an off-line, possibly globally optimal mode.

A distinction of both NMPC and RL was observed and successfully realized in a hybrid approach, a variant of the Guided Policy Search algorithm [112]. The approach was able to learn obstacle avoidance policies for a quadrotor [186]. It adopted a collection of model predictive control roll-outs obtained under full state observation and trained a deep control policy that required only the on-board sensors of the vehicle.

A combination of RL and NMPC in one framework has the benefit of allowing RL to gather the required experience without damaging a many-degree-of-freedom system, where NMPC acts as a safe-guard. The experience is used by RL to compensate the difference between the internal model of the system and the real one. While any model-based nominal controller is suitable for acting as a safeguard, the complexity of humanoid robots motivates to use advanced methods of NMPC for this role.

This can be compared to adaptive internal model control (IMC) [32] from a control theoretic viewpoint. The implementation requires an explicit model of the plant to be used as part of the controller. However, in adaptive IMC, the structure of the unknown system is determined offline, while its parameters can be inferred by online parameter estimation [107]. A particular shortcoming is that the structure needs to be identified precisely, otherwise a model-plant mismatch remains.

In principle, any model-free RL algorithm such as [87, 156–158] can be used. However, lack of safety measures, i.e., the prevention of actions that cause damage to the system, for example, not falling in bipedal locomotion, and sample complexity of the algorithms limits their application to real systems.

Learning the forward model of the system demonstrates the lowest number of interactions with it [63, 88]. Learning the inverse model [28, 91] assumes that the model can connect successive states prescribed by the nominal controller.

When the approximate model of the system is available, it is possible to pre-train the initial policy, which can speed up learning. In [46] a two-step sequential approach is proposed. First, an iterative linear-quadratic-Gaussian algorithm is used to design an initial policy. Then, the policy is refined using the PI^2 algorithm on the real system. Another approach is to iteratively learn the difference model of the measured state and the state obtained on the approximate model and adopt this difference model for improving the policy [66, 153]. Finally, the authors in [148] use an ensemble of slightly perturbed model parameters to learn a robust policy.

Learning involving offline planning or human-expert demonstrations [1, 112, 143, 155] constrains the problem space, thus reducing hardware damage. This option requires either a hand-coded suboptimal policy or a skilled human operator.

For a bipedal robot where any failure can be catastrophic, the above methods are not suited even given a good starting policy, because it is likely that RL will result in at least several failures during subsequent policy improvement episodes. It is possible to guarantee safe learning when one can either predict repercussions of bad actions [123] or has a backup policy to lead the system back to safe states [54, 67].

Up to the present and in contrast to this thesis, the comparative analysis and benchmarking of both model-based and model-free methods of optimal control are not present in the current literature. Furthermore, the investigation of a beneficial and constructive combination of both approaches to optimal control for more than simple examples or, needless to say, even the implementation on the respective robotic platform in real-time as proposed in this thesis.

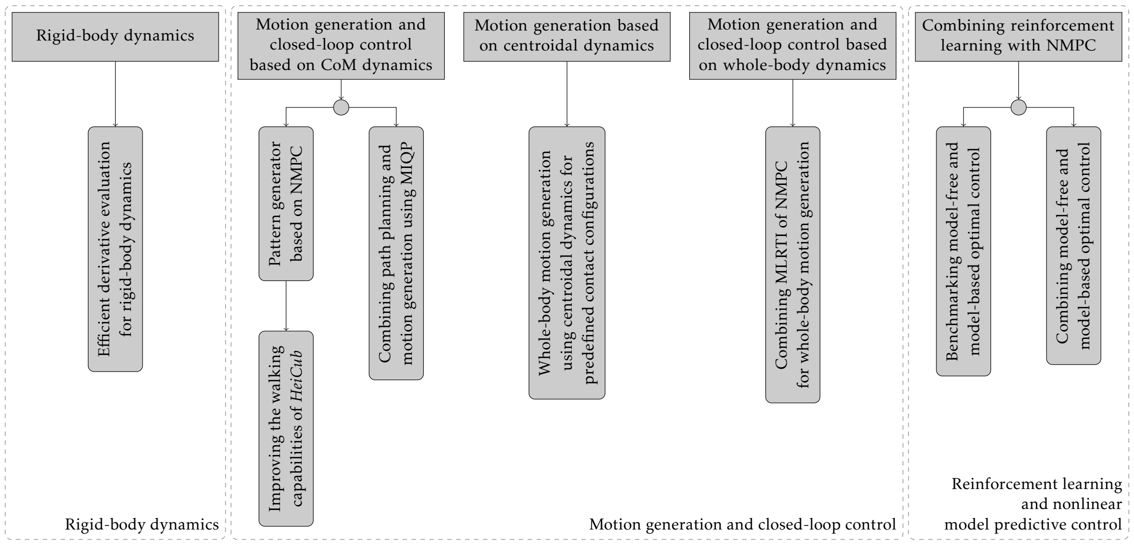

Rigid-body dynamics
Efficient derivative evaluation for rigid-body dynamics
Rigid-body dynamics
Motion generation and closed-loop control based on CoM dynamics
Pattern generator based on NMPC
Improving the walking capabilities of HeiCub
Combining path planning and motion generation using MIQP
Motion generation based on centroidal dynamics
Whole-body motion generation using centroidal dynamics for predefined contact configurations
Motion generation and closed-loop control based on whole-body dynamics
Combining MLRTI of NMPC for whole-body motion generation
Motion generation and closed-loop control
Combining reinforcement learning with NMPC
Benchmarking model-free and model-based optimal control
Combining model-free and model-based optimal control
Reinforcement learning and nonlinear model predictive control

Figure 0.2: Overview of the contributions of the thesis to different subjects of nonlinear model predictive control for whole-body motion generation of humanoids.

Aims and Contributions of this Thesis

The aim of this thesis is to to make one new step towards real-time feasible NMPC for the optimal closed-loop motion generation of humanoids modeled as multi-body systems (MBSs) being realizable on today's humanoid robots. To this end, this thesis presents novel results and advances over previously established techniques in a number of areas.

The author of this thesis contributed to several publications during the process of creation of this work. In the remainder of this section, we give a brief overview of the respective contributions, refer to the publications in which they appeared, describe their content, and clarify the contribution of the author of this thesis.

Rigid Body Dynamics

Within the scope of this thesis, we contributed to the efficient handling of rigid-body dynamics (RBD) in the context of optimal control. We propose novel methods for the efficient evaluation of derivatives of the required RBD quantities for the solution of optimal control problems for motion generation of humanoids. We describe this contribution and its advances over related work in the following paragraph.

Efficient Derivative Evaluation for Rigid Body Dynamics

The state of the art in today's robotics applications for the evaluation of common RBD quantities are recursive algorithms exploiting the kinematic tree topology of the robot modeled as MBS, c.f. [48]. While this is enough for simulation purposes, derivative-based optimization, as, for example, in direct optimal control, requires at least first-order derivative information.

The approaches for the evaluation of derivative information presented in the literature proof that a significant speed up in the evaluation of derivative information can be gained for the well-know recursive algorithms in contrast to performing numerical differentiation (FD). This speed up is possible due analyzing the internals of the recursive RBD algorithms and implementing dedicated code for the derivative evaluation by either manually derivation or automatically by applying algorithmic differentiation (AD). However, to this end, MBS subject to kinematic constraints and collisions that require matrix factorization techniques to evaluate the RBD have not been addressed before.

In this thesis, we propose novel algorithmic augmentation of state-of-the-art recursive algorithms in order to additionally evaluate the first-order forward derivatives. By following the principles AD and extending the theory to vector-valued elementary operations, we propose to treat the recursive algorithms as concatenation of elementary operations. In this way, the efficient analytic derivative of each elementary operations can then be successively propagated by means of the chain rule. This yields the correct derivatives of some of the well-known algorithms, e.g. the recursive *Newton-Euler* algorithm (RNEA) or the composite rigid-body algorithm (CRBA). Furthermore, we propose an extension of the approach to the linear algebra of contact dynamics, which involves matrix factorization to solve both the respective descriptor form and the conservation of angular momentum equation on collisions of the considered MBS.

The proposed algorithmic augmentations are implemented in the freely-available open source library *Rigid Body Dynamics Library* (RBDL) [49, 50], which realizes the state-of-the-art algorithms in an efficient template-based *C++* code. The proposed approach is thoroughly tested against its FD counterpart on benchmark examples. The applicability of the developed derivative evaluation for direct optimal control is shown for a lifting motion of a human model. Furthermore, we demonstrate the benefit of appropriate sparsity exploitation.

We based parts of our journal article [105] on the respective chapter 2 of this thesis. In this article, we present how an state-of-the-art code for the evaluation of RBD can be augmented following the principle of AD to also evaluate the derivatives. My contribution is the derivation of the theoretical part of the code transformation relying on the principle of AD as well as major parts of the technical implementation. I set up all the benchmark problems as well as the optimal control applications. My technical implementation is available as a fork[11] of the well-known RBDL of *Felis*, c.f. [50]. I was responsible for major parts of the article, the theoretical background, the efficient implementation of AD code of the core RBD algorithms and the comparison of the results in numerical benchmark examples as well as their applications in the optimal control context. Chapter 2 contains further mathematical clarification and the mandatory theorems to proof the correctness of the technical implementation.

Motion Generation based on Reduced and Whole-body Dynamic Models

In the context of this thesis, the evaluation of different model reduction approaches of MBS were performed. This led to the development of motion generation algorithms based on these model reductions, which are suitable for an actual realization on the robotic platform in real-time. We describe each contribution and its advances over related work in the following paragraphs.

Center of Mass Dynamics

In this thesis, we propose a framework for the generation of whole-body motions and closed-loop control of humanoids based on walking pattern generators (WPGs). We propose a reformulation of the state-of-the-art WPGs, which separate the optimization of positional and rotational degrees of freedom (DoFs) of the linear inverted pendulum

[11] https://bitbucket.org/mkudruss/rbdl/src/dev/

model (LIPM) and the foot placement. In contrast to this two-step approaches to WPG, our combined but nonlinear reformulation is able to find simultaneously foot-step positions and orientations. Additionally, the proposed formulation allows to directly include obstacle avoidance into the formulation by the special nonlinear constraints.

While the treatment of nonlinearities in WPG is seen to hinder their execution in real-time, we propose a tailored solver of the nonlinear problem based on NMPC real-time iterations that provide fast feedback computation for a practical realization. In this way, we show that the whole motion generation chain including our WPG formulation runs in real time on the embedded hardware of the humanoid robot *HRP-2* of *Laboratory for Analysis and Architecture of Systems*, Toulouse, France. Furthermore, we propose an adaption of the algorithm to run in real time on the external computing hardware of the humanoid robot *HeiCub* of *Heidelberg University*, Heidelberg, Germany.

Parts of the work of this thesis were published in our journal article [133]. We present an extension of the state of the art in linear model predictive control (LMPC)-based motion generation. By combining linear and rotational DoF of the simplified model a nonlinear least-squares problem is derived. Following this, we propose a tailored algorithm based on NMPC principles, efficiently implemented and evaluated on the robotic platform of *HRP-2*. My contribution is the derivation of the NMPC algorithm, the adaption of a suitable solver and the efficient implementation in *Python*. The experiments presented in the article were conducted by the co-authors of *Laboratory for Analysis and Architecture of Systems*, Toulouse, France.

Other parts of this work were published in our conference article [163]. In this article, we present the improvement of the walking capabilities of *HeiCub* by employing a closed-loop control concept based on the above mentioned pattern generator based on NMPC presented in our journal paper [133]. The improvements are documented by comparing key performance indicators of the novel control scheme against the state of the art, and show a significant superiority of the novel scheme. I contributed to this article by adapting the pattern generator for closed-loop control concept and the implementation on the robot platform *HRP-2* to the humanoid robot *HeiCub*. Furthermore, I contributed to the experiments jointly conducted with the co-authors of *Heidelberg University*, Heidelberg, Germany.

Additionally, this thesis presents a novel mixed-integer formulation for a pattern generator based on the same principles as the previously mentioned contributions. While our previous work required the hard-coding of contact sequences and support phases, the formulation presented within this work implements fully automatic foot step placement based on a simplified complementarity formulation together with an automatic contact surface selection for cluttered environments as well as an improved obstacle avoidance strategy.

Centroidal Dynamics

In this thesis, we propose a framework for the generation of whole-body motions of humanoids for predefined multi-contact supports based on the centroidal dynamics of the humanoid. We formulate a reduced version of the multi-contact centroidal dynamics of a humanoid by means of an ellipsoid model. In this way, we are able to separate the effects of actuated and non-actuated DoFs on the change of total momentum at center of mass (CoM) level, which reduces the dimension of state space drastically.

The reduced model incorporates only the DoFs of the floating-base as states, hence reducing the overall computational complexity of the approach and qualifying for being real-time feasible. The derived reduced model includes the major effects on the under-actuated part and generalizes in the number of contacts. Following this, we extend the range of tasks, e.g. level ground walking, walking non-flat floor, multi-contact like using the handrail during stair climbing.

We apply the approach in order to generate a whole-body motion for the humanoid robot platform *HRP-2*, where we realize a stair climbing task with additional handrail support. In this way, for the first time, we made *HRP-2* climb stairs of 15 cm height repeatedly. We show, that by leveraging handrail support the overall motor power consumption is reduced by 25 %.

Parts of this work were published in our proceedings article [106]. In this article, we propose an approach to whole-body multi-contact motion generation based on the centroidal dynamics of a humanoid. We achieved real-time feasibility by further reducing the complexity of the centroidal dynamics by separating terms that depend on the joint motions. In this way, optimal CoM trajectories as well as contact forces are computed by means of optimal control (OC). I was responsible for deriving the reduced centroidal dynamics model, formulating the respective optimal control problem (OCP) for walking and stair climbing as well as fine-tuning the motion for execution on the robot *HRP-2*. The experimental results were conducted by the co-authors at *Laboratory for Analysis and Architecture of Systems*, Toulouse, France.

Nonlinear Model Predictive Control for Motion Generation of Humanoids

The vast complexity of the OCPs based on the whole-body dynamics of the respective humanoid still hinders the application of these formulations in a online closed-loop control context. While it is possible to apply reduced versions of the dynamic model in combination with other strategies, the application of whole-body dynamics for NMPC have not yet been realized on the robotic platform or at least not yet in real-time.

In this thesis, we propose the combination of advanced methods of multi-level real-time iteration (MLRTI) for NMPC to realize fast feedback control for whole-body motion generation. From the analysis of the computational time of a task, we show that the feedback phase of NMPC can be performed many times faster than the rather slow preparation phase that computes the evolution of the respective dynamic system. From this knowledge, we present a thread-based implementation that separates both phases such that fast feedback using the last linearization can be queried while the preparation phase is evolving the dynamic system.

In this way, we are able to implement real-time whole-body motion generation for the humanoid robot *Leo* of *Delft University of Technology*, Delft, the Netherlands. The contribution of this thesis is the derivation, the efficient implementation of the NMPC strategy as well as the realization on the robot itself. The experiments on the humanoid robot *Leo* were jointly conducted with *Koryakovskiy* at *Delft University of Technology*.

Combining Nonlinear Model Predictive Control & Reinforcement Learning

Within the scope of this thesis, we propose to benchmark model-based and model-free methods of optimal control for motion generation with respect to model-plant mismatch, deriving a hybrid control strategy and evaluating it on the actual hardware platform. In contrast to the state of the art, our conclusions go beyond the validation of similarity of solutions or computational benefits for ideal models. We provide numerical evidence that under uncertainties, situations may arise in which one or the other method can be favorable for performance. We describe each contribution and its advances over related work in the following paragraphs.

Benchmarking Nonlinear Model Predictive Control and Reinforcement Learning

A major contribution of the thesis is the investigation of a beneficial combination of model-free reinforcement learning (RL) and model-based NMPC for the control humanoid robots. In this thesis, we propose a quantitative comparison of reinforcement learning and nonlinear model predictive control subject to model uncertainties. Using a guiding benchmark example of a cart-pendulum swing-up and balance task, we show differences, trade-offs and pitfalls of the specific problem formulations. In this way, we quantitatively determine break-even points at which the superiority of one method over the other changes.

Parts of this work were published in our journal article [103]. In this article, we conduct a comprehensive comparison of the performance model-free RL and model-based NMPC on an exemplary benchmark example. The presented comparison quantifies the performance of the methods subject to parametric and structural uncertainties in terms of two different criteria, namely the similarity of trajectories and the resulting rewards. For the comparison, a standard benchmark problem was chosen: a cart-pendulum swing-up and balance task. Here, it is demonstrated that NMPC has advantages over RL if uncertainties can be eliminated through identification of the system parameters. Otherwise, there exists a break-even point after which model-free RL performs better than NMPC with an inaccurate model. We find that a combination of both RL and NMPC can be beneficial for real systems being subject to uncertainties.

Koryakovskiy and I equally contributed to this publication. While *Koryakovskiy* implemented the RL problems, my contribution is the implementation of the NMPC problems. The benchmarking was jointly realized by *Koryakovskiy* and *Kudruss*, where all NMPC-related parts were done by me. The discussion of a common problem formulation in order to solve the same problems in the framework of both RL and NMPC as well as the definition of the quantities of interest in the form of key performance indicators (KPIs) were derived in joint work.

Combining Nonlinear Model Predictive Control and Reinforcement Learning

In this thesis, we propose to combine reinforcement learning and nonlinear model predictive control to compensate model uncertainties following our findings from accessing methods of reinforcement learning and nonlinear model predictive control. Therefore, we propose two combination schemes in order to compensate model-plant mismatch induced by uncertainties, which preserve the *Markov* property. The first scheme learns a compensatory control on top of NMPC, while the second scheme learns a compensatory signal from the difference between model-predicted and actual transition. Each of the proposed schemes leverages a model-based controller, here NMPC. We shed light on the mathematical properties of the proposed schemes and establish the theoretical foundation of the approaches by means of two theorems. The same RL framework in which the benchmarking was realized is applied to verify the benefits of combination of both model-free and model-based methods of optimal control. The superior scheme is found from numerical experiments, which we then realized on the robot *Leo* for validation of the approach. Parts of this work were published in our journal article [104].

Thesis Overview

This thesis is structured in three major parts according to the overview of its contributions depicted in Figure 0.2, covering the contributions to rigid-body dynamics (RBD), whole-body motion generation of humanoids, and the combination of nonlinear model predictive control (NMPC) and reinforcement learning (RL).

In chapter 1, we give an introduction to the underlying methods of this thesis. We first introduce the considered robotic hardware throughout in the remainder of this work. Following this, we present how according to the state of the art a dynamic model of the robots can be derived by means of RBD. Locomotion as multi-stage optimal control problem (OCP) as well as the formulation of an NMPC problem considered in this thesis. We present the direct optimal control approach based on multiple shooting as presented by *Bock* in detail. The numerical solution of the proposed OCPs in the form of structured nonlinear programming employing sequential quadratic programming (SQP) methods is presented. From this, we introduce the state-of-the-art methods of NMPC. We explain how we consider the ordinary differential equations (ODEs) of the multi-body system (MBS) of the humanoids in the formulation of the OCP, and conclude with pointers to the succeeding content of this thesis

The work related to RBD is presented in chapter 2. In chapter 2, we present the efficient evaluation of first-order forward derivative information for recursive algorithms for RBD of MBS with tree-topology by leveraging methods of algorithmic differentiation (AD).

The work concerning whole-body motion generation for humanoids with either reduced models or the whole-body dynamics of a humanoid to RBD are presented in chapters 3, 4 and 5.

In chapter 3, we present the work on motion generation based on center of mass (CoM) dynamics. This includes the derivation of the linear inverted pendulum model (LIPM) from the RBD of MBS, the formulation of a reactive pattern generator based on NMPC as well as its application for motion generation of *HRP-2* and *HeiCub*, and the formulation of a pattern generator based on a mixed-integer quadratic program (MIQP) and its application for motion generation of *HeiCub*.

In chapter 4, we present our reduced formulation of the centroidal dynamics together with a framework that enables the generation of whole-body motions for the humanoid robot *HRP-2*.

In chapter 5, we present our novel approach to whole-body motion generation based on a thread-based scheduling of multi-level real-time iteration (MLRTI) of NMPC. Following this, we show results obtained from experiments with the robot *Leo* of *Delft University of Technology*.

In chapter 6, we first give a brief introduction into the methodology of RL. Afterwards, we present our efforts on benchmarking model-based and model-free methods of optimal control by introducing a quantitative and comprehensive comparison of their performance for systems subject to model-plant mismatch in the form of a protocol of a computational study that is performed on a benchmark example. From the knowledge of the conducted computational study, we additionally propose different schemes on the combination of both RL and NMPC in order to address the cooping with model-plant mismatch. This is followed by an evaluation of the schemes in simulation and a validation on the robot *Leo*.

1 From Robots to Rigid Body Mechanics and Optimal Control

In the following chapter, we give an introduction to the concepts on which this thesis is based. We strive to encode the bipedal locomotion of humanoids in the form of an optimal control problem (OCP). The basis of this approach is the description of the humanoid as multi-body system (MBS) subject to impacts and kinematic constraints such that the respective rigid-body dynamics (RBD) result in a discontinuous hybrid dynamic system. By taking into account the specific properties of the human gait, we can formulate bipedal locomotion as multi-stage OCP subject to ordinary differential equations (ODEs), which encode the RBD of the humanoid, and special switching conditions linking the model stages, which encode the discontinuities due to impacts. From this fundamental formulation, we are able to derive different approaches that enables us to solve the problem online on the respective hardware. By hardware, we mean the humanoid robots that are supplied by partners for this thesis and act as target platforms on which the algorithms are executed for testing and demonstration. The derived feedback control concepts are based on nonlinear model predictive control (NMPC), a closed-loop control strategy that solves an open-loop OCP over a finite horizon on-line in order to compute a feedback control from the current system state online in real time.

We introduce the basic concepts of the human gait that are required for this thesis in Section 1.1. Following this, in Section 1.2, we give an overview of the robotic hardware considered in this thesis. We introduce the state of the art of modeling humanoid robots as MBSs in the form of a discontinuous hybrid dynamic system in Section 1.3. Following this, we present how the bipedal gait of humanoids can be formulated as an OCP considering the discontinuous hybrid dynamics of the respective RBD in Section 1.4. Furthermore, we introduce the direct approach of optimal control in the form of multiple shooting in order to discretize the resulting OCPs and briefly discuss how to solve the highly structured nonlinear programs (NLPs) by means of structure exploting sequential quadratic programming (SQP) methods. Finally, we present the fundamentals of advanced methods of NMPC based on the real-time iterations of SQP. Next, we revisit this methodology for system and parameter estimation in the form of moving horizon estimation (MHE) techniques. We conclude this chapter by recapping the presented ideas.

1.1 Bipedal Walking of Humanoids

In Figure 1.1, the human gait is schematically visualized, where we refer to [142] for a detailed overview of human gait characteristics. According to *Vukobratović* in [177], regardless of the structure or the number degrees of freedom (DoFs) of the system under consideration bipedal legged locomotion is characterized by the possibility of rotation of the overall system about one of the foot edges, equivalent to an unpowered (passive) DoF, symmetry of gait or periodicity, both related to regular gait only, and an alternating appearance of single- and double-support phases.

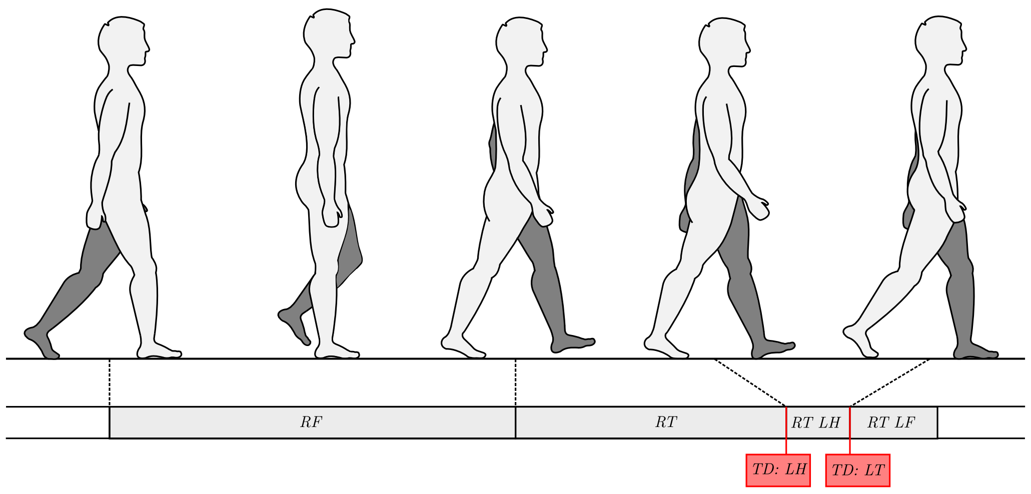


Figure 1.1: The gait cycle of a human represented by a single step of a full stride together with the respective phases, events and contact configuration. From an initial configuration, the single support phase with the right foot fully in contact with the ground (RF) is followed by a change in contact configuration when the right foot rolls over from heel to toe (RF). After the instantaneous touchdown event of the left heel (TD: LH), the double support phase is established with two legs and two contact points on the ground (RT LH) until in another event the left toe establishes contact as well (TD: LT). The step is finalized in double support phase with two contacts (RT LF).

Revisiting Figure 1.1, a gait consists of two different contact configurations. The statically unstable single-support phase comprising the phases (RF, RT), during which one feet is in contact with the ground and the other is swung from the back to the front position. The legs define an open kinematic loop during the single-support or swing phase. This is followed by the statically stable double-support phase, i.e., both feet are in contact with the ground. Here, the legs define a closed kinematic chain. The contact configuration is essential for the system in order to realize locomotion with respect to its environment. As only through contact, an interaction with the environment is possible.

While Figure 1.1 shows only free walking on flat ground with absence of obstacles, human bipedal locomotion is much more versatile than this simple task presents. Humans can freely walk while evading obstacles and coping with different ground conditions, e.g. slippery floor, ice, gravel or sand. Bipedal locomotion of humans is successful also in constrained environments, e.g. utilizing footholds, balancing on bars, or traversing uneven or cluttered terrain. Bipedal locomotion also includes walking with additional hand contacts, such as additional hand support on tables, walls or handrails. The most important task however is to preserve the dynamic balance (or stability) of the considered system during the gait, i.e., not to fall.

The human gait, as realization of an efficient and robust locomotion strategy and a distinctive feature of humanity, represents an ongoing source of stimulation for various fields of research. This involves theoretical studies, motion analysis, simulation as well as practically realized systems from simple planar mechanisms to today's humanoid robots, examples of the most advanced robotic realizations of bipedal locomotion. While the human gait is very robust and humans can withstand pushes as well as large perturbation without falling during walking, this level of confidence is yet to be implemented on the respective hardware platforms of today's humanoid robots.

1.2 Considered Humanoid Robot Hardware Platforms

The ability to perform bipedal walking in an autonomous way is a crucial characteristic of humanoid robots. Following the goal of becoming assistants or helpers for humans or addressing other challenges, e.g. in disaster response scenarios, bipedal humanoid robots are better suited than wheeled robots to move in a human environment, including walking over stairs, navigating rougher terrain and moving over small obstacles. Additionally, robots with a fully anthropomorphic form are also much more likely be to be socially accepted as a companion than other robotic realizations.

In order to consider robotic control algorithms to be successful, an ideal *in-silico* evaluation in simulation is not enough, such that their realization on the actual hardware to validate their performance is mandatory. Therefore, one of the goals of this thesis is that the developed algorithms do not only consider practical knowledge of expert robotic practitioners but, with their help, could be executed on the target platform whenever possible. The interested reader is referred to [160] for an overview of the history as well as the state of the art of robotics. In the remainder of this thesis, three actual robotic hardware platforms will appear, i.e., *HRP-2*, *HeiCub* and *Leo*. We give a brief introduction to each of the humanoid robots in the following section respectively.

1.2.1 The Humanoid Robot *HRP-2* N^{o}4

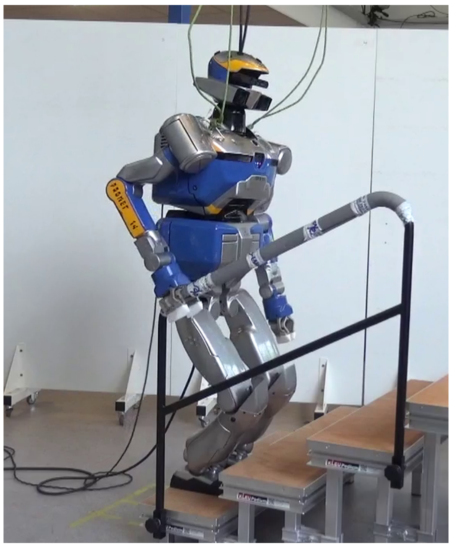


Figure 1.2: HRP-2 robot of *Laboratory for Analysis and Architecture of Systems* during a stair climbing task from our conference article [106].

Within this thesis, one of the humanoid robotic platform, in the focus of this thesis, is *HRP-2* N^{o} 14 located at *Laboratory for Analysis and Architecture of Systems* (CNRS-LAAS) in Toulouse, c.f. [85, 89]. *HRP-2* is a medium human-sized robot of 1.58 m with 30 DoFs. *METI* Japan developed this robot within the *Humanoid Robotics* project, c.f. [75]. We refer the reader to [89] for a detailed technical description of the robot.

The mechanical structure of *HRP-2* can be assumed to be perfectly rigid as presented by [75, 89]. The joints of *HRP-2* are equipped with brushed DC motors. Gearboxes with high reduction ratios are employed on each joint. In this way, higher order dynamic coupling effects to a pure rotational inertia augmentation of the given joint can be neglected, c.f. [93].

The low-level control at joint level of *HRP-2* is realized via high-gain position control in order to achieve tracking of given trajectory profiles. An important part of the control concept of the robot is the passive elasticity in the ankle joint. The ankle-foot mechanism represents a passive spring-damper system according to [132]. During the execution of motions, this ankle-foot mechanism can act as additional stabilizer, c.f. [185].

The robot *HRP-2* acts as an example for the application of our developed motion generation approaches. The developed motion generation approaches based on center of mass (CoM) and centroidal dynamics are realized on *HRP-2* for validation in Chapter 3 and 4 respectively.

1.2.2 The Humanoid Robot *HeiCub*

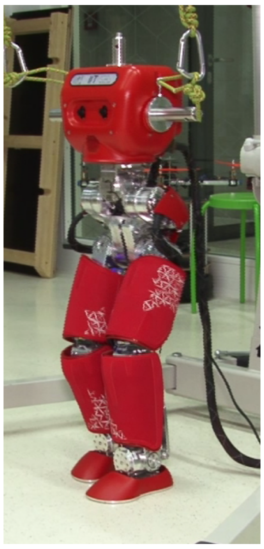


Figure 1.3: *HeiCub* robot of *Heidelberg University* during a walking task.

The *Robotics Lab* of *Heidelberg University* has a reduced version of the *iCub* humanoid robot, called *HeiCub*. The robot was manufactured at the *Fondazione Istituto Italiano di Tecnologia* (IIT) in the context of the *KoroiBot* project. *HeiCub* is a research platform specially designed for walking experiments and is derived from the platform *iCub* [120] IIT.

In contrast to the full *iCub*, *HeiCub* is smaller, i.e., 0.97 m in height. The robot has a total weight of 26.4 kg. The reduced version *HeiCub* consists only of the torso, the pelvis and the legs of *iCub*. While the full platform has 53 DoFs, *HeiCub* only has a total of 15 DoFs. Six DoFs are located in each leg and the remaining three are located in the torso.

The legs have been redesigned to be more powerful. In this way, their functionality is extended to enable advanced motions of the robot, e.g. squatting or walking. They have a length of 0.51 m. The feet are 0.2 m long and 0.1 m wide. Furthermore, both *iCub* and *HeiiCub* can be equipped with deformable and pressure sensitive soft soles that can compensate for uneven ground surfaces. These soles can also be used to measure ground reaction forces to locate the center of pressure (CoP) for closed-loop control.

The robot is equipped with various sensors for perception and proprioception. The robot carries two cameras for stereo vision in the upper torso. In contrast to the full *iCub* that wears the inertial measurement units in its head, *HeiCub* has them installed in the torso. Six-axis force torque sensors as well as optical encoders are installed in each joint of the respective robot leg. *HeiCub*'s legs have a tactile skin allowing that allows to sense applied external forces.

In order to control the robot a PC-104 equipped with an *Intel* CPU is mounted in the torso of *HeiCub*. The operating system is *Linux*, where the firmware implements custom Ethernet or CAN protocol in order to communicate low-level joint control or the sensors. The robot relies on external computational power based on network connection. The middle ware *Yet Another Robot Platform*, c.f. [55], is used to enable transparent communication between robot, simulation and external hardware.

The humanoid robot is used to validate the motion generation approaches based on CoM dynamics presented in Chapter 3.

1.2.3 The Humanoid Robot *Leo*

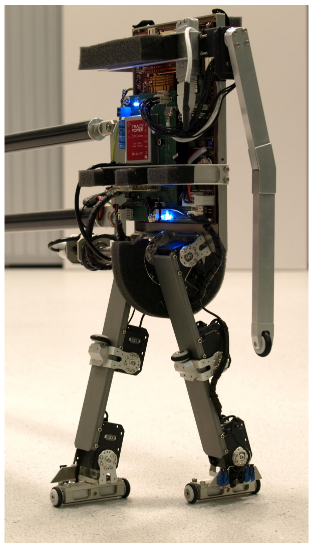


Figure 1.4: *Leo* robot of *Delft University of Technology* during a walking task.

In Figure 1.4, the robot *Leo* of TU Delft is depicted. *Leo* was originally built to perform reinforcement learning experiments directly on the hardware, c.f. [155].

For walking experiments and to enforce a 2D motion around a center platform, robot *Leo* is attached to a boom. *Leo* is small humanoid robot with a height of 50 cm. Its total mass is 1.7 kg. In order to prevent damage from falls of the robot in a wide range of configurations, foam bumpers are installed on each side of the top of the torso as well as between the motors located at the hip. *Leo* has seven DoFs in total, three in each leg at ankle, knee and hip as well as one motor in the shoulder joint. Each of the DoFs is actuated and the actuation is realized by *XM430* servo motors from *Dynamixel* with a maximum torque 3 N m. A reduction of the wear of the in-build motor gearboxes is realized by additional elastic couplings.

Leo is equipped with an *VIA Eden* on-board embedded computer with 1.2 GHz CPU and 1 GB RAM. The operating system on-board of the robot implements a fixed sampling time of 30 ms. The communication between on-board computer and motors is realized via RS-485 serial ports. The motors are used in voltage control mode. They report their position by encoder values for closed-loop control and their temperature for compensation. Joint velocities are retrieved from the position signal by means of numerical differentiation. In order to reduce the noise, the results are filtered by a second-order *Butterworth* filter.

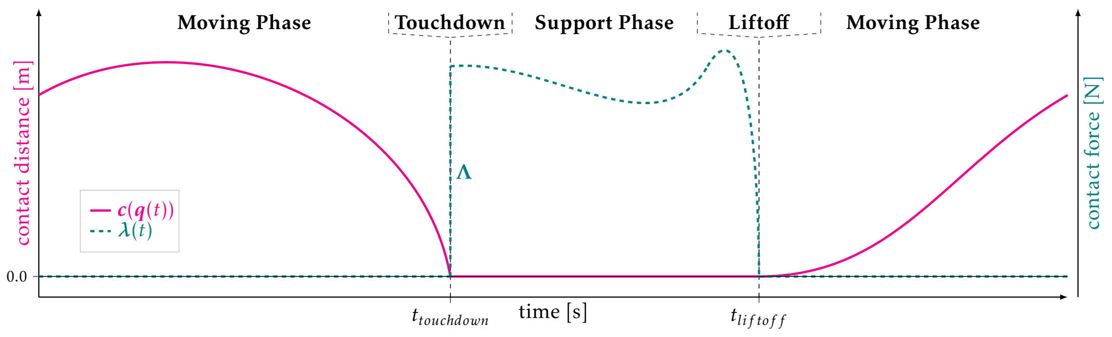


Figure 1.5: Visualization of the impulsive hybrid dynamics of multi-body system with unilateral constraints. In a moving phase, as long as the contact is not established, i.e., the distance to the contact surface (solid line) is larger than zero ($\boldsymbol{c} > 0$), no contact force (dashed line), can be applied ($\boldsymbol{\lambda} \equiv 0$). A touchdown takes place as soon as a contact is established at $t_{touchdown}$ ($\boldsymbol{c}(t_{touchdown}) = 0$), then an impulsive force $\boldsymbol{\Lambda}$ enforces a non-penetration of the contact surface and stops the respective motion ($\dot{\boldsymbol{c}}(t^{+}_{touchdown}) = 0$). During the support phase, a non-zero contact force $\boldsymbol{\lambda} > 0$ is applied, while the contact distance remains zero ($\boldsymbol{c} \equiv 0$). As soon as the contact force vanishes at $t_{liftoff}$ ($\boldsymbol{\lambda}(t_{liftoff}) = 0$) the contact can be released and can leave the contact surface ($\boldsymbol{c} \leqslant 0$) again. During the moving phase the force stays zero, while the contact can freely move.

The 2D robot *Leo* is used as benchmark platform for the motion generation using advanced methods of NMPC as presented in Chapter 5. Furthermore, we validate the hybrid control strategies combining methods of NMPC and RL on robot *Leo*, c.f. chapter 6.

1.3 Humanoids as Rigid-body Models

Revisiting Figure 1.1 of the introduction of bipedal locomotion in Section 1.1, it is clear that the interaction of the humanoid with its environment plays a crucial role for its locomotion abilities. Assuming the mechanical structures of the above mentioned humanoid robots to be perfectly rigid, the state of the art models them as MBS and the respective dynamic models can be derived by means of RBD. We refer to [19, 48] for the technical details and follow our introduction of the required definitions from [105] in the remainder of this section.

In Figure 1.5, the hybrid discontinuous nature of the RBD of MBS subject to unilateral contacts is depicted. The figure shows two model stages, Moving Phase and Support Phase, each representing different contact configurations of an MBS. The model stages differ in the dynamics of the MBS resulting from the different contact configurations representing the hybrid nature of the MBS dynamics. We describe the dynamics of a MBS in contact with its environment by means of kinematic constraints in the following sections. The model stages are connected by two infinitesimal events, Touchdown and Liftoff, indicating the transition from one contact configuration of the MBS to another one. In contrast to the Liftoff event, before the Touchdown event the system's contact motion perpendicular to the contact surface is immediately stopped such that a jerk is transmitted through the structure of the robot. This discontinuity on velocity level represents the impulsive nature of the dynamics of an MBS subject to unilateral constraints, which we introduce in detail in a later . Afterwards, the contact's perpendicular motion with respect to the contact surface is fixed such that it does not move during the support phase except the contact is released or the contact begins to slip due to lack of friction. The behavior of the system in Figure 1.5, i.e., that either a force is applied or the contact can move, represents the complementarity between force and contact described later in more detail.

Multi-Body Systems and Contact Dynamics

In the case of humanoid robots, we model them as underactuated MBS using a free-floating base to model the position and orientation of the robot with respect to its environment. Furthermore, we assume the MBS to consist of n^{B} connected rigid bodies. Each connection consists of two bodies and the connecting joint which defines their relative motion, e.g. prismatic, rotational joints or their combination to multi-DoFs, where we assume each joint to be actuated, i.e., can generate a generalized τ along or around the motion axis. We exclude kinematic loops, except due to contacts with the environment, and presume a *root* body that can be identified as the torso of the robot. The position and orientation of the root body with respect to a global coordinate system is modeled as the above mentioned free-floating base with additional six non-actuated DoFs. The total n^{DoF} DoFs of the MBS is the sum of the six non-actuated DoFs of the free-floating base and the actuated n^{Act} DoFs defined via joints. In this way, the humanoid can be described as MBS with a kinematic tree topology by enumerating each body from the root up to n^{B} consecutively and applying the same numeration for the DoFs.

Furthermore, the MBS under consideration is subject to a number of rigid contacts that model the interaction with the environment. The contact configuration of the MBS may change over time for, e.g., a walking humanoid robot. In particular, we consider scleronomous holonomic constraints of the form $\boldsymbol{g}_i(\boldsymbol{q}(t)) \geqslant \boldsymbol{0}$, $1 \leqslant i \leqslant n^{\mathrm{G}}$, with $\boldsymbol{g}_i : \mathbb{R}^{n^{\mathrm{DoF}}} \to \mathbb{R}$ and generalized coordinates $\boldsymbol{q} : \mathbb{R} \to \mathbb{R}^{n^{\mathrm{DoF}}}$ describing the configuration of the MBS of the humanoid. The contact functions $\boldsymbol{g}_i$ model the distance between contact points and the environment, or, in the case of auto-collision, the MBS itself. For brevity of exposition, we omit dependencies on time t wherever this is obvious.

A contact becomes active when this distance function becomes zero. For a given time instant, we define the index set I of active constraints

$$\mathrm{I} := \{i \mid \boldsymbol{g}_i(\boldsymbol{q}) = 0,\ 1 \leqslant i \leqslant n^{\mathrm{G}}\}.$$

We denote by $n^{\mathrm{I}} := |\mathrm{I}|$ the size of the set I, by $\boldsymbol{g}_{\mathrm{I}}$ the vector-valued function $(\boldsymbol{g}_i)_{i\in\mathrm{I}}$, and by $\boldsymbol{J}_{\mathrm{I}}(\boldsymbol{q}) := \frac{\mathrm{d}\boldsymbol{g}_{\mathrm{I}}}{\mathrm{d}\boldsymbol{q}} \in \mathbb{R}^{n_{\mathrm{I}} \times n^{\mathrm{DoF}}}$ its Jacobian with respect to $\boldsymbol{q}$.

The dynamics of a constrained MBS for a fixed constraint set I are then described by

$$\boldsymbol{H}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{c}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \boldsymbol{S}^T \boldsymbol{\tau} + \boldsymbol{J}_{\mathrm{I}}(\boldsymbol{q})^T \boldsymbol{\lambda}_{\mathrm{I}}, \tag{1.1a}$$

$$\boldsymbol{g}_{\mathrm{I}}(\boldsymbol{q}) = \boldsymbol{0}, \tag{1.1b}$$

with generalized velocities and accelerations $\dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}} : \mathbb{R} \to \mathbb{R}^{n^{\mathrm{DoF}}}$, generalized forces $\boldsymbol{\tau} : \mathbb{R} \to \mathbb{R}^{n^{\mathrm{Act}}}$, a symmetric positive definite joint space inertia matrix or mass matrix $\boldsymbol{H} : \mathbb{R}^{n^{\mathrm{DoF}}} \to \mathbb{R}^{n^{\mathrm{DoF}} \times n^{\mathrm{DoF}}}$, $\boldsymbol{c} : \mathbb{R}^{n^{\mathrm{DoF}}} \times \mathbb{R}^{n^{\mathrm{DoF}}} \to \mathbb{R}^{n^{\mathrm{DoF}}}$ the vector of nonlinear effects, e.g. *Coriolis*, centrifugal, and gravity terms, and a selection matrix $\boldsymbol{S} \in \mathbb{R}^{n^{\mathrm{Act}} \times n^{\mathrm{DoF}}}$ mapping the joint torques $\boldsymbol{\tau}$ to the actuated DoFs of the system. Active contacts $\boldsymbol{g}_{\mathrm{I}}$ result in an external force $\boldsymbol{\lambda}_{\mathrm{I}} : \mathbb{R} \to \mathbb{R}^{n^{\mathrm{G}}}$ acting on the system by means of the transposed contact Jacobian $\boldsymbol{J}_{\mathrm{I}}$.

Assumption We assume that the contact Jacobian $\boldsymbol{J}_{\mathrm{I}}(\boldsymbol{q})$ has full row rank for any constraint set I, i.e.,

$$\operatorname{rank}(\boldsymbol{J}_\mathrm{I}(\boldsymbol{q})) = n^\mathrm{I}. \tag{1.2}$$

△

Remark Rank deficiency of the contact Jacobian indicates redundant constraints. In this case, it is always possible to replace the current constraint set by a subset that satisfies Assumption 1.1.

It is well known that equation (1.1) is a differential algebraic equation (DAE) of index three that can be reduced to index one under smoothness assumptions. Applied to the equation of motion (1.1), this yields a second-order index 1 DAE system with $\boldsymbol{\gamma}_\mathrm{I}(\boldsymbol{q},\dot{\boldsymbol{q}}) = \frac{\mathrm{d}\boldsymbol{J}_\mathrm{I}(\boldsymbol{q})}{\mathrm{d}t}\dot{\boldsymbol{q}}$ in the form of

$$\boldsymbol{H}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{c}(\boldsymbol{q},\dot{\boldsymbol{q}}) = \boldsymbol{S}^T\boldsymbol{\tau} + \boldsymbol{J}_\mathrm{I}(\boldsymbol{q})^T\boldsymbol{\lambda}_\mathrm{I} \tag{1.3a}$$

$$\boldsymbol{J}_\mathrm{I}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{\gamma}_\mathrm{I}(\boldsymbol{q},\dot{\boldsymbol{q}}) = \boldsymbol{0} \tag{1.3b}$$

which can be equivalently written in the so-called descriptor form, i.e.,

$$\begin{bmatrix} \boldsymbol{H}(\boldsymbol{q}) & \boldsymbol{J}_\mathrm{I}(\boldsymbol{q})^T \\ \boldsymbol{J}_\mathrm{I}(\boldsymbol{q}) & 0 \end{bmatrix} \begin{bmatrix} \ddot{\boldsymbol{q}} \\ -\boldsymbol{\lambda}_\mathrm{I} \end{bmatrix} = \begin{bmatrix} \boldsymbol{S}^T\boldsymbol{\tau} - \boldsymbol{c}(\boldsymbol{q},\dot{\boldsymbol{q}}) \\ -\boldsymbol{\gamma}_\mathrm{I}(\boldsymbol{q},\dot{\boldsymbol{q}}) \end{bmatrix}, \tag{1.4}$$

where additionally the invariants of the kinematic constraints due to (1.1b) have to be fulfilled, i.e.,

$$\boldsymbol{g}_\mathrm{I}(\boldsymbol{q}) = \boldsymbol{0}, \tag{1.5a}$$

$$\boldsymbol{J}_\mathrm{I}(\boldsymbol{q})\dot{\boldsymbol{q}} = \boldsymbol{0}. \tag{1.5b}$$

While mathematically equivalent to (1.1), in the case of numerical integration equation (1.3a) is not fully respected and might result in an accumulation of errors. Therefore, especially when dealing large step sizes and/or long simulation durations, the application of *Baumgarte* stabilization according to *Ascher et al.* [5] can counteract these effects.

However, this is only required for the simulation of MBS on long horizons, which is not the case for the OCP formulation we strive for due to its discretization based on multiple shooting. In this case, the horizon is separated into smaller sub-intervals, where we enforce the invariants (1.5) as constraints on the initial value. In this way, the errors can not accumulate along the total horizon and a drift causing the kinematic constraints to be violated is not observable. The details for this are derived in Section 1.5.1.

Force-Contact Complementarity

In the case of unilateral constraints, i.e., $\boldsymbol{g}(\boldsymbol{q}) \geqslant \boldsymbol{0}$, the complementarity between contact and the respective contact force has to be respected. This means that on top of the equation of motion of either index 3 (1.1) or index 1 (1.4) the following complementarity condition has to hold

$$\boldsymbol{\lambda}_\mathrm{I}^T\boldsymbol{g}_\mathrm{I}(\boldsymbol{q}) = 0 \tag{1.6a}$$

$$\boldsymbol{\lambda}_\mathrm{I}, \boldsymbol{g}_\mathrm{I}(\boldsymbol{q}) \geqslant 0, \tag{1.6b}$$

where we refer again to [19] for details concerning the simulation of these systems.

Modeling of Contact Events

System (1.1) governs the MBS as long as all constraints I remain active, and all constraints in the complement remain inactive. Changes in the contact set I during the motion of the MBS give rise to implicit switches between different equation of dynamics that have to be addressed properly.

Advanced impact models, e.g. [173, 174], consider friction, simultaneous impacts of multiple rigid-bodies as well as physical more meaningful impacts as fully inelastic collisions in the form of separate compression and expansion phases during collision.

Here, we propose to treat each contact configuration of the MBS represented by a fixed set I of active constraints as separate model stage combined by a contact event infinitesimal in time accounting for the impact. Then, on each model stage, the dynamics of the MBS with a fixed constraint set can be treated according to (1.1) or (1.3). The detection and confirmation of switches, i.e., changes of the constraint set, is achieved by monitoring the invariants on position and velocity level as well as the applied contact force $\boldsymbol{\lambda}$.

The infinitesimal contact event accounts for the instantaneous state change upon contact. Whenever a new contact event occurs for any t, i.e., $g_i(\boldsymbol{q}(t)) = 0$ and $\dot{g}_i(\boldsymbol{q}(t)) \leqslant 0$ for some $i \notin \mathrm{I}$, we add the contact index i to the set I and solve for conservation of momentum for the new constraint set $\tilde{\mathrm{I}} = \mathrm{I} \cup \{i\}$, of rigid MBS [48], i.e.,

$$\begin{bmatrix} \boldsymbol{H}(\boldsymbol{q}) & \boldsymbol{J}_{\tilde{\mathrm{I}}}(\boldsymbol{q})^T \\ \boldsymbol{J}_{\tilde{\mathrm{I}}}(\boldsymbol{q}) & 0 \end{bmatrix} \begin{bmatrix} \dot{\boldsymbol{q}}^+ \\ \boldsymbol{\Lambda}_{\tilde{\mathrm{I}}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{H}(\boldsymbol{q})\dot{\boldsymbol{q}}^- \\ \boldsymbol{v}_{\tilde{\mathrm{I}}}^- \end{bmatrix}, \tag{1.7}$$

where $\boldsymbol{\Lambda}_{\tilde{\mathrm{I}}} \in \mathbb{R}^{n^{\mathrm{G}}}$ is the instantaneous force impulse, $\dot{\boldsymbol{q}}^- \in \mathbb{R}^{n^{\mathrm{DoF}}}$ and $\dot{\boldsymbol{q}}^+ \in \mathbb{R}^{n^{\mathrm{DoF}}}$ are the joint velocities before and after the collision. The normal velocities of the contact before and after impact are $\boldsymbol{v}_{\tilde{\mathrm{I}}}^-, \boldsymbol{v}_{\tilde{\mathrm{I}}}^+ \in \mathbb{R}^{n_{\mathrm{I}}}$; the latter will be zero in this work.

Note that the relation of linear and angular momentum

$$\boldsymbol{H}(\ddot{\boldsymbol{q}}^+ - \ddot{\boldsymbol{q}}^-) = -\lim_{\Delta t \to 0} \int_t^{t+\Delta t} \boldsymbol{J}(s)^T \boldsymbol{\lambda}(s) \mathrm{d}s \tag{1.8}$$

from [179], as well as the collision hypothesis, respectively, becomes (1.7) because we treat the collision as if it happened instantaneously on a transition stage of infinitesimal duration.

1.4 Bipedal Locomotion as Optimal Control Problem

In the previous sections, we derived the principle characteristics of bipedal locomotion of of humanoid. We described humanoid robots as MBS and introduced the mathematics of releasing and establishing contacts with the environment as well as the consequences for the equation of motion. Revisiting the contact sequence as shown in Figure 1.1 a straight-forward way to encode optimal walking of a humanoid into an optimization problem is to formulate a multi-stage OCP with state-dependent discontinuities at the respective stage borders subject to periodicity constraints.

Definition (Multi-stage Optimal Control Problem) Let $\mathcal{T} = \bigcup_{0 \leqslant j < n^{\mathrm{m}}} \mathcal{T}_j \subset \mathbb{R}$ be a finite time horizon partitioned into $n^{\mathrm{m}} \in \mathbb{N}$ different model stages on $\mathcal{T}_j = [t_j, t_{j+1}]$: The respective multi-stage optimal control problem is formulated as

$$\min_{\substack{x(\cdot),u(\cdot),\\ p,\{t_{i,j}\}}} \quad \sum_{0\leqslant j<n^{\mathrm{m}}} \int_{t_j}^{t_{j+1}} l_j(\boldsymbol{x}(t),\boldsymbol{u}(t),\boldsymbol{p})\,\mathrm{d}t + m_j(\boldsymbol{x}(t_{j+1}),\boldsymbol{p}) \tag{1.9a}$$

$$\text{s.t.} \quad \dot{\boldsymbol{x}}(t) = \boldsymbol{f}_j(\boldsymbol{x}(t),\boldsymbol{u}(t),\boldsymbol{p}), \qquad \forall t\in\mathcal{T}_j, \quad 0\leqslant j<n^{\mathrm{m}}, \tag{1.9b}$$

$$\boldsymbol{x}(t_j) = \Delta_{j-1}(\boldsymbol{x}(t_{j-1}),\boldsymbol{p}), \qquad 1\leqslant j<n^{\mathrm{m}}, \tag{1.9c}$$

$$0 \leqslant \boldsymbol{g}_j(\boldsymbol{x}(t),\boldsymbol{u}(t),\boldsymbol{p}), \qquad \forall t\in\mathcal{T}_j, \quad 0\leqslant j<n^{\mathrm{m}}, \tag{1.9d}$$

$$0 \leqq \sum_{t_{i,j}} \boldsymbol{h}_{i,j}(\boldsymbol{x}(t_{i,j}),\boldsymbol{p}), \quad \{t_{i,j}\}\subset\mathcal{T}_j, \quad 0\leqslant j<n^{\mathrm{m}}. \tag{1.9e}$$

Here, we strive to find a control trajectory $\boldsymbol{u} : \mathcal{T} \to \mathbb{R}^{n^{\mathrm{u}}}$ such that an objective function composed of stage-wise Lagrange terms $l_j : \mathbb{R}^{n^{\mathrm{x}}} \times \mathbb{R}^{n^{\mathrm{u}}} \times \mathbb{R}^{n^{\mathrm{p}}} \to \mathbb{R}$, $0 \leqslant j < n^{\mathrm{m}}$ and Mayer terms $m_j : \mathbb{R}^{n^{\mathrm{x}}} \times \mathbb{R}^{n^{\mathrm{p}}} \to \mathbb{R}$. The state trajectory $\boldsymbol{x} : \mathcal{T} \to \mathbb{R}^{n^{\mathrm{x}}}$ of the dynamic system is defined by stage-wise systems of ODE with right hand sides $\boldsymbol{f}_j : \mathbb{R}^{n^{\mathrm{x}}} \times \mathbb{R}^{n^{\mathrm{u}}} \times \mathbb{R}^{n^{\mathrm{p}}} \to \mathbb{R}^{n^{\mathrm{x}}}$, $0 \leqslant j < n^{\mathrm{m}}$ that may depend on the global model parameters $p \in \mathbb{R}^{n^{\mathrm{p}}}$ of the system. Discontinuities at the model stage borders $t_j \in \mathbb{R}$, $1 \leqslant j < n^{\mathrm{m}}$ are defined by transition functions $\Delta_j : \mathbb{R}^{n^{\mathrm{x}}} \times \mathbb{R}^{n^{\mathrm{u}}} \times \mathbb{R}^{n^{\mathrm{p}}} \to \mathbb{R}^{n^{\mathrm{x}}}$, $0 \leqslant j < n^{\mathrm{m}}$. In addition, mixed state-control constraints $\boldsymbol{g}_j : \mathbb{R}^{n^{\mathrm{x}}} \times \mathbb{R}^{n^{\mathrm{u}}} \times \mathbb{R}^{n^{\mathrm{p}}} \to \mathbb{R}^{n^{\mathrm{c}}}$, $0 \leqslant j < n^{\mathrm{m}}$ on the horizon and contributions $\boldsymbol{h}_{i,j} : \mathbb{R}^{n^{\mathrm{x}}} \times \mathbb{R}^{n^{\mathrm{p}}} \to \mathbb{R}^{n^{\mathrm{r},i,j}}$, $0 \leqslant j < n^{\mathrm{m}}$ to coupled (in-)equality point constraints on a finite number of $N_j + 1$ time points $\{t_{i,j}\} \subset \mathcal{T}_j$, $0 \leqslant i \leqslant N_j$, are imposed on model stages $0 \leqslant j < n^{\mathrm{m}}$ of the problem. $\triangle$

In Definition 1.3, the evolution of the MBS is described by a system of ODEs (1.9b). This dynamic process is affected by an input in the form of control function $\boldsymbol{u}(t)$ at any time $t \in \mathcal{T}$. In general, the control input can be assumed to be at least measurable. However, to not complicate the formulation and avoid detailed function space considerations, we restrict the control inputs to be piecewise polynomial, i.e., the set of all control inputs is given by $\boldsymbol{u} \in \mathcal{U} = \left\{\boldsymbol{u} : \mathcal{T} \to \mathbb{R}^{n^{\mathrm{u}}} | \forall t \in \mathcal{T}\ \exists\ \mathcal{I} \subset \mathcal{T}_j \subseteq \mathcal{T}, 0 < j \leqslant n^{\mathrm{m}} : \boldsymbol{u}\big|_{\mathcal{I}} \equiv \boldsymbol{v} \in \mathbb{R}[t]\right\}$, where $\mathbb{R}[t]$ denotes the polynomial ring over $\mathbb{R}$ in t. While we restricted the assumptions on the OCP to improve readability, we refer the reader to [29] for a rigorous discourse of OCP based on functional analysis.

In order to guarantee existence and uniqueness of $\boldsymbol{x}$, we assume the right-hand side of the ODEs to be piecewise *Lipschitz* continuous. Following this, the evolution of the system state is described by means of $\boldsymbol{x}(t)$ at any $t \in \mathcal{T}$ and we define the set of all state trajectories by $\mathcal{X} = \{\boldsymbol{x} : \mathcal{T} \to \mathbb{R}^{n^{\mathrm{x}}}\}$.

The admissible sets for the state and control functions are restricted by the constraint functions in (1.9d) and (1.9e), where the first comprises, for example, mixed path and control constraints, restrict the set of initial values $\boldsymbol{x}(t_0)$, and contain boundary conditions for the trajectories, and the latter comprises, for example, periodicity constraints, and decoupled constraints such as initial values and terminal values.

Multi-body Systems in the Optimal Control Problem Formulation In order to treat bipedal locomotion of humanoids , as described in Section 1.1, in the context of optimal control, we describe how the impulsive hybrid dynamics of MBS subject to unilateral kinematic constraints, as described in Section 1.3, can be embedded in the OCP (1.9).

For each contact configuration given by a fixed constraint set I_j, $0 \leqslant j < n^{\mathrm{m}}$, we embed the respective equation of motion in the form of (1.4) as ODE into the OCP (1.9b). By introducing both joint positions and velocities as differential states $\boldsymbol{x} := \left[\boldsymbol{q}^T, \dot{\boldsymbol{q}}^T\right]^T$ and formulating the second-order differential equation given by (1.4) as an ODE by order reduction, which integrates the acceleration $\ddot{\boldsymbol{q}}$ to the actual joint positions, given by

$$\dot{\boldsymbol{x}}(t) = \begin{bmatrix}\dot{\boldsymbol{q}}(t)\\ \dot{\boldsymbol{v}}(t)\end{bmatrix} = \begin{bmatrix}\boldsymbol{v}(t)\\ \boldsymbol{a}(t) = \mathrm{FD}(\boldsymbol{q}(t), \boldsymbol{v}(t), \boldsymbol{\tau}(\boldsymbol{u}(t)); \mathrm{I}_j)\end{bmatrix} = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{p}), t \in \mathcal{T}_j, \quad 0 \leqslant j < n^{\mathrm{m}}, \tag{1.10a}$$

where, for correctness of notation, we renamed the generalized velocities and accelerations by $\boldsymbol{v} = \dot{\boldsymbol{q}}$ and $\boldsymbol{a} = \ddot{\boldsymbol{q}}$. The accelerations are retrieved by means of an forward dynamics (FD) solution operator $\mathrm{FD} : \mathbb{R}^{n^{\mathrm{DoF}}} \times \mathbb{R}^{n^{\mathrm{DoF}}} \times \mathbb{R}^{n^{\mathrm{Act}}} \to \mathbb{R}^{n^{\mathrm{DoF}}} \times \mathbb{R}^{n^{\mathrm{G}}}$, which maps joint positions, velocities and torques onto joint accelerations and contact reaction forces given the current contact situation I, that is defined by

$$(\ddot{\boldsymbol{q}}, \boldsymbol{\lambda}) = \mathrm{FD}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{\tau}; \mathrm{I}). \tag{1.11}$$

The operator FD is the solution operator of the linear system given by the index-1 formulation (1.4). The resulting joint accelerations $\ddot{\boldsymbol{q}}$ are then twice integrated to get joint positions $\boldsymbol{q}$ and velocities $\dot{\boldsymbol{q}}$.

There exist two principal direct techniques for solving the linear system (1.4) and we refer the reader for a detailed overview on solution techniques of range- and null-space approaches to [10, 48].

Following the fixed contact sequence, every stage except the final one is followed by an infinitesimal *transition stage*. In particular, such state jumps Δ_j (1.9c) are caused by impacts resulting from inelastic collisions modeled via (1.7).

The detection and confirmation of switches, i.e., changes of the constraint set, is achieved by formulating a suitable end-point constraint (1.9e) per stage, given by the contact invariants of (1.5) and/or by constraints on the contact force $\boldsymbol{\lambda}$. While we assumed that the order of model stages is known and fixed, the respective switching times t_j, i.e., the time instants in which the contact events occur, may be subject to optimization. Furthermore, friction cones are directly treated into the OCP by (1.9d) instead of handling them in the ODE (1.9b).

1.4.1 Nonlinear Model Predictive Control

NMPC is a closed-loop control strategy in which the feedback control is computed from the current system state by solving an open-loop optimal control problem over a single finite prediction horizon on-line, therefore also denoted as receding horizon control. In contrast to the OCP problem (1.9) and for brevity of exposition, we focus on a single stage time horizon $\mathcal{T} = [0, T]$ for the NMPC problem and limit the problem class to tracking NMPC problems.

Definition (Nonlinear Model Predictive Control Problem) Given estimates of the current state $\hat{\boldsymbol{x}} \in \mathbb{R}^{n^{\mathrm{x}}}$ and model parameters $\hat{\boldsymbol{p}} \in \mathbb{R}^{n^{\mathrm{p}}}$ of the dynamic system, the NMPC problem under consideration is of the form of

$$
\begin{aligned}
\min_{\boldsymbol{x}(\cdot),\boldsymbol{u}(\cdot),\boldsymbol{p}} \quad & \int_0^T \|\boldsymbol{\ell}(\boldsymbol{x}(t),\boldsymbol{u}(t),\boldsymbol{p}) - \bar{\boldsymbol{\ell}}(t,\boldsymbol{p})\|_{\boldsymbol{W}}^2 \, \mathrm{d}t && (1.12a)\\
& \quad + \|\boldsymbol{e}(\boldsymbol{x}(T),\boldsymbol{p}) - \bar{\boldsymbol{e}}(\boldsymbol{p})\|_{\tilde{\boldsymbol{W}}}^2 && (1.12b)\\
\text{s.t.} \quad & \dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t),\boldsymbol{u}(t),\boldsymbol{p}), \qquad t \in \mathcal{T}, && (1.12c)\\
& 0 = \boldsymbol{x}(0) - \hat{\boldsymbol{x}}_0, && (1.12d)\\
& 0 = \boldsymbol{p} - \hat{\boldsymbol{p}}, && (1.12e)\\
& 0 \leqslant \boldsymbol{g}(\boldsymbol{x}(t),\boldsymbol{u}(t),\boldsymbol{p}), \qquad t \in \mathcal{T}, && (1.12f)\\
& 0 \leqq \sum_{t_i} \boldsymbol{h}_i(\boldsymbol{x}(t_i),\boldsymbol{p}), \quad \{t_i\} \subset \mathcal{T}. && (1.12g)
\end{aligned}
$$

In this problem, the deviation of a non-linear objective $\boldsymbol{\ell} : \mathbb{R}^{n^x} \times \mathbb{R}^{n^u} \times \mathbb{R}^{n^p} \to \mathbb{R}^{n^l}$ from a given setpoint $\bar{\boldsymbol{\ell}}$, as well as a penalty for an end-point $\boldsymbol{e} : \mathbb{R}^{n^x} \times \mathbb{R}^{n^p} \to \mathbb{R}^{n^e}$ deviation from a given setpoint $\bar{\boldsymbol{e}}$ is to be minimized with respect to a weighted L^2-norm with positive definite weighting matrices $\boldsymbol{W} \in \mathbb{R}^{n^l \times n^l}$ and $\tilde{\boldsymbol{W}} \in \mathbb{R}^{n^e \times n^e}$. The remaining quantities are the same as for the OCP formulation given in Definition 1.3. △

For the NMPC problem, the initial value and parameter embedding constraints (1.12d, 1.12e) are of special interest as they enter the problem linearly and can be exploited to solve NMPC problems in real-time.

1.4.2 Moving-Horizon Estimation

Model validations, both online and offline, are possible through state and parameter estimation techniques based on time discrete nonlinear least-squares problems. In the NMPC context, MHE methods, c.f. [107], are employed to provide state and parameter estimates for $\hat{\boldsymbol{x}}_0 \in \mathbb{R}^{n^x}$, $\hat{\boldsymbol{p}} \in \mathbb{R}^{n^p}$ in (1.12). We apply the same specialization of the problem formulation as for the NMPC problem (1.12) for better readability, i.e., we define it for a single stage time horizon $\mathcal{T} = [-T, 0]$ and focus on ODEs only.

The MHE problem minimizes the error between the model response $\boldsymbol{y} : \mathbb{R}^{n^x} \times \mathbb{R}^{n^u} \times \mathbb{R}^{n^p} \to \mathbb{R}^{n^h}$ and N measurements $\boldsymbol{\eta} = \boldsymbol{y}(\boldsymbol{x}(t),\boldsymbol{u}(t),\boldsymbol{p}^\star) + \boldsymbol{\epsilon}(t)$, from the real system defined by the true but unknown parameters $\boldsymbol{p}^\star$, subject to an additive measurement error $\boldsymbol{\epsilon}(t) \sim \boldsymbol{\mathcal{N}}(0, \Xi)$, with zero-mean and covariance matrix $\Xi = \operatorname{diag}(\xi_0, \ldots, \xi_{n^h})$. The quantities of interest here are the state and parameter estimates for $\hat{\boldsymbol{x}}_0$ and $\hat{\boldsymbol{p}}$ in (1.12), which are required for (1.12d) and (1.12e).

Definition (Moving-Horizon Estimation Problem) The MHE problem under consideration is given by

$$
\begin{aligned}
\min_{\boldsymbol{x}(\cdot),\boldsymbol{p}} \quad & \sum_{k=-N}^{0} \|\boldsymbol{y}(\boldsymbol{x}(t_k),\boldsymbol{u}(t_k),\boldsymbol{p}) - \boldsymbol{\eta}_k\|_{\Xi_k}^2 + \left\| \begin{bmatrix} \boldsymbol{x}(-T) - \bar{\boldsymbol{x}}_{-T} \\ \boldsymbol{p} - \bar{\boldsymbol{p}} \end{bmatrix} \right\|_{\boldsymbol{P}}^2 && (1.13a)\\
\text{s.t.} \quad & \dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t),\boldsymbol{u}(t),\boldsymbol{p}), \quad t \in \mathcal{T}, && (1.13b)\\
& 0 \leqslant \boldsymbol{g}(\boldsymbol{x}(t),\boldsymbol{u}(t),\boldsymbol{p}), \quad t \in \mathcal{T}, && (1.13c)
\end{aligned}
$$

where the controls $\boldsymbol{u}$ are fixed to the values that have already been applied to the system in the past on $[-T, 0]$. The initial values $\bar{\boldsymbol{x}}_{-T}$, $\bar{\boldsymbol{p}}$ in (1.13a) weighted by the covariance matrix $\boldsymbol{P} \in \mathbb{R}^{(n^x+n^p)\times(n^x+n^p)}$ are used to incorporate *a-priori* information into the problem. The remaining quantities are the same as for the OCP formulation given in Definition 1.3. △

1.5 The Direct Multiple Shooting Method for Optimal Control

In order to solve the infinite-dimensional OCP as well as the NMPC and MHE problems, we follow a direct and simultaneous approach to optimal control. Therefore, we derive in the following section the direct multiple shooting method. This method is then used to discretize, parameterize, and finally solve the optimal control problem (1.9) or its NMPC/MHE counterpart (1.12) in practice.

1.5.1 The Direct Approach to Optimal Control

The principle of direct approach to optimal control is to *first discretize* the infinite dimensional controls, *then optimize* the resulting structured finite optimization problem. Following this, we commence by first discretizing the control $\boldsymbol{u}(\cdot)$ on a prescribed time grid $0 = \mathrm{t}_0 < \mathrm{t}_1 < \ldots < \mathrm{t}_N = n^{\mathrm{m}}$ comprised of N intervals. The end point t_N is chosen such that each model stage is mapped onto a unit interval, and the grid points are assumed to comprise the integers, which correspond to model stage boundary points. This grid also partitions the physical time horizon $\mathcal{T} = [0, T] = [0, 0 + \sum_{k=0}^{n^{\mathrm{m}}-1} h_k]$ into N intervals, where the $h_i := t_{i+1} - t_i$ denote the physical durations of each model stage. Defining ι to be the index of the stage containing t, i.e., $\iota = \lfloor \mathrm{t} \rfloor$ the time transformation is given by

$$t(\mathrm{t}) := t_0 + \sum_{k=0}^{\iota-1} h_k + h_\iota(\mathrm{t} - \iota), \quad \mathrm{t} \in [0, n^{\mathrm{m}}] \subset \mathbb{R}.$$

The control $\boldsymbol{u}_j(\cdot)$ is discretized by means of base functions $\boldsymbol{b}_j$, which themselves are parametrized by $n^{\mathrm{q},j}$ control parameters $\boldsymbol{q}_{i,j,\iota} \in \mathbb{R}$, which yields

$$\boldsymbol{u}_j(\mathrm{t})\Big|_{[\mathrm{t}_i, \mathrm{t}_{i+1})} := \boldsymbol{b}_j(\mathrm{t}, \boldsymbol{q}_{i,j,1}, \ldots, \boldsymbol{q}_{i,j,n^{\mathrm{q},j}}) \in \mathbb{R} \tag{1.14}$$

on time intervals $[\mathrm{t}_i, \mathrm{t}_{i+1})$ for $0 \leqslant i \leqslant N$ and for controls $1 \leqslant j \leqslant n^{\mathrm{u}}$. The typical choices for the base functions are piecewise constant controls with $n^{\mathrm{q},j} = 1$

$$\boldsymbol{b}_j(\mathrm{t}, \boldsymbol{q}_{i,j}) = \boldsymbol{q}_{i,j}, \quad \mathrm{t} \in [\mathrm{t}_i, \mathrm{t}_i + 1), \tag{1.15}$$

or piecewise linear controls with $n^{\mathrm{q},j} = 2$

$$\boldsymbol{b}_j(\mathrm{t}, \boldsymbol{q}_{i,j}) = \frac{(\mathrm{t} - \mathrm{t}_i)\, \boldsymbol{q}_{i,j,2} + (\mathrm{t}_{i+1} - \mathrm{t})\, \boldsymbol{q}_{i,j,1}}{(\mathrm{t}_{i+1} - \mathrm{t}_i)}, \quad \mathrm{t} \in [\mathrm{t}_i, \mathrm{t}_i + 1). \tag{1.16}$$

Moreover, one can impose continuity conditions in the grid points t_i on the control functions in the case of $n^{\mathrm{q},j} \geqslant 2$ by

$$\boldsymbol{u}_j\Big|_{[\mathrm{t}_i, \mathrm{t}_{i+1})}(\mathrm{t}_{i+1}) = \boldsymbol{b}_j(\mathrm{t}_{i+1}, \boldsymbol{q}_{i,j}) \overset{!}{=} \boldsymbol{b}_j(\mathrm{t}_{i+1}, \boldsymbol{q}_{i+1,j}) = \boldsymbol{u}_j\Big|_{[\mathrm{t}_{i+1}, \mathrm{t}_{i+2})}(\mathrm{t}_{i+1}), \quad 0 \leqslant i < N - 1.$$

1.5.2 The Direct Multiple Shooting Method

In this thesis, we recap the direct multiple shooting method for optimal control as presented in [13]. Further, the idea is to parametrize the state trajectory $\boldsymbol{x}(\cdot)$ as well. This is achieved by introducing $N + 1$ state variables $\boldsymbol{s}_i \in \mathbb{R}^{n^{\mathrm{x}}}$ on the time grid points t_i, and by solving initial value problems (IVPs) separately on each time interval:

$$\frac{\mathrm{d}x(t)}{\mathrm{dt}} = h_\iota \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{b}(t, \boldsymbol{q}_i), \boldsymbol{p}), \quad t = t(\mathrm{t}), \quad \mathrm{t} \in [\mathrm{t}_i, \mathrm{t}_{i+1}], \tag{1.17a}$$

$$0 = \boldsymbol{x}(t(\mathrm{t}_i)) - \boldsymbol{s}_i. \tag{1.17b}$$

As stated above, we denote by h_ι again the duration of the model stage to which the time interval $[\mathrm{t}_i, \mathrm{t}_{i+1}]$ belongs. We denote the differential evolutions (or solutions) of the IVPs by $\boldsymbol{x}(\mathrm{t}_{i+1}; \mathrm{t}_i, \boldsymbol{s}_i, \boldsymbol{q}_i, \boldsymbol{p}, h_\iota)$, where by ; we denote an implicit dependence on the quantities after the semicolon that come into play when we derive the required derivatives later on.

Analogously to the discretization scheme of the controls, the continuity of the differential state trajectory in the solution of the discretized problem is guaranteed by imposing additional matching conditions at the nodes of the time grid t_i by

$$0 = \boldsymbol{s}_{i+1} - \boldsymbol{x}(\mathrm{t}_{i+1}; \mathrm{t}_i, \boldsymbol{s}_i, \boldsymbol{q}_i, \boldsymbol{p}, h_\iota), \qquad \forall 0 \leqslant i < N \text{ with } \mathrm{t}_{i+1} \notin \mathbb{N}, \tag{1.18a}$$

that also covers the model stage transitions (1.9c) in grid points t_{i+1} coinciding with model stage boundaries,

$$0 = \boldsymbol{s}_{i+1} - \Delta_\iota(\boldsymbol{x}(\mathrm{t}_{i+1}; \mathrm{t}_i, \boldsymbol{s}_i, \boldsymbol{q}_i, \boldsymbol{p}, h_\iota), \boldsymbol{p}), \quad \forall 0 \leqslant i < N \text{ with } \mathrm{t}_{i+1} \in \mathbb{N}. \tag{1.18b}$$

Again, by index ι we denote the index of the model stage ending in t_{i+1}. The doubting reader has to recap that the discretization points t_i

1.5.3 Structured Nonlinear Programming

After the full discretization and parametrization of all infinite-dimensional quantities in the OCP (1.9) or the NMPC counterpart, a large but structured NLP is derived. It is of the form of

$$\min_{\boldsymbol{s},\boldsymbol{q},\boldsymbol{p},\boldsymbol{h}} \quad \sum_{i=0}^{N-1} \tilde{l}_i(\boldsymbol{s}_i, \boldsymbol{q}_i, \boldsymbol{p}, h_\iota) + \tilde{m}(\boldsymbol{s}_N, \boldsymbol{p}, h_N) \tag{1.19a}$$

$$\text{s.t.} \quad 0 = \boldsymbol{s}(\mathrm{t}_{i+1}; \mathrm{t}_i, \boldsymbol{s}_i, \boldsymbol{q}_i, \boldsymbol{p}, h_\iota) - \boldsymbol{s}_{i+1}, \quad 0 \leqslant i < N, \tag{1.19b}$$

$$0 \leqslant \tilde{\boldsymbol{g}}(\mathrm{t}_i, \boldsymbol{s}_i, \boldsymbol{q}_i, \boldsymbol{p}, h_\iota), \qquad 0 \leqslant i \leqslant N, \tag{1.19c}$$

$$0 \leqq \tilde{\boldsymbol{h}}_j(\mathrm{t}_i, \boldsymbol{s}_i, \boldsymbol{q}_i, \boldsymbol{p}, h_\iota), \qquad 0 \leqslant i \leqslant N. \tag{1.19d}$$

For the brevity of exposition, we define discretized counterparts $\tilde{l}_i$, $\tilde{m}$, $\tilde{\boldsymbol{g}}$, and $\tilde{\boldsymbol{h}}_j$ of the respective functions in (1.9) that also account for the control discretization (1.14) and possibly control continuity conditions. The continuity conditions of the differential state trajectory (1.19b) require the solution of the IVPs (1.18a), which shooting approaches achieve by adaptively discretizing the IVPs in time by making use of state-of-the-art ODE solvers, for example [2].

Depending on the chosen discretization scheme the NLP (1.19) can be solved efficiently by exploiting this structure by dedicated SQP methods. The structure arises mainly due to the local support of the control function discretization and in particular for the multiple shooting approach due to the equality constraints (1.19b). The latter can be even eliminated by applying a partial null-space approach, referred to as condensing and partial reduction, c.f. [13, 109] for the details.

First and possibly second-order derivatives, which required by the SQP method, then involve the computation of sensitivities of the solution of the dynamics with respect to all dependencies according to the principle of internal numerical differentiation, cf. [2, 12].

By eliminating the redundant variables of the differential states $\boldsymbol{s}_1^k, \ldots, \boldsymbol{s}_N^k$ by means of the above mentioned partial reduction or condensing, we summarize the remaining DoFs in an iterate $\boldsymbol{w}^k$ in the kth iteration of an SQP method, which concatenates the sub-vectors $\boldsymbol{w}^k = (\boldsymbol{p}^k, h^k, \boldsymbol{s}_0^k, \boldsymbol{q}^k)$. SQP methods then employs a second-order model of the NLP (1.19) in the form of a quadratic program (QP) in each iteration to compute an increment $\Delta\boldsymbol{w}$. For the kth iteration the QP is of the form of

$$\begin{aligned} \min_{\Delta w} \quad & \tfrac{1}{2}\Delta\boldsymbol{w}^T\boldsymbol{B}(\boldsymbol{w}^k)\Delta\boldsymbol{w} + \Delta\boldsymbol{w}^T\boldsymbol{b}(\boldsymbol{w}^k) && \text{(1.20a)} \\ \text{s.t.} \quad & 0 = \boldsymbol{c}(\boldsymbol{w}^k) + \boldsymbol{C}(\boldsymbol{w}^k)\Delta\boldsymbol{w}, && \text{(1.20b)} \\ & 0 \leqslant \boldsymbol{d}(\boldsymbol{w}^k) + \boldsymbol{D}(\boldsymbol{w}^k)\Delta\boldsymbol{w}, && \text{(1.20c)} \end{aligned}$$

where by $\boldsymbol{B}(\boldsymbol{w}^k)$ we denote the Hessian of the Lagrangian of (1.19) in $\boldsymbol{w}^k$ and $\boldsymbol{b}(\boldsymbol{w}^k)$ denotes the gradient of the objective(1.19a). Typically, SQP methods do not compute the numerically costly Hessian information directly, but employ a symmetric and usually positive definite approximation, c.f. [138] for details. The constraints of the NLP (1.19b,1.19d) and (1.19c,1.19d) are summarized into equality- and inequality constraints and by (1.20b) and (1.20c) we denote their local linearizations, respectively.

In general, the resulting QP will be small and dense. Therefore, state-of-the-art active-set QP solvers can be used for their solution, e.g. *QPSOL* [60], *QPOPT* [59], *qpOASES* [51, 52], or *QORE* [154]. In the remainder of this, we use the QP solver *QPOPT* if not stated otherwise.

The solution of the respective second-order approximation $\Delta\boldsymbol{w}^k = (\Delta\boldsymbol{p}^k, \Delta h^k, \Delta\boldsymbol{s}_0^k, \Delta\boldsymbol{q}^k)$ is used as a search direction to find the next SQP iterate $\boldsymbol{w}^{k+1}$. In order to guarantee the finding of local minimum from any initial guess, globalization of the convergence of the SQP method can be achieved by either trust-region or line-search methods, c.f. [138].

1.5.4 Real-Time Iterations for NMPC and MHE

State-of-the-art NLP-based NMPC methods rely on the so-called real-time iteration scheme due to [36, 37]. The idea is to use the local contractivity properties of *Newton*'s method and only apply one SQP iteration, which requires the solution of a single QP, to provide feedback in real-time. This is achieved by careful initialization of the SQP method. The SQP iteration can be separated into three distinct phases, i.e. preparation, feedback and transition.

1. *Preparation:* In the preparation phase the evaluation of states and derivatives as well as the structure exploitation takes place. This way a QP is set up except for the values for the current initial system state $\hat{y}_0 \in \mathbb{R}^{n^y}$ and of the model parameters $\hat{p} \in \mathbb{R}^{n^p}$, which enter the NMPC problem (1.12) linearly.
   In particular, the data $\boldsymbol{B}(\boldsymbol{w}^k)$, $\boldsymbol{b}(\boldsymbol{w}^k)$, $\boldsymbol{C}(\boldsymbol{w}^k)$, $\boldsymbol{d}(\boldsymbol{w}^k)$ and $\boldsymbol{D}(\boldsymbol{w}^k)$ of the QP (1.20) is evaluated as well as the parts of $\boldsymbol{c}(\boldsymbol{w}^k)$ that do not correspond to discretizations of $\hat{\boldsymbol{x}}_0$ and $\hat{\boldsymbol{p}}$.
2. *Feedback:* As soon as the missing estimates for $\hat{\boldsymbol{x}}_0$ and $\hat{\boldsymbol{p}}$ are available the feedback control is computed in the feedback phase by solving the QP.
3. *Transition:* After the control is fed back to the system the structure exploitation is rolled back, the horizon is shifted and the Lagrange multipliers are updated.

Every feedback phase is then again followed by a preparation phase of the next time step.

In NMPC, there are two characteristically times that have to be taken into account: sampling time and feedback delay. The feedback delay is the time between the availability of the measurements of system state $\hat{\boldsymbol{x}}_0$ and parameters $\hat{\boldsymbol{p}}$ until the subsequent computation of the feedback control while the sampling time determines the cycle length between subsequent measurements. It is important to ensure that feedback times are short so that the system state and parameters do not deviate too much from the measurements that are used to determine to feedback control. The feedback time in this real-time iteration scheme is essentially comprised by cost to solve one quadratic program, while the computational heavy parts constituting of evaluation of QP data already have been performed in the preparation phase before the measurements are known. The sampling time in this scheme is the sum of the times of the preparation, feedback and transition phase.

This way computational expensive parts can be separated from time-critical ones and the computational delay of the feedback is reduced to only the time required to solve a single QP. Approaches similar to those used to achieve real-time feedback control for NMPC can be used to solve the MHE problem.

1.6 Summary

In this chapter, we introduced the characteristics of bipedal locomotion of humanoids and gave an explanation why the locomotion capabilities of a biological system are considered to be optimal. We gave an overview of the considered robotic hardware platforms and present how a dynamic model can be derived by treating the robotic systems as a rigid bodies attached to a floating base that describes the position and orientation of the system in space. We introduced an optimal control problem formulation used to analyze and synthesize optimal walking motions. Furthermore, we stated formulations for the online and closed-loop control counterparts of the optimal control problem in the form of non-linear model predictive control (NMPC) and moving horizon estimation (MHE) problem formulations. The state of the art in solving the optimal control problem formulations as well as advanced methods of NMPC were revisited.

2 Efficient Derivative Evaluation for Rigid Body Dynamics

Simulation, optimization and control of robotic and bio-mechanical systems depend on a mathematical model description, typically a rigid-body system connected by joints with tree topology. For these kind of models efficient algorithms exist to compute their rigid-body dynamics (RBD) , e.g. computing forward or inverse dynamics of the model. The methods of choice for the evaluation of common RBD quantities are recursive algorithms acting on the kinematic tree topology of the considered robot, c.f. [48]. Furthermore, these models are subject to both kinematic and loop constraints. Both employ tailored linear algebra to solve the respective descriptor form of the equation of motion, which allows the evaluation of the dynamics of the constrained system.

While the efficient evaluation of RBD is sufficient for simulation purposes, methods of model-based optimal control, e.g. whole-body control as well as stability estimation and correction, or machine learning, another emerging trend, rely on the efficient derivative evaluation for RBD of complex multi-body systems (MBSs) subject to constraints and collisions. The required derivative information is often approximated by numerical differentiation (FD). However, they greatly benefit from accurate gradients, which promote faster convergence, smaller iteration counts, and improved handling of nonlinearities or ill-conditioning of the problem formulations, which are particularly observed when kinematic constraints are involved.

In this chapter, we propose the algorithmic augmentation of state-of-the-art recursive algorithms in order to evaluate the first-order forward derivatives, which are required for derivative-based optimization as for example in the direct approach of optimal control. Major parts of our journal paper [105] are based on the contents of this chapter.

We follow the principles algorithmic differentiation (AD) and treat the recursive algorithms as concatenation of elementary operations, where for each elementary operation an analytic derivative can be derived. Following this, the underlying computational graph of the former nominal function evaluation can be transformed into a computational graph that additionally evaluates the derivatives by propagating directional derivatives successively by means of the chain rule, as shown on an example in Figure 2.1. The efficiency is further improved by sparsity exploitation. We validate and benchmark the implementation against its FD counterpart for a lifting motion of a human model.

Furthermore, we propose the extension of the approach to the linear algebra of contact dynamics, which involves matrix factorization to solve both the respective descriptor form and the conservation of angular momentum equation on collisions of the MBS.

The augmentations are implemented in the freely available open source library *Rigid Body Dynamics Library* (RBDL) [49, 50], which realizes the state-of-the-art algorithms in an efficient template-based *C++* code. The proposed approach is thoroughly tested against its FD counterpart on benchmark examples. The applicability of the developed derivative evaluation for direct optimal control is shown on a lifting motion of a human model.

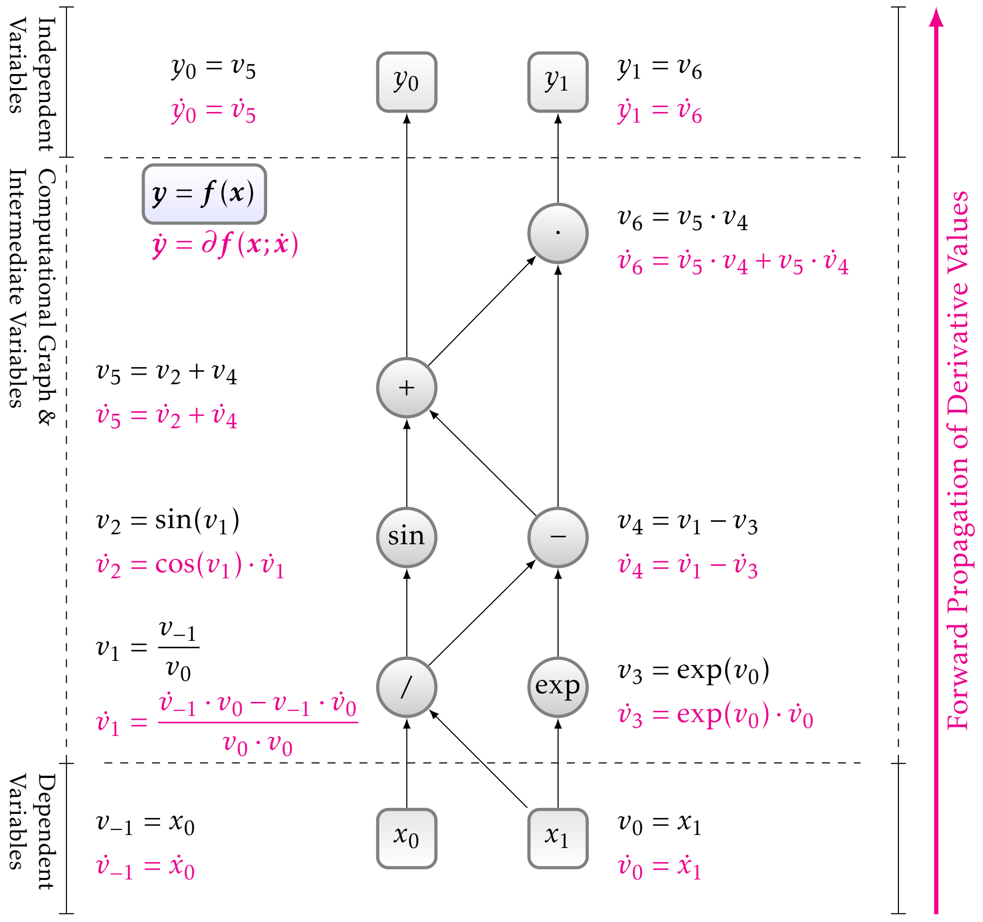

Independent Variables
Computational Graph & Intermediate Variables
Dependent Variables
Forward Propagation of Derivative Values
$y_0 = v_5$
$\dot{y}_0 = \dot{v}_5$
$y_1 = v_6$
$\dot{y}_1 = \dot{v}_6$
$y = f(x)$
$\dot{y} = \partial f(x;\dot{x})$
$v_6 = v_5 \cdot v_4$
$\dot{v}_6 = \dot{v}_5 \cdot v_4 + v_5 \cdot \dot{v}_4$
$v_5 = v_2 + v_4$
$\dot{v}_5 = \dot{v}_2 + \dot{v}_4$
$v_2 = \sin(v_1)$
$\dot{v}_2 = \cos(v_1) \cdot \dot{v}_1$
$v_4 = v_1 - v_3$
$\dot{v}_4 = \dot{v}_1 - \dot{v}_3$
$v_1 = \frac{v_{-1}}{v_0}$
$\dot{v}_1 = \frac{\dot{v}_{-1} \cdot v_0 - v_{-1} \cdot \dot{v}_0}{v_0 \cdot v_0}$
$v_3 = \exp(v_0)$
$\dot{v}_3 = \exp(v_0) \cdot \dot{v}_0$
$v_{-1} = x_0$
$\dot{v}_{-1} = \dot{x}_0$
$v_0 = x_1$
$\dot{v}_0 = \dot{x}_1$
y_0
y_1
$\cdot$
$+$
sin
$-$
$/$
exp
x_0
x_1

Figure 2.1: Schematic of forward mode of algorithmic differentiation (AD) for a multivariate function $\boldsymbol{y} = f(\boldsymbol{x})$ that computes independent values $\boldsymbol{y} \in \mathbb{R}^m$ from dependent variables $\boldsymbol{x} \in \mathbb{R}^n$. The derived computational graph represents the nominal evaluation of the explicit expression $y_0 = \sin(\frac{x_0}{x_1}) + \frac{x_0}{x_1} - \exp(x_1)$ and $y_1 = y_0 \cdot \left(\frac{x_0}{x_1} - \exp(x_1)\right)$, with independent values $\boldsymbol{y} = \left[\begin{smallmatrix} y_0 \\ y_1 \end{smallmatrix}\right]$ and dependent variables $\boldsymbol{x} = \left[\begin{smallmatrix} x_0 \\ x_1 \end{smallmatrix}\right]$. By propagating directions $\dot{x}_0, \dot{x}_1 \in \mathbb{R}$ from the dependent variables (bottom) through the computational graph up to the independent variables (top) the first order forward (directional) derivative $\dot{\boldsymbol{y}} = \partial f(\boldsymbol{x};\dot{\boldsymbol{x}})$ (magenta) can be efficiently computed.

2.1 Evaluation of Derivatives

In this section, we introduce the relevant background and notational details to give the necessary context of the work.

Definition (Differentiability) Let $f : \mathcal{U} \subseteq \mathbb{R}^n \to \mathbb{R}^m$, $\mathcal{U}$ open and $\boldsymbol{x} \in \mathcal{U}$. f is *(Fréchet) differentiable* at $\boldsymbol{x}$, if a linear map $\mathrm{D}f(\boldsymbol{x}) : \mathbb{R}^n \to \mathbb{R}^m$ exists s.t.

$$f(\boldsymbol{x}+\boldsymbol{v}) = f(\boldsymbol{x}) + \mathrm{D}f(\boldsymbol{x})\boldsymbol{v} + \boldsymbol{r}(\boldsymbol{x};\boldsymbol{v})$$

for all $\boldsymbol{v}$ s.t. $\boldsymbol{x}+\boldsymbol{v} \in \mathcal{U}$ and $\frac{\boldsymbol{r}(\boldsymbol{x};\boldsymbol{v})}{\|\boldsymbol{v}\|} \underset{\boldsymbol{v}\to\boldsymbol{0}}{\longrightarrow} \boldsymbol{0}$. $\triangle$

Definition (Derivative) Let $f : \mathcal{U} \subseteq \mathbb{R}^n \to \mathbb{R}^m$, $\mathcal{U}$ open subset of $\mathbb{R}^n$ and $\boldsymbol{x} \in \mathcal{U}$. Then the uniquely defined operator $\mathrm{D}\boldsymbol{f}(\boldsymbol{x})$ is denoted by

$$\begin{aligned} \mathrm{D}\boldsymbol{f} : \mathcal{U} &\to \mathcal{L}is(\mathbb{R}^n, \mathbb{R}^m) \\ \boldsymbol{x} &\mapsto \mathrm{D}\boldsymbol{f}(\boldsymbol{x})(\cdot) \end{aligned}$$

and is called the (first-order) derivative of $\boldsymbol{f}$ at $\boldsymbol{x}$, where the function space of linear isomorphisms between two real-valued vector spaces is denoted by $\mathcal{L}is(\cdot,\cdot)$. One can use equivalently

$$\mathrm{D}\boldsymbol{f}(\boldsymbol{x})(\boldsymbol{v}) = \frac{\partial \boldsymbol{f}(\boldsymbol{x})}{\partial \boldsymbol{x}}\boldsymbol{v} = \mathrm{D}\boldsymbol{f}(\boldsymbol{x})\boldsymbol{v}.$$ △

Definition (Directional Derivative) Let $\boldsymbol{f} : \mathcal{U} \subseteq \mathbb{R}^n \to \mathbb{R}^m$ and $\boldsymbol{x} \in \mathcal{U}$, $\mathcal{U}$ open subset of $\mathbb{R}^n$, and $\boldsymbol{v} \in \mathbb{R}^n \setminus \boldsymbol{0}$. There exists a sufficiently small $h > 0$ s.t. $\boldsymbol{x} + h\boldsymbol{v} \in \mathcal{U}$. If the function $h \to \boldsymbol{f}(\boldsymbol{x} + h\boldsymbol{v})$ is differentiable at $h = 0$, then the derivative

$$\partial \boldsymbol{f}(\boldsymbol{x};\boldsymbol{v}) = \lim_{h\to 0} \frac{\boldsymbol{f}(\boldsymbol{x}+h\boldsymbol{v}) - \boldsymbol{f}(\boldsymbol{x})}{h}$$

is called the (first-order) directional derivative of $\boldsymbol{f}$ at $\boldsymbol{x}$ along the direction $\boldsymbol{v}$. △

Remark For any vector- or matrix-valued function $\boldsymbol{f}$ note that its directional derivative $\partial \boldsymbol{f}(\boldsymbol{x};\boldsymbol{d})$ at $\boldsymbol{x}$ along an arbitrary direction $\boldsymbol{v} \neq \boldsymbol{0}$ is of the same dimension as the nominal function evaluation. However, special properties of the range space can be lost on the derivative itself.

For example, a rotational matrix $\boldsymbol{E} \in \mathcal{SO}(3)$ parametrized by means of a scalar $p \in \mathbb{R}$ is given by the matrix-valued function $\boldsymbol{E} : \mathbb{R} \to \mathbb{R}^{3\times 3}$, $p \mapsto \boldsymbol{E}(p)$, c.f. (2.16). The respective directional derivative $\partial \boldsymbol{E}(p;d)$ is again a (3×3) matrix, but no longer an element of $\mathcal{SO}(3)$.

Proposition (Chain Rule) Let $\boldsymbol{f} : \mathcal{U} \subseteq \mathbb{R}^n \to \mathbb{R}^m$, $\boldsymbol{x} \mapsto \boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x})$ and $\boldsymbol{g} : \mathcal{V} \subseteq \mathbb{R}^m \to \mathbb{R}^l$, $\boldsymbol{y} \mapsto \boldsymbol{z} = \boldsymbol{g}(\boldsymbol{y})$ be differentiable and $\boldsymbol{f}(\mathcal{U}) \subseteq \mathcal{V}$. Then $\boldsymbol{g} \circ \boldsymbol{f} : \mathcal{U} \subseteq \mathbb{R}^n \to \mathbb{R}^l$ is differentiable in $\boldsymbol{x} \in \mathcal{U}$. The derivative is given as

$$\partial(\boldsymbol{g} \circ \boldsymbol{f})(\boldsymbol{x}) = \partial \boldsymbol{g}(\boldsymbol{f}(\boldsymbol{x})) \circ \partial \boldsymbol{f}(\boldsymbol{x}).$$ △

Proof A proof can be found in any textbook covering multivariate analysis, e.g. [4].

2.1.1 Symbolic Derivatives

The idea of computing symbolic derivatives both by hand or by computer algebra packages, e.g. *Mathematica*[4], *Maple*[5] [26] or *SymPy*[6], arises from the circumstance that one is used to look and work with mathematical formulas and expressions given in symbolic form. On the one hand, deriving and implementing derivatives manually is always possible. However, it is both error prone and tedious, and therefore not recommended.

On the other hand, even though good computer algebra tools exist for these cases, there are two known drawbacks from which symbolic computations suffer. First, it can be easily observed, that symbolic expressions often grow exponentially fast, especially when there are recursion or loops involved in their evaluation, c.f. the *Speelpenning*'s problem in [61] or the ray tracing example in [180]. Second, when using symbolic differentiation naively, e.g. for computing the gradient of a function evaluation, we require a single symbolic expression for each partial derivative which is hardly memory efficient. On top of it, these expression will most likely share common sub-expressions or intermediate values which are then re-evaluated for every partial derivative leading also to an inefficient evaluation of the derivative information.

[4] https://www.wolfram.com/mathematica/
[5] https://www.maplesoft.com/products/Maple/
[6] http://www.sympy.org

While this view on symbolic differentiation is more focused on the pitfalls and drawbacks of the approach, there exist examples that apply these techniques successfully. For example, methods for improving the efficiency of symbolic derivative can be found in [64]. In [50], a comparison of two RBD libraries is shown. The software package *RBDL* relies on the efficient evaluation of RBD based on spatial algebra while the library *DYNAMOD* [99] uses the same formulation and leverages the symbolic expression handling of *Maple* to extract optimized *C* code. The comparison shows that the optimized symbolic expressions are able to outperform the numerical evaluation in special cases. Another example is *Drake* [169], which uses the capabilities of *Simulink* and *Matlab* to handle complex expressions and code generation.

2.1.2 Numerical Derivatives

The most common and straightforward way to approximate derivative information of a function $\boldsymbol{f} : \mathcal{U} \subseteq \mathbb{R}^n \to \mathbb{R}^m$ can be achieved by means of the *Taylor* series expansion of $\boldsymbol{f}$ in $\boldsymbol{x} \in \mathcal{U}$ along $\boldsymbol{d} \in \mathcal{U}$, i.e.,

$$\boldsymbol{f}(\boldsymbol{x} + h\boldsymbol{d}) = \boldsymbol{f}(\boldsymbol{x}) + h\partial\boldsymbol{f}(\boldsymbol{x};\boldsymbol{d}) + O\left(h^2\right),$$

such that $\boldsymbol{x} + h\boldsymbol{d} \in \mathcal{U}$ for sufficiently large $h > 0$. Following this, the directional derivative of function $\boldsymbol{f}$ in $\boldsymbol{x}$ along the direction $\boldsymbol{d}$ can be approximated by means of a one-sided (forward) finite difference scheme

$$\partial_{+h}\boldsymbol{f}(\boldsymbol{x};\boldsymbol{d}) = \frac{\boldsymbol{f}(\boldsymbol{x} + h\boldsymbol{d}) - \boldsymbol{f}(\boldsymbol{x})}{h} + O(h).$$

Analogously, one can derive an improved central finite difference scheme by

$$\partial_{\pm h}\boldsymbol{f}(\boldsymbol{x};\boldsymbol{d}) = \frac{\boldsymbol{f}(\boldsymbol{x} + h\boldsymbol{d}) - \boldsymbol{f}(\boldsymbol{x} - h\boldsymbol{d})}{2h} + O\left(h^2\right).$$

The effort to evaluate the approximation of the directional derivative $\partial\boldsymbol{f}(\boldsymbol{x};\boldsymbol{d}) \in \mathbb{R}^m$ along a single direction $\boldsymbol{d}$ is 2 times the effort required for the nominal function evaluation for both finite differencing scheme. In order to compute the full Jacobian matrix $\mathrm{D}\boldsymbol{f}(\boldsymbol{x}) \in \mathbb{R}^{m\times n}$, the one-sided differences requires $1 + n$ times and the central difference scheme $2n$ times the effort of the nominal evaluation of the function $\boldsymbol{f}$.

While FD can easily be implemented on top of the evaluation of the nominal evaluation, the clear drawback of the approach is the loss of precision of the approximation due to truncation errors as the precision critically depends on the magnitude $h\|d\|$. For a small choice of h the dominant error is due to cancellation, while for a large h the truncation of higher terms is responsible for the loss of significant digits. In this way, even though the recommended perturbations of $h\|d\| = \sqrt{eps}$ for the one-sided and $h\|d\| = \sqrt[3]{eps}$ for the central scheme are used, where *eps* denotes the actual machine precision[7], the accuracy of the derivative information that can be achieved is half of the significant digits of $\boldsymbol{f}(\boldsymbol{x})$ for the one-sided and two thirds for central difference scheme at the maximum.

[7] The machine precision *eps* is defined as the smallest positive floating–point number representable by the actual machine such that $1 + eps \neq 1$ holds. For *double precision* arithmetic the resulting machine precision is $eps = 2^{-53} \approx 1.11 \cdot 10^{-16}$ and $eps = 2^{-23} \approx 1.19 \cdot 10^{-7}$ for *single precision* arithmetic.

2.1.3 Automatic Differentiation

In contrast to FD, AD does not treat the evaluation procedure of a function f at point $\boldsymbol{x}$ as a black box, but relies on an actual implementation of the evaluation procedure in terms of a computer program or computational graph. AD exploits the fact that every computer program executes a sequence of elementary arithmetic operations such as additions or elementary functions, e.g. $\exp$, $\cos$ or $\sqrt{\cdot}$. By mechanically applying the chain rule of derivative calculus repeatedly to these operations, the original evaluation procedure is augmented such that derivatives are evaluated simultaneously with the evaluation of the nominal function f. In this way, derivatives of arbitrary order can be computed automatically, and accurate to working precision.

The decomposition of function f as a concatenation of elemental functions can be interpreted by means of the following Definition.

Definition (Factorable Function) Let $\mathcal{L}$ be a finite set of real-valued elemental functions $\varphi_i : \mathbb{R}^n \to \mathbb{R}, \boldsymbol{x} \mapsto \varphi(\boldsymbol{x})$. A function $\boldsymbol{f} : \mathbb{R}^n \to \mathbb{R}^m, \boldsymbol{x} \mapsto \boldsymbol{f}(\boldsymbol{x})$ is called a *factorable function* iff there exists a finite sequence $\{\varphi_{1-n}, \ldots, \varphi_k\}$, $k \geqslant m$, such that it holds:

1. For $1 \leqslant i \leqslant n$ the function $\varphi_{i-n} = \pi_i^n$ is the projection on the i-th component of the evaluation point $\boldsymbol{x} \in \mathbb{R}^n$,
2. For $1 \leqslant i \leqslant m$ the function $\varphi_{k-m+i} = \pi_i^m$ is the projection on the i-th component of the evaluation result $\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x}) \in \mathbb{R}^m$,
3. For $1 \leqslant i \leqslant k-m$ the function φ_i is constant or a concatenation of one or more functions φ_j with $1-n \leqslant j \leqslant i-1$, i.e., of functions preceding φ_i in the concatenation sequence. △

In general, the elemental functions φ_i can be vector-valued functions as well, however this requires then to include linear algebra operations on vectors or matrices as elemental operations. For simplicity, we assume scalar valued elementary functions in this section.

Following the above definition, a factorable function f can be represented by a sequence $\{\varphi_{1-n}, \ldots, \varphi_k\}$ of elemental functions. In this way, an evaluation procedure or a computational graph of the function f given inputs $\boldsymbol{x}$ can be evaluated by algorithm 2.1. The algorithm uses the notation of a slice operator, e.g., $\boldsymbol{x}[1:n]$, and the dependence relation $\prec$, e.g., $v_i = \varphi_i(v_{j \prec i})$, specified by the following definitions.

Definition (Slice operator and indexing, $\boldsymbol{i:j:k}$) The slice operator generalizes the indexing of an vector, s.t. several segments or elements can be extracted at once. The operator is denoted by $i:j:k$, where $i \in \mathbb{N}$ is the lower (inclusive) bound , $j \in \mathbb{N}$ the upper (inclusive) bound and $k \in \mathbb{N}$ the optional step. A slice of a vector $\boldsymbol{x}$ is defined as follows:

$$\boldsymbol{x}_{[i:j:k]} = [x_i, x_{i+k}, x_{i+2k}, \ldots, x_{i+lk}], \tag{2.1}$$

where $l \in \mathbb{N}$ is the largest integer s.t. $i + lk \leqslant j$. For $\boldsymbol{x} \in \mathbb{R}^n$ the indices i, j, k are optional in the slice of a vector $\boldsymbol{x}[i:j:k]$ and by omitting them it is equivalent to setting $i = 1$, $j = n+1$, $k = 1$, e.g. $\boldsymbol{x} = \boldsymbol{x}_{[:]}$. Alternatively, when used as an operator, for example for $\dot{\boldsymbol{A}}, \dot{\boldsymbol{B}}, \dot{\boldsymbol{C}} \in \mathbb{R}^{p\times m\times m}$ and $\boldsymbol{A}, \boldsymbol{B} \in \mathbb{R}^{m\times m}$ slicing is used as an implicit loop over the dimension p s.t.

$$\dot{\boldsymbol{C}}[:] = \dot{\boldsymbol{A}}[:]\boldsymbol{B} + \boldsymbol{B}\dot{\boldsymbol{C}}[:] \Leftrightarrow \dot{\boldsymbol{C}}_i = \dot{\boldsymbol{A}}_i\boldsymbol{B} + \boldsymbol{B}\dot{\boldsymbol{C}}_i,\ i = 1, \ldots, p. \tag{2.2}$$

△

Definition (Precedence Relation $\prec$) The precedence relation $j \prec i$ indicates that the intermediate quantity v_i depends directly on the intermediates v_j for indices $j < i$, i.e., at least one element of the tuple $v_i = \varphi_j(v_{k\prec j})$ is an argument of the function φ_i. Following this, each instruction φ_i can be interpreted by a node in a directed acyclic graph, the computational graph. There is an edge from φ_i to φ_j iff $j \prec i$. △

Algorithm 2.1: Zero-order forward sweep of automatic differentiation.

```
input  : $\varphi_{1-n}, \dots, \varphi_k, \boldsymbol{x}$
output : $\boldsymbol{y} = f(\boldsymbol{x})$
1 $v_{[1-n:0]} = \boldsymbol{x}_{[1:n]}$
2 for $i = 1 : k$ do
3     $v_i = \varphi_i(v_{j<i})$
4 end
5 $\boldsymbol{y}_{[1:m]} = v_{[k-m+1:k]}$
```

Algorithm 2.2: First-order forward sweep of automatic differentiation.

```
input  : $\varphi_{1-n}, \dots, \varphi_k, \boldsymbol{x}, \dot{\boldsymbol{x}}$
output : $\boldsymbol{y} = f(\boldsymbol{x}), \dot{\boldsymbol{y}} = \partial f(\boldsymbol{x}; \dot{\boldsymbol{x}})$
1 $v_{[1-n:0]} = \boldsymbol{x}_{[1:n]}$
2 $\dot{v}_{[1-n:0]} = \dot{\boldsymbol{x}}_{[1:n]}$
3 for $i = 1 : k$ do
4     $v_i = \varphi_i(v_{j<i})$
5     $\dot{v}_i = \sum_{j<i} \frac{\partial \varphi_i}{\partial v_j}(v_{j<i})\, \dot{v}_j$
6 end
7 $\boldsymbol{y}_{[1:m]} = v_{[k-m+1:k]}$
8 $\dot{\boldsymbol{y}}_{[1:m]} = \dot{v}_{[k-m+1:k]}$
```

2.1.4 Forward Mode of Automatic Differentiation

The forward mode of automatic differentiation is used to augment the evaluation procedure of $f(\boldsymbol{x})$ implemented in algorithm 2.1 in order to additionally compute a directional derivative $\dot{\boldsymbol{y}} = \frac{\partial f}{\partial \boldsymbol{x}}(\boldsymbol{x})\dot{\boldsymbol{x}} \equiv \partial f(\boldsymbol{x}; \dot{\boldsymbol{x}})$.

The idea is to propagate tangential information by means of mechanically applying the chain rule to every intermediate evaluation in the form of

$$\dot{v}_i = \sum_{j<i} \frac{\partial \varphi_i}{\partial v_j}(v_{j<i})\, \dot{v}_j \equiv \sum_{j<i} \partial_{v_j} \varphi_i(v_{j<i}; \dot{v}_j).$$

Here, for each elementary operation φ_i it is assumed that there exists the respective elemental tangent operation given by $\frac{\partial \varphi_i}{\partial v_j}(v_{j<i})\dot{v}_j$. In contrast to symbolic differentiation, the intermediate partials $v_i = \frac{\partial \varphi_i}{\partial v_j}(u_i)\dot{u}_i$ are not accumulated as expression of growing complexity and then evaluated but first evaluated as floating point numbers inplace and then accumulated. The resulting evaluation procedure is presented in algorithm 2.1. Algorithm 2.2 can be extended to compute multiple directional derivatives in one sweep.

Remark (Order of Evaluation) In algorithm 2.2, each operation of the nominal evaluation procedure is followed by the respective tangent propagation. Mathematically speaking, the order of evaluation of nominal and first-order propagation does not influence the result. However, in code, problems can occur due to overwriting of variables in memory. For example, in the evaluation of a simple multiplication like $y = x \cdot y$, the respective first-order forward sweep yields the procedure

$$\begin{aligned} &y = x \cdot y, \\ &\dot{y} = \dot{x} \cdot y + x \cdot \dot{y}. \end{aligned}$$

Here, we observe that the value of y is already overwritten in the first line and then applied in the second line, which will render the resulting derivative to be wrong systematically.

Changing the order of nominal and derivative evaluation in the code is one way to prevent this. However, this can lead to the case where expensive computations required for the nominal operations has to be executed twice, e.g. in the evaluation of $y = \sqrt{x}$. For this example, the

Table 2.1: Computational effort and quality of derivatives $\partial \boldsymbol{f}(\boldsymbol{x};\boldsymbol{d})$ for a single direction $\boldsymbol{d} \in \mathbb{R}^n$ or $p \in \mathbb{N}$ directions $\boldsymbol{D} \in \mathbb{R}^{n \times p}$ measured in nominal evaluations of $\boldsymbol{f}(\boldsymbol{x})$, i.e., #eval($\boldsymbol{f}(\boldsymbol{x})$), and the expected accuracy measured in significant digits of the nominal accuracy of the actual machine precision[7] *eps*.

	Effort [#eval($\boldsymbol{f}(\boldsymbol{x})$)]		Precision in
Type	$\boldsymbol{d} \in \mathbb{R}^{\cdot}$	$\boldsymbol{D} \in \mathbb{R}^{n \times p}$	machine prec. *eps*
$\partial_{+h}\boldsymbol{f}(\boldsymbol{x};\cdot)$	2	$1 + p$	$\gtrsim \sqrt{eps}$
$\partial_{\pm h}\boldsymbol{f}(\boldsymbol{x};\cdot)$	2	$1 + 2p$	$\gtrsim \sqrt[3]{eps^2}$
$\partial_{ad}\boldsymbol{f}(\boldsymbol{x};\cdot)$	$\leqslant \frac{5}{2}$	$\leqslant 1 + \frac{3}{2}p$	$\gtrsim eps$

first-order forward sweep then yields two possible procedures

$$\begin{array}{l} \dot{y} = \dfrac{\dot{x}}{2\sqrt{x}}, \\ y = \sqrt{x} \end{array} \quad \text{vs.} \quad \begin{array}{l} y = \sqrt{x} \\ \dot{y} = \dfrac{\dot{x}}{2y} \end{array}$$

Herein, we will always evaluate the nominal operations first in the presentation of algorithms. In code, we will first compute the derivative then evaluate the nominal value when no drawback is expected on the efficiency.

In contrast to the approximation error occuring of FD, the derivatives computed via AD are exact within machine precision, which is supported by *rule* 4 of the textbook [61] by *Griewank*:

> *The Jacobian-vector (and Jacobian transposed vector) products calculated in the forward (and reverse) mode, respectively, correspond to the exact values for an evaluation procedure whose elementals are perturbed at the level of the machine precision.*

According to [61], the theory of AD suggests that the effort to compute the directional derivative $\partial \boldsymbol{f}(\boldsymbol{x};\boldsymbol{d}) \in \mathbb{R}^m$ along a single direction $\boldsymbol{d} \in \mathbb{R}^n$ is bounded by $\frac{5}{2}$ times the effort required for the nominal function evaluation $\boldsymbol{f}(\boldsymbol{x})$. The evaluation of the full Jacobian matrix $\mathrm{D}\boldsymbol{f}(\boldsymbol{x}) \in \mathbb{R}^{m \times n}$ or more generally the effort of propagating a number of p directions $\boldsymbol{D} \in \mathbb{R}^{n \times p}$ in order to compute $\partial \boldsymbol{f}(\boldsymbol{x};\boldsymbol{D}) \in \mathbb{R}^{m \times p}$ the effort is bounded by $1 + \frac{3}{2}p$ times the effort of the nominal evaluation of the function $\boldsymbol{f}$, i.e., it scales linearly in number of directions p. Therefore, in the worst case, theory suggests that they are slightly more expensive than finite difference approximations. Practical applications show that this is most often not the case. We conclude this section with a summary of the time complexity and accuracy observations in Table 2.1.

2.2 Evaluation of Rigid-Body Dynamics using Recursive Algorithms

In Chapter 1, we introduced the descriptor form of the equation of motion of a MBS in contact with its environment, see equation (1.4). Revisiting the appearing quantities, e.g. $\boldsymbol{H}(\boldsymbol{q})$, $\boldsymbol{c}(\boldsymbol{q},\dot{\boldsymbol{q}})$, $\boldsymbol{J}(\boldsymbol{q})$ and $\boldsymbol{\gamma}(\boldsymbol{q},\dot{\boldsymbol{q}})$, where $\boldsymbol{q}$, $\dot{\boldsymbol{q}}$ denote the generalized joint positions and velocities, each of these vector-valued quantities has to be evaluated considering the kinematic tree topology of the respective MBS. In order to solve the actual equation of

λ_i	Parent body index for joint i connecting body i with body λ_i.
$\boldsymbol{S}_i$	Motion space matrix for joint i
$\boldsymbol{v}_i$	Spatial velocity of body i
$\boldsymbol{c}_i$	Velocity dependent acceleration term of body i
$\boldsymbol{a}_i$	Spatial acceleration of body i
$\boldsymbol{f}_i$	Spatial force of body i acting on parent body λ_i via joint i
${}^{i}\boldsymbol{X}_{\lambda_i}$	Spatial transformation from the parent of body i to body i
$\boldsymbol{X}_{\mathrm{T}i}$	Spatial transformation from the parent of body i to the frame of joint i
$\boldsymbol{I}_i$	Spatial inertia of body i
$\boldsymbol{I}_i^c$	Composite body inertia of body i

Table 2.2: Variable definitions of a loop-free rigid multi-body model.

motion, its descriptor form has to be solved by using linear algebra routines including matrix decomposition. In the following, we give a brief overview of the relevant parts of the theory of spatial algebra for rigid-body dynamics required for our work. Afterwards, we present the main recursive algorithms based on spatial algebra to compute the quantities of interest of the equation of motion. Finally, we recap the appropriate linear algebra in order to solve the descriptor form and then introduce our approach to efficiently compute the derivative of the solution with respect to its inputs.

2.2.1 Primer on Spatial Algebra for Rigid-Body Dynamics

In the following sections, we lean on the notation presented in [50] by *Felis*. Therefore, we follow [50] considering mathematical notation and presentation and will refer to this publication concerning the details that are omitted for the sake of brevity. The textbook of *Featherstone* [48] gives an in depth introduction of these concepts.

In order to support the understanding of the following sections, we revisit the overview of variable definitions from [50] in Table 2.2 and give a more detailed explanation on the quantities that play an important role in the remainder of this chapter.

We assume a MBS consisting of a set of $n^{\mathrm{B}} \in \mathbb{N}$ rigid bodies that are interconnected by joints, where each joint defines the relative motion between two bodies. Furthermore, we restrict the MBS to be of a kinematic tree topology, i.e., there exists no kinematic loops (except of possible loops due to contacts with its environment) and there exists a *root* body from which all branches are pure kinematic chains (no loops, always two bodies connected by a single joint).

Considering this configuration of the MBS, we enumerate each body from the root ($i = 0$) up to n^{B} consecutively. In order to mathematically access the above defined topology, we define the following index λ_i that denotes the parent body for joint i for all $i = 1, \ldots, n^{\mathrm{B}}$, where λ_i.

There are two fundamental elements of spatial algebra, spatial motions $\hat{\boldsymbol{v}} \in \mathsf{M}^6$ and spatial forces $\hat{\boldsymbol{f}} \in \mathsf{F}^6$. Motion vectors, for example, describe spatial velocities of bodies, i.e., spatial velocity $\hat{\boldsymbol{v}} \in \mathsf{M}^6$ is able to describe both the linear as well as the angular velocity of a rigid body given by

$$\hat{\boldsymbol{v}} = \begin{bmatrix} \boldsymbol{\omega} \\ \boldsymbol{v}_O \end{bmatrix} = \begin{bmatrix} \omega_x, \omega_y, \omega_z, v_{Ox}, v_{Oy}, v_{Oz}, \end{bmatrix}^T , \tag{2.3}$$

where $O \in \mathbb{R}^3$ denotes an arbitrary reference point, while $\boldsymbol{\omega} \in \mathbb{R}^3$ describes the angular velocity of the body about an axis that passes through O and $\boldsymbol{v}_O \in \mathbb{R}^3$ describes the linear velocity of a body point that currently coincides with reference point O. In contrast to this, spatial forces force components acting on the rigid body, e.g. $\hat{\boldsymbol{f}} \in \mathsf{F}^6$ describes both linear and angular forces given by

$$\hat{\boldsymbol{f}} = \begin{bmatrix} \boldsymbol{n}_O \\ \boldsymbol{f} \end{bmatrix} = \begin{bmatrix} n_{Ox}, n_{Oy}, n_{Oz}, f_x, f_y, f_z, \end{bmatrix}^T, \tag{2.4}$$

where $\boldsymbol{n}_O \in \mathbb{R}^3$ describes the total moment about a body point that currently coincides with reference point O and $\boldsymbol{f} \in \mathbb{R}^3$ the linear force along an axis passing through O. A mapping $\mathcal{D}_O : \mathbb{R}^6 \to \mathsf{M}^6$ and $\mathcal{E}_O : \mathbb{R}^6 \to \mathsf{F}^6$ can be defined by using *Plücker* bases $\mathcal{D}_O, \mathcal{E}_O$, c.f. [47].

The motion quantities $\boldsymbol{v}_i$, $\boldsymbol{c}_i$, $\boldsymbol{a}_i$ of spatial velocity, velocity dependent acceleration and spatial acceleration of body i are quantities of the spatial motion space M^6. The force $\boldsymbol{f}_i$ of body i acting on the parent body λ_i is a quantity of spatial force space F^6.

Spatial algebra defines a cross product for motion vectors by $\times$ and another for forces by $\times^*$. For $\hat{\boldsymbol{v}} \in \mathsf{M}^6$, they read

$$\hat{\boldsymbol{v}}\times = \begin{bmatrix} \boldsymbol{\omega}\times & \boldsymbol{0} \\ \boldsymbol{v}_O\times & \boldsymbol{\omega}\times \end{bmatrix}, \tag{2.5a}$$

$$\hat{\boldsymbol{v}}\times^* = \begin{bmatrix} \boldsymbol{\omega}\times & \boldsymbol{v}_O\times \\ \boldsymbol{0} & \boldsymbol{\omega}\times \end{bmatrix}. \tag{2.5b}$$

Please note, that the adjoint cross product is not skew symmetric in contrast to its nominal counterpart.

As mentioned above, the joints define the relative motion of one body with respect to the other. Furthermore, each joint i defines the respective motion degrees of freedom (DoFs) by its motion space matrix $\boldsymbol{S}_i$. Here, we restrict ourselves to single-DoF joints for convenience, but multi-DoF joints can easily derived by looking up the details in [48]. Following this, the motions subspace matrices are given by

$$\boldsymbol{S}_{Rx} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad \boldsymbol{S}_{Ry} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad \boldsymbol{S}_{Rz} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \tag{2.6a}$$

$$\boldsymbol{S}_{Tx} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \quad \boldsymbol{S}_{Ty} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \quad \boldsymbol{S}_{Tz} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}. \tag{2.6b}$$

Given the motion subspace matrices as above, we can express the velocities and accelerations of the ith joint as

$$\boldsymbol{v}_{\mathrm{J}i} = \boldsymbol{S}_i \dot{\boldsymbol{q}}_i \tag{2.7}$$

$$\boldsymbol{a}_{\mathrm{J}i} = \dot{\boldsymbol{S}}_i \dot{\boldsymbol{q}}_i + \boldsymbol{S}_i \ddot{\boldsymbol{q}}_i \tag{2.8}$$

$$= \left(\frac{\mathrm{d}\boldsymbol{S}_i}{\mathrm{d}t} + \boldsymbol{v}_i \times \boldsymbol{S}_i \right) \dot{\boldsymbol{q}}_i + \boldsymbol{S}_i \ddot{\boldsymbol{q}}_i, \tag{2.9}$$

where the cross product term arises from the effect of the motion of the respective frame of body i on the change motion subspace. A formal definition of the time derivative with respect to a moving frame is given in Definition 2.13. However, this value is always zero for single degree of freedom joints as investigated in this thesis. The cross term is sometimes referred to as *velocity dependent spatial acceleration term* and denoted by $\boldsymbol{c}_{\mathrm{J}i}$ in the remainder of this chapter.

Another quantity of interest is spatial inertia which itself defines a mapping between motion and force spaces $\hat{\boldsymbol{I}} : \mathsf{M}^6 \to \mathsf{F}^6$. The spatial inertia term of body i $\boldsymbol{I}_i$ and the i–th composite body inertia term $\boldsymbol{I}_i^c$ play a minor role in the following sections as they are constant with respect to the quantities we evaluate the derivatives at. However, we give a formal definition for completeness.

Definition (Spatial Inertia $\hat{\boldsymbol{I}}$) The spatial inertia $\hat{\boldsymbol{I}}$ can be expressed as a 6×6 matrix:

$$\hat{\boldsymbol{I}} = \begin{bmatrix} \boldsymbol{I}_C + m\boldsymbol{c} \times \boldsymbol{c} \times^T & m\boldsymbol{c}\times \\ m\boldsymbol{c}\times^T & m\boldsymbol{1} \end{bmatrix}, \tag{2.10}$$

where $\boldsymbol{I}_C$ is the inertia of the body at its center of mass, $\boldsymbol{c}$ is the 3-D coordinate vector of the center of mass, and m is the mass of the body.

For a body with spatial velocity $\hat{\boldsymbol{v}}$ and spatial inertia $\hat{\boldsymbol{I}}$ we can compute the spatial momentum as $\hat{\boldsymbol{h}} = \hat{\boldsymbol{I}}\hat{\boldsymbol{v}}$. One should note that $\hat{\boldsymbol{h}} \in \mathsf{F}^6$. △

Definition (Spatial Inertia Projections) We define the following projections $\hat{\boldsymbol{I}}\big|_{\boldsymbol{I}_C}$, $\hat{\boldsymbol{I}}\big|_{\mathrm{m}\boldsymbol{c}}$ and $\hat{\boldsymbol{I}}\big|_{\mathrm{m}}$ of a spatial inertia term to be projections on the respective blocks given by

$$\hat{\boldsymbol{I}}\big|_{\boldsymbol{I}_C} = \boldsymbol{P}_{\boldsymbol{I}_C}^T \hat{\boldsymbol{I}} \boldsymbol{P}_{\boldsymbol{I}_C} = \boldsymbol{I}_C, \tag{2.11}$$

$$\hat{\boldsymbol{I}}\big|_{\mathrm{m}\boldsymbol{c}} = \times^{-1}(\boldsymbol{P}_{m\boldsymbol{c}\times}^T \hat{\boldsymbol{I}} \boldsymbol{P}_{m\boldsymbol{c}\times}) = \mathrm{m}\boldsymbol{c}, \tag{2.12}$$

$$\hat{\boldsymbol{I}}\big|_{\mathrm{m}} = \boldsymbol{P}_{\mathrm{m}}^T \hat{\boldsymbol{I}} \boldsymbol{P}_{\mathrm{m}} = \mathrm{m}, \tag{2.13}$$

and $\times^{-1}(\cdot)$ retrieves the vector quantity from an $\times$ operator according to $\times^{-1}(\boldsymbol{c}\times) = \boldsymbol{c}$. △

The critical quantities in terms of derivative evaluations are the spatial or *Plücker* transformations denoted by $\boldsymbol{X}$, e.g. ${}^i\boldsymbol{X}_{\lambda_i}$, $\boldsymbol{X}_{\mathrm{T}i} \in \mathbb{R}^{6\times 6}$ the different spatial transformations from parent to the body or the joint, which describe the transformation between the Cartesian coordinate frames located at each body describing their position and orientation with respect to to their parent frame.

Definition (Spatial Transformation) Let A and B be two coordinate frames, where B is translated by $\boldsymbol{r} \in \mathbb{R}^3$ and rotated by the orthonormal matrix $\boldsymbol{E} \in \mathcal{SO}(3)$ relatively to A, then the respective spatial transformation $\boldsymbol{X}(\boldsymbol{E}, \boldsymbol{r})$ is given by

$${}^B\boldsymbol{X}_A(\boldsymbol{E}, \boldsymbol{r}) = \begin{bmatrix} \boldsymbol{E} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{E} \end{bmatrix} \begin{bmatrix} \boldsymbol{1} & \boldsymbol{0} \\ -\boldsymbol{r}\times & \boldsymbol{1} \end{bmatrix} = \begin{bmatrix} \boldsymbol{E} & \boldsymbol{0} \\ -\boldsymbol{E}\boldsymbol{r}\times & \boldsymbol{E} \end{bmatrix}. \tag{2.14}$$

Here, $\mathcal{SO}(3)$ denotes the special orthogonal group of rotational matrices and by $\boldsymbol{r}\times$ we denote the skew symmetric matrix of the translational vector $\boldsymbol{r}$ given by

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} \times = \begin{bmatrix} 0 & -z & y \\ z & 0 & -x \\ -y & x & 0 \end{bmatrix}. \tag{2.15}$$

△

In our case, $\boldsymbol{E}$ is either $\boldsymbol{1}$ or one of the matrices $\boldsymbol{E}_{Rx}$, $\boldsymbol{E}_{Ry}$, $\boldsymbol{E}_{Rz}$, where the index indicates a rotational matrix around the respective axis parametrized by a scalar q, e.g.

$$\boldsymbol{E}_{Rx}(q) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(q) & \sin(q) \\ 0 & -\sin(q) & \cos(q) \end{bmatrix}. \tag{2.16}$$

Spatial transformations of the form of (2.14) are used to transform elements of motion space M^6. For elements of spatial force space F^6 the adjoint operator ${}^B\boldsymbol{X}_A^*$ has to be applied, which is given by

$${}^B\boldsymbol{X}_A^* = \begin{bmatrix} \boldsymbol{E} & -\boldsymbol{E}\boldsymbol{r}\times \\ \boldsymbol{0} & \boldsymbol{E} \end{bmatrix}. \tag{2.17}$$

The special structure of the spatial transformations enables to implement structure exploiting linear algebra techniques to efficiently evaluate the respective operations like matrix-matrix or matrix-vector products, c.f. [48].

2.2.2 Derivatives of Spatial Transformations

First, we need to introduce the problem arising from differentiation in moving coordinates. For constant basis vectors are constants, the component-wise derivative of the coordinate vector is the solution. Otherwise, the motion of the basis vectors needs to be taken into account as presented by the following definition.

Definition Let A be the *Plücker* coordinate system associated with a Cartesian frame. The frame is moving with a spatial velocity of $\boldsymbol{v}_A$. Given spatial vectors $\boldsymbol{m} \in \mathsf{M}^6$ and $\boldsymbol{f} \in \mathsf{F}^6$ then A^m and A^f denote their respective coordinate vector representation in A. The coordinate vectors that represent $\dot{\boldsymbol{m}}$ and $\dot{\boldsymbol{f}}$ in the coordinates of A are given respectively by

$${}^A\left(\frac{\mathrm{d}\boldsymbol{m}}{\mathrm{d}t}\right) = \frac{\mathrm{d}^A\boldsymbol{m}}{\mathrm{d}t} +^A \boldsymbol{v}_A \times {}^A\boldsymbol{m}, \tag{2.18}$$

$${}^A\left(\frac{\mathrm{d}\boldsymbol{f}}{\mathrm{d}t}\right) = \frac{\mathrm{d}^A\boldsymbol{f}}{\mathrm{d}t} +^A \boldsymbol{v}_A \times^* {}^A\boldsymbol{f}, \tag{2.19}$$

where, for the sake of brevity, we further define for any coordinate vector ${}^Av$ representing v in coordinates of A

$${}^A\dot{\boldsymbol{v}} :=^A \left(\frac{\mathrm{d}\boldsymbol{v}}{\mathrm{d}t}\right), \tag{2.20}$$

$${}^A\mathring{\boldsymbol{v}} := \frac{\mathrm{d}^A\boldsymbol{v}}{\mathrm{d}t}. \tag{2.21}$$

△

Now, we introduce manipulations of spatial transformations as elementary operations in the AD sense. Spatial transformations are products of two matrices in special form, which leads to Proposition 2.14.

Proposition (Spatial Transformation Derivative) Let A and B be frames with spatial velocities $\boldsymbol{v}_A, \boldsymbol{v}_B \in \mathsf{M}^6$, for any motion vector $\boldsymbol{m} \in \mathsf{M}^6$ or force vector $\boldsymbol{f} \in \mathsf{F}^6$, the derivative of the spatial transforms are given by

$$\partial_t({}^B\boldsymbol{X}_A)^A\boldsymbol{m} = {}^B(\boldsymbol{v}_A - \boldsymbol{v}_B) \times {}^B\boldsymbol{X}_A{}^A\boldsymbol{m}, \tag{2.22a}$$
$$\partial_t({}^A\boldsymbol{X}_B^*)^B\boldsymbol{f} = {}^B(\boldsymbol{v}_B - \boldsymbol{v}_A) \times^* {}^A\boldsymbol{X}_B^{*B}\boldsymbol{f}. \tag{2.22b}$$

If A and B are connected by a single-DoF joint defined by motion matrix $\boldsymbol{S}$, i.e., the $\boldsymbol{X}$ is parametrized by $q(t)$, this yields

$$\partial_t({}^B\boldsymbol{X}_A(q(t)); \dot{q})\boldsymbol{m} = [({}^B\boldsymbol{X}_A\boldsymbol{m})\times]\boldsymbol{S}\dot{q}, \tag{2.23a}$$
$$\partial_t({}^A\boldsymbol{X}_B^*(q(t)); \dot{q})\boldsymbol{f} = {}^A\boldsymbol{X}_B^*[\times^*\boldsymbol{f}]\boldsymbol{S}\dot{q}, \tag{2.23b}$$

where we define $[\times^*\boldsymbol{f}]$ as the matrix encoding the operation $\cdot \times^* \boldsymbol{f}$ because $\times^*$ is not skew symmetric, see (2.5). △

Proof A proof is given in [48] for the first case, the second follows analogously, i.e., by applying Definition 2.13, we receive the following equations

$${}^A\dot{\boldsymbol{f}} = {}^A\boldsymbol{X}_B^*\,{}^B\dot{\boldsymbol{f}}, \tag{2.24}$$
$${}^A\mathring{\boldsymbol{f}} = \left(\frac{\mathrm{d}}{\mathrm{d}t}{}^A\boldsymbol{X}_B^*\right){}^B\boldsymbol{f} + {}^A\boldsymbol{X}_B^*\,{}^B\mathring{\boldsymbol{f}}, \tag{2.25}$$
$${}^B\dot{\boldsymbol{f}} = {}^B\mathring{\boldsymbol{f}} + {}^B\boldsymbol{v}_B \times {}^B\boldsymbol{f}, \tag{2.26}$$
$${}^A\dot{\boldsymbol{f}} = {}^A\mathring{\boldsymbol{f}} + {}^A\boldsymbol{v}_A \times {}^A\boldsymbol{f}. \tag{2.27}$$

Following this, the derivative is then derived via

$$\left(\frac{\mathrm{d}}{\mathrm{d}t}{}^A\boldsymbol{X}_B^*\right){}^B\dot{\boldsymbol{f}} = {}^A\mathring{\boldsymbol{f}} - {}^A\boldsymbol{X}_B^*\,{}^B\mathring{\boldsymbol{f}} \tag{2.28}$$
$$= {}^A\mathring{\boldsymbol{f}} - {}^A\boldsymbol{v}_A \times {}^A\boldsymbol{f} - {}^A\boldsymbol{X}_B^*\left({}^B\dot{\boldsymbol{f}} + {}^B\boldsymbol{v}_B \times {}^B\boldsymbol{f}\right) \tag{2.29}$$
$$= \underbrace{{}^A\dot{\boldsymbol{f}} - {}^A\boldsymbol{X}_B^*\,{}^B\dot{\boldsymbol{f}}}_{=\boldsymbol{0}} + {}^A\boldsymbol{X}_B^*\,{}^B\boldsymbol{v}_B \times {}^B\boldsymbol{f} - {}^A\boldsymbol{v}_A \times {}^A\boldsymbol{X}_B^*\,{}^B\boldsymbol{f} \tag{2.30}$$
$$= \left({}^A\boldsymbol{X}_B^*\,{}^B\boldsymbol{v}_B \times - {}^A\boldsymbol{v}_A \times {}^A\boldsymbol{X}_B^*\right){}^B\boldsymbol{f}. \tag{2.31}$$

As spatial coordinate vector ${}^B\boldsymbol{f}$ is arbitrary, this yields the result

$$\frac{\mathrm{d}}{\mathrm{d}t}\left({}^A\boldsymbol{X}_B^*\right) = {}^B(\boldsymbol{v}_A - \boldsymbol{v}_B) \times {}^A\boldsymbol{X}_B^*. \tag{2.32}$$

Assuming the spatial transform to be parametrized by means of a motion subspace matrix according to (2.6) and replacing the relative velocity by the joint velocity of (2.7), then the result becomes

$$\frac{\mathrm{d}}{\mathrm{d}t}\left({}^A\boldsymbol{X}_B(q(t))^*; \dot{q}\right){}^B\boldsymbol{f} = {}^A\boldsymbol{X}_B^*\left(\boldsymbol{S}\dot{q} \times^* {}^B\boldsymbol{f}\right). \tag{2.33}$$

For the nominal spatial transform the same result can be easily shown but is left out here. The claim follows from replacing the cross product by a matrix operation in order to reformulate the expression as efficient matrix-vector product according to (2.23a), (2.23b). □

Remark Note that the derivative of a spatial transform from Proposition 2.14 is no longer a spatial transform, $\boldsymbol{X} \equiv \boldsymbol{X}(\boldsymbol{E}, \boldsymbol{r})$, because the identity $\boldsymbol{E}^T\boldsymbol{E} = \boldsymbol{1}$ does not hold for $\partial\boldsymbol{E}$ in general, i.e.,

$$\partial({}^B\boldsymbol{X}_A)(\boldsymbol{E}, \boldsymbol{r}; \partial\boldsymbol{E}, \partial\boldsymbol{r}) \neq \boldsymbol{X}(\partial\boldsymbol{E}, \partial\boldsymbol{E}\boldsymbol{r} + \boldsymbol{E}\partial\boldsymbol{r}). \tag{2.34}$$

The property of the codomain is lost because the directional derivative $\partial\boldsymbol{E}(p;d) \in \mathbb{R}^{3\times3}$ of $\boldsymbol{E} : \mathbb{R}^p \to \mathbb{R}^{3\times3}$ mapping to $\mathcal{SO}(3)$ is not necessarily an element of $\mathcal{SO}(3)$ anymore.

Proposition (Derivative of Spatial Inertia Transformation) Let A and B be frames with spatial velocities $\boldsymbol{v}_A, \boldsymbol{v}_B \in \mathsf{M}^6$ and $\boldsymbol{I}$ the spatial inertia of a body in A, the derivative of the transformation ${}^A\boldsymbol{I} = {}^A\boldsymbol{X}_B^*\, {}^B\boldsymbol{I}\, {}^B\boldsymbol{X}_A$ of $\boldsymbol{I}$ from B to A is given by

$$\partial_t({}^A\boldsymbol{I}) = [{}^A(\boldsymbol{v}_B - \boldsymbol{v}_A)\times^*]^B\boldsymbol{I} - {}^B\boldsymbol{I}[{}^A(\boldsymbol{v}_B - \boldsymbol{v}_A)\times].$$

Proof Claim follows from [48] and using Proposition 2.14. Assuming the spatial transform to be parametrized by means of a motion subspace matrix according to (2.6) and replacing the relative velocity by the joint velocity of (2.7), the result is derived analogously as in Proposition 2.14. □

Re-using the structure of the nominal spatial transforms allows the derivative information to be efficiently stored and computed in motion and force space efficiently. Because the associated derivatives are straightforward consequences of the results just derived, they serve as elementary operations in the AD methodology.

2.3 Evaluation of Rigid-Body Dynamics

In this section, we briefly recap the well-known descriptor form of the equation of motion of a MBS subject to external contacts expressed via kinematic constraints as presented in detail in Section 1.3, which is given by

$$\begin{bmatrix} \boldsymbol{H}(\boldsymbol{q}) & \boldsymbol{J}(\boldsymbol{q})^T \\ \boldsymbol{J}(\boldsymbol{q}) & 0 \end{bmatrix} \begin{bmatrix} \ddot{\boldsymbol{q}} \\ -\boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \boldsymbol{S}^T\boldsymbol{\tau} - \boldsymbol{c}(\boldsymbol{q},\dot{\boldsymbol{q}}) \\ -\boldsymbol{\gamma}(\boldsymbol{q},\dot{\boldsymbol{q}}) \end{bmatrix} \tag{2.35}$$

with generalized positions, velocities, accelerations and forces $\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}, \boldsymbol{\tau} : \mathbb{R} \to \mathbb{R}^{n^{\text{DoF}}}$, symmetric positive definite joint space inertia $\boldsymbol{H} : \mathbb{R}^{n^{\text{DoF}}} \to \mathbb{R}^{n^{\text{DoF}}\times n^{\text{DoF}}}$, the nonlinear effects vector $\boldsymbol{c} : \mathbb{R}^{n^{\text{DoF}}} \times \mathbb{R}^{n^{\text{DoF}}} \to \mathbb{R}^{n^{\text{DoF}}}$, e.g. *Coriolis*, centrifugal, and gravity terms, and a selection matrix $\boldsymbol{S} \in \mathbb{R}^{n^{\text{DoF}}\times n^{\text{DoF}}}$ mapping the joint torques $\boldsymbol{\tau}$ to the actuated DoFs. We focus on scleronomous holonomic constraints of the form $\boldsymbol{g}(\boldsymbol{q}(t)) = 0$ with $\boldsymbol{g} : \mathbb{R}^{n^{\text{DoF}}} \to \mathbb{R}^{n^{\text{G}}}$, where a contact force $\boldsymbol{\lambda} : \mathbb{R} \to \mathbb{R}^{n^{\text{G}}}$ acts on the system by means of the transposed contact Jacobian $\boldsymbol{J}$. The term $\boldsymbol{\gamma}(\boldsymbol{q},\dot{\boldsymbol{q}}) = \frac{\partial\dot{g}(q)}{\partial t}\dot{\boldsymbol{q}}$ is called the contact Hessian. Loop constraints can be modeled by the same means and yielding the same equation as above, see [48] for more details.

The same holds for infinitesimal contact events in the form of equation

$$\begin{bmatrix} \boldsymbol{H}(\boldsymbol{q}) & \boldsymbol{J}(\boldsymbol{q})^T \\ \boldsymbol{J}(\boldsymbol{q}) & 0 \end{bmatrix} \begin{bmatrix} \dot{\boldsymbol{q}}^+ \\ \boldsymbol{\Lambda} \end{bmatrix} = \begin{bmatrix} \boldsymbol{H}(\boldsymbol{q})\dot{\boldsymbol{q}}^- \\ \boldsymbol{v}^- \end{bmatrix}, \tag{2.36}$$

where $\boldsymbol{\Lambda} \in \mathbb{R}^{n^{\text{G}}}$ is the instantaneous force impulse, the normal velocities of the contact before is $\boldsymbol{v}^- \in \mathbb{R}^{n^{\text{G}}}$, and superscripts $\pm$ denote time instants before and after the event. We refer to Section 1.3 for the details on the contact event.

Each of the involved vector- or matrix-valued quantities, e.g. $\boldsymbol{H}(\boldsymbol{q})$, $\boldsymbol{c}(\boldsymbol{q},\dot{\boldsymbol{q}})$, $\boldsymbol{J}(\boldsymbol{q})$ and $\boldsymbol{\gamma}(\boldsymbol{q},\dot{\boldsymbol{q}})$ has to be evaluated considering the kinematic topology of the respective MBS and the descriptor form of the equation of motions has to be solved by using linear algebra routines employing matrix decomposition techniques. We refer to [10] for general approaches to solve saddle point problems like (2.35), i.e., null or range (*Schur*) space approaches. The latter are especially appealing as structure exploiting algorithms exist to compute the inverse of the joint space inertia matrix $\boldsymbol{H}$, c.f. [48].

2.3.1 Recursive *Newton–Euler* Algorithm

Algorithm 2.3: Recursive *Newton–Euler* Algorithm (RNEA)

```
input  : $q$, $\dot{q}$, $\ddot{q}$
output : $\lambda = \mathrm{RNEA}(q, \dot{q}, \ddot{q})$
1  $v_0 = 0$
2  $a_0 = -a_g$
3  for $i = 1, \ldots, n_B$ do
4      $\left[X_{\mathrm{J}i}, S_i, v_{\mathrm{J}i}, c_{\mathrm{J}i}\right] = \mathrm{jcalc}(\mathrm{jtype}(i), q_i, \dot{q}_i)$
5      ${}^iX_{\lambda_i} = X_{\mathrm{J}i} X_{\mathrm{T}i}$
6      $v_i = {}^iX_{\lambda_i} v_{\lambda_i} + v_{\mathrm{J}i}$
7      $c_i = c_{\mathrm{J}i} + v_i \times v_{\mathrm{J}i}$
8      $a_i = {}^iX_{\lambda_i} a_{\lambda_i} + c_i + S_i \ddot{q}_i$
9      $h_i = I_i v_i$
10     $f_i = I_i a_i + v_i \times^* h_i - f_i^x$
11 end
12 for $i = n_B, \ldots, 1$ do
13     $\tau_i = S_i^T f_i$
14     if $\lambda_i \neq 0$ then
15         $f_{\lambda_i} = f_{\lambda_i} + {}^{\lambda_i}X_i^* f_i$
16     end
17 end
```

The equation of motion in (2.35) requires that the nonlinear effects term $c(q, \dot{q})$, which summarizes the effects of *Coriolis*, centrifugal, and gravity terms, is efficiently computed. For each involved quantities an efficient recursive algorithm is available, for the nonlinear effects one can use an adaption of the recursive *Newton-Euler* algorithm (RNEA).

RNEA computes the inverse dynamics (ID) of a MBS with tree topology in $\mathcal{O}(n^{\mathrm{B}})$. For the setup of (2.35), we apply the RNEA to compute the $c(q, \dot{q})$ by setting $\ddot{q} = 0$. RNEA comprises three steps:

1. compute position, velocity and acceleration of each single body,
2. compute the net force causing the acceleration of the body, and
3. back propagation of the force that is applied in each joint.

Algorithm 2.3 gives a pseudo code version of RNEA. Here, *jcalc* refers to a function that computes all joint specific quantities, the ith spatial transform $X_{\mathrm{J}i}$, motion subspace matrix S_i according to (2.6), and the velocity cross term from (2.8), while *jtype* denotes the respective single-DoF joint as defined in (2.8). The consideration of external forces f_i^x which are assumed to be already expressed in the respective joint frame, is possible in general. However, we assume them to be constant for the algorithm for the sake of brevity. We apply the principle of AD and obtain the resulting pseudo code in Algorithm 2.4. The correctness of the derivative is captured in Theorem 2.17.

Theorem 2.17 (Derivative of RNEA) Let $q, \dot{q}, \ddot{q} \in \mathbb{R}^{n^{\mathrm{B}}}$ define the configuration, velocity and acceleration of a MBS of tree topology and $\partial q, \partial \dot{q}, \partial \ddot{q} \in \mathbb{R}^{n^{\mathrm{B}} \times \mathrm{D}}$ the respective directions. When the respective general joint forces $\tau \in \mathbb{R}^{n^{\mathrm{B}}}$ are computed by means of inverse dynamics, i.e., $\tau = \mathrm{RNEA}(q, \dot{q}, \ddot{q})$, where recursive *Newton-Euler* algorithm (RNEA) refers to the application of Algorithm 2.3, then respective directional derivative of the inverse dynamics with respect to above

Algorithm 2.4: RNEA Derivative

```
input  : $\boldsymbol{q}, \partial\boldsymbol{q}, \dot{\boldsymbol{q}}, \partial\dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}, \partial\ddot{\boldsymbol{q}}$
output : $\boldsymbol{\tau}, \partial\boldsymbol{\tau} = \partial\mathrm{RNEA}(\boldsymbol{q}, \partial\boldsymbol{q}, \dot{\boldsymbol{q}}, \partial\dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}, \partial\ddot{\boldsymbol{q}})$
$\boldsymbol{v}_0 = \boldsymbol{0}$                                                  // Nominal
$\partial\boldsymbol{v}_0[:] = \boldsymbol{0}$
$\boldsymbol{a}_0 = -\boldsymbol{a}_g$                                                // Nominal
$\partial\boldsymbol{a}_0[:] = \boldsymbol{0}$
for $i = 1, \ldots, n_B$ do
    $\left[\boldsymbol{X}_{\mathrm{J}i}, \boldsymbol{S}_i, \boldsymbol{v}_{\mathrm{J}i}, \boldsymbol{c}_{\mathrm{J}i}\right] = \mathrm{jcalc}(\mathrm{jtype}(i), \boldsymbol{q}_i, \dot{\boldsymbol{q}}_i)$   // Nominal, ED reuses quantities
    ${}^{i}\boldsymbol{X}_{\lambda_i} = \boldsymbol{X}_{\mathrm{J}i}\boldsymbol{X}_{\mathrm{T}i}$                   // Nominal, ED reuses quantities
    $\boldsymbol{v}_i = {}^{i}\boldsymbol{X}_{\lambda_i}\boldsymbol{v}_{\lambda_i}$                                  // Nominal
    $\partial\boldsymbol{v}_i = \boldsymbol{v}_i \times \boldsymbol{S}_i\partial\boldsymbol{q}_i + {}^{i}\boldsymbol{X}_{\lambda_i}\partial\boldsymbol{v}_{\lambda_i} + \boldsymbol{S}_i\partial\dot{\boldsymbol{q}}$
    $\boldsymbol{v}_i = \boldsymbol{v}_i + \boldsymbol{v}_{\mathrm{J}i}$                            // Nominal code continued
    $\boldsymbol{c}_i = \boldsymbol{c}_{\mathrm{J}i} + \boldsymbol{v}_i \times \boldsymbol{v}_{\mathrm{J}i}$                                  // Nominal
    $\partial\boldsymbol{c}_i = \boldsymbol{v}_i \times \boldsymbol{S}_i\partial\dot{\boldsymbol{q}}_i - \boldsymbol{v}_{\mathrm{J}i} \times \partial\boldsymbol{v}_i$
    $\boldsymbol{a}_i = {}^{i}\boldsymbol{X}_{\lambda_i}\boldsymbol{a}_{\lambda_i} + \boldsymbol{c}_i + \boldsymbol{S}_i\ddot{\boldsymbol{q}}_i$                          // Nominal
    $\partial\boldsymbol{a}_i = \partial\boldsymbol{a}_i \times \boldsymbol{S}_i\partial\boldsymbol{q}_i + {}^{i}\boldsymbol{X}_{\lambda_i}\partial\boldsymbol{a}_{\lambda_i} + \partial\boldsymbol{c}_i + \boldsymbol{S}_i\partial\ddot{\boldsymbol{q}}_i$
    $\boldsymbol{a}_i = \boldsymbol{c}_i + \boldsymbol{S}_i\ddot{\boldsymbol{q}}_i$                            // Nominal code continued
    $\boldsymbol{h}_i = \boldsymbol{I}_i\boldsymbol{v}_i$                                          // Nominal
    $\boldsymbol{f}_i = \boldsymbol{I}_i\boldsymbol{a}_i + \boldsymbol{v}_i \times^* \boldsymbol{h}_i - \boldsymbol{f}_i^x$                       // Nominal
    $\partial\boldsymbol{f}_i = \boldsymbol{I}_i\partial\boldsymbol{a}_i + \partial\boldsymbol{v}_i \times^* \boldsymbol{I}_i\boldsymbol{v}_i + \boldsymbol{v}_i\boldsymbol{h}_i \times^* \partial\boldsymbol{v}_i$
end
for $i = n_B, \ldots, 1$ do
    $\boldsymbol{\tau}_i = \boldsymbol{S}_i^T\boldsymbol{f}_i$                                          // Nominal
    $\partial\boldsymbol{\tau}_i = \boldsymbol{S}_i^T\partial\boldsymbol{f}_i$
    if $\lambda_i \neq 0$ then
        $\boldsymbol{f}_{\lambda_i} = \boldsymbol{f}_{\lambda_i} + {}^{\lambda_i}\boldsymbol{X}_i^*\boldsymbol{f}_i$                         // Nominal
        $\partial\boldsymbol{f}_{\lambda_i} = \partial\boldsymbol{f}_{\lambda_i} + {}^{\lambda_i}\boldsymbol{X}_i^*\left[\times^*\boldsymbol{f}_i\right]\boldsymbol{S}_i\partial\boldsymbol{q}_i + \partial\boldsymbol{f}_i$
    end
end
```

directions, that is

$$\partial\boldsymbol{\tau} = \partial\text{RNEA}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}; \partial\boldsymbol{q}, \partial\dot{\boldsymbol{q}}, \partial\ddot{\boldsymbol{q}}), \tag{2.37}$$

with derivative $\partial\boldsymbol{\tau} \in \mathbb{R}^{n^{\mathrm{B}} \times \mathrm{D}}$, is given by Algorithm 2.4. △

Proof Proposition 2.14 suggests how to compute the derivative of the algorithm by propagating $\partial\boldsymbol{q}$, $\partial\dot{\boldsymbol{q}}$, and $\partial\ddot{\boldsymbol{q}}$ by means of derivative calculus respectively. In line 6 of the algorithm, we do not need to compute the derivative of the spatial transform in advance. We are then able to apply Proposition 2.14 by multiplying with the directions as matrix-vector products efficiently, e.g., in line 9, 14, and 25. The correctness then follows from applying derivative calculus and exploiting the linearity of the cross product. □

Remark Naively applying AD will lead to the same correct results but leads to an implementation that relies heavily on loops instead of the application of efficient matrix-vector products. For example, line 6 in Algorithm 2.3 yields naively

$$\boldsymbol{v}_i[:] = \partial^i\boldsymbol{X}_{\lambda_i}[:]\boldsymbol{v}_{\lambda_i} + {}^i\boldsymbol{X}_{\lambda_i}\partial\boldsymbol{v}_{\lambda_i}[:] + \partial\boldsymbol{v}_{\mathrm{J}i}[:].$$

In order to distinguish the two correct implementations, we refer to the naive implementation by AD and the efficient evaluation scheme by efficient algorithmic differentiation (ED).

2.3.2 Composite Rigid-Body Algorithm

Algorithm 2.5: Composite Rigid-Body Algorithm

```
input  : $\boldsymbol{q}$,
output : $\boldsymbol{H} = \text{CRBA}(\boldsymbol{q})$
1  $\boldsymbol{H} = \boldsymbol{0}$
2  for $i = 1, \dots, n_B$ do
3      $[\boldsymbol{X}_{\mathrm{J}i}, \boldsymbol{S}_i] = \text{jcalc}(\text{jtype}(i), \boldsymbol{q}_i, \dot{\boldsymbol{q}}_i)$
4      ${}^i\boldsymbol{X}_{\lambda_i} = \boldsymbol{X}_{\mathrm{J}i}\boldsymbol{X}_{\mathrm{T}i}$
5      $\boldsymbol{I}_i^c = \boldsymbol{I}_i$
6  end
7  for $i = n_B, \dots, 1$ do
8      if $\lambda_i \neq 0$ then
9          $\boldsymbol{I}_{\lambda_i}^c = \boldsymbol{I}_{\lambda_i}^c + {}^{\lambda_i}\boldsymbol{X}_i^* \boldsymbol{I}_i^c\, {}^i\boldsymbol{X}_{\lambda_i}$
10     end
11     $\boldsymbol{F} = \boldsymbol{I}_i^c \boldsymbol{S}_i$
12     $\boldsymbol{H}_{ii} = \boldsymbol{S}_i^T \boldsymbol{F}$
13     $j = i$
14     while $\lambda_j \neq 0$ do
15         $\boldsymbol{F} = {}^{\lambda_i}\boldsymbol{X}_j^* \boldsymbol{F}$
16         $j = \lambda_j$
17         $\boldsymbol{H}_{ij} = \boldsymbol{F}^T \boldsymbol{S}_j$
18         $\boldsymbol{H}_{ji} = \boldsymbol{H}_{ij}^T$
19     end
20 end
```

The joint space inertia matrix $\boldsymbol{H}(\boldsymbol{q})$ can be efficiently computed by means of the composite rigid-body algorithm. It computes the non-zero entries by recursively assembling the required composite rigid-body inertia $\boldsymbol{I}^c$ backwards up to the root body. Algorithm 2.5 gives a pseudo code version of CRBA. In Algorithm 2.6, we present the derivative of CRBA as *pseudo* code by applying the principle of AD. The correctness of the derivative is captured in Theorem 2.19.

Algorithm 2.6: Composite Rigid-Body Algorithm Forward Sweep.

```
input  : $\boldsymbol{q}$, $\partial\boldsymbol{q}$
output : $\boldsymbol{H}, \partial\boldsymbol{H} = \partial\text{CRBA}(\boldsymbol{q};\partial\boldsymbol{q})$
1  $\boldsymbol{H} = \boldsymbol{0}$                                                     // Nominal
2  $\partial\boldsymbol{H}[:] = \boldsymbol{0}$
3  for $i = 1,\ldots,n_B$ do
4      $\left[\boldsymbol{X}_{\mathrm{J}i}, \boldsymbol{S}_i\right] = \text{jcalc}(\text{jtype}(i), \boldsymbol{q}_i, \dot{\boldsymbol{q}}_i)$                  // Nominal
5      ${}^i\boldsymbol{X}_{\lambda_i} = \boldsymbol{X}_{\mathrm{J}i}\boldsymbol{X}_{\mathrm{T}i}$                    // Nominal, ED reuses quantities
6      $\boldsymbol{I}^c_i = \boldsymbol{I}_i$                                                // Nominal
7      $\partial\boldsymbol{I}^c_i[:] = \boldsymbol{0}$
8  end
9  for $i = n_B,\ldots,1$ do
10     if $\lambda_i \neq 0$ then
11         $\boldsymbol{I}^c = {}^{\lambda_i}\boldsymbol{X}^*_i\boldsymbol{I}^c_i{}^i\boldsymbol{X}_{\lambda_i}$                                  // Nominal
12         $\boldsymbol{I}^c_{\lambda_i} = \boldsymbol{I}^c_{\lambda_i} + \boldsymbol{I}^c$                                     // Nominal
13         $\partial\boldsymbol{I}^c_{\lambda_i}[:] = \partial\boldsymbol{I}^c_{\lambda_i}[:] + [{}^i\boldsymbol{S}_i\partial\boldsymbol{q}[:]\times^*]\boldsymbol{I}^c - \boldsymbol{I}^c[{}^i\boldsymbol{S}_i\partial\boldsymbol{q}[:]\times] + {}^{\lambda_i}\boldsymbol{X}^*_i\partial\boldsymbol{I}^c_i[:]{}^i\boldsymbol{X}_{\lambda_i}$
14     end
15     $\boldsymbol{F} = \boldsymbol{I}^c_i\boldsymbol{S}_i$                                             // Nominal
16     $\partial\boldsymbol{F}[:] = \partial\boldsymbol{I}^c_i[:]\boldsymbol{S}_i$                       // $\partial\boldsymbol{S}_i = \boldsymbol{0}$ for 1-DoF joints
17     $\boldsymbol{H}_{ii} = \boldsymbol{S}^T_i\boldsymbol{F}$                                            // Nominal
18     $\partial\boldsymbol{H}_{ii}[:] = \boldsymbol{S}^T_i\partial\boldsymbol{F}[:]$                      // $\partial\boldsymbol{S}_i = \boldsymbol{0}$ for 1-DoF joints
19     $j = i$
20     while $\lambda_i \neq 0$ do
21         $\boldsymbol{F} = {}^{\lambda_i}\boldsymbol{X}^*_j\boldsymbol{F}$                                       // Nominal
22         $\partial\boldsymbol{F} = {}^{\lambda_i}\boldsymbol{X}^*_j(\partial\boldsymbol{F} - [\times^*\boldsymbol{F}]\boldsymbol{S}_j\partial\boldsymbol{q})$               // Matrix-vector product
23         $j = \lambda_i$                                                 // Nominal
24         $\boldsymbol{H}_{ij} = \boldsymbol{F}^T\boldsymbol{S}_j$                                          // Nominal
25         $\boldsymbol{H}_{ji} = \boldsymbol{H}^T_{ij}$                                             // Nominal
26         $\partial\boldsymbol{H}_{ij}[:] = \partial\boldsymbol{F}^T[:]\boldsymbol{S}_j$                   // $\partial\boldsymbol{S}_i = \boldsymbol{0}$ for 1-DoF joints
27         $\partial\boldsymbol{H}_{ji}[:] = \partial\boldsymbol{H}^T_{ij}[:]$
28     end
29 end
```

Theorem 2.19 (Derivative of CRBA) Let $\boldsymbol{q} \in \mathbb{R}^{n^{\mathrm{B}}}$ define the configuration of a MBS of tree topology and $\partial\boldsymbol{q} \in \mathbb{R}^{n^{\mathrm{B}}\times\mathrm{D}}$ the respective directions. When the joint space inertia matrix $\boldsymbol{H} \in \mathbb{R}^{n^{\mathrm{B}}\times n^{\mathrm{B}}}$ is computed using composite rigid-body algorithm (CRBA) given by Algorithm 2.5, i.e., $\boldsymbol{H} = \text{RNEA}(\boldsymbol{q},\dot{\boldsymbol{q}},\ddot{\boldsymbol{q}})$, then respective directional derivative with respect to above directions, given by

$$\partial\boldsymbol{H} = \partial\text{CRBA}(\boldsymbol{q};\partial\boldsymbol{q}), \tag{2.38}$$

with derivative $\partial\boldsymbol{H} \in \mathbb{R}^{\mathrm{D}\times n^{\mathrm{B}}\times n^{\mathrm{B}}}$, is given by Algorithm 2.6. △

Proof The correctness of above claim follows Proposition 2.14, 2.16 and derivative calculus by propagating $\partial \boldsymbol{q}$ respectively. In this way, the derivative of line 11, 12 is given by line 13 according to Proposition 2.16. Line 22 is then a direct application of Proposition 2.14. The rest follows from derivative calculus. □

Remark Due to the fact that the derivative of the joint space inertia matrix itself becomes a tensor, even for ED we require to perform implicit loops in order to compute the respective derivative. However, even though the implementation looks like the naive application of AD principle, the results still show superiority of ED.

2.3.3 Derivatives of Contact Dynamics

Algorithm 2.7: Solution of Descriptor Form Derivative.

```
input  : $\boldsymbol{q}$, $\partial\boldsymbol{q}$, $\dot{\boldsymbol{q}}$, $\partial\dot{\boldsymbol{q}}$, $\boldsymbol{\tau}$, $\partial\boldsymbol{\tau}$, I
output: $\ddot{\boldsymbol{q}}$, $\partial\ddot{\boldsymbol{q}}$, $\boldsymbol{\lambda}$, $\partial\boldsymbol{\lambda} = \partial\mathrm{FD}(\boldsymbol{q},\partial\boldsymbol{q},\dot{\boldsymbol{q}},\partial\dot{\boldsymbol{q}},\boldsymbol{\tau},\partial\boldsymbol{\tau},\mathrm{I})$
1 $\boldsymbol{H},\partial\boldsymbol{H} = \partial\mathrm{CRBA}(\boldsymbol{q},\partial\boldsymbol{q})$
2 $\boldsymbol{c},\partial\boldsymbol{c} = \partial\mathrm{RNEA}(\boldsymbol{q},\partial\boldsymbol{q},\dot{\boldsymbol{q}},\partial\dot{\boldsymbol{q}},\boldsymbol{0},\boldsymbol{0})$
3 $\boldsymbol{K} = \begin{bmatrix}\boldsymbol{H} & \boldsymbol{J}^T\\ \boldsymbol{J} & 0\end{bmatrix}, \partial\boldsymbol{K} = \begin{bmatrix}\partial\boldsymbol{H} & \partial\boldsymbol{J}^T\\ \partial\boldsymbol{J} & 0\end{bmatrix},$
4 $\boldsymbol{b} = \begin{bmatrix}\boldsymbol{S}^T\boldsymbol{\lambda}-\boldsymbol{c}\\ -\boldsymbol{\gamma}\end{bmatrix}, \partial\boldsymbol{b} = \begin{bmatrix}\boldsymbol{S}^T\partial\boldsymbol{\lambda}-\partial\boldsymbol{c}\\ -\partial\boldsymbol{\gamma}\end{bmatrix},$
5 $\boldsymbol{x} = (\ddot{\boldsymbol{q}}^T,\boldsymbol{\lambda}^T)^T = \mathrm{solve}(\boldsymbol{K},\boldsymbol{b}),$
6 $(\partial\ddot{\boldsymbol{q}}^T,\partial\boldsymbol{\lambda}^T)^T = \mathrm{solve}(\boldsymbol{K},\partial\boldsymbol{b}-\partial\boldsymbol{K}\boldsymbol{x}),$
```

In the following section, we derive the derivative of the solution of linear system with respect to its inputs. Recapping equations (2.35) and (2.36), these insights become important for the computation of constrained forward dynamics as the required dynamic quantities are computed by solving the respective linear system. In this way, the evaluation of the derivative of the solution of linear system can be accelerated considerably. This will be demonstrated computationally in Section 2.4.2.

Definition (Solution of a Linear System) Consider $\boldsymbol{A} \in \mathbb{R}^{N\times N}$, $\boldsymbol{B} \in \mathbb{R}^{N\times M}$ and $\boldsymbol{X} \in \mathbb{R}^{N\times M}$ that solves the linear equation $\boldsymbol{A}\,\boldsymbol{X} = \boldsymbol{B}$, where we denote the solution operator $\boldsymbol{X} = \boldsymbol{A}^{-1}\,\boldsymbol{B}$ by

$$\boldsymbol{X} = \mathrm{solve}(\boldsymbol{A},\boldsymbol{B}). \tag{2.39}$$

△

Proposition (Derivative of a Linear System) The derivative of the solution of the setting of Def. 2.21 with respective directions ∂A, ∂X and ∂B is $0 = \partial\boldsymbol{A}\boldsymbol{X} + \boldsymbol{A}\partial\boldsymbol{X} - \partial\boldsymbol{B}$. Recapping (2.39), the derivative is given by

$$\partial\boldsymbol{X} = \mathrm{solve}(\boldsymbol{A},\partial\boldsymbol{B}-\partial\boldsymbol{A}\boldsymbol{X}). \tag{2.40}$$

△

Proof *Taylor*-expansion of each input quantity up to order one and coefficient-wise comparison yield the claim. □

Remark (2.40) is sometimes stated as $\partial\boldsymbol{X} = \boldsymbol{A}^{-1}(\partial\boldsymbol{B} - \partial\boldsymbol{A}\boldsymbol{A}^{-1}\boldsymbol{B})$, which may be misleading and often yields inefficient implementations.

While the dynamic quantities in the form of joint space inertia matrix $\boldsymbol{H}$ and nonlinear effect $\boldsymbol{c}$ were already described in previous sections, the kinematic quantities in equations (2.35) and (2.36) will be briefly explained in the following paragraph. The rows of contact Jacobian $\boldsymbol{J}$ for contact frame A of body j is computed by transforming the joint motion space matrices to the coordinate frame A by

$$^{A}\hat{\boldsymbol{J}}(\boldsymbol{q}) = {}^{A}\boldsymbol{X}_0 \sum_{i \in \kappa(j)} {}^{0}\boldsymbol{X}_i(\boldsymbol{q}_i)\boldsymbol{S}_i. \tag{2.41}$$

The contact Hessian $\boldsymbol{\gamma}(\boldsymbol{q},\dot{\boldsymbol{q}})$, as the time derivative of contact Jacobian $\boldsymbol{J}$ follows (2.41) in its computation. The computation of the respective derivatives follows again from Proposition 2.14 and derivative calculus.

For the solution of the descriptor form (2.35) to compute the resulting generalized joint accelerations $\ddot{\boldsymbol{q}}$ and contact forces $\boldsymbol{\lambda}$, common linear algebra techniques for the solution of saddle-point systems [10] or special tailored structure-exploiting methods for MBS [48] can be employed. The solution methods readily at hand can be separated into three approaches, *direct*, *range-space* and *null-space*. They are subject to different trade-offs, where especially the model topology as well as number of constraints and constrained bodies can impact the performance and the choice of matrix decomposition techniques affects the numerical conditions. Therefore the performance and the numerical stability may vary depending on these factors. We refer again to [10] for common linear algebra techniques for the solution of saddle-point systems or to [48] for structure-exploiting methods of linear algebra tailored for MBS respectively.

Consider, for example, *direct* to solve (2.35). It solves the full *Karush-Kuhn-Tucker* system by means of a suitable decomposition of the composed left hand side and simultaneously computes $\ddot{\boldsymbol{q}}$ and $\boldsymbol{\lambda}$. While this may be slow for large systems, as it does not exploit the structure from the lower right block of zeros, we will only present this approach as the derivative can be derived analogously for the other approaches. Proposition 2.22 allows to reuse the matrix decomposition in the operation *solve* from Definition 2.21, see again Algorithm 2.7. Following this, we apply the principle of AD in order to define Algorithm 2.7. The correctness of the derivative is captured in Theorem 2.24.

Theorem 2.24 (Derivative of the Solution of Constrained Dynamics) Let $\boldsymbol{q},\dot{\boldsymbol{q}} \in \mathbb{R}^{n^{\mathrm{B}}}$ define the configuration and velocity of a MBS of tree topology, while $\partial\boldsymbol{q}, \partial\dot{\boldsymbol{q}}\mathbb{R}^{n^{\mathrm{B}}\times \mathrm{D}}$ are the respective directions. Furthermore, let $\boldsymbol{\tau} \in \mathbb{R}^{n^{\mathrm{B}}}$ be the currently applied generalized joint forces with respective directions $\partial\boldsymbol{\tau} \in \mathbb{R}^{n^{\mathrm{B}}\times \mathrm{D}}$, and let the index set $\mathrm{I} \subset \mathbb{N}$ define the currently active kinematic constraints of the MBS. The forward dynamics of a constrained system are computed by the solution of the linear system $\boldsymbol{K}\boldsymbol{x} = \boldsymbol{b}$ where the solution is given by the solution operator (2.39), i.e., $\boldsymbol{x} = \operatorname{solve} K, b$. We define $\boldsymbol{x} = (\ddot{\boldsymbol{q}}^T, -\boldsymbol{\lambda}^T)^T$ that collects joint accelerations $\ddot{\boldsymbol{q}} \in \mathbb{R}^{n^{\mathrm{B}}}$ and contact forces $\boldsymbol{\lambda} \in \mathbb{R}^{n^{\mathrm{G}}}$ the matrix $\boldsymbol{K} \in \mathbb{R}^{(n^{\mathrm{B}}+n^{\mathrm{G}})\times(n^{\mathrm{B}}+n^{\mathrm{G}})}$ representing the left-hand side of the equation of motion (2.35), and vector $\boldsymbol{b} \in \mathbb{R}^{n^{\mathrm{B}}+n^{\mathrm{G}}}$ representing the right-hand side of equation (2.35), i.e.,

$$\boldsymbol{K} = \begin{bmatrix} \boldsymbol{H}(\boldsymbol{q}) & \boldsymbol{J}^T(\boldsymbol{q}) \\ \boldsymbol{J}(\boldsymbol{q}) & 0 \end{bmatrix}, \qquad \boldsymbol{b} = \begin{bmatrix} \boldsymbol{S}^T\boldsymbol{\tau} - \boldsymbol{c}(\boldsymbol{q},\dot{\boldsymbol{q}}) \\ -\boldsymbol{\gamma}(\boldsymbol{q},\dot{\boldsymbol{q}}) \end{bmatrix} \tag{2.42}$$

with joint space inertia matrix $\boldsymbol{H} = \mathrm{CRBA}(\boldsymbol{q})$ evaluated via CRBA and contact Jacobian $\boldsymbol{J}(\boldsymbol{q})$, selection matrix $\boldsymbol{S} \in \mathbb{R}^{n^{\mathrm{B}}\times n^{\mathrm{Act}}}$, the nonlinear effects term $\boldsymbol{c} = \mathrm{RNEA}(\boldsymbol{q},\dot{\boldsymbol{q}},\boldsymbol{0})$ evaluated via RNEA and contact Hessian $\boldsymbol{\gamma}(\boldsymbol{q},\dot{\boldsymbol{q}})$. The directional derivative of the constrained forward dynamics with respect to the respective directions is then given by Algorithm 2.7. $\triangle$

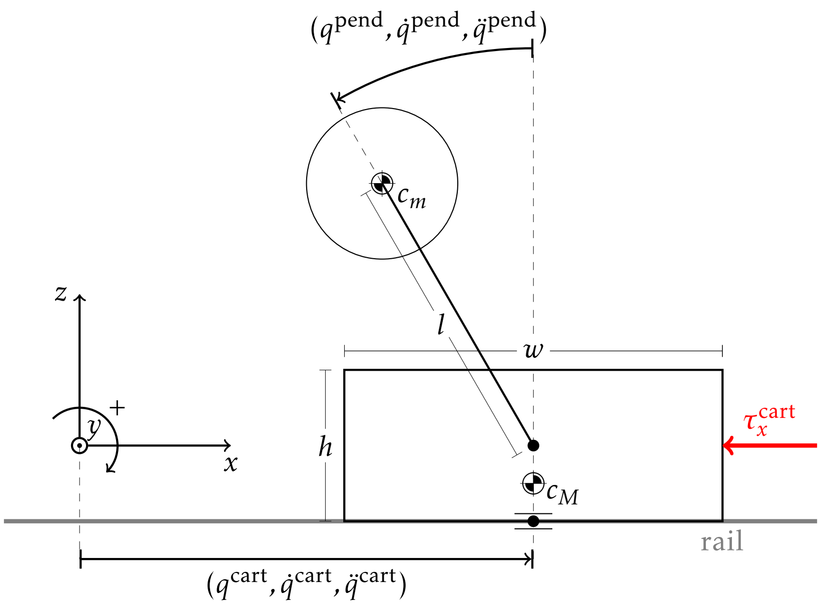

$(q^{\text{pend}}, \dot{q}^{\text{pend}}, \ddot{q}^{\text{pend}})$
c_m
z
l
w
$+$
y
x
h
τ_x^{cart}
c_M
rail
$(q^{\text{cart}}, \dot{q}^{\text{cart}}, \ddot{q}^{\text{cart}})$

Figure 2.2: An inverted pendulum on a movable cart.

Proof The correctness of the directional derivative follows from Theorem 2.19 that verifies line 1 of Algorithm 2.7 and Theorem 2.17 verifies line 2. The rest of the argumentation is then given by Proposition 2.21 for the solution of the linear system in lines 3-6. This yields the result then. □

2.4 Results

In the following section, we present the results obtained from implementing and benchmarking the derivative evaluation of recursive algorithms for RBD based on FD, AD and ED. First, we tested the quality of the resulting derivative information against analytically derived derivatives of the RBD of two cart-pendulum models, one in minimal coordinates and one defined via a bilateral kinematic constraint. Second, we benchmarked the computational times required to evaluate derivatives using AD and FD on a set of different MBS. Finally, we show results from the application in an optimal control setting solving a bending and lifting task of a bio-mechanical model and compare solution performance using FD and AD.

2.4.1 Evaluation of Derivative Quality

In order to quantify the differences between derivative information computed via FD, AD, and ED against analytic derivatives, we consider a cart-pendulum benchmark model in two versions: one with kinematic constraints and one without kinematic constraints in minimal coordinates. It is depicted in Figure 2.2.

The contact model is defined as a free-floating box with three DoFs, x, y position and orientation around y-axis, and one DoF for the rotational motion of the pendulum relative to the cart, where the kinematic constraint restricts the motion of the root box in z direction as well as prevents a rotation around the y axis. The contact can be resolved directly into a minimal coordinate formulation with two DoFs representing then our contact-free model.

We evaluated 10 000 uniform random samples of the inputs $\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}, \boldsymbol{\tau} \in [-\pi, \pi]^{n^{\text{DoF}}}$ and the respective directions $\boldsymbol{D} \in \mathbb{R}^{n^{\text{DoF}} \times 3n^{\text{DoF}}}$ for both models and each algorithm, i.e., CRBA,

Table 2.3: Derivative quality of numerical (FD ∂_{+h}, FDC $\partial_{\pm h}$) and automated (AD, ED) against analytic derivatives (AN).

	Cart-Pendulum in Minimal Coordinates, Error against AN							
	CRBA		RNEA		NEFFECTS		ABA	
	min	max	min	max	min	max	min	max
FD	$3.03 \cdot 10^{-9}$	$2.75 \cdot 10^{-8}$	$8.81 \cdot 10^{-8}$	$1.15 \cdot 10^{-6}$	$5.62 \cdot 10^{-8}$	$3.42 \cdot 10^{-7}$	$1.33 \cdot 10^{-7}$	$1.49 \cdot 10^{-6}$
FDC	$1.15 \cdot 10^{-12}$	$9.71 \cdot 10^{-11}$	$2.16 \cdot 10^{-10}$	$1.42 \cdot 10^{-9}$	$8.76 \cdot 10^{-11}$	$1.19 \cdot 10^{-9}$	$3.29 \cdot 10^{-10}$	$6.56 \cdot 10^{-9}$
ED	$2.22 \cdot 10^{-16}$	$2.22 \cdot 10^{-16}$	$1.78 \cdot 10^{-15}$	$1.95 \cdot 10^{-14}$	$6.66 \cdot 10^{-16}$	$1.15 \cdot 10^{-14}$	$1.78 \cdot 10^{-15}$	$4.26 \cdot 10^{-14}$
AD	$2.22 \cdot 10^{-16}$	$2.22 \cdot 10^{-16}$	$1.78 \cdot 10^{-15}$	$1.33 \cdot 10^{-14}$	$5.00 \cdot 10^{-16}$	$8.88 \cdot 10^{-15}$	$1.78 \cdot 10^{-15}$	$4.26 \cdot 10^{-14}$

	Cart-Pendulum with Contacts, Error against AN									
	CRBA		RNEA		NEFFECTS		ABA		FDCONTACTS	
	min	max	min	max	min	max	min	max	min	max
FD	$2.74 \cdot 10^{-8}$	$1.04 \cdot 10^{-7}$	$1.54 \cdot 10^{-6}$	$7.78 \cdot 10^{-6}$	$1.62 \cdot 10^{-6}$	$8.04 \cdot 10^{-6}$	$3.99 \cdot 10^{-7}$	$9.11 \cdot 10^{-6}$	$1.19 \cdot 10^{-6}$	$6.57 \cdot 10^{-6}$
FDC	$7.10 \cdot 10^{-11}$	$7.21 \cdot 10^{-10}$	$2.37 \cdot 10^{-9}$	$1.54 \cdot 10^{-8}$	$2.25 \cdot 10^{-9}$	$1.61 \cdot 10^{-8}$	$4.87 \cdot 10^{-10}$	$1.01 \cdot 10^{-8}$	$1.41 \cdot 10^{-9}$	$7.94 \cdot 10^{-8}$
ED	$2.22 \cdot 10^{-16}$	$1.22 \cdot 10^{-15}$	$3.11 \cdot 10^{-14}$	$2.17 \cdot 10^{-13}$	$3.73 \cdot 10^{-14}$	$1.79 \cdot 10^{-13}$	$1.42 \cdot 10^{-14}$	$4.98 \cdot 10^{-13}$	$7.33 \cdot 10^{-15}$	$1.71 \cdot 10^{-13}$
AD	$2.22 \cdot 10^{-16}$	$1.33 \cdot 10^{-15}$	$2.13 \cdot 10^{-14}$	$1.58 \cdot 10^{-13}$	$2.13 \cdot 10^{-14}$	$1.72 \cdot 10^{-13}$	$1.42 \cdot 10^{-14}$	$4.98 \cdot 10^{-13}$	$8.88 \cdot 10^{-15}$	$1.28 \cdot 10^{-13}$

RNEA, NEFFECTS, ABA and FDCONTACTS. The latter is not run for the minimal coordinate model. In the case of kinematic constraints, one has to avoid singular contact Jacobians $\boldsymbol{J}$, which would induce huge errors in the nominal as well as the derivative information. Thus, we limit the rotation of the cart-pendulum model around its y-axis to the interval $[-1, 1]$.

The dynamics were evaluated in double precision. Therefore, the expected accuracies are $\sqrt{\epsilon} \approx 1.054 \cdot 10^{-8}$ for FD and $\sqrt[3]{\epsilon^2} \approx 2.310 \cdot 10^{-11}$ for FDC, see Tab. 2.1. In contrast, we expect the AD and ED derivatives to be around ϵ.

Figures 2.3a shows the resulting errors for the cart-pendulum model in minimal coordinates, while Figure 2.3b shows those for the cart-pendulum model with kinematic constraints.

The approximation errors cluster for all algorithms. For both models, CRBA showed the best accuracy, while ABA showed the worst. Both AD and ED show an accuracy of $\approx 1 \cdot 10^{-14}$. FD and FDC behave similar: mean accuracies of $\approx 5 \cdot 10^{-6}$ and $\approx 1 \cdot 10^{-8}$ are achieved in the worst case for the contact model. We note that error propagation can easily account for a loss of 2-3 orders of magnitude in precision, c.f. similar observations in [180], and occurs for all modes of derivative evaluation. The results confirm that AD improves the derivative accuracy substantially, in our setup by up to 5 orders of magnitude.

2.4.2 Benchmarking the Derivative Evaluation Runtime Performance

We have evaluated the runtimes of AD, ED as well as FD, FDC using a 10 DoF multi-pendulum model, similar to the one from [50], for an increasing number of propagated directions n for the algorithms FDCONTACTS, NEFFECTS, CRBA and the solution of (2.35). FDC evaluates slowest, in line with the expectation $1 + 2n$. FD performs a little worse than the expectation of $1 + n$. The AD implementation performs similar to FD for FDCONTACTS, and between FD and FDC for NEFFECTS and CRBA. In the latter case, it exceeds the theoretical expectation of $1 + 1.5n$. For these three algorithms, the structure exploitation implemented in ED pays off and ED outperforms AD and FD. The biggest benefit of AD and ED over a black-box FD evaluation is obtained in the solution of (2.35). The reuse of the factorization significantly decreases the runtime. AD and ED take less than a quarter of the time of FD. The results are shown in Figure 2.4.

2.4.3 Application to Optimal Control of a Bio-mechanical Model

We have used our AD implementation to optimally control a two-phase bending and lifting motion for an 8 DoF human model in the sagittal plane, see [115], and with varying weights attached to the hand during lifting. We followed a direct and all-at-once approach and solved the resulting NLP using an SQP method. Figure 2.5 (a) – (c) show its convergence from the same initialization for different masses to lift in the second phase. The SQP method consistently required fewer iterations using AD compared to FD in all cases (up to 10 % for (b)). Entry into the rapid full step local convergence phase happened earlier in all cases, yielding a faster descent.

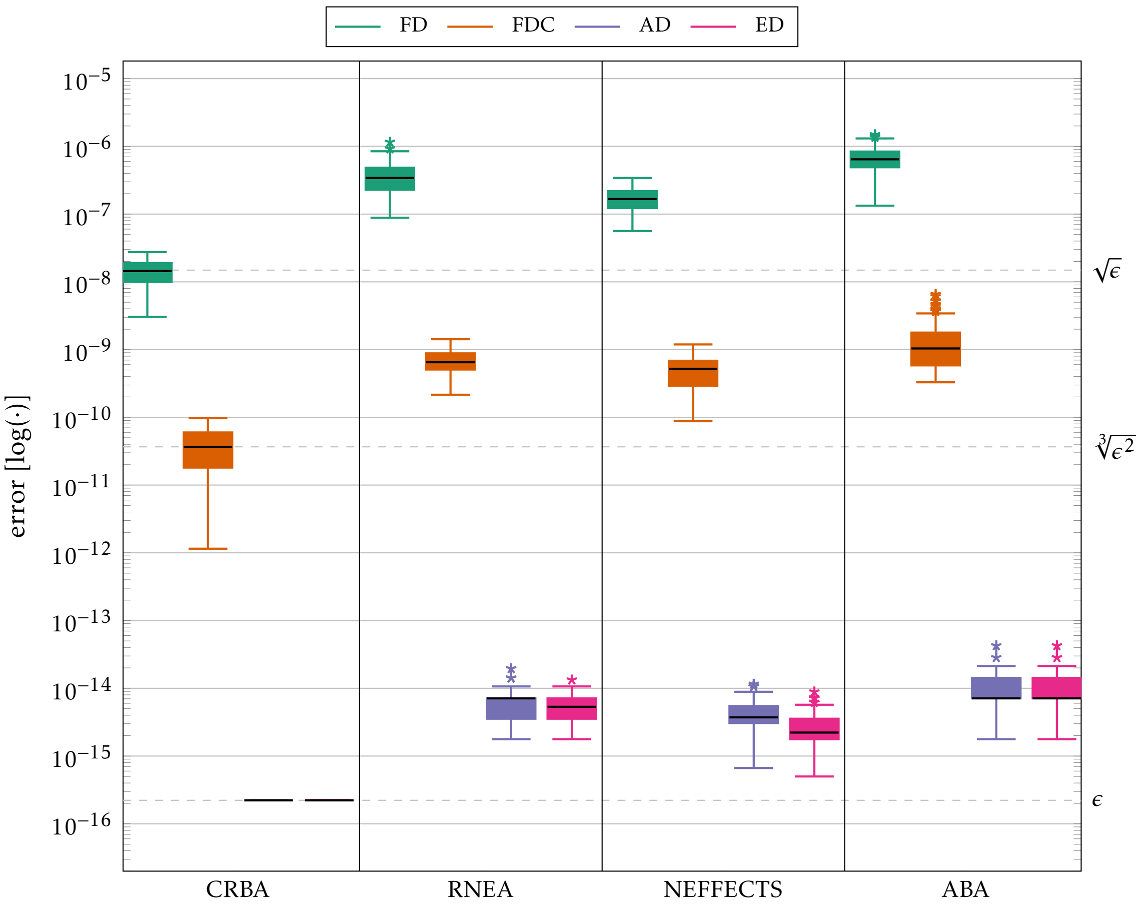
2.5 Summary

In this Chapter, we proposed a new AD approach to evaluate derivative information of recursive algorithms for rigid-body dynamics (RBD). The proposed approach was implemented and we demonstrated its applicability for optimal control by numerical examples. Furthermore, we presented the theoretical expectations of the efficiency from the literature and benchmarked our performance against common numerical differentiation (FD) approximations of the derivatives.

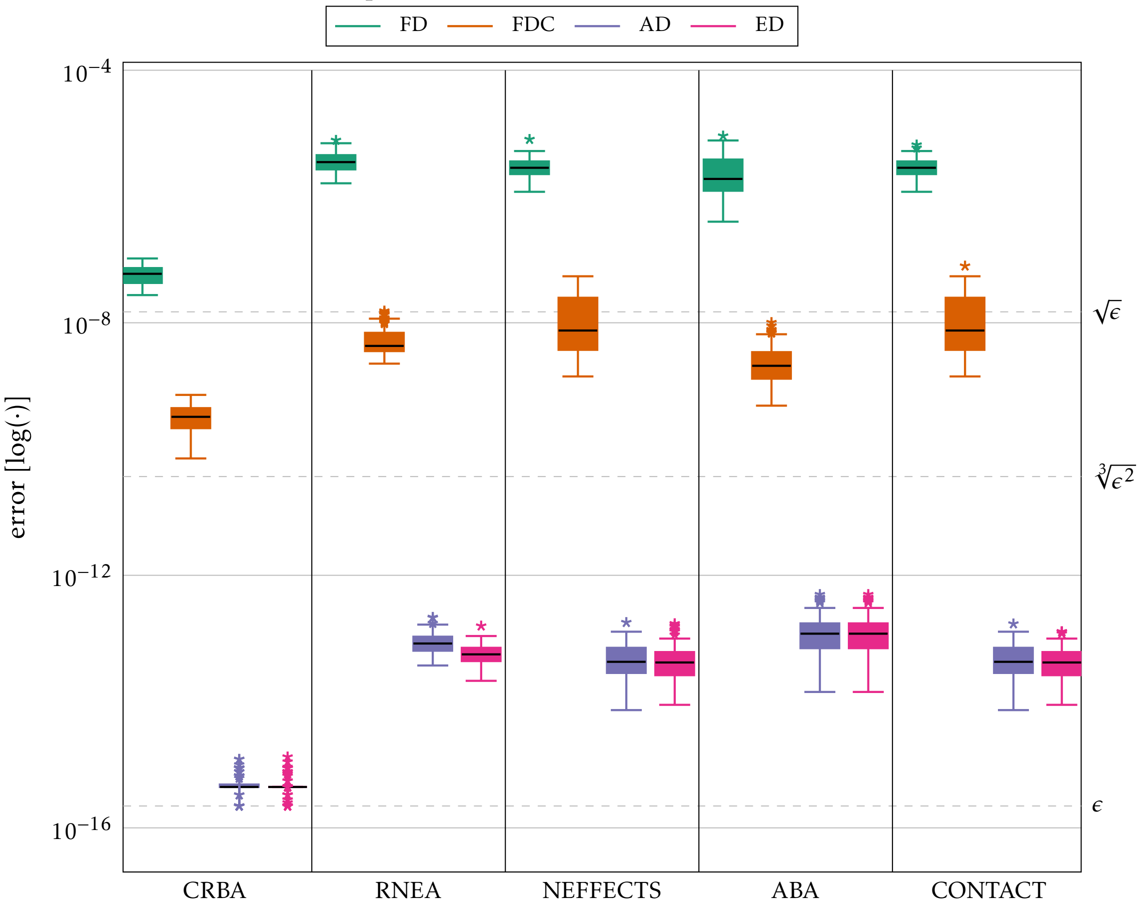
Each of the investigated derivative evaluation modes followed the theoretical expectations when tested against analytic derivatives concerning the quality of the derivative information. Following the theoretical boundaries, the implementations algorithmic differentiation (AD) and efficient algorithmic differentiation (ED) both exhibited an expected performance up to machine precision.

Benchmarking the speed of the respective derivative evaluations, we showed that the naive implementation of AD, which is based on mechanical application of the chain rule, performs between FD and finite central differences in most of the considered cases and then slightly better than the theoretical expectation. Furthermore, we observe that ED, which utilizes the structure of the RBD algorithms to avoid unnecessary computations, outperforms the FD computation in terms of speed in all of the considered cases. We highlight that employing Proposition 2.22 in the linear system solution speeds up AD and ED considerably.

In contrast to the presented literature that focuses on recursive *Newton-Euler* algorithm (RNEA), we augmented the algorithms composite rigid-body algorithm (CRBA) as well as Articulated Body Algorithm (ABA). The latter being considered as *complex* in [23] due to the occurring tensor-valued quantities. Additionally, none of the existing publications treats the contact case in multi-body system (MBS) that represents the core finding of this Chapter.



(a) Cart-pendulum model in minimal coordinates.



(b) Cart-pendulum model with constraints.

Figure 2.3: Errors against analytic derivatives of FD, FDC, AD and ED for different algorithms and both cart-pendulum models.

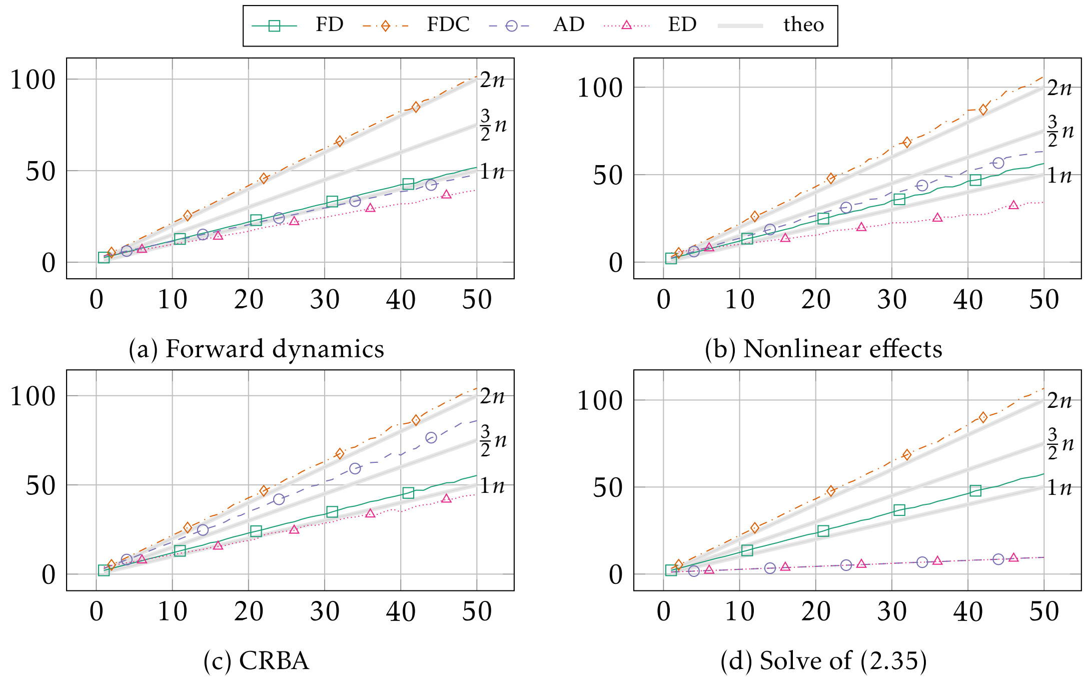
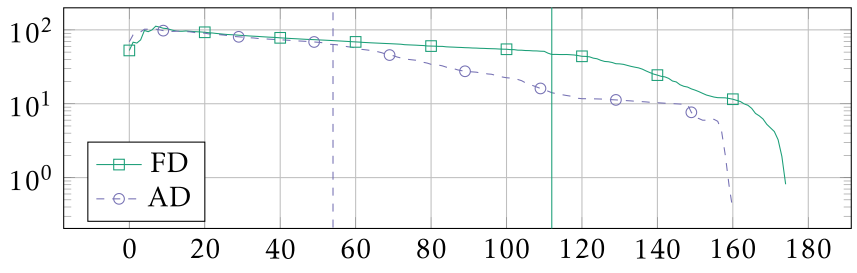
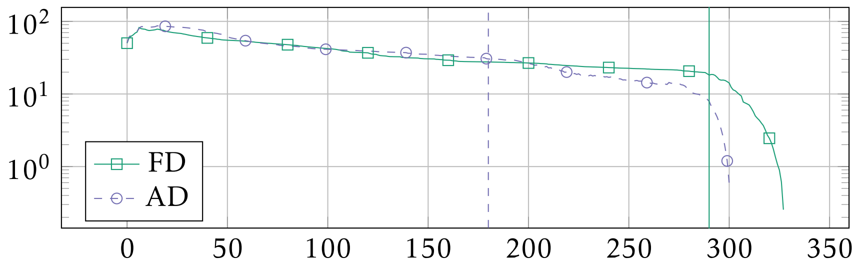


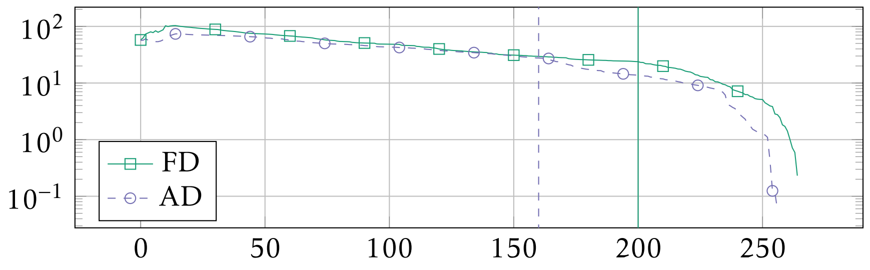
Figure 2.4: Effort $[n \times \mathrm{eval}(\boldsymbol{f}(\boldsymbol{x}))]$ against number of propagated directions $[n]$ of FD, FDC, AD and ED for a multi-pendulum model.



(a) 3 kg



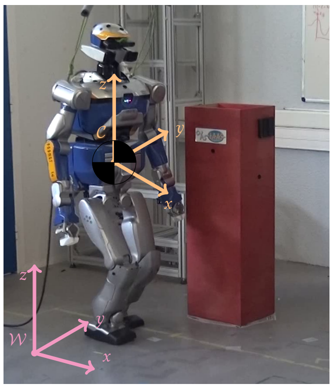
(b) 6 kg



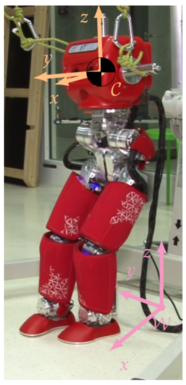
(c) 9 kg

Figure 2.5: Self-convergence plots of SQP performance using AD and FD for different masses for the biomechanical OCP. The vertical axis depicts distance of current iterate $\boldsymbol{x}^n$ to final iterate $\boldsymbol{x}^\star$ [$\|\boldsymbol{x}^n - \boldsymbol{x}^\star\|_2$] versus iteration counter [n]. Vertical lines indicate that the algorithms enter the full step local convergence phase.

3 Motion Generation Based on Center of Mass Dynamics



(a) *HRP-2* avoiding obstacles in the scene.



(b) *HeiCub* walking with closed-loop control.

Figure 3.1: The algorithms derived within this thesis were applied to improve the walking capabilities of the humanoid robots *HRP-2* as well as *HeiCub*.

The following chapter presents methods and algorithms for motion generation using simplified models based solely on the center of mass (CoM) and the zero-moment point (ZMP) motion, which were developed during the work on this thesis. They are applied to improve the walking capabilities of the two humanoid robots *HRP-2* of *Laboratory for Analysis and Architecture of Systems* (CNRS-LAAS) as well as *HeiCub* of *Heidelberg University* (UHEI).

The contents of the following chapter are based on our journal article [133] and our conference article [163] as well as unpublished work on an extension concerning the combination of motion planning and generation based on a mixed-integer formulation.

The goal of motion generation for humanoids is to realize *human-like* task performance on the robot while considering the real-time constraints dictated by the robotic platform. Therefore, state-of-the-art methods achieving real-time motion generation on today's robotic hardware apply a reduced view on the whole-body dynamics of a humanoid by only considering the motion of the CoM and of the ZMP or center of pressure (CoP)[1]. The use of a reduced model is a practical solution on platforms with limited computational capabilities for example *HRP-2*. However, the drawback of employing models reduced to the CoM and ZMP motion is that effects related to the inertia of the whole-body motion are ignored or simplified. This leads to a systematic error hindering the execution of the computed motion on the robot. However, this mismatch can be drastically reduced by applying a dynamic filter (DF). Another important topic is collision avoidance during walking. While stable walking on clear level ground can already be achieved by robots

[1] In the case of a dynamic stable motion, the computed point quantities of both concepts, zero-moment point (ZMP) as well as center of pressure (CoP), are identical and coincide, cf. [177]. Here, we prefer to use ZMP over CoP as this is a more common term in motion generation, while CoP is more common in motion analysis.

nowadays, including obstacle avoidance into the motion generation formulation can drastically improve the range of movement tasks a robot can undertake.

Two walking pattern generators (WPGs) were developed during this thesis to include them into the motion generation framework. The approaches derived in the remainder of this chapter were all tested on the robotic hardware platforms *HRP-2* and *HeiCub* and were proven to run in real-time.

The following contributions were made in this thesis:

- We propose a combined but nonlinear reformulation of a two-step walking WPG.
- Our formulation is able to find simultaneously foot-step positions and orientations.
- The formulation can directly introduce nonlinear constraints, e.g. used for obstacle avoidance.
- The nonlinear problem is solved using nonlinear model predictive control (NMPC) real-time iterations for fast feedback computation.
- The model-plant mismatch resulting from model reduction is compensated by a DF.
- The whole algorithm runs in real time on the embedded hardware of the humanoid robot *HRP-2*.
- The algorithm runs in real time on the external computing hardware of the humanoid robot *HeiCub*.
- We propose a mixed-integer formulation realizing a combination of motion planning and generation.
- This allows to include any obstacles, which projection on the ground can be approximated by convex polygons, as well as step stones, i.e., restriction of the foot placement by regions described by convex polygons.
- The mixed-integer formulation was tested in simulation for a *HeiCub* model.

In Figure 3.2, we visualize the general work flow for motion generation using WPGs based on CoM dynamics. The core of the approach is the WPG that computes CoM, feet and ZMP trajectories based on a reduced model in the form of the linear inverted pendulum model (LIPM) as it will be derived in Section 3.1. The inputs from the control perspective are user inputs for example a reference linear and angular velocity at CoM level for tracking. Additionally, sensory information can be included in order to integrate collision avoidance for example as presented in Section 3.2.3 and 3.3.2. For proper setup, the WPG may include additional robot specific details and/or a kinematic model of the robot to be able to formulate constraints guaranteeing dynamic balance and proper foot step placement.

In the first step, the WPG computes discrete CoM jerk values $\dddot{\boldsymbol{c}}$ and foot steps $\boldsymbol{f}$ (, if not prescribed,) on a finite time horizon from the given user inputs by solving a linear-quadratic regulator (LQR) problem as [84], a convex optimization problem in the form of a quadratic program (QP) as [71] or even a nonlinear optimization problem in the form of a nonlinear program (NLP) as it will be presented in Section 3.2 or a mixed-integer optimization problem as in [78] as well as the work presented in Section 3.3. The application of an *Euler* integration scheme enables then to compute CoM trajectories $\boldsymbol{c}$ from the CoM jerks and interpolation of the foot steps $\boldsymbol{f}$ from the foot step planning on the horizon allows to define 3D trajectories by means of fifth-order polynomials.

The resulting CoM trajectories computed have to *filtered* to retrieve $\tilde{\boldsymbol{c}}$ using a DF as presented in Section 3.4.1 in order to compensate the error introduced from using the LIPM as dynamic model instead of the whole-body dynamics of the humanoid. The *filtered* CoM trajectories altogether with the foot trajectories are then input to a (generalized) inverse kinematics algorithm. In this way, reference joint trajectories $\boldsymbol{q}, \dot{\boldsymbol{q}}$ are computed

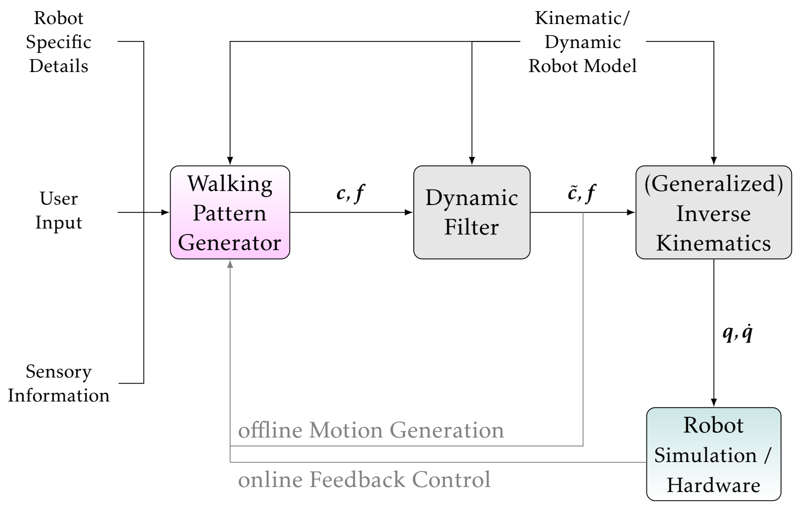

Robot Specific Details
Kinematic/ Dynamic Robot Model
User Input
Walking Pattern Generator
c, f
Dynamic Filter
$\tilde{c}, f$
(Generalized) Inverse Kinematics
$q, \dot{q}$
Sensory Information
offline Motion Generation
online Feedback Control
Robot Simulation / Hardware

Figure 3.2: Visualization of the general control scheme of the walking pattern generator (WPG) framework derived in this thesis. From robot specific details, a kinematic description of the robot and input from the user a WPG computes center of mass (CoM) and feet trajectories $\boldsymbol{c}$, $\boldsymbol{f}$ represented by polynomials. The CoM trajectories have to be adapted by a so-called *dynamic filter* in order to compensate the error introduced by the reduced model used inside the WPG. From these trajectories an inverse kinematics framework computes joint angles to be realized by the low-level control on the robot. For offline motion generation the filtered CoM trajectories are fed back to the WPG and for online motion generation the respective sensory information of the robot is used to close the loop.

that can be applied directly to the low-level joint control of the robot in order to realize a dynamically feasible walking trajectory.

In open-loop control, the WPG is then reinitialized with the current reference velocity input and the corrected initial states retrieved from the dynamic filter $\tilde{\boldsymbol{c}}$ as done in [71, 133]. In closed-loop control, an estimate of the actual CoM position is retrieved from the robot and fed back to the WPG as presented in [149, 163].

In the following sections we will explain all relevant building blocks of the scheme depicted in Figure 3.2. First, the reduced model is derived in detail in Section 3.1. Second, the WPGs derived in this thesis are then explained in more detail in separate sections 3.2 and 3.3. This is followed by an explanation of the remaining building blocks in Section 3.4. Finally, we show the respective results of the different approaches followed by the conclusion of this chapter in the form of a summary.

3.1 Derivation of the Reduced Dynamic Model

In this Section, a reduced model of the whole-body dynamics of a humanoid is derived as first presented by *Kajita* in [83]. The idea of limiting the dynamic model to the motion of the CoM and the ZMP is motivated by the schematic drawing of Figure 3.3, which shows the pendulum like movement of the CoM with respect to the ZMP as well as the limitation of the evolution of the CoM inside a plane parallel to the ground. The application of the model requires the following assumptions to be made.

Assumption (linear inverted pendulum model Assumptions) We assume in order to approximate the whole-body dynamics of a multi-body system by means of the linear inverted pendulum model (LIPM):

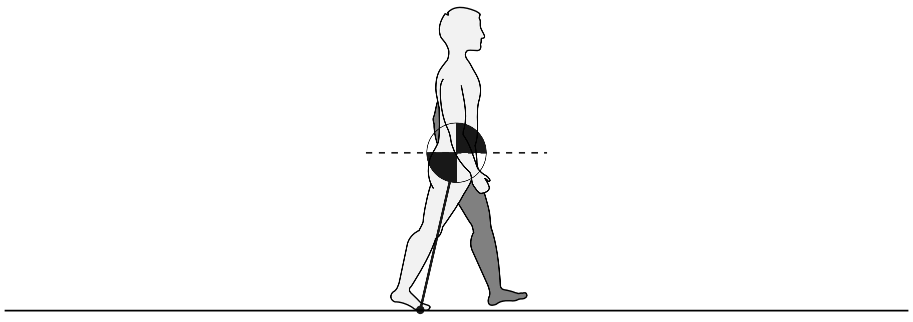


Figure 3.3: Visualization of pendulum model derived from human motion during gait. The motion of the center of mass (CoM) is very characteristic for a humanoid and has attracted much attention in motion analysis. For a human, the CoM motion in the sagittal plane can roughly be described by that of cycloid due to the pendulum-like motion during swing phase where the zero-moment point (ZMP) or center of pressure acts as its pivot on the ground. The linear inverted pendulum model limits the evolution of the CoM motion to a plane in parallel to the ground and in addition links CoM and ZMP by a friction-less prismatic joint.

- The angular momentum due to the motion of all robot parts is supposed to be zero.
- The CoM of the humanoid evolves on a horizontal plane parallel to the ground.
- The normal vectors of each contact point/surface have to be collinear.
- The sum of all contact forces can be re-expressed as a single unique force at the ZMP. $\triangle$

As a consequence, it is possible to restrict the number of all degrees of freedom (DoFs) of the humanoid to three DoFs, which are the projection of the robot CoM (x, y)-position on the ground plane and its root frame orientation θ around the vertical z-axis[2]. The justification of this is presented in detail by *Wieber* in [181]. We briefly summarize the derivation in the remainder of this section. However, we follow the notional convention for the MBS dynamics introduced in Section 1 for convenience.

The equation of motion for a MBS in contact with its environment can be found in Chapter 1. Revisiting the equation, the part defining the motion of the robot incorporating also external forces is given by

$$\boldsymbol{H}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{c}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \boldsymbol{S}^T \boldsymbol{\tau} + \boldsymbol{J}^T \boldsymbol{\lambda}, \tag{3.1}$$

where $\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}} : \mathbb{R} \to \mathbb{R}^{n^{\text{DoF}}}$ denote the generalized positions, velocities, accelerations, $\boldsymbol{\tau} : \mathbb{R} \to \mathbb{R}^{n^{\text{Act}}}$ denotes the generalized forces of the MBS while the selection matrix $\boldsymbol{S} \in \mathbb{R}^{n^{\text{DoF}} \times n^{\text{Act}}}$ maps the generalized forces on the actuated DoFs. The external force $\boldsymbol{\lambda} \in \mathbb{R}^{n^{\text{G}}}$ acts on the system by means of the kinematic Jacobian $\boldsymbol{J} := \partial_{\boldsymbol{q}} \boldsymbol{g}(\boldsymbol{q})$ of a kinematic constraint $\boldsymbol{g}(\boldsymbol{q}) = 0$. The joint space inertia or mass matrix is denoted by $\boldsymbol{H} \in \mathbb{R}^{n^{\text{DoF}} \times n^{\text{DoF}}}$ and the nonlinear effects are given by $\boldsymbol{c} \in \mathbb{R}^{n^{\text{DoF}}}$. We refer the reader to Chapter 1 or [48] for the details.

Under the assumption that the dynamic model of humanoid always considers the root body, e.g. the pelvis or trunk, to be modeled as a free-flyer or floating base, we assume a coordinate frame, i.e., a right-handed Cartesian coordinate system, explicitly describing

[2] Here, we use the coordinate convention such that the Cartesian frame attached to the root body of the robot (pelvis for *HRP-2*, upper torso *HeiCub*) points its x-axis towards the front of the robot and the z-axis upwards, c.f. Figure 3.1a and 3.1b.

the position and orientation of the root body to the world coordinate system in terms of non-joint related DoFs, i.e.,

$$q = \begin{bmatrix} x_0 \\ \theta_0 \\ \hat{q} \end{bmatrix}, \tag{3.2}$$

where the position of the coordinate system is denoted by $x_0 \in \mathbb{R}^3$ and the orientation is described by a rotational matrix $E \in \mathcal{SO}(3)$ from the set of ortho-normal matrices parametrized by means of θ_0 (, $\theta_0 \in \mathbb{R}^3$ for example using *Euler* angles,), while $\hat{q} \in \mathbb{R}^{n^{\mathrm{DoF}}}$ are the respective minimal joint coordinates of the humanoid. This situation is already considered in Equation (3.1) by the selection matrix S mapping the actual joint torques to the actuated degrees of freedom leaving the free-flyer or floating base free from any actuation as in (3.1).

Following this, the equation of motion of a MBS in contact from Equation (3.1) can be split into two equations

$$H_0(q)\ddot{q} + c_0(q,\dot{q}) = \qquad J_0^T\lambda, \tag{3.3a}$$
$$H_1(q)\ddot{q} + c_1(q,\dot{q}) = \tau + J_1^T\lambda, \tag{3.3b}$$

where the first describes the non-actuated dynamics and the latter the actuated part.

For the next step, we formally introduce the two concepts of CoM and ZMP as well as the respective static and dynamic balance criteria.

Definition (Center of Mass (CoM), $c \in \mathbb{R}^3$) The CoM for any system at any given time is the unique point of the average position of the mass of the system, where the weighted relative position of the distributed mass sums to zero.

In the case of a MBS consisting of $n^{\mathrm{B}} \in \mathbb{N}$ rigid bodies, the center of mass c of the respective MBS in a given configuration can be expressed as by means of the composite rigid-body inertia ${}^0I^{\mathrm{tot}} = \sum_{i=1}^{n^{\mathrm{B}}} {}^0X_i^{\star} I_i {}^iX_0$ such that the CoM is given by

$$c = \frac{{}^0I^{\mathrm{tot}}\big|_{\mathrm{m}c}}{{}^0I^{\mathrm{tot}}\big|_{\mathrm{m}}}, \tag{3.4}$$

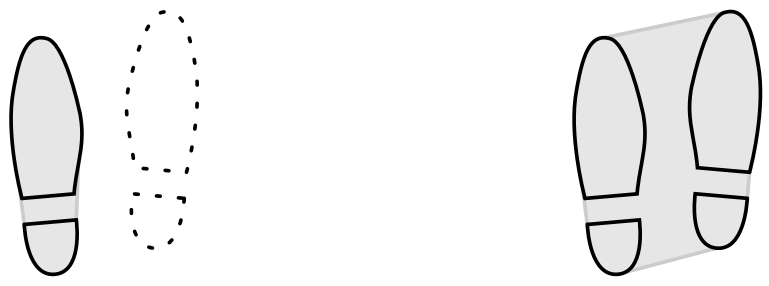
where I_i is the spatial inertia of the ith body and $I|_{(\cdot)}$ is the projection onto the respective quantity according to Definition 2.11. ${}^0X_i$ denotes the spatial transform mapping the spatial information in the ith body frame to a given reference frame according to Definition 2.12. △

Remark (Center of Gravity (CoG)) For a uniform gravitational field, an alternative view on the CoM as the point where the resultant torque due to gravity vanishes, which is sometimes also referred to as Center of Gravity (CoG). In this thesis, the two concepts are to be seen as identical and therefore we always refer to CoM by convention.

Definition (Support Polygon w.r.t. a Set of Contact Points I, $\mathcal{S}_{\mathrm{I}}$) Let $\mathrm{I} \subset \mathbb{R}^3$ be a set of co-planar contact points such that all contact points lie within a contact surface defined by normal vector $n \in \mathbb{R}^3$. The support polygon $\mathcal{S}_{\mathrm{I}} \subset \mathbb{R}^3$ is then defined as the respective convex hull containing all contact points, i.e.,

$$\mathcal{S}_{\mathrm{I}} := \operatorname{conv}(\mathrm{I}) = \left\{ \sum_{i=1}^{|\mathrm{I}|} \alpha_i x_i \,\middle|\, \forall i : \alpha_i \geq 0 \wedge \sum_{i=1}^{|\mathrm{I}|} \alpha_i = 1 \wedge x_i \in \mathrm{I} \right\}. \tag{3.5}$$

△



(a) Support polygon of single support phase. (b) Support polygon of double support phase.

Figure 3.4: Visualization of the support polygon (gray area) for different phases of the human gait, where the support foot is indicated by solid black lines and the swing foot by dashed black lines.

Remark For a humanoid, the polygon of support is the contact area of the current support foot during single support phase, c.f. Figure 3.4a. In the case of two feet on level ground, the polygon of support is the ground area covered by both feet as well as the area in between them, c.f. Figure 3.4a.

Definition (Static Stability) Given any multi-body system that is in contact with its environment by means of the set of contact points $\mathrm{I} \subset \mathbb{R}^3$ such that all contact points lie within a contact surface defined by normal vector $\boldsymbol{n} \in \mathbb{R}^3$. The multi-body system is then considered to be *statically stable* if and only if the projection along the gravity of its center of mass, defined according to Definition 3.2, lies within the respective projection of the polygon of support $\mathcal{S}_{\mathrm{I}} \subset \mathbb{R}^3$ defined according to Definition 3.4. △

The technical concept of ZMP was first introduced in 1968 by *Vukobratović*, see [175]. However, the name ZMP itself was presented first in [176]. Since then the ZMP has become an essential quantity for evaluating the stability or dynamic balance for both motion analysis as well as motion generation. For the latter in such a way that ZMP-based gait synthesis has been the only approach for a long time. The idea of the ZMP is to define an single indicator for the dynamic balance of the robot. Following this, we look for the unique point in which all forces and moments acting on the robot can be replaced by one single force and moment.

Definition (Total Change of Momentum, $\dot{\boldsymbol{h}}_{tot}$) Given a multi-body system consisting of n^{B} bodies, then the total change of angular momentum $\dot{\boldsymbol{h}}_{tot}$ with respect to the world frame indicated by index 0 is given by

$$ {}^{0}\dot{\boldsymbol{h}}_{tot} = \sum_{i=1}^{n^{\mathrm{B}}} {}^{0}\boldsymbol{X}_i^{\star *} (\boldsymbol{I}_i \boldsymbol{a}_i + \boldsymbol{v} \times^{*} \boldsymbol{I}_i \boldsymbol{v}_i), \tag{3.6} $$

where $\boldsymbol{I}_i$ is the spatial inertia of the ith body according to Definition 2.10 and ${}^{0}\boldsymbol{X}_i$ denotes the spatial transform mapping the spatial information in the ith body frame to a given reference frame according to Definition 2.12. $\boldsymbol{v}_i$, $\boldsymbol{a}_i$ denote the spatial velocity and acceleration of the ith body with respect to the local reference frame. △

Definition (Zero-Moment Point (ZMP)) Let $\mathcal{S} \subset \mathbb{R}^3$ be a convex set on a planar contact surface given by its normal vector $\boldsymbol{n} \in \mathbb{R}^3$ such that the contact friction is sufficiently high in order to prevent the contact from slipping. The ZMP is then defined, according to [177], as that point on the contact surface at which the net moment of the inertial forces and the gravity forces has no component along the horizontal axes with respect to the contact surface.

Choosing the reference frame to be located at the CoM, see Definition 3.2, of the respective MBS, the resulting external force acting at the CoM is then given by

$$^{CoM}\hat{\boldsymbol{f}}_{ext} = ^{CoM}\dot{\boldsymbol{h}}_{tot} - \mathrm{m}_{tot}^{CoM}\boldsymbol{g},$$

where the spatial force vector $^{CoM}\hat{\boldsymbol{f}}_{ext} \in \mathsf{F}^6$ denotes the respective external forces acting at the CoM, $\mathrm{m}_{tot} \in \mathbb{R}$ is the total mass of the MBS, $^{CoM}\boldsymbol{g} \in \mathsf{F}^6$ denotes the gravitational force acting on the CoM, and the superscript indicates the respective reference frame of the quantity (if required).

From the projection of these external forces onto the contact surface $^{S}\hat{\boldsymbol{f}}_{ext} = {}^{S}\boldsymbol{X}_{CoM}^{CoM}\hat{\boldsymbol{f}}_{ext}$ with $^{S}\hat{\boldsymbol{f}}_{ext} = [\,\boldsymbol{n}_0^T\; \boldsymbol{f}^T\,]$, where $\boldsymbol{n}_0, \boldsymbol{f} \in \mathbb{R}^3$ denote the total moment about the point that coincides with O and the linear force along an axis passing through O, the ZMP is then computed by

$$\boldsymbol{z}^{\mathcal{S}_{\mathrm{I}}} = \frac{\boldsymbol{n} \times \boldsymbol{n}_0}{\boldsymbol{n} \cdot \boldsymbol{f}}. \tag{3.7}$$

△

Definition (Sufficient Criterion for Dynamic Stability (based on ZMP)) Given any multi-body system that is in contact with its environment by means of the set of contact points $\mathrm{I} \subset \mathbb{R}^3$ such that all contact points lie within a contact surface defined by normal vector $\boldsymbol{n} \in \mathbb{R}^3$ and such that the contact friction is sufficiently high (in order to prevent the contacts from slipping). If the projection along gravity of the ZMP of the MBS is within the interior of the projected support polygon $\mathcal{S}_{\mathrm{I}}$, i.e.,

$$\boldsymbol{z}^{\mathcal{S}_{\mathrm{I}}} \in \mathring{\mathcal{S}}_{\mathrm{I}}, \tag{3.8}$$

then the current motion is *dynamically stable*. △

Remark (On other Stability Criteria) The ZMP criterion is not a necessary condition for dynamic stability While there is not yet a criterion known that can describe the stability of the motion of humans and is both necessary and sufficient, other stability criteria are commonly applied to generate stable motions or analyze motion data. Examples are the *Capture Point* of *Pratt* presented in [146], the foot placement estimator from *Wight* presented in [182] or criteria based on *Lyapunov* stability as presented by *Mombaur* in [125]. Recent work uses centroidal dynamics from [139] to define dynamic stable configurations, for example [31]. The latter will be explained in more detail in Chapter 4 presenting the work of the author of this thesis on this topic.

The projection of the dynamics to the CoM of the underlying MBS can be found in detail in [181]. Following this, we end up with the *Newton-Euler* equations for the CoM, i.e.,

$$\mathrm{m}_{tot}\,(\ddot{\boldsymbol{c}} + \boldsymbol{g}) = \sum_{i\in\mathrm{I}} \boldsymbol{\lambda}_i, \tag{3.9a}$$

$$\mathrm{m}_{tot}\,\boldsymbol{c} \times (\ddot{\boldsymbol{c}} + \boldsymbol{g}) + \dot{\boldsymbol{l}} = \sum_{i\in\mathrm{I}} \boldsymbol{p}_i \times \boldsymbol{\lambda}_i, \tag{3.9b}$$

where m_{tot} denotes the total mass of the humanoid, $\boldsymbol{g} \in \mathbb{R}^3$ is the gravitational force acting on the system, $\boldsymbol{c} \in \mathbb{R}^3$ denotes the coordinate vector of the CoM, $\boldsymbol{l} \in \mathbb{R}^3$ denotes the angular momentum of the system and $\boldsymbol{p}_i \in \mathbb{R}^3$, $\boldsymbol{\lambda}_i \in \mathbb{R}^3$ denote the respective lever arms with respect to the contact points and contact forces given by the contact set $\mathrm{I} \subset \mathbb{N}$.

Equation (3.9) can be further simplified by neglecting the change of angular momentum term $\dot{\boldsymbol{l}} \equiv \boldsymbol{0}$ and re-expressing the sum of contact forces as single unique force $\boldsymbol{\lambda} = \sum_i \boldsymbol{\lambda}_i$ at the ZMP according to Assumption 3.1, i.e.,

$$\mathrm{m}_{tot}\,(\ddot{\boldsymbol{c}}+\boldsymbol{g}) = \boldsymbol{\lambda}_i, \tag{3.10a}$$

$$\mathrm{m}_{tot}\,\boldsymbol{c}\times(\ddot{\boldsymbol{c}}+\boldsymbol{g}) = \sum_i \boldsymbol{p}\times\boldsymbol{\lambda}. \tag{3.10b}$$

By expressing $\boldsymbol{p} = \begin{bmatrix} p_x & p_y & p_z \end{bmatrix}^T$ and dividing by the total mass m_{tot} the equation can be simplified to the two well-known equations (the third is omitted due to linear dependence) relating the ZMP motion to the *CoM* motion given by

$$p_x = c_x - \frac{c_z}{\ddot{c}_z + \mathrm{g}}\ddot{c}_x \tag{3.11a}$$

$$p_y = c_y - \frac{c_z}{\ddot{c}_z + \mathrm{g}}\ddot{c}_y. \tag{3.11b}$$

For a CoM motion evolving on a plane parallel to the ground, i.e., $c_z \equiv \text{const.} \implies \ddot{c}_z \equiv 0$, the equation is further reduced to the *ZMP* equations

$$p_x = c_x - \frac{c_z}{\mathrm{g}}\ddot{c}_x \tag{3.12a}$$

$$p_y = c_y - \frac{c_z}{\mathrm{g}}\ddot{c}_y. \tag{3.12b}$$

Finally, the ZMP equations (3.12) or the so-called *cart-table* model according to [84] serves as the reduced model in the WPG context.

3.2 Walking Pattern Generator based on Nonlinear Model Predictive Control

The work presented in this section is based on our journal article [133]. We propose a WPG based on NMPC together with a tailored implementation, and its application for motion generation of the robotic platforms *HRP-2* and *HeiCub*. This work pursues the work on *walking without thinking* presented in [71] and it extends it in several ways. In contrast to former articles, the WPG presented here is able to compute feasible foot step positions and orientations simultaneously while locally avoiding collisions with obstacles.

The general workflow of the implementation of a WPG presented herein is schematically depicted in Figure 3.2. The core of the proposed scheme is the WPG itself, which implements a tailored solver based on advanced methods of NMPC described in Chapter 1.

In the following sections, first, we discuss the discretization of the CoM dynamics in Section 3.2.1. Second, we introduce the formulation of the automatic foot step placement in Section 3.2.2. Finally, we formulate the NMPC problem in Section 3.2.3 and the respective solution algorithm in Section 3.2.4.

3.2.1 Discretization of Center-of-Mass Dynamics

WPGs use not only a common reduced model as derived in Section 3.1 but they also share a common discretization scheme for the CoM motion derived as presented in the remainder of this section. We follow [71] for the formulation of the discretized CoM dynamics and embed it in mathematical formulation and notational style of this thesis.

As presented in Section 3.1, we restrict the whole-body dynamics of the humanoid to that of the LIPM and can therefore restrict the CoM motion to its motion in the x,y-plane.

In accordance with the optimal control problem (OCP) (1.9) formulation presented in Chapter 1, we formulate the evolution of the position of the CoM c^ν, where we use $\nu \in \{x,y\}$ to simplify the notation, in the world frame as ordinary differential equation (ODE) on a given finite preview horizon $\mathcal{T} := [0,T]$ in the form of

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix} c^\nu(t) \\ \dot{c}^\nu(t) \\ \ddot{c}^\nu(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}\begin{bmatrix} c^\nu(t) \\ \dot{c}^\nu(t) \\ \ddot{c}^\nu(t) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}\dddot{c}^\nu(t), \quad t \in \mathcal{T}, \tag{3.13}$$

where we strive for finding optimal controls in the form of input jerk $\dddot{c}^\nu$. We subdivide the given horizon $\mathcal{T}$ equidistantly by means of a constant sampling period h into $n \in \mathbb{N}$ subintervals $\mathcal{T}_k = [kh,(k+1)h]$, $k = 0,1,\ldots,n-1$, such that $nh = T$ is the length of the preview horizon.

In order to formulate a finite dimensional optimization problem and to obtain smooth trajectories, such that the accelerations are at least continuous, we discretize the CoM jerk $\dddot{c}^\nu$ by means of a piecewise constant control on each interval, i.e.,

$$\dddot{c}^\nu(t) \equiv \text{const.}, \quad t \in \mathcal{T}_j, \quad j = 0,1,\ldots,n-1. \tag{3.14}$$

Following this, a time-stepping scheme, which maps the current state of the frame $\boldsymbol{c}_k^\nu$ to the future states, can be derived by manually performing the integration on the subintervals, it reads

$$\boldsymbol{c}_{k+j}^\nu = \boldsymbol{A}^j\, \boldsymbol{c}_k^\nu + \sum_{i=0}^{j-1} \boldsymbol{A}^i\, \boldsymbol{B}\, \dddot{c}_{k+i}^\nu, \quad j = 0,1,\ldots,n, \tag{3.15}$$

$$\boldsymbol{c}_k^\nu = \begin{bmatrix} c_k^\nu \\ \dot{c}_k^\nu \\ \ddot{c}_k^\nu \end{bmatrix}, \quad \boldsymbol{A} = \begin{bmatrix} 1 & h & h^2/2 \\ 0 & 1 & h \\ 0 & 0 & 1 \end{bmatrix}, \quad \boldsymbol{B} = \begin{bmatrix} h^3/6 \\ h^2/2 \\ h \end{bmatrix}. \tag{3.16}$$

The evolution of the CoM over the preview horizon as well as the respective derivatives are then described by means of the initial state $\boldsymbol{c}_k^\nu$ and the CoM jerk $\dddot{\boldsymbol{C}}_{k+1}^\nu$ by

$$\boldsymbol{C}_{k+1}^\nu = \boldsymbol{P}_{ps}\boldsymbol{c}_k^\nu + \boldsymbol{P}_{pu}\dddot{\boldsymbol{C}}_k^\nu, \tag{3.17}$$

$$\dot{\boldsymbol{C}}_{k+1}^\nu = \boldsymbol{P}_{vs}\boldsymbol{c}_k^\nu + \boldsymbol{P}_{vu}\dddot{\boldsymbol{C}}_k^\nu, \tag{3.18}$$

$$\ddot{\boldsymbol{C}}_{k+1}^\nu = \boldsymbol{P}_{as}\boldsymbol{c}_k^\nu + \boldsymbol{P}_{au}\dddot{\boldsymbol{C}}_k^\nu, \tag{3.19}$$

where $\boldsymbol{C}_{k+1}^\nu$, $\dot{\boldsymbol{C}}_{k+1}^\nu$, $\ddot{\boldsymbol{C}}_{k+1}^\nu$ and $\dddot{\boldsymbol{C}}_{k+1}^\nu \in \mathbb{R}^n$ are given by

$$\boldsymbol{C}_{k+1}^\nu = \begin{bmatrix} c_{k+1}^\nu, & \ldots & c_{k+n}^\nu \end{bmatrix}^T,$$
$$\dot{\boldsymbol{C}}_{k+1}^\nu = \begin{bmatrix} \dot{c}_{k+1}^\nu & \ldots & \dot{c}_{k+n}^\nu \end{bmatrix}^T,$$
$$\ddot{\boldsymbol{C}}_{k+1}^\nu = \begin{bmatrix} \ddot{c}_{k+1}^\nu & \ldots & \ddot{c}_{k+n}^\nu \end{bmatrix}^T,$$
$$\dddot{\boldsymbol{C}}_{k+1}^\nu = \begin{bmatrix} \dddot{c}_{k+1}^\nu & \ldots & \dddot{c}_{k+n}^\nu \end{bmatrix}^T,$$

where we exceptionally use matrix notation for the compound vectors due to a notational clash. The respective matrices in the expression, $\boldsymbol{P}_{ps}, \boldsymbol{P}_{vs}, \boldsymbol{P}_{as} \in \mathbb{R}^{n\times 3}$ and $\boldsymbol{P}_{au}, \boldsymbol{P}_{pu}, \boldsymbol{P}_{vu} \in \mathbb{R}^{n\times n}$, are given by

$$\boldsymbol{P}_{ps} = \begin{bmatrix} 1 & h & {h^2}/{2} \\ \vdots & \vdots & \vdots \\ 1 & nh & {n^2h^2}/{2} \end{bmatrix}, \qquad \boldsymbol{P}_{pu} = \begin{bmatrix} {h^3}/{6} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ {(1+3n+3n^2)h^3}/{6} & \cdots & {h^3}/{6} \end{bmatrix},$$

$$\boldsymbol{P}_{vs} = \begin{bmatrix} 0 & 1 & h \\ \vdots & \vdots & \vdots \\ 0 & 1 & nh \end{bmatrix}, \qquad \boldsymbol{P}_{vu} = \begin{bmatrix} {h^2}/{2} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ {(1+2n)h^2}/{2} & \cdots & {h^2}/{2} \end{bmatrix}$$

$$\boldsymbol{P}_{as} = \begin{bmatrix} 0 & 0 & 1 \\ \vdots & \vdots & \vdots \\ 0 & 0 & 1 \end{bmatrix}, \qquad \boldsymbol{P}_{au} = \begin{bmatrix} h & \cdots & 0 \\ \vdots & \ddots & \vdots \\ h & \cdots & h \end{bmatrix}.$$

An equivalent scheme can be formulated for ZMP evolution using the LIPM as derived in Section 3.1 from (3.12), it reads

$$z_{k+j}^{\nu} = \begin{bmatrix} 1 & 0 & -{c^z}/{\mathrm{g}} \end{bmatrix} \boldsymbol{c}_{k+j}^{\nu}, \quad j = 0, 1 \dots n,$$

where c^z denotes the constant *CoM* height between the ground and the CoM, and g denotes the norm of the gravity vector $\boldsymbol{g}$. The expression for the evolution of the ZMP with respect to the initial CoM states and the CoM jerk on the finite horizon is then given by

$$\boldsymbol{Z}_{k+1}^{\nu} = \boldsymbol{P}_{zs}\boldsymbol{c}_k^{\nu} + \boldsymbol{P}_{zu}\dddot{\boldsymbol{C}}_k^{\nu}, \tag{3.20}$$

where $\boldsymbol{P}_{zs} \in \mathbb{R}^{n\times 3}$ $\boldsymbol{P}_{zu} \in \mathbb{R}^{n\times n}$ are defined by

$$\boldsymbol{P}_{zs} = \begin{bmatrix} 1 & h & {h^2}/{2} - {c^z}/{\mathrm{g}} \\ \vdots & \vdots & \vdots \\ 1 & nh & {n^2h^2}/{2} - {c^z}/{\mathrm{g}} \end{bmatrix}, \qquad \boldsymbol{P}_{zu} = \begin{bmatrix} {h^3}/{6} - {hc^z}/{\mathrm{g}} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ {(1+3n+3n^2)h^3}/{6} - {hc^z}/{\mathrm{g}} & \cdots & {h^3}/{6} - {hc^z}/{\mathrm{g}} \end{bmatrix}.$$

3.2.2 Automatic Foot Step Placement

In contrast to former approaches, which only track a precomputed and feasible ZMP reference trajectory $\boldsymbol{z}^{ref}$, advanced WPGs derived from [71] implement adaptive placement of the feet in order to ensure the balance of the robot even under external perturbations. To this end, consider a frame $\mathcal{F}$ attached to the support foot center, with its current position and orientation on the ground given by f_k^{ν} with $\nu \in \{x, y, \theta\}$, where by θ we denote the orientation of the foot frame with respect to the world frame.

While the usage of a single point mass as model prohibits the definition of an orientation, in [71] a frame attached to the CoM in accordance with the floating base of the robot is defined and the orientation of this frame and the feet directions are optimized. Here, we only optimize the foot step orientations from which the orientation of the robot free-flyer, denoted by c^{θ}, is computed as described in the remainder of this section. Let $c^{\theta}(t)$, $f^{\theta,\mathrm{L}}(t)$ and $f^{\theta,\mathrm{R}}(t)$ be respectively the orientation of the free-flyer, the left foot and the right foot at any time $t \in \mathcal{T}$. Hence $c^{\theta}(t)$ is by convention the average orientation of both feet at any time given by

$$\begin{bmatrix} c^{\theta}(t) \\ \dot{c}^{\theta}(t) \\ \ddot{c}^{\theta}(t) \end{bmatrix} = \begin{bmatrix} \frac{1}{2}(f^{\theta,\mathrm{L}}(t) + f^{\theta,\mathrm{R}}(t)) \\ \frac{1}{2}(\dot{f}^{\theta,\mathrm{L}}(t) + \dot{f}^{\theta,\mathrm{R}}(t)) \\ \frac{1}{2}(\ddot{f}^{\theta,\mathrm{L}}(t) + \ddot{f}^{\theta,\mathrm{R}}(t)) \end{bmatrix}, \quad t \in \mathcal{T}.$$

We describe the future steps as free variable of the optimization problem and denote them by

$$\boldsymbol{F}^{\nu}_{k+1} = \begin{bmatrix} f^{\nu}_{k+1} & f^{\nu}_{k+2} & \dots & f^{\nu}_{k+n} \end{bmatrix}^T, \tag{3.21}$$

for which we can derive a similar mapping relating the initial support foot position f^{ν}_k and the actual free foot step positions $\boldsymbol{F}^{\nu}_k \in \mathbb{R}^{n_f}$ to the support foot position at each time step $\boldsymbol{F}^{\nu}_{k+1} \in \mathbb{R}^n$ by

$$\boldsymbol{F}^{\nu}_{k+1} = \boldsymbol{v}_{k+1} f^{\nu}_k + \boldsymbol{V}_{k+1} \boldsymbol{F}^{\nu}_k, \tag{3.22}$$

where the vector $\boldsymbol{v}_{k+1} \in \mathbb{R}^n$ and matrix $\boldsymbol{V}_{k+1} \in \mathbb{R}^{n \times n_f + 1}$ indicate which step falls in the sampling interval $\mathcal{T}_k$ and are given by

$$\boldsymbol{v}_{k+1} = \begin{bmatrix} 1 \\ \vdots \\ 1 \\ 0 \\ \vdots \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \qquad \boldsymbol{V}_{k+1} = \begin{bmatrix} 0 & 0 & & 0 & 0 \\ \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & & 0 & 0 \\ 1 & 0 & & 0 & 0 \\ \vdots & \vdots & & \vdots & \vdots \\ 1 & 0 & & 0 & 0 \\ 0 & 1 & & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & & 1 & 0 \\ 0 & 0 & & 0 & 1 \\ \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & & 0 & 1 \end{bmatrix}, \tag{3.23}$$

where sampling times correspond to rows and steps to columns, and by n_f we denote the maximum number of double support phases in the preview.

3.2.3 Nonlinear Model Predictive Control Problem Formulation

In this section, we formulate the NMPC problem that enables to simultaneously optimize both the position and the orientation by means of the respective free variables presented in the previous sections.

In [71], the orientation problem is solved separately from the position problem in order to circumvent the nonlinear case. However, solving the orientation separately and then injecting the solution into the position problem amounts to solve a different problem than the nonlinear combination of both.

This problem is especially important when dealing with a cluttered environment, where obstacles have to be avoided, as it is considered within this work, because then the orientation cannot be computed separately from the position anymore. Following this, we derive, analyze the nonlinear problem formulation and propose an appropriate approach for its solution.

In order to derive the problem formulation in the form of an NLP, we first will give a description of the objective function and the necessary constraints for the ZMP, foot position constraints as well as the collision avoidance. From the composition of the problem, we then derive a tailored solver for the problem in Section 3.2.4 and finally talk about its implementation.

The Objective Function

The objective function for the NLP used to formulate the discretized NMPC problem is given by a linear combination of four distinct terms in the form of

$$\min_{\boldsymbol{U}_k} \frac{\omega_1}{2} j_1(\boldsymbol{U}_k) + \frac{\omega_2}{2} j_2(\boldsymbol{U}_k) + \frac{\omega_3}{2} j_3(\boldsymbol{U}_k) + \frac{\omega_4}{2} j_4(\boldsymbol{U}_k), \tag{3.24}$$

where $\omega_i \geqslant 0$ for $i = 1,2,3,4$ denote the weights of the respective terms of the objective function and $\boldsymbol{U}_k$ denote the free variables of the problem given as

$$\boldsymbol{U}_k^{x,y} = \begin{bmatrix} \dddot{\boldsymbol{C}}_k^x \\ \boldsymbol{F}_k^x \\ \dddot{\boldsymbol{C}}_k^y \\ \boldsymbol{F}_k^y \end{bmatrix}, \quad \boldsymbol{U}_k^\theta = \boldsymbol{F}_k^\theta, \quad \boldsymbol{U}_k = \begin{bmatrix} \boldsymbol{U}_k^{x,y} \\ \boldsymbol{U}_k^\theta \end{bmatrix}. \tag{3.25}$$

$j_1(\boldsymbol{U}_k)$ is the cost function term related to linear velocity tracking

$$j_1(\boldsymbol{U}_k) = ||\dot{\boldsymbol{C}}_{k+1}^x - \boldsymbol{v}_{k+1}^{x,ref}||_2^2 + ||\dot{\boldsymbol{C}}_{k+1}^y - \boldsymbol{v}_{k+1}^{y,ref}||_2^2.$$

$j_2(\boldsymbol{U}_k)$ is the cost function term related to angular velocity tracking

$$j_2(\boldsymbol{U}_k) = ||\boldsymbol{F}_{k+1}^\theta - \int \boldsymbol{v}_{k+1}^{\theta,ref} \, \mathrm{d}t||_2^2.$$

$j_3(\boldsymbol{U}_k)$ is the cost function term related to minimizing the distance between the ZMP and the projection of the ankle onto the sole

$$j_3(\boldsymbol{U}_k) = ||\boldsymbol{F}_{k+1}^x - \boldsymbol{Z}_{k+1}^x||_2^2 + ||\boldsymbol{F}_{k+1}^y - \boldsymbol{Z}_{k+1}^y||_2^2. \tag{3.26}$$

$j_4(\boldsymbol{U}_k)$ is the cost function term related to minimizing the applied CoM jerk

$$j_4(\boldsymbol{U}_k) = ||\dddot{\boldsymbol{C}}_{k+1}^x||_2^2 + ||\dddot{\boldsymbol{C}}_{k+1}^y||_2^2.$$

The above minimization problem can then be expressed in the canonical form of

$$\min_{\boldsymbol{U}_k} \quad \frac{1}{2}\boldsymbol{U}_k^T \boldsymbol{Q}_k \boldsymbol{U}_k + \boldsymbol{p}_k^T \boldsymbol{U}_k \tag{3.27a}$$

$$\text{with} \quad \boldsymbol{Q}_k = \begin{bmatrix} \boldsymbol{Q}_k^{x,y} & 0 \\ 0 & \boldsymbol{Q}_k^{\theta} \end{bmatrix}, \tag{3.27b}$$

$$\boldsymbol{Q}_k^{\theta} = \omega_2 \mathbb{I}_{n_f}, \tag{3.27c}$$

$$\boldsymbol{p}_k = \begin{bmatrix} \boldsymbol{p}_k^{x,y} \\ \boldsymbol{p}_k^{\theta} \end{bmatrix}, \tag{3.27d}$$

$$\boldsymbol{p}_k^{\theta} = \omega_2 \left(\begin{bmatrix} 1 & \dots & n_f \end{bmatrix} T_{step}\, \boldsymbol{v}_{k+1}^{\theta,ref} + \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} \boldsymbol{F}_k^{\theta} \right) \tag{3.27e}$$

The reader is kindly referred to [71] for the defintion of $\boldsymbol{Q}^{x,y}$ and $\boldsymbol{p}^{x,y}$. The matrix $\boldsymbol{Q}_k^{\theta}$ and $\boldsymbol{p}_k^{\theta}$ are derived because we use a slightly different approach than [71] to deal with the orientation.

The Constraints

In order to guarantee applicability of the planned motion, we have to enforce different constraints on it. First, the balance of the robot has to be ensured by defining proper constraints onto the ZMP trajectories. Second, because we implement automatic foot step placement, we have to guarantee kinematic feasibility of the foot steps in order to realize on the robot. Additionally, we implement obstacle avoidance by introducing nonlinear constraints to the problem. The following exposition is based on [71].

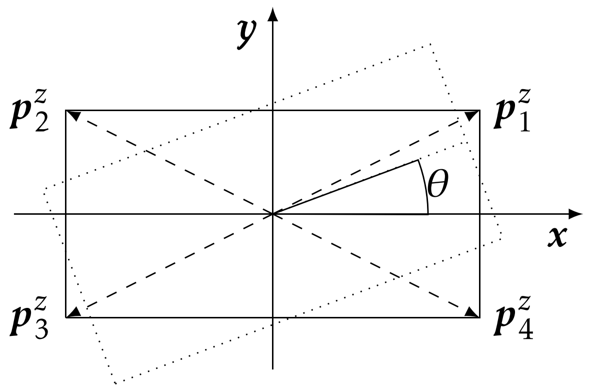


Figure 3.5: Shape of the foot with the position vector $\boldsymbol{p}_i^z$ describing the support polygon and θ representing its orientation from our article [133].

Balance Constraints In order to guarantee a dynamic stable motion according to Definition 3.9, the ZMP has to remain strictly inside the support polygon, c.f. [181]. The respective polygon for the humanoid robot *HRP-2* is depicted in Figure 3.5. Only a single foot has to be modeled as a support polygon, because

- *HRP-2* has symmetrical feet and
- the sampling period of the problem is designed in a way that no double support phase occurs at any sampling time.

We denote the set of linear inequalities representing the convex polygon by means of the matrix $\boldsymbol{A}_{ZMP} \in \mathbb{R}^{5n \times 2(n+n_f)}$ and vector $\boldsymbol{b}_{ZMP} \in \mathbb{R}^{5n}$. The ZMP at instant k, denoted by $\boldsymbol{z}_k = [\, z_k^x \; z_k^y \,]^T$, (c.f. Section 3.1) lies inside the support polygon if and only if the following linear inequalities hold

$$\boldsymbol{A}_{ZMP}\boldsymbol{R}(f_k^\theta)(\boldsymbol{z}_k - \boldsymbol{f}_k) \leqslant \boldsymbol{b}_{ZMP}, \tag{3.28}$$

where the support foot position at time instant k is given by $\boldsymbol{f}_k = [\, f_k^x \; f_k^y \,]^T$, the 2D rotation matrix $\boldsymbol{R}(f_k^\theta)$ is defined as

$$\boldsymbol{R}(f_k^\theta) = \begin{bmatrix} \cos(f_k^\theta) & \sin(f_k^\theta) \\ -\sin(f_k^\theta) & \cos(f_k^\theta) \end{bmatrix}. \tag{3.29}$$

We further separate the matrix product $\boldsymbol{A}_{ZMP}\boldsymbol{R}(f_k^\theta)$ into sub-matrices depending on the respective optimization variables by

$$\boldsymbol{A}_{ZMP}\boldsymbol{R}(f_k^\theta) = \begin{bmatrix} \boldsymbol{A}_{cop,k}^{x,\theta} & \boldsymbol{A}_{cop,k}^{y,\theta} \end{bmatrix}. \tag{3.30}$$

Using Equation (3.22), the constraint for each time step of the horizon is defined by

$$\boldsymbol{D}_{k+1}(\boldsymbol{U}_k^\theta) \begin{bmatrix} \boldsymbol{Z}_{k+1}^x - \boldsymbol{v}_{k+1} f_k^x - \boldsymbol{V}_{k+1}\boldsymbol{F}_k^x \\ \boldsymbol{Z}_{k+1}^y - \boldsymbol{v}_{k+1} f_k^y - \boldsymbol{V}_{k+1}\boldsymbol{F}_k^y \end{bmatrix} \leqslant \boldsymbol{b}_{ZMP\,k+1}, \tag{3.31}$$

with

$$\boldsymbol{b}_{ZMP,k+1} = \begin{bmatrix} \boldsymbol{b}_{ZMP}^T & \dots & \boldsymbol{b}_{ZMP}^T \end{bmatrix}^T \tag{3.32}$$

and

$$\boldsymbol{D}_{k+1}(\boldsymbol{U}_k^\theta) = \begin{bmatrix} \boldsymbol{A}_{cop,k+1}^{x,\theta} & 0 & \boldsymbol{A}_{cop,k+1}^{y,\theta} & 0 \\ & \ddots & & \ddots \\ 0 & \boldsymbol{A}_{cop,k+n}^{x,\theta} & 0 & \boldsymbol{A}_{cop,k+n}^{y,\theta} \end{bmatrix}. \tag{3.33}$$

From Equation (3.31), the canonical form of the constraint is

$$\boldsymbol{A}_{cop,k}(\boldsymbol{U}_k^\theta)\, \boldsymbol{U}_k^{x,y} \leqslant \overline{\boldsymbol{U}_{cop,k}}, \tag{3.34}$$

where $\boldsymbol{A}_{cop,k}(\boldsymbol{U}_k^\theta)$ is a matrix depending on $\boldsymbol{U}_k^\theta$ which makes this constraint nonlinear and by $\overline{\boldsymbol{U}_{cop,k}}$ we define the upper bound of the linear inequality. The last steps of the derivation are detailed in [71].

Foot Step Feasibility Constraints In order to guarantee a feasible foot step placement, we require a description of the kinematic and dynamic capabilities of the robot under consideration in form of a convex hull. As presented in [71], this can be achieved by kinematically sampling the task space of the feet and checking if the performed step would make the robot fall. In this way, a convex hull representing feasible foot placement for *HRP-2* can be derived as it is shown in Figure 3.6. The set of linear inequalities representing this convex polygon is defined by means of the matrix $\boldsymbol{A}_{foot}$ and $\boldsymbol{b}_{foot}$. Instead of distinguishing between the left and the right support foot by means of indices r

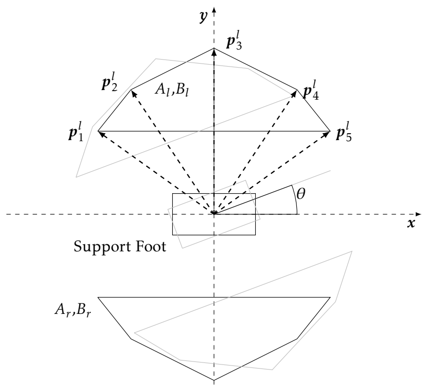


Figure 3.6: Shape of the selected convex polygon boundary of the foot placement from the article [133].

or l, we use the lower index $foot$ because the problem is symmetrical. The linear inequality formulating proper foot step placement by means of the convex hull in Figure 3.6 is given as

$$\boldsymbol{A}_{foot}\boldsymbol{R}(\theta)(\boldsymbol{f}_{k+1}-\boldsymbol{f}_k) \leqslant \boldsymbol{b}_{foot}. \tag{3.35}$$

In the exact manner as in Equation (3.34), the vector and matrices derived in Equation (3.34) are used to express this constraint for each previewed foot step according to [71]. The canonical form of the constraint is then given by

$$\boldsymbol{A}_{foot,k}(\boldsymbol{U}_k^{\theta})\,\boldsymbol{U}_k^{x,y} \leqslant \overline{\boldsymbol{U}_{foot,k}}\,, \tag{3.36}$$

where $\boldsymbol{A}_{foot,k}(\boldsymbol{U}_k^{\theta})$ depends on $\boldsymbol{U}_k^{\theta}$ like the ZMP constraint matrix $\boldsymbol{A}_{cop,k}(\boldsymbol{U}_k^{\theta})$, which again renders this constraint to be nonlinear while $\overline{\boldsymbol{U}_{foot,k}}$ denotes an upper bound.

Foot Orientation Constraint One additional constraint considers the maximum and minimum angle between both feet

$$-\theta_{thresh} \leqslant \boldsymbol{F}_{k+1}^{\theta} - \boldsymbol{F}_k^{\theta} \leqslant \theta_{thresh}, \tag{3.37}$$

which can be stated in canonical form of

$$\underline{\boldsymbol{U}_{\theta,k}} \leqslant \boldsymbol{A}_{\theta}\boldsymbol{U}_k^{\theta} \leqslant \overline{\boldsymbol{U}_{\theta,k}} \tag{3.38}$$

(3.39)

where the matrix $\boldsymbol{A}_\theta \in \mathbb{R}^{\times}$ and the vectors $\underline{\boldsymbol{U}_{\theta,k}}$, $\overline{\boldsymbol{U}_{\theta,k}} \in \mathbb{R}$ are given by

$$\boldsymbol{A}_\theta = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -1 & 1 & \ddots & \vdots \\ 0 & \ddots & \ddots & 0 \\ \ddots & 0 & -1 & 1 \end{bmatrix},$$

$$\overline{\boldsymbol{U}_{\theta,k}} = \begin{bmatrix} \theta_{thresh} + f_k^\theta & \theta_{thresh} & \dots & \theta_{thresh} \end{bmatrix}^T,$$

$$\underline{\boldsymbol{U}_{\theta,k}} = \begin{bmatrix} -\theta_{thresh} + f_k^\theta & -\theta_{thresh} & \dots & -\theta_{thresh} \end{bmatrix}^T.$$

In practice the bound $\theta_{thresh} = 0.05\,\text{rad}$ is used, which takes the hardware limits into account. At this stage, the optimization problem allows the robot to place its feet anywhere inside the convex hull at any moment. In [71], the velocity of the foot is limited by bounding the feasible foot step area that corresponds to a maximum velocity. We chose to use the same idea extended to all the foot steps degrees of freedom. This significantly decreases the variation of accelerations before foot landing.

Local Obstacle Avoidance Constraints To this end, we only consider obstacles, whose projections on the ground can be approximated as convex polygon. We choose to further approximate a given obstacle by means of a circle given by $\mathcal{C} = \{\boldsymbol{p} \in \mathbb{R}^2, (p^x - x_0)^2 + (p^y - y_0)^2 = \mathrm{R}^2\}$, where $\boldsymbol{p} = [p^x\ p^y]^T$. This choice simplifies the derivation, allows to easily introduce additional security margins for collisions avoidance, and the resulting constraint does not depend on the orientation of the foot steps this way. Here, x_0 and y_0 are coordinates of the center of the obstacle with respect to the world frame and R is the radius of the circular approximation. A collision free walking pattern for the given time horizon is given if and only if all previewed foot steps are not within any of the circular approximations. For the jth previewed step, at iteration $k+j$ the constraint is expressed by

$$\mathrm{R}^2 + \mathrm{m}^2 \leqslant = \left(f_{k+j}^x - x_0\right)^2 + \left(f_{k+j}^y - y_0\right)^2, \tag{3.40}$$

where m is a security margin taking into account the swept volume of the robot. This constraint can be formulated in canonical form in the form of

$$\underline{\boldsymbol{U}_{obs,j}} \leqslant \boldsymbol{U}_k^T \boldsymbol{H}_{obs,j} \boldsymbol{U}_k + \boldsymbol{A}_{obs,j} \boldsymbol{U}_k \tag{3.41}$$

with a selection matrix $\boldsymbol{H}_{obs,j} \in \mathbb{R}^{\times}$ and a vector $\boldsymbol{A}_{obs,j}$ depending on x_0 and y_0.

3.2.4 The Solver

In order to solve the problem with quadratic objective function but nonlinear constraints, this section is dedicated to derive a tailored optimization algorithm for the given problem class. The nonlinearity of the constraint and the still quadratic objective classifies the former QP as a general NLP with a certain structure that can efficiently be exploited. While former variants of WPG could leverage LQR solvers to solve the optimization problem as presented in [83] or convex optimization for the separated solution of the position and orientation problem as used in [71], we have to solve a nonlinear least squares optimization problem here. The least-squares problem can be state in general form of

$$
\begin{aligned}
\min_{\boldsymbol{U}_k} \quad & \frac{1}{2}\|\boldsymbol{\ell}(\boldsymbol{U}_k)\|_2^2 && (3.42a)\\
\text{s.t.} \quad & \underline{\boldsymbol{d}} \leqslant \boldsymbol{d}(\boldsymbol{U}_k) \leqslant \overline{\boldsymbol{d}}. && (3.42b)
\end{aligned}
$$

In general, derivative-based methods of optimization are the methods of choice for NLPs as presented in Chapter 1.5.3. Here, the least squares structure can be exploited to solve Equation (3.42) more efficiently using a generalized *Gauß-Newton* method. Starting with an initial guess $\boldsymbol{U}_0$ the method iterates $\boldsymbol{U}_{k+1} = \boldsymbol{U}_k + \Delta\boldsymbol{U}_k$, where the increment $\Delta\boldsymbol{U}_k$ is obtained from the solution of the following quadratic approximation of the former nonlinear problem (3.42), which reads

$$
\begin{aligned}
\min_{\Delta\boldsymbol{U}_k} \quad & \frac{1}{2}\|\boldsymbol{\ell}_{k-1} + \nabla\boldsymbol{\ell}_k^T \Delta\boldsymbol{U}_k\|_2^2 && (3.43a)\\
\text{s.t.} \quad & \underline{\boldsymbol{d}} - \boldsymbol{d}_k \leqslant \nabla\boldsymbol{d}_k^T \Delta\boldsymbol{U}_k \leqslant \overline{\boldsymbol{d}} - \boldsymbol{d}_k, && (3.43b)
\end{aligned}
$$

with $\boldsymbol{\ell}_k := \boldsymbol{\ell}(\boldsymbol{U}_k)$, $\boldsymbol{d}_k := \boldsymbol{d}(\boldsymbol{U}_k)$, and $\nabla\boldsymbol{\ell}_k^T := \frac{\mathrm{d}\boldsymbol{\ell}}{\mathrm{d}\boldsymbol{U}}(\boldsymbol{U}_k)$ the transposed gradient with respect to $\boldsymbol{U}_k$, which coincides with the respective Jacobian. Reformulating Equation (3.43) into canonical form reads

$$
\begin{aligned}
\min_{\Delta\boldsymbol{U}_k} \quad & \frac{1}{2}\Delta\boldsymbol{U}_k^T \tilde{\boldsymbol{Q}}_k \Delta\boldsymbol{U}_k + \tilde{\boldsymbol{p}}_k^T \Delta\boldsymbol{U}_k && (3.44a)\\
\text{s.t.} \quad & \underline{\tilde{\boldsymbol{U}_k}} \leqslant \tilde{\boldsymbol{A}}_k \Delta\boldsymbol{U}_k \leqslant \overline{\tilde{\boldsymbol{U}_k}}, && (3.44b)
\end{aligned}
$$

with the Hessian matrix $\tilde{\boldsymbol{Q}}_k$ and the gradient vector $\tilde{\boldsymbol{p}}_k$ given by

$$
\tilde{\boldsymbol{Q}}_k = \boldsymbol{Q}_k, \quad \tilde{\boldsymbol{p}}_k = \begin{bmatrix} \frac{1}{2}(\boldsymbol{U}_{k-1}^{x,y})^T \boldsymbol{Q}_k^{x,y} + \boldsymbol{p}_k^{x,y} \\ \frac{1}{2}(\boldsymbol{U}_{k-1}^{\theta})^T \boldsymbol{Q}_k^{\theta} + \boldsymbol{p}_k^{\theta} \end{bmatrix},
$$

as well as the linear inequalities given by

$$
\tilde{\boldsymbol{A}}_k = \begin{bmatrix} \boldsymbol{A}_{cop,k}(\boldsymbol{U}_{k-1}^{\theta}) & \nabla_{\boldsymbol{U}_k^{\theta}}^T \boldsymbol{A}_{cop,k}|_{\boldsymbol{U}_{k-1}^{\theta}} \boldsymbol{U}_{k-1}^{x,y}) \\ \boldsymbol{A}_{foot,k}(\boldsymbol{U}_{k-1}^{\theta}) & \nabla_{\boldsymbol{U}_k^{\theta}}^T \boldsymbol{A}_{foot,k}|_{\boldsymbol{U}_{k-1}^{\theta}} \boldsymbol{U}_{k-1}^{x,y}) \\ 0 & A_{\theta} \\ H_{obs,j}\boldsymbol{U}_{k-1} + A_{obs,j} & 0 \end{bmatrix},
$$

$$
\underline{\tilde{\boldsymbol{U}_k}} = \begin{bmatrix} -\infty \\ -\infty \\ \underline{\boldsymbol{U}_{\theta,k}} \\ \underline{\boldsymbol{U}_{obs,j}} \end{bmatrix} - \boldsymbol{h}_{k-1}, \overline{\tilde{\boldsymbol{U}_k}} = \begin{bmatrix} \overline{\boldsymbol{U}_{cop,k}} \\ \overline{\boldsymbol{U}_{foot,k}} \\ \overline{\boldsymbol{U}_{\theta,k}} \\ +\infty \end{bmatrix} - \boldsymbol{h}_{k-1},
$$

$$
\boldsymbol{h}_{k-1} = \begin{bmatrix} \boldsymbol{A}_{cop,k}(\boldsymbol{U}_{k-1}^{\theta})\, \boldsymbol{U}_{k-1}^{x,y} \\ \boldsymbol{A}_{foot,k}(\boldsymbol{U}_{k-1}^{\theta})\, \boldsymbol{U}_{k-1}^{x,y} \\ \boldsymbol{A}_{\theta}\, \boldsymbol{U}_{k-1}^{\theta} \\ \boldsymbol{U}_{k-1}^T \boldsymbol{H}_{obs,j} \boldsymbol{U}_{k-1} + \boldsymbol{A}_{obs,j} \boldsymbol{U}_{k-1} \end{bmatrix},
$$

$$
\forall j \in 1, \ldots, n_f.
$$

In principle, at each time instant of the control loop the resolution of the nonlinear problem requires the use of the above sequential quadratic programming (SQP) method to be solved until a given as convergence criterion is reached, c.f. [138]. However, in this work, we apply the idea of the NMPC *real-time iterations* due to [15, 35] as they were presented in Chapter 1.5.4. By carefully initializing the applied SQP method and by preserving the state from the last iteration, the computational effort can be reduced to solving a single QP (one iteration of the respective SQP method) at each time instant of the control loop. Furthermore, the computational process is separated into three phases, two of which can be completed in advance without knowledge of the actual process state. In this way, the feedback delay can be drastically reduced. Therefore, instead of solving Equation (3.42), we recalculate its linearization once at each iteration of the control loop and solve only the QP (3.44) once in each iteration.

For the solution itself, off-the-shelf QP solver for the respective QP derived in (3.44) can be applied. Here, we employ *qpOases* for a solution in real-time as this solver implements an online active set strategy. We refer to [53] for the respective details. The achieved results with this implementation are then shown in Section 3.5.

3.3 Walking Pattern Generator based on Mixed-integer Programming

In the following sections, we propose a reformulation of a mixed-integer approach presented in [78]. In contrast to this work, our formulation can be generalized in terms of number of contacts and implements the simplified contact complementarity proposed in our article [106] as mixed-integer complementarity constraints. Furthermore, we improve the “reactivity” of the approach presented in [133] by developing an evasion strategy based on the alteration of the reference velocities according to a potential field induced by obstacles in the environment. The key feature of this novel approach is the implementation of an automatic contact surface selection for the foot steps, which combines the planning of foot steps from [33] together with the simplified motion generation, which is based on the LIPM used in pattern generation in general.

Revisiting the mixed-integer approach presented in [78], the key features are

1. automatic foot step placement,
2. no prescription of gait phase and order, and
3. no prescription of step timing necessary.

However, we could also identify three major flaws in the approach, i.e.

1. it is based on virtual/non-physical variables from which walking emerges,
2. it does not generalize in terms of contacts, which prevents extension to more versatile application of the approach, and
3. it did not implement auto collision avoidance.

Therefore, a major reformulation is required to improve the existing features, while addressing the above mentioned flaws.

Our approach strives to formulate the walking and obstacle avoidance problem as mixed-integer quadratic program (MIQP). The difference to a canonical QP as presented in (1.20) of Chapter 1 or (3.43) in Section 3.2.4 is that a subset of the free optimization variables are discrete variables.

In order to do so, we first describe the respective constraints that enables a walking motion to evolve from the formulation as well as constraints that guarantee stability and kinematic feasibility of the resulting motion. Second, we introduce the collision avoidance strategy of finding obstacle-free areas (OFAs) and our reference velocity deflection algorithm. Third, because some of the constraints require the augmentation of the objective function by additional terms and the introduction of additional variables, we talk about the objective function. Key concept of the formulation is the collision avoidance that is realized as planning the motion in the OFAs in the scene and reference velocity deflection on the obstacles to provide guidance for evasion. Finally, we derive a MIQP that enable our formulation to be efficiently solved by the off-the-shelf solver *gurobi* [65].

3.3.1 Constraint Formulation

In this section, we recap the essential constraints for the MIQP formulation we roughly described above. While we refer to Section 3.2.3, when possible, here we describe the key constraints that are different from other approaches, which are mostly the proper definition of the feasible set of the integer variables as well as the complementarity constraint that enable the evolution of a gait pattern.

Force-Contact Complementarity as Mixed-Integer Constraints

Here, we apply the LIPM as presented in Section 3.1 as dynamic model and follow the same discretization of CoM and ZMP trajectories as presented in Section 3.2.1.

Following the definition of a MBSs in unilateral contact with its environment that was already presented in Chapter 1, an essential aspect of the contact dynamics is the force-contact complementarity of Section 1.3. The force-contact complementarity models a mutual exclusivity of contact force and mode of the contact (active or inactive), given by Equation (1.6).

Revisiting the formulation presented in [106], using the reduced model given by (3.10) the force-torque complementarity can be reduced to a simplified version, i.e.,

$$\|\boldsymbol{\lambda}_i(t)\| \cdot \|\dot{\boldsymbol{p}}_i(t)\| = 0, \ \forall i \in \mathrm{I}, \ t \in \mathcal{T}, \tag{3.45a}$$

where $\boldsymbol{\lambda}_i$ denotes the contact force at contact point $\boldsymbol{p}_i$ and $\dot{\boldsymbol{p}}_i$ the respective velocity of the contact point.

In order to define a mixed-integer problem and because the contact forces are not part of the LIPM formulation, we can replace the contact forces by a binary contact indicator $\alpha_i \in \{0, 1\}$ and Equation (3.45) becomes

$$\alpha_i(t)\|\dot{\boldsymbol{p}}_i(t)\| = 0, \ \forall i \in \mathrm{I}, \ t \in \mathcal{T}. \tag{3.46a}$$

The contact indicator itself is defined by

$$\alpha_i(t) = \begin{cases} 1, \text{ if } i\text{th contact is established} \\ 0, \text{ else} \end{cases} . \tag{3.47}$$

In accordance with the discretization of the CoM dynamics, we discretize the contact position $\boldsymbol{p}_i(t) \in \mathbb{R}^2$ on the contact surface along the tangential axes $\nu \in \{x, y\}$ on the horizon $\mathcal{T} \in \mathbb{N}$ by means of piecewise constant controls defined on n intervals $\mathcal{T}_j$, $j = 0, \ldots, n-1$, such that

$$p_i^\nu(t) \equiv \text{const.}, \ t \in \mathcal{T}_j, \ j \in 0, 1, \ldots, n-1, \ i \in \mathrm{I}, \tag{3.48}$$

where we define for the ith contact point the position along axis ν the discretized trajectory over the complete time horizon $\mathcal{T}$ by

$$\boldsymbol{P}_{i,k}^{\nu} = \left(p_{i,k+1}^{\nu}, p_{i,k+2}^{\nu}, \dots, p_{i,k+n}^{\nu}\right), \tag{3.49}$$

and the current position of this contact point is given by $p_{i,k}^{\nu}$. In this way, the contact point velocity is not explicitly defined but given by a finite difference scheme, i.e., $\dot{p}_{i,k}^{\nu} := \frac{p_{i,k}^{\nu}-p_{i,k-1}^{\nu}}{t_k-t_{k-1}}$.

Analogously, we discretize the binary contact indicator α_i on the horizon $\mathcal{T}$ by piecewise constant controls and define for contact point i

$$\boldsymbol{A}_{i,k} = \left(\alpha_{i,k+1}, \alpha_{i,k+2}, \dots, \alpha_{i,k+n}\right), \tag{3.50}$$

where the current contact configuration is then given by $\alpha_{i,k}$.

By limiting the contact set to only two foot contacts $\mathcal{F} = \{\mathrm{L}, \mathrm{R}\}$ for the left and right foot center f_{L}, f_{R} respectively, as described in Section 3.2.2, the walking motion can then emerge from enforcing at least one foot contact to be established by

$$1 \leqslant \sum_{i \in \mathcal{F}} \alpha_{i,k} \leqslant 2, \; k = 1, \dots, n, \; i \in \mathcal{F}. \tag{3.51}$$

A proper definition of the walking task for the above chosen discretization requires additional constraints to be defined. First, if the contact indicator α_i is active in two consecutive time intervals, then the respective contact position is not allowed to change, i.e.,

$$\alpha_{i,k-1} + \alpha_{i,k} = 2 \Rightarrow f_{i,k-1}^{\nu} = f_{i,k}^{\nu}, \; i \in \{\mathrm{L}, \mathrm{R}\}, \; \nu \in \{x, y\}, \; k = 1, \dots, n-1. \tag{3.52}$$

Additionally, always one foot has to be fixed to the ground, this is given by

$$\alpha_{\mathrm{L},k} + \alpha_{\mathrm{R},k} \geq 1, \; k = 0, 1, \dots, n-1. \tag{3.53}$$

In order to prevent an immediate change of the current support foot, we define

$$\alpha_{i,k} = 0 \Rightarrow \alpha_{j,k+1} = 1, \; i, j \in \{\mathrm{L}, \mathrm{R}\}, \; i \neq j. \tag{3.54}$$

Zero-Moment Point Constraints

According to the stability criterion given in Definition 3.9, we define ZMP constraints such that the ZMP stays in the respective support polygon as visualized in Figure 3.7.

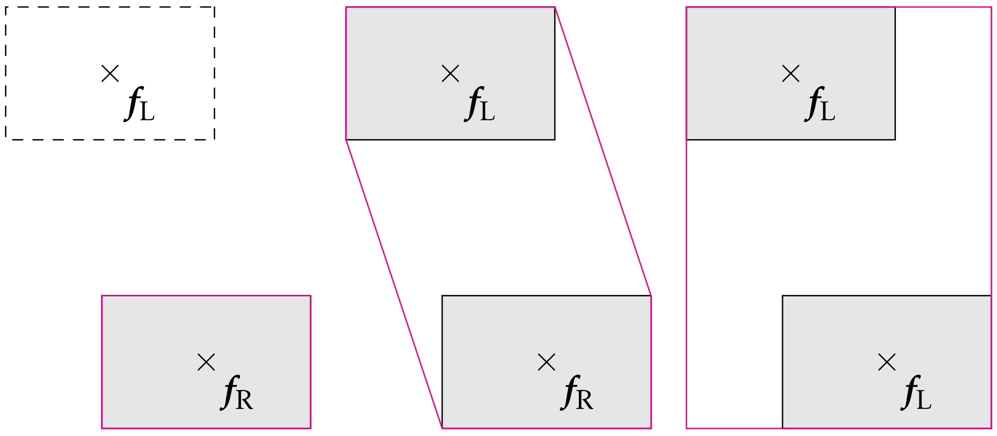
For the single support phase, c.f. Figure 3.7a, we formulate the corresponding polygon of support for foot p_j^{ν}, $\nu \in \{x, y\}$ and $j \in \{\mathrm{L}, \mathrm{R}\}$ by

$$(\boldsymbol{P}_{zu}\boldsymbol{U}_k^{\nu})_i - \frac{1}{2} f_{j,k+i}^{\nu} \leq \frac{1}{2}\mathrm{f} - (\boldsymbol{P}_{zs} c_k^{\nu})_i, \tag{3.55a}$$

$$-(\boldsymbol{P}_{zu}\boldsymbol{U}_k^{\nu})_i + \frac{1}{2} f_{j,k+i}^{\nu} \leq \frac{1}{2}\mathrm{f} + (\boldsymbol{P}_{zs} c_k^{\nu})_i, \tag{3.55b}$$

where f is the foot length for $\nu = x$ or the foot width for $\nu = y$ respectively. The constraints of (3.55) are then only active, when the respective foot is also support foot, i.e., $\alpha_{\mathrm{L}} = 1$ or $\alpha_{\mathrm{R}} = 1$.

While it is straightforward to define the polygon of support for the single support phase, for the double support phase, c.f. Figure 3.7b, this would result in nonlinear



(a) Single support phase (b) Double support phase (c) Support over estimation

Figure 3.7: Different support situations for a walking humanoid robot with feet center positions as crosses for left and right foot $f_L, f_R \in \mathbb{R}^2$. Feet placed on the ground are indicated by black boxes filled in gray, while feet not in contact with the ground are indicated by dashed black lines. The support polygon for the current configuration is indicated by lines in magenta.

constraints. Following this, we chose to use an over estimation of the support polygon as depicted in Figure 3.7c. In order to do so, we introduce the distance between both foot positions $d^{\nu}_{\mathrm{LR}} = \|\boldsymbol{f}^{\nu}_{\mathrm{L}} - \boldsymbol{f}^{\nu}_{\mathrm{R}}\|_1$ in L^1-norm. To be able to formulate a MIQP problem later on, we relax L^1-norm by introducing additional variables $d^{\mathrm{LR},x}$ for the double support phase ($\alpha_{\mathrm{L}} = \alpha_{\mathrm{R}} = 1$). We discretize d^{ν}_{LR} on the horizon $\mathcal{T}$ by a piecewise constant function and define their evolution on the horizon by

$$\boldsymbol{D}^{\nu}_{\mathrm{LR},k} = \left(d^{\nu}_{\mathrm{LR},k+1}, d^{\nu}_{\mathrm{LR},k+2}, \ldots, d^{\nu}_{\mathrm{LR},k+n'}\right). \tag{3.56}$$

We then formulate the following constraints for $\nu \in \{x, y\}$

$$d^{\nu}_{\mathrm{LR},k+i} \geq +(f^{\nu}_{\mathrm{L}} - f^{\nu}_{\mathrm{R}}) \tag{3.57a}$$
$$d^{\nu}_{\mathrm{LR},k+i} \geq -(f^{\nu}_{\mathrm{L}} - f^{\nu}_{\mathrm{R}}). \tag{3.57b}$$

The relaxation is complete when we additionally augment the objective such that $d^{lr,x}$ and $d^{lr,y}$ are minimized which is achieved by adding the term

$$j_6 = (\boldsymbol{D}^{x}_{\mathrm{LR},k+1})^T \boldsymbol{D}^{x}_{\mathrm{LR}} + (\boldsymbol{D}^{y}_{\mathrm{LR},k+1})^T \boldsymbol{D}^{y}_{\mathrm{LR}}. \tag{3.58}$$

The absolute value of the distance between both feet enables us to define the ZMP constraints for $\nu \in \{x, y\}$ by

$$z^{\nu}_{k+1} \leq \frac{1}{2}(f^{\nu}_{\mathrm{L}} + f^{\nu}_{\mathrm{R}}) + \frac{1}{2}d^{lr,\nu} + \frac{1}{2}\mathrm{f}, \tag{3.59a}$$
$$z^{\nu}_{k+1} \geq \frac{1}{2}(f^{\nu}_{\mathrm{L}} + f^{\nu}_{\mathrm{R}}) - \frac{1}{2}d^{lr,\nu} - \frac{1}{2}\mathrm{f}, \tag{3.59b}$$

where $z^{\nu}_{k+i} = (\boldsymbol{P}_{zs} c^{\nu}_k + \boldsymbol{P}_{zu} \boldsymbol{U}^{\nu}_k)_i$ (3.59) and by $(\cdot)_i$ we denote the ith column of the respective vector quantity.

Maximum Swing Foot Velocity

In order to guarantee the physical consistency of the velocity of the swing foot, we introduce constraints such that the foot motion is limited per time step. Analogously to Section 3.3.1, we introduce additional variables d_{L}^{ν}, $\nu \in \{x,y\}$ that model the traveled distance of a foot by

$$d_{\mathrm{L}}^{\nu} = \|f_{\mathrm{L},k}^{\nu} - f_{\mathrm{L},k+1}^{\nu}\|_1. \tag{3.60}$$

The respective vector quantities are denoted by $\boldsymbol{D}_{\mathrm{i}}^{\nu}$ for $\nu \in \{x,y\}$ and $i \in \{\mathrm{L},\mathrm{R}\}$ and minimized in the objective function in the form of

$$j_7 = (\boldsymbol{D}_{\mathrm{L},k+1}^{x})^T \boldsymbol{D}_{\mathrm{L}}^{x} + (\boldsymbol{D}_{\mathrm{L},k+1}^{y})^T \boldsymbol{D}_{\mathrm{L}}^{y} + (\boldsymbol{D}_{\mathrm{R},k+1}^{x})^T \boldsymbol{D}_{\mathrm{R}}^{x} + (\boldsymbol{D}_{\mathrm{R},k+1}^{y})^T \boldsymbol{D}_{\mathrm{R}}^{y}. \tag{3.61}$$

This formulation then yields the following linear constraints

$$d_{\mathrm{L},k}^{\nu} \geq +(f_{\mathrm{L},k}^{x} - f_{\mathrm{L},k+1}^{\nu}), \tag{3.62}$$
$$d_{\mathrm{L},k}^{\nu} \geq -(f_{\mathrm{L},k}^{x} - f_{\mathrm{L},k+1}^{\nu}). \tag{3.63}$$

Now we can simply use these substituted absolute differences $d^{l,x}$ to limit the step length per time interval by

$$\sqrt{(d_{\mathrm{L},k}^{x})^2 + d_{\mathrm{L},k}^{y})^2} \leqslant d_{\mathrm{L},k}^{x} + d_{\mathrm{L},k}^{y} \leq L_{step} = v_{max} \cdot \Delta t, \tag{3.64}$$

where v_{max} denotes the maximum swing foot velocity, Δt the time interval and L_{step} the resulting maximum step length per time interval. We can then define the constraints in linear form by

$$d_{\mathrm{L},k}^{x} + d_{\mathrm{L},k}^{y} \leq L_{step}, \tag{3.65}$$
$$d_{\mathrm{R},k}^{x} + d_{\mathrm{R},k}^{y} \leq L_{step}. \tag{3.66}$$

Here, we use $v_{max} = 0.5\,\mathrm{m\,s^{-1}}$ or $L_{step} = 0.05\,\mathrm{m}$.

Foot Step Feasibility Constraints

The foot position constraints can be equivalently formulated as presented in Section 3.2.3. In this way, we want the foot center positions $\boldsymbol{p}_{\mathrm{L}}$, $\boldsymbol{p}_{\mathrm{R}}$ to lie within the foot position feasible area, which can be given by Figure 3.6 for example, such that Equation 3.35 has to hold. However, we have to take care that the respective constraint is only enforced on the current swing foot motion. This is achieved by defining a vanishing constraint using the respective support foot indicator α_j for $j \in \{\mathrm{L},\mathrm{R}\}$ i.e.,

$$0 \leqslant (1-\alpha_j) \cdot (\boldsymbol{A}_{foot_j}(\boldsymbol{f}_{j,k+1} - \boldsymbol{f}_{j,k}) - \boldsymbol{b}_{foot_j}). \tag{3.67}$$

Here, we drop the dependence on the orientation of the feet, because we do not consider the foot orientations in this formulation.

3.3.2 Obstacle Avoidance

In contrast to our approach presented in Section 3.2, where we define convex regions that shall not be entered by the robot, here, we chose to plan the motion directly in the obstacle free space of a scene cluttered by obstacles. This can efficiently be done by finding a covering set of convex regions of the corresponding scene. The WPG formulation then only plans footsteps in these convex regions, by formulating them together with indicator variables as vanishing constraints in the MIQP formulation. Furthermore, the approach enables not only the avoiding of obstacles in the path of the robot but also to include foot holds into the planning. Additionally, we enhance the *reactivity* of the collision avoidance strategy by deflecting the reference velocity on the obstacle to find a guiding path through cluttered terrain.

First, we recall our assumptions on the scene description, i.e., how we assume that obstacles are recognized and describe the format that is required for the actual avoidance algorithms. Second, we present the *IRIS* algorithm that is heavily used by our strategy to find obstacle free areas in a scene. Third, we introduce our algorithm of *Obstacle Free Area Generation* in detail. Finally, we conclude the section with a detailed description of the reference velocity deflection algorithm.

Scene and Obstacle Description Here, we assume the robot to be part of a cluttered environment such that obstacles hinder the robot to find a suitable path towards a goal. In order to identify these obstacles, we can apply stereo vision [45, 82] , from a depth sensor [20] or LIDAR technology [108] to receive a point cloud or depth map from the current scene.

From a given depth map, we assume that obstacles surrounding the robot in the current scene can be identified and sufficiently approximated by means of a set of polytopes $\mathcal{O}$, i.e., all convex regions that are not accessible by the robot and are to be avoided, given by

$$\mathcal{O} = \bigcup_{i \in \mathcal{O}} \mathcal{O}_i, \tag{3.68a}$$

$$\mathcal{O}_i = \left\{ \boldsymbol{x} \in \mathbb{R}^2 | \boldsymbol{A}_i \boldsymbol{x} \leqslant \boldsymbol{b}_i \right\}, i \in \mathcal{O}, \tag{3.68b}$$

where we are only interested in the projection of the shapes onto level ground, i.e., the x, y-plane. We refer to $\mathcal{O}$ as both an index set as well a an description of the actual convex shapes.

IRIS Algorithm From a given set of polytopes describing obstacles in a scene, the goal of our obstacle avoidance strategy is to identify large convex OFAs. In [34], *Deits* presents the *IRIS* algorithm, an abbreviation for Iterative Regional Inflation by Semi-Definite Programming (SDP). The algorithm is able to find iteratively polytopic obstacle free areas in both 2D and 3D. Given a number of obstacles $\mathcal{O}$ and a seed point $\boldsymbol{s} \in \mathbb{R}^2$, the method $IRIS.InflateRegion(\boldsymbol{s},\mathcal{O})$ successively performs two steps: 1) finding separating hyperplanes by solving a convex optimization problem and 2) finding an inscribed ellipsoid by means of semi-definite programming. The hyperplanes separate the interior of the convex region, in which the seed point lies, from all adjacent obstacles. In this way, two mathematical descriptions of the obstacle free area are returned in the form of the polytope, where the respective set of hyperplanes is returned as a matrix and a vector, as well as the inscribed ellipsoid described by the respective positive definite and symmetric matrix and an offset vector.

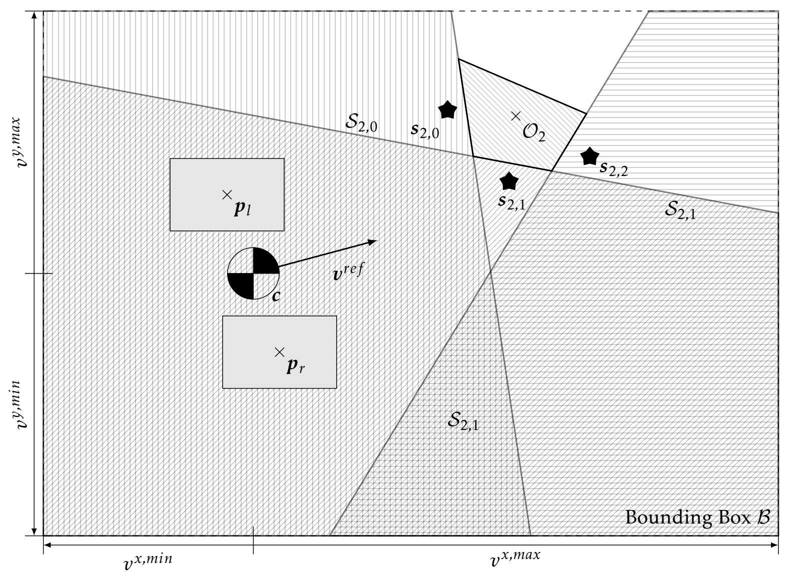


Figure 3.8: A visualization of the OFA search algorithm. From the current CoM position $\boldsymbol{c}$ a box $\mathcal{B}$ is defined from the maximal reachable space given by maximum and minimum velocities $\boldsymbol{v}^{min}, \boldsymbol{v}^{max}$. Only the obstacles $\mathcal{O}$ inside the bounding box are considered. For each obstacle $\mathcal{O}_i \in \mathcal{O}$, from the seed points of the obstacle, denoted by $\boldsymbol{s}_{i,\cdot}$ and visualized as black asterisks, the largest obstacle-free convex region is computed by applying the *IRIS* algorithm. The obstacle-free convex areas $\mathcal{S}_{i,\cdot}$ are then returned for the planning.

An implementation of the *IRIS* algorithm can be found here [3]. The convex optimization problems are solved by *CVXGEN*[4], c.f. [119]. The semi-definite optimization problem is solved using the *MOSEK* optimization software, c.f. [170].

Searching for Obstacle-Free Areas The idea of searching for OFAs is depicted in Figure 3.8. We assume that a description of all obstacles in a scene is given by the set $\mathcal{O}$. Given the current CoM position $\boldsymbol{c}$, we can limit the consideration of obstacles to the maximal reachable space of the robot $\mathcal{B}$, which is defined by its maximum and minimum possible velocities $\boldsymbol{v}^{min}, \boldsymbol{v}^{max} \in \mathbb{R}^2$. Here, $\boldsymbol{v}^{\nu,max}, \boldsymbol{v}^{\nu,max}$ for $\nu \in \{x, y\}$ can be computed from the time discretization together with the maximum swing foot velocity $\boldsymbol{v}^{swing,max}$, given by

$$\boldsymbol{v}^{swing} = \Delta t \cdot n \cdot \boldsymbol{v}^{swing,max}, \tag{3.69}$$

$$v^{\nu,max} = \max_j\{\boldsymbol{v}_j^\nu\} + \boldsymbol{v}^{swing}, \tag{3.70}$$

$$v^{\nu,min} = \min_j\{\boldsymbol{v}_j^\nu\} + \boldsymbol{v}^{swing}, \tag{3.71}$$

where $\boldsymbol{v}_j$ are the nodes of the valid foot position polygon. The bounding box $\mathcal{B}$ is then defined relative to the CoM position $\boldsymbol{c}$ as depicted in Figure 3.8, i.e., for $\nu \in \{x, y\}$

$$\max_\nu(\mathcal{B}) = c^\nu + v^{\nu,max}, \tag{3.72}$$

$$\min_\nu(\mathcal{B}) = c^\nu + v^{\nu,min}, \tag{3.73}$$

[3] https://github.com/rdeits/iris-distro [4] https://cvxgen.com/docs/index.html

Algorithm 3.1: Obstacle-Free Area Search

```
  input  : $\boldsymbol{c} \in \mathbb{R}^2$, $\boldsymbol{v}^{min}, \boldsymbol{v}^{max} \in \mathbb{R}^2$, $\mathcal{O}$, $\mathcal{B}$
  output : $\mathcal{S}$
1 $\boldsymbol{H} = \boldsymbol{0}$
2 $\mathcal{B} = ComputeBoundingBox(\boldsymbol{c}, \boldsymbol{v}^{min}, \boldsymbol{v}^{max})$
3 $\tilde{\mathcal{O}} = \mathcal{O} \cap \mathcal{B}$
4 for $i = 1, \dots, |\tilde{\mathcal{O}}|$ do
5     for $j = 1, \dots, EdgesOf(\mathcal{O}_i)$ do
6         $\mathcal{S}_{i,j} = IRIS.InflateRegion(\boldsymbol{s}_{i,j}, \mathcal{B}, \mathcal{O})$
7         $\mathcal{S} = \mathcal{S} \cup \left\{\mathcal{S}_{i,j}\right\}$
8     end
9 end
```

where the minimum velocity $v^{v,min}$ can also be negative, except the robot is not able to walk backwards. We then consider only the obstacles $\tilde{\mathcal{O}} = \mathcal{O} \cap \mathcal{B}$ that lie inside the bounding box $\mathcal{B}$.

In order to find all OFAs inside the bounding box, for each obstacle $\mathcal{O}_i \in \tilde{\mathcal{O}}$ and for each of the seed points $\boldsymbol{s}_{i,j}$ of the ith obstacle, we compute the largest obstacle-free convex region by applying the *IRIS* algorithm presented above. The seed points are located at the center of each edge of the ith obstacle as depicted in Figure 3.8. The resulting obstacle-free convex areas $\mathcal{S}_{i,j}$ are then returned for the planning in the form of $\boldsymbol{A}_{i,j}, \boldsymbol{b}_{i,j}$ according to their mathematical description (3.68b).

Introducing binary variables $\theta^o_{l,k+i} \in \{0,1\}$ that indicates whether the respective foot $\boldsymbol{f}_l$ is in obstacle-free are $\mathcal{S}_i$ at timestep $k+i$. We denote all OFAs on the whole prediction horizon by $\boldsymbol{\mathcal{S}}$ and for each time interval by $\boldsymbol{\mathcal{S}}_i$ for $k = 0, \dots, n-1$. The respective variables $\theta^o_{l,k+i}$ per time step i and foot l are accumulated in the vectors $\boldsymbol{T}_{l,i}$ for all o in $\boldsymbol{\mathcal{S}}_i$.

Furthermore, to ensure that a support foot (i.e. respecting $\alpha = 1$) is always in exactly a single OFA, we introduce the constraint

$$\sum_{o \in \boldsymbol{\mathcal{S}}_i} \theta^o_{l,k+i} + (1 - \alpha^l_{k+i}) = 1, \tag{3.74}$$

together with the constraint for each time step $i = 0, \dots, n-1$, each support foot $l \in$ L, R and for all OFAs $o \in \boldsymbol{\mathcal{S}}_i$ at the current time step, that is given by

$$0 \leqslant (1 - \theta^o_{l,k+i}) \cdot (\boldsymbol{A}_o \boldsymbol{f}_{l,k+i} - \boldsymbol{b}_o). \tag{3.75}$$

Algorithm 3.1 presents a version of the respective algorithm that enables the identification of obstacle free areas in a scene in pseudo code.

Modifying the Reference Velocity In addition to the guaranteed collision-free motion that arises from planning in the above presented obstacle-free space of the current scene, we found that the WPG requires additional guidance around obstacle to avoid getting stuck at unluckily placed obstacles. Therefore, the basic idea is to augment the tracking of the original reference velocity $\boldsymbol{v}^{ref}$ by an additional term $\boldsymbol{v}_U$, which is tangential to a radial potential $\mathcal{U}$ around the obstacle $\mathcal{O}$ as depicted in Figure 3.9.

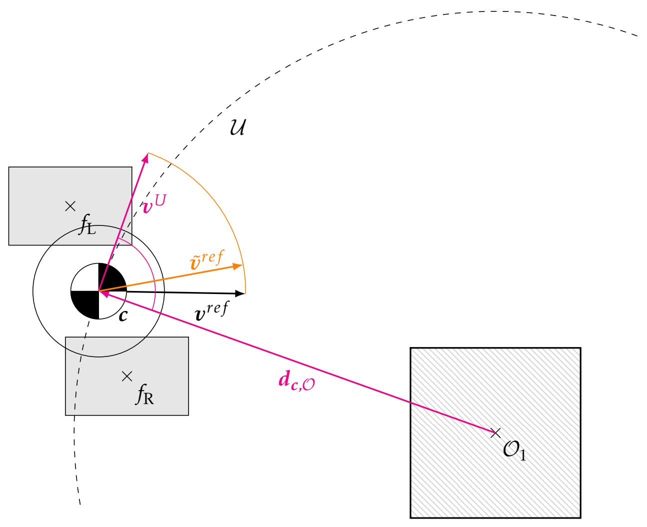


Figure 3.9: A visualization of the reference velocity deflection at obstacles. Instead of tracking the original reference velocity $\boldsymbol{v}^{ref}$, we augment the respective reference velocity by an additional term $\boldsymbol{v}^U$ that is tangential to a radial potential $\mathcal{U}$ around the obstacle $\mathcal{O}$. The new reference velocity $\tilde{\boldsymbol{v}}^{ref}$ is then a linear combination of $\boldsymbol{v}^{ref}$ and $\boldsymbol{v}^U$ dependent on the distance between CoM and the obstacle. This implements guidance around obstacle for the robot depicted as foot steps f_{L}, f_{R} and its center of mass $\boldsymbol{c}$.

In order to derive the deflection, we briefly recap our assumptions. The deflection shall be based on the distance of the CoM to the obstacle $\mathcal{O}$, given by

$$\mathrm{dist}(\boldsymbol{c},\mathcal{O}) = \|\boldsymbol{d}_{\boldsymbol{c},\mathcal{O}}\|_2^2, \tag{3.76}$$

$$\boldsymbol{d}_{\boldsymbol{c},\mathcal{O}} \approx \boldsymbol{o} - \boldsymbol{c}, \tag{3.77}$$

where we are satisfied with an approximation of the actual distance to the obstacle by the distance to its center $\boldsymbol{o} \in \mathbb{R}^2$, which is slightly larger but negligible.

In order to define a velocity term $\boldsymbol{v}_U$, which is tangential to the contour lines of the radial potential field $\mathcal{U}$, it must hold

$$\boldsymbol{v}^U \perp \boldsymbol{d}_{\boldsymbol{c},\mathcal{O}}, \tag{3.78}$$

such that we can derive from (3.78)

$$\boldsymbol{v}^U = \rho \boldsymbol{d}_\perp, \tag{3.79}$$

$$\boldsymbol{d}_\perp = \begin{bmatrix} d^y_{c,\mathcal{O}} \\ -d^x_{c,\mathcal{O}} \end{bmatrix}, \tag{3.80}$$

where ρ is a scaling factor equalizing the length of $\boldsymbol{v}^U$, such that $\boldsymbol{v}^U = \boldsymbol{v}^{ref}$, i.e.,

$$|\rho| = \frac{\|\boldsymbol{v}^{ref}\|_2}{\|\boldsymbol{d}_{\boldsymbol{c},\mathcal{O}}\|_2}. \tag{3.81}$$

Here, the sign of ρ additionally depends on the sign of the angle between $\boldsymbol{d}_{\boldsymbol{c},\mathcal{O}}$ and $\boldsymbol{v}^{ref}$, which is to be chosen such that $\measuredangle(\boldsymbol{v}^{ref}, \boldsymbol{v}^U) < \frac{\pi}{2}$ with

$$\measuredangle(\boldsymbol{v}^{ref}, \boldsymbol{v}^U) = \arccos \frac{(\boldsymbol{v}^{ref})^T \boldsymbol{v}^U}{\|\boldsymbol{v}^{ref}\| \, \|\boldsymbol{v}^U\|}. \tag{3.82}$$

This is achieved by the heuristic

$$\operatorname{sign}(\rho) = \begin{cases} 1, & \text{if } (\boldsymbol{v}^{ref})^T \boldsymbol{d}_\perp > 0, \\ -1, & \text{else.} \end{cases} \tag{3.83}$$

The computational costs for this sign switching are negligible such that the efficiency does not suffer from guessing.

In order to compute an augmented reference velocity $\tilde{\boldsymbol{v}}^{ref}$, we chose to blend the original reference velocity by means of a linear combination in the form of

$$\tilde{\boldsymbol{v}}^{ref} = (1-\kappa)\boldsymbol{v}^{ref} + \kappa \boldsymbol{v}^U, \tag{3.84}$$

where $\kappa \in [0,1]$ measures the distance between the CoM $\boldsymbol{c}$ and the obstacle $\mathcal{O}$ in the form of exponential decay, i.e.,

$$\kappa = \exp(-\frac{\|\boldsymbol{d}_{\boldsymbol{c},\mathcal{O}}\|_2^p}{s_{\boldsymbol{c},\mathcal{O}}}), \tag{3.85}$$

where $p \in \mathbb{N}$ controls the slope of the radial potential $\mathcal{U}$ and $s_{\boldsymbol{c},\mathcal{O}} \geqslant 0$ denotes a user-set safety margin that is given for $d \geqslant 0$ and $k \in [0,1]$ by

$$s_{\boldsymbol{c},\mathcal{O}} = -\frac{d}{\ln(k)}, \tag{3.86}$$

where for example one receives for a security margin of $d = 0.1\,\text{m}$ and a decay rate of $k = 0.1$ a value of $s_{\boldsymbol{c},\mathcal{O}} = 0.043\,\text{m}$ for the abstract security margin.

Remark (Properties of the Reference Velocity Augmentation) An important property of the computation above is that the norm of the augmented reference velocity $\tilde{\boldsymbol{v}}^{ref}$ does never exceed the norm of the original term $\boldsymbol{v}^{ref}$, i.e.,

$$\begin{aligned} \|\tilde{\boldsymbol{v}}^{ref}\| &= \|(1-\kappa)\boldsymbol{v}^{ref} + \kappa \boldsymbol{v}^U\| \\ &\leq (1-\kappa)\|\boldsymbol{v}^{ref}\| + \kappa\|\boldsymbol{v}^U\| \\ &= (1-\kappa)\|\boldsymbol{v}^{ref}\| + \kappa\|\boldsymbol{v}^{ref}\| \\ &= \|\boldsymbol{v}^{ref}\|. \end{aligned}$$

This has the advantage that in the worst case the robot gets slower close to the obstacle.

Remark (Fixing the Avoidance Direction) In the case that the robot walks directly towards the center of the obstacle, the above formulation leads to an oscillating of the avoidance direction. Following this, the robot will not avoid the obstacle at all and gets slower till he stops.

In order to prevent this, we fix the direction of avoidance from the above formulation if the respective center of the obstacle $\boldsymbol{o}$ is in a certain area Ξ of the walking direction $\boldsymbol{v}^{ref}$, given by

$$(\tilde{\boldsymbol{v}}_\perp^{ref})^T \boldsymbol{o} \leq (\tilde{\boldsymbol{v}}_\perp^{ref})^T (\boldsymbol{p} + \xi \tilde{\boldsymbol{v}}_\perp^{ref}), \tag{3.87a}$$

$$-(\tilde{\boldsymbol{v}}_\perp^{ref})^T \boldsymbol{o} \leq -(\tilde{\boldsymbol{v}}_\perp^{ref})^T (\boldsymbol{p} - \xi \tilde{\boldsymbol{v}}_\perp^{ref}), \tag{3.87b}$$

where $\tilde{\boldsymbol{v}}_{\perp}^{ref}$ are the normalized orthogonal vectors of $\tilde{\boldsymbol{v}}^{ref}$. $\boldsymbol{o}$ and $\boldsymbol{p}$ are the respective center of the obstacle or the feet and ξ is a parameter to adjust the size of Ξ.

In the case that Equation (3.87) is satisfied, the robot will pass the obstacle on the right side per default. This strategy can be easily extended by considering either the velocity of an obstacle to efficiently evade it or the position of other obstacles in the current scene.

3.3.3 The Objective Function

The objective function for the MIQP as given by Definition 3.13 is defined as a weighted linear combination in the form of

$$\min_{\boldsymbol{U}_k} \omega_1 j_1(\boldsymbol{U}_k)+\omega_2 j_2(\boldsymbol{U}_k)+\omega_3 j_3(\boldsymbol{U}_k)+\omega_4 j_4(\boldsymbol{U}_k)+\omega_5 j_5(\boldsymbol{U}_k)+\omega_{6,7} j_6(\boldsymbol{U}_k)+\omega_{6,7} j_7(\boldsymbol{U}_k), \quad (3.88)$$

where $\omega_i \geqslant 0$ for $i = 1,2,3,4$ denote the weights of the respective terms of the objective function. The free variables are denoted by $\boldsymbol{U}_k$ and are given in the form of

$$\boldsymbol{U}_k^{x,y} = \begin{bmatrix} \boldsymbol{U}_k^{x,y} \\ \boldsymbol{U}_k^{d} \\ \boldsymbol{U}_k^{S} \end{bmatrix} \quad (3.89)$$

with

$$\boldsymbol{U}_k^{x,y} = \begin{bmatrix} \dddot{\boldsymbol{C}}_k^x \\ \boldsymbol{F}_k^x \\ \dddot{\boldsymbol{C}}_k^y \\ \boldsymbol{F}_k^y \\ \boldsymbol{A}_{\mathrm{L},k} \\ \boldsymbol{A}_{\mathrm{R},k} \end{bmatrix}, \qquad \boldsymbol{U}_k^{d} = \begin{bmatrix} \boldsymbol{D}_{\mathrm{LR},k}^x \\ \boldsymbol{D}_{\mathrm{LR},k}^y \\ \boldsymbol{D}_{\mathrm{L},k}^x \\ \boldsymbol{D}_{\mathrm{L},k}^y \\ \boldsymbol{D}_{\mathrm{R},k}^x \\ \boldsymbol{D}_{\mathrm{R},k}^y \end{bmatrix}, \qquad \boldsymbol{U}_k^{S} = \begin{bmatrix} \boldsymbol{T}_{\mathrm{L},1} \\ \vdots \\ \boldsymbol{T}_{\mathrm{L},n} \\ \boldsymbol{T}_{\mathrm{R},1} \\ \vdots \\ \boldsymbol{T}_{\mathrm{R},n} \end{bmatrix}, \quad (3.90)$$

where $\boldsymbol{U}_k^{x,y}$ denote the DoFs related to the CoM motion and the foot placement, where $\boldsymbol{U}_k^{d}$ consists of the variables used to compute l_1 approximation of certain quantities, and $\boldsymbol{U}_k^{S}$ denote the DoFs related to the OFAs from the collision avoidance strategy presented in Section 3.3.2.

The different objective terms are defined in the following. $j_1(\boldsymbol{U}_k)$ is the cost function term related to linear velocity tracking

$$j_1(\boldsymbol{U}_k) = \|\dot{\boldsymbol{C}}_{k+1}^x - \boldsymbol{v}_{k+1}^{x,ref}\|_2^2 + \|\dot{\boldsymbol{C}}_{k+1}^y - \boldsymbol{v}_{k+1}^{y,ref}\|_2^2,$$

where in this case the given reference velocity $\boldsymbol{v}_{k+1}^{ref}$ is altered according to the deflection strategy presented in Section 3.3.2

$j_2(\boldsymbol{U}_k)$ is the cost function term related to minimizing the distance between the ZMP and the projection of the ankle onto the sole

$$j_2(\boldsymbol{U}_k) = \|\boldsymbol{F}_{k+1}^x - \boldsymbol{Z}_{k+1}^x\|_2^2 + \|\boldsymbol{F}_{k+1}^y - \boldsymbol{Z}_{k+1}^y\|_2^2. \quad (3.91)$$

$j_3(\boldsymbol{U}_k)$ denotes the cost function term that centers the CoM to lie close between the foot position centers

$$j_3(\boldsymbol{U}_k) = \|\tfrac{1}{2}(\boldsymbol{F}_{\mathrm{L},k+1}^x + \boldsymbol{F}_{\mathrm{L},k+1}^y) - \boldsymbol{C}_{k+1}^x\|_2^2. \quad (3.92)$$

$j_4(\boldsymbol{U}_k)$ is the cost function term related to minimizing the applied CoM jerk

$$j_4(\boldsymbol{U}_k) = \|\dddot{\boldsymbol{C}}^x_{k+1}\|_2^2 + \|\dddot{\boldsymbol{C}}^y_{k+1}\|_2^2.$$

$j_5(\boldsymbol{U}_k)$ is the cost function term responsible for favor the single support phase over the double support phase with growing reference velocity $\boldsymbol{v}^{ref}$ given by

$$j_5(\boldsymbol{U}_k) = \|\boldsymbol{A}_{\mathrm{L},k}\|_2^2 + \|\boldsymbol{A}_{\mathrm{R},k}\|_2^2,$$

where the weight ω_5 is divided by the norm of the reference velocity $\boldsymbol{v}^{ref}$ as long as $\left\|\boldsymbol{v}^{ref}\right\| \geqslant \epsilon$ and else is chosen to be zero. In this way, single support phase is dominant when the norm of the reference velocity is positive, but when it is close to zero the algorithm favors to stay in double support phase.

In order to formulate the ZMP constraints by means of the disctance between both foot centers $\boldsymbol{f}_{\mathrm{L}}$, $\boldsymbol{f}_{\mathrm{R}}$, we added additional terms (3.58), (3.61) to the objective function that are weighted by a single weight $\omega_{6,7}$. Here, we implemented the respective objective functions $j_6(\boldsymbol{U}_k)$, $j_7(\boldsymbol{U}_k)$ in the form of

$$j_6(\boldsymbol{U}_k) = \|\boldsymbol{D}^x_{\mathrm{LR},k}\|_2^2 + \|\boldsymbol{D}^y_{\mathrm{LR},k}\|_2^2, \tag{3.93}$$

$$j_7(\boldsymbol{U}_k) = \|\boldsymbol{D}^x_{\mathrm{L},k}\|_2^2 + \|\boldsymbol{D}^y_{\mathrm{L},k}\|_2^2 + \|\boldsymbol{D}^x_{\mathrm{R},k}\|_2^2 + \|\boldsymbol{D}^y_{\mathrm{R},k}\|_2^2. \tag{3.94}$$

3.3.4 Mixed-integer Quadratic Programming Formulation

We strive to formulate the walking and obstacle avoidance problem as MIQP. A formal description is given by the following Definition.

Definition (Mixed–Integer Quadratic Program) A mixed-integer quadratic program is a constrained finite–dimensional optimization problem of the form

$$\min_{\boldsymbol{w}\in\mathbb{R}^{\mathrm{n}}} \quad \tfrac{1}{2}\boldsymbol{w}^T\boldsymbol{B}\boldsymbol{w} + \boldsymbol{w}^T\boldsymbol{b} \tag{3.95a}$$

$$\text{s.t.} \quad 0 = \boldsymbol{C}\boldsymbol{w} - \boldsymbol{c}, \tag{3.95b}$$

$$0 \leqslant \boldsymbol{w}, \tag{3.95c}$$

$$w_i \in \mathbb{Z}, i \in \mathcal{I}, |\mathcal{I}| \leqslant n, \tag{3.95d}$$

where $\boldsymbol{B} \in \mathbb{R}^{\mathrm{n}\times\mathrm{n}}$ is a positive definite, symmetric matrix with $\boldsymbol{b} \in \mathbb{R}^{\mathrm{n}}$, constraint matrix $\boldsymbol{C} \in \mathbb{R}^{\mathrm{m}\times\mathrm{n}}$ with $\boldsymbol{c} \in \mathbb{R}^{\mathrm{m}}$ and the $\mathcal{I}$ indicates a subset of the free optimization variables $\boldsymbol{w} \in \mathbb{R}^{\mathrm{n}}$. △

In order to do so, we introduced binary variables in the form of the DoFs $\boldsymbol{A}_{\mathrm{L},k}$, $\boldsymbol{A}_{\mathrm{R},k}$, $\boldsymbol{T}_{\mathrm{L},1}, \ldots, \boldsymbol{T}_{\mathrm{L},n}$, $\boldsymbol{T}_{\mathrm{R},1}, \ldots, \boldsymbol{T}_{\mathrm{R},n}$. These variables encode decisions on the foot placement and represent free variables for the optimizer. Depending on the state of these variables, i.e., being either 1 or 0, they can enable a constraint or disable it for the MIQP formulation. These special constraints are so-called complementarity and vanishing constraints and can be reformulated as linear constraints for the MIQP formulation of Definition 3.13 by using the big-M method. We briefly introduce this procedure before stating the problem formulation finally.

Big-M Formulation

The big-M method is a technique that can be used to reformulate a special form of complementarity and vanishing constraints into linear inequalities in the mixed-integer programming context. Here, we are interested in a special linear version

$$0 = \alpha \cdot \boldsymbol{w}, \tag{3.96a}$$
$$\alpha \in \{0,1\}, \tag{3.96b}$$
$$0 \leqslant \boldsymbol{w}, \tag{3.96c}$$
$$\boldsymbol{w} = \boldsymbol{A}\boldsymbol{v} - \boldsymbol{b}, \tag{3.96d}$$

where α is a binary variable indicating a mode of the system and the linear constraint given by matrix $\boldsymbol{A} \in \mathbb{R}^{\mathrm{m}\times\mathrm{n}}$ with full row rank and vectors $\boldsymbol{x} \in \mathbb{R}^{\mathrm{n}}$, $\boldsymbol{w}, \boldsymbol{b} \in \mathbb{R}^{\mathrm{m}}$.

In this special case, we apply the big-M method to reformulate the complementarity constraints (3.96) into linear constraints by chosing as sufficiently large $\mathrm{M} \geqslant 0$ and by reformulating (3.96a) into two constraints, i.e.,

$$0 \leqslant \boldsymbol{w} + (1-\alpha)\cdot \mathrm{M}, \tag{3.97a}$$
$$0 \leqslant (1-\alpha)\cdot \mathrm{M} - \boldsymbol{w}. \tag{3.97b}$$

In this way, for $\alpha = 0$, the big-M formulation basically renders the linear constraint, given by $\boldsymbol{w}$, to be always fulfilled. In contrast to this, for $\alpha = 1$, the big-M formulation renders the linear constraint to be an equality constraint again.

For vanishing constraints of the form

$$0 \leqslant \alpha \cdot \boldsymbol{w}, \tag{3.98a}$$
$$\alpha \in \{0,1\}, \tag{3.98b}$$
$$\boldsymbol{w} = \boldsymbol{A}\boldsymbol{v} - \boldsymbol{b}, \tag{3.98c}$$

the constraint (3.98a) can be reformulated by means of the big-M method by a single constraint given by

$$0 \leqslant \boldsymbol{w} + (1-\alpha)\cdot \mathrm{M}. \tag{3.99}$$

Remark The tightness of the relaxation strongly depends on the choice of M. Even if one applies the strongest possible choice, it can still be quite weak. The big advantage of the method is its ease of use and the possibility to linearly relax nonlinear constraints in this case without increasing the problem much.

In this way, we receive the following problem formulation.

$$\min_{\boldsymbol{U}_k} \quad \frac{1}{2}\|\boldsymbol{\ell}(\boldsymbol{U}_k)\|_2^2 \tag{3.100a}$$
$$\text{s.t.} \quad \underline{\boldsymbol{d}} \leqslant \boldsymbol{d}(\boldsymbol{U}_k) \leqslant \overline{\boldsymbol{d}}. \tag{3.100b}$$

Problem Formulation

In Section 3.3.3, the binary variables are already introduced and distinguished from the real-valued counterparts such that the integrality according to (3.95d) is clear and the free optimization variables are given.

Following Definition 3.13, the objective function has to be of quadratic form. As presented in Section 3.3.3, all objectives are already given as quadratic terms. In order to formulate them in canonical form, the quantities like $\boldsymbol{C}_{k+1}(\boldsymbol{U}_k)$ have to be resolved by

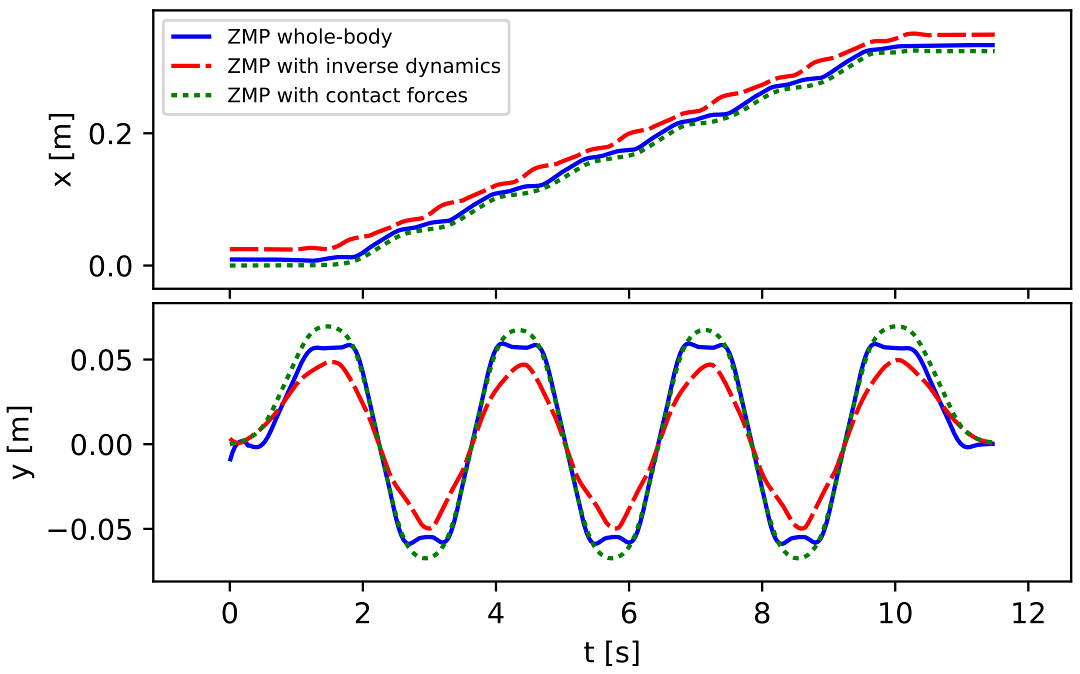


Figure 3.10: Trajectories of different zero-moment point approximations. The whole-body ZMP in solid blue according to Definition 3.8, the approximation of the ZMP using inverse dynamics in dashed red, and the ZMP approximation using forward dynamics with contact forces from [121].

means of the matrix quantities $\boldsymbol{P}_{ps}$, $\boldsymbol{P}_{pu}$ and the initial value $\boldsymbol{c}_k$. The objective terms are mostly already derived in this form in [71] or in Section 3.2.3 or already in the respective form, such that we skip the in detail derivation but refer the reader to the respective sections or references for details.

Considering the linear (3.95b) and box constraints (3.95c), we see that most of the constraints are already given in the correct form, i.e., (3.51), (3.52), (3.53), (3.54), (3.65), (3.75). Whereas some of them, i.e., (3.46), (3.55), (3.59), (3.67), (3.74) , require a reformulation according to the big-M method. Following this, the setup of each MIQP is straightforward.

3.4 Realization of the Motion on the Robot

In Figure 3.2, we visualize the work flow for motion generation using WPGs based on CoM dynamics. Here, we describe the last two building blocks required in order to realize the motions computed by the WPGs. This involves, on the one hand, the compensation of model-plant mismatch between the simplified model of the LIPM and the whole-body dynamics of the full robot in the form of a DF. On the other hand, WPGs compute feet and CoM trajectories only and therefore have to be post-processed to be realized on the robot. This can be either realized by pure inverse kinematics or a generalized inverse kinematics framework allowing for additional stabilizing control on the robot.

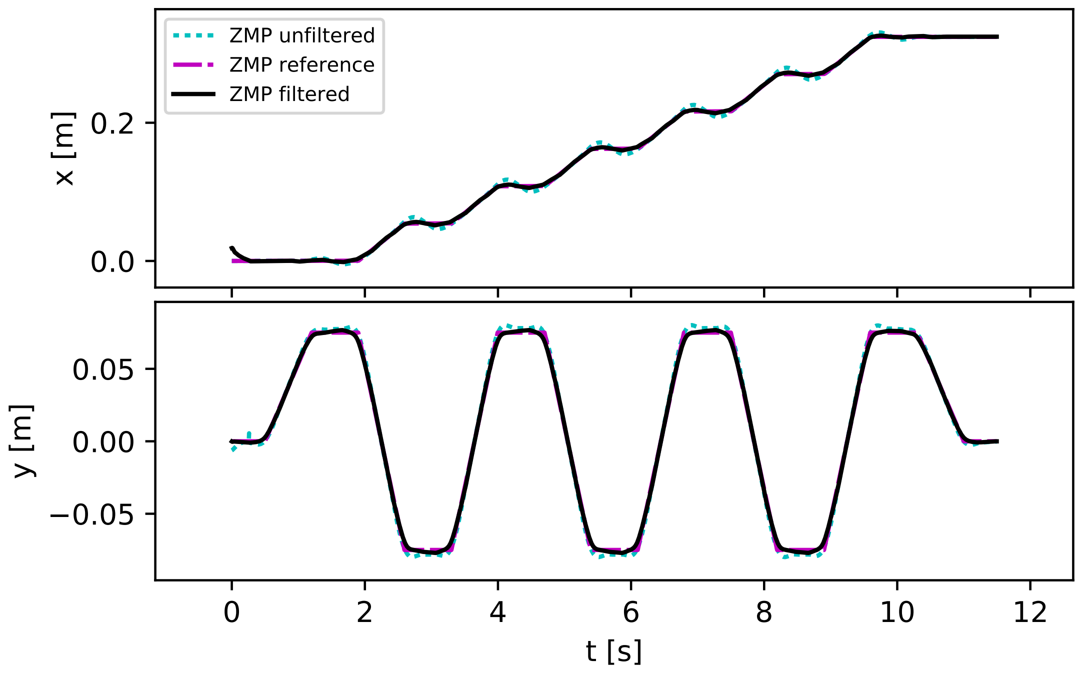


Figure 3.11: Results of the dynamic filter on the zero-moment point trajectories of *HeiCub*. The unfiltered whole-body ZMP trajectory in dotted teal, the ZMP reference in dashed magenta, and the resulting filtered ZMP in solid black.

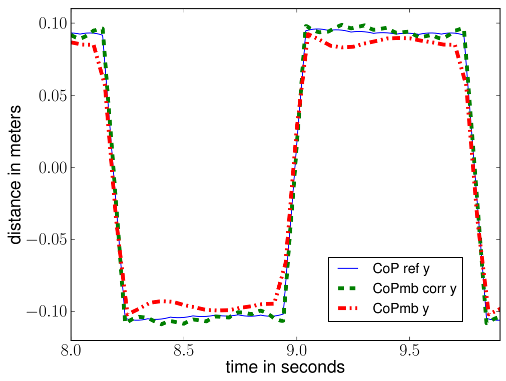


Figure 3.12: Results of the dynamic filter applied on the zero-moment point trajectories of *HRP-2* from our article [133]. In solid blue, the reference ZMP computed by the solver. In dash-dot-dot red, the ZMP multi-body. In dashed green, the ZMP multi-body recomputed after correction.

Algorithm 3.2: Dynamic Filter

```
input  : $c, \dot{c}, \ddot{c}, f, \dot{f}, \ddot{f}, z$
output : $\tilde{c}$
1 $q, \dot{q}, \ddot{q}$ = InverseKinematicsWithDerivatives($c, \dot{c}, \ddot{c}, f, \dot{f}, \ddot{f}$)
2 $z^\star$ = ComputeZeroMomentPoint($q, \dot{q}, \ddot{q}$)
3 $\Delta z = z^\star - z$
4 $\Delta\tilde{c}$ = PreviewControl($\Delta z, f$)
5 $\tilde{c} = c + \Delta\tilde{c}$
```

3.4.1 Dynamic Filter

The reduction of the dynamics to a simple model like the LIPM in the WPG makes the computed ZMP trajectories to be different from ones reconstructed from the joint trajectories finally realized on the robot, which can be either computed from a whole-body dynamic model or the resulting ZMP trajectories acquired from sensory information. Therefore, the implementation of the algorithm from [71] that was successfully implemented on the *HRP-2* in the *Japan Robotic Laboratory* (JRL), turned out to be unstable for its first test on another *HRP-2* robot located at CNRS-LAAS. An approach in order to cope with this difficulty was first introduced by *Kajita* [83] and referred to as the *dynamic filter*.

This filter aims to compensate this error on the ZMP level by adapting the computed CoM motion in order to minimize the error. The problem can be stated as minimization problem in the form of

$$\min_{c} \quad \frac{1}{2}\|\Delta z(c)\|_2^2, \tag{3.101}$$

where the error $\Delta z = z^\star - z$ is the difference between the *real* ZMP based on a dynamic model of the robot and the ZMP planned by the WPG. The filter as presented in [83] reuses the WPG presented in the same article in order to do so. The WPG is based on a LQR formulation that computes a CoM trajectory from a given ZMP trajectory. The linearity of the LQR formulation allows to apply the same computation of the error Δz resulting in a correcting increment Δc. Algorithm 3.2 shows an implementation of the algorithm in pseudo code form. An alternative formulation is presented in [162] by *Stasse*, where the DF is described as *Newton*'s methods applied to the equality $\Delta z(c) = 0$.

In general, this method does not guarantee the convergence, and might suffer from numerical instability. However, it has proven its efficiency for the specific problem of bipedal locomotion, c.f. [136]. Indeed in practice one iteration of the DF is sufficient to reduce considerably the error on the ZMP, as shown in Figure 3.12.

Remark (Zero-Moment Point Approximation using Inverse Dynamics) For robots, which fulfill the assumption that the inertial effects of legs and arms can be neglected, as for example *HRP-2*, an approximation of the ZMP can be easily computed by means of unconstrained inverse dynamics, such that line 2 reads

$$\tilde{z} = \text{RNEA}(q, \dot{q}, \ddot{q}),$$

and the respective increments are then computed with $\tilde{z}$ instead of $z^\star$. However, for robots not fulfilling this property, as for example *HeiCub*, this approximation leads to a critical error, such that the resulting trajectories cannot be applied for execution on the robot itself, c.f. Figure 3.10.

In [121], an alternative approximation of the whole-body ZMP is proposed. The author uses the inverse dynamics to compute an approximation of the joint torques $\boldsymbol{\tau}$ that are then used to compute FD with a defined contact configuration determined from the WPG to approximate the ZMP from the resulting contact forces. However, the resulting ZMP trajectories are still only an approximation to the real ZMP trajectory from the whole-body dynamics of the humanoid, c.f. Figure 3.10.

3.4.2 Generalized Inverse Kinematics

In order to realize a stable whole-body motion on the actual robot from the computed CoM, feet trajectories and ZMP trajectories, we need to compute the required reference inputs on joint level. For *HRP-2*, whose joint actuation is realized via position controlled DC motors, we have to compute reference joint angles and velocities $\boldsymbol{q}, \dot{\boldsymbol{q}}$. While these trajectories can be easily computed by means of inverse kinematics [9, 164] and then applied to the real robot as in our work [163], a better and more stable performance on the robot can be achieved by using a generalized inverse kinematics framework that adds some low-level stabilizing control on the robot. The generalized inverse kinematics used in this thesis implements the work presented in [116, 150, 152]. An alternative approach based on the same idea of hierarchical convex optimization is presented in [113].

Pure Inverse Kinematics The above mentioned pure inverse kinematics approach implemented in [163] implements two numeric algorithms to solve the inverse kinematics problem. Following the approach of *Sugihara* presented in [164], for a set of N given end-effector poses, i.e., specified positions and orientations, a residual can be defined by

$$\boldsymbol{e}_i(\boldsymbol{q}) = \begin{cases} {}^d\boldsymbol{p}_i - \boldsymbol{p}_i(\boldsymbol{q}) \\ \boldsymbol{R}_i^T(\boldsymbol{q})\, \boldsymbol{a}({}^d\boldsymbol{R}_i \boldsymbol{R}_i^T(\boldsymbol{q})) \end{cases}, \quad i = 1, \ldots, N, \tag{3.102}$$

where, for the ith end-effector and given generalized joint positions $\boldsymbol{q} \in \mathbb{R}^{n^{\text{DoF}}}$, we define the desired and current positions by vectors ${}^d\boldsymbol{p}_i, \boldsymbol{p}_i \in \mathbb{R}^3$ and represent the desired and current orientation by rotational matrices ${}^d\boldsymbol{R}_i, \boldsymbol{R}_i \in \mathcal{SO}(3)$. The function $\boldsymbol{a}(\cdot)$ computes the angular velocity from a rotational matrix, where the result might have to be transformed back into world coordinates depending on the adopted convention. From a given initial configuration of the robot $\boldsymbol{q}_0$ the joint angles required to realize the given desired end-effector poses can be computed by solving a NLP minimization problem of the form

$$\min_{\boldsymbol{q}} \quad \frac{1}{2}\|\boldsymbol{e}(\boldsymbol{q})\|_2^2 \tag{3.103a}$$
$$\text{s.t.} \quad \underline{\boldsymbol{q}} \leqslant \boldsymbol{q} \leqslant \overline{\boldsymbol{q}}, \tag{3.103b}$$

where $\boldsymbol{e} = (\boldsymbol{e}_0^T, \ldots, \boldsymbol{e}_N^T)^T$ are the stacked residuals of all end-effector poses and $\underline{\boldsymbol{q}}$ and $\overline{\boldsymbol{q}}$ are the joint limits of the robot to be respected during optimization. While the original work [164] *Sugihara* applies a tailored *Levenberg-Marquardt* method to solve the unconstrained version of the inverse kinematics problem of Equation (3.103), the constrained problem requires state-of-the-art NLP solvers to be solved, e.g. the interior point optimizer *IPOPT* [178] as in [163].

Generalized Inverse Kinematics The generalized inverse kinematics or more precisely the operational-space inverse dynamics approaches try to find joint accelerations $\ddot{\boldsymbol{q}}$, joint torques $\boldsymbol{\tau}$ as well as contact forces $\boldsymbol{\lambda}$ in order to fulfill a hierarchy of N task functions $\boldsymbol{e}_i(\boldsymbol{q})$, $i = 1,\dots,N$, e.g. desired end-effector poses according to Equation (3.102), on acceleration level such that $\ddot{\boldsymbol{e}}_i -^d \ddot{\boldsymbol{e}}_i = 0$, where by ${}^d\ddot{\boldsymbol{e}}$ we denote a certain goal and $\ddot{\boldsymbol{e}}$ is the second order derivative given by means of the task function Jacobian $\boldsymbol{J}_i$, i.e.,

$$\ddot{\boldsymbol{e}}_i = \boldsymbol{J}_i\ddot{\boldsymbol{q}} + \dot{\boldsymbol{J}}_i\dot{\boldsymbol{q}}. \tag{3.104}$$

For a single task the operational-space inverse dynamics problem can be formulated in the form of a canonical QP, which reads

$$\begin{aligned}
\min_{\ddot{\boldsymbol{q}},\boldsymbol{\tau},\boldsymbol{\lambda}} \quad & \frac{1}{2}\|\ddot{\boldsymbol{e}} -^d \ddot{\boldsymbol{e}}\|_2^2 && \text{(3.105a)}\\
\text{s.t.} \quad & \boldsymbol{H}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{c}(\boldsymbol{q},\dot{\boldsymbol{q}}) = \boldsymbol{S}^T\boldsymbol{\tau} + \boldsymbol{J}^T\boldsymbol{\lambda}, && \text{(3.105b)}\\
& \boldsymbol{J}\ddot{\boldsymbol{q}} + \dot{\boldsymbol{J}}\dot{\boldsymbol{q}} = \boldsymbol{0}, && \text{(3.105c)}\\
& \underline{\boldsymbol{q}} \leqslant \boldsymbol{q} \leqslant \overline{\boldsymbol{q}}. && \text{(3.105d)}
\end{aligned}$$

In [151] this concept is generalized and a hierarchy of different tasks and equality as well as inequality constraints is then formulated by means of strict hierarchies denoted by $\prec$, such that the change of a lower priority task does not effect a higher one. This is achieved by solving series of subsequent QPs, minimizing a system of slack variables, such that each of the QPs is operating in the Null-space of the constraints of the lower priority. Highest priority is always given to the dynamic feasibility constraint (3.105b) and the contact constraints (3.105c) followed by the motion tasks. In this way, it is not only possible to track end-effector poses but also guarantees a dynamic feasible whole-body motion, such that the generalized inverse kinematics framework can act as instantaneous stabilizing control on the robot.

3.5 Results

In the following section, we first present the results of the NMPC-based WPG of Section 3.2 applied to the robot *HRP-2* of CNRS-LAAS as presented in [133]. Second, we show how the WPG could be adapted to the robot *HeiCub* of UHEI and present results obtained from running the WPG in closed-loop control on the robot as presented in [163]. Finally, we present results for the WPG based on the MIQP formulation of Section 3.3 in simulation.

3.5.1 Results of the Walking Pattern Generator based on NMPC on *HRP-2*

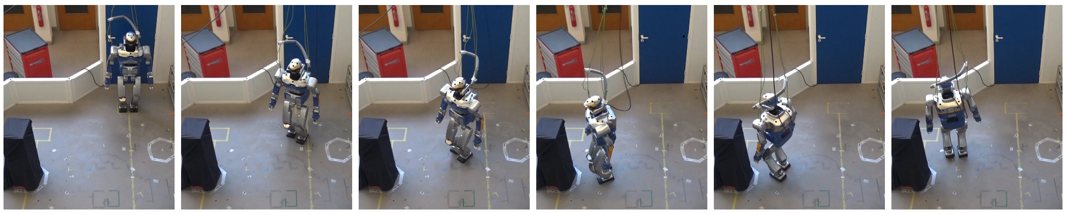


Figure 3.13: Experiment on the HRP2 robot using the setup B from our article [133].

In this Section, we present two experiments performed on the humanoid robot *HRP-*

2. The scenarios of the two experiments are chosen in order to create local situations where a foot-step planner using a discrete set of foot-step transitions may fail. Here, we only consider a given reference velocity that controls the overall motion of the robot. In the first experiment, the reference velocity drives the robot towards an obstacle which can be avoided to the collision avoidance strategy integrated in the WPG formulation of Section 3.2. In the second experiment the same strategy is employed to guarantee a collision free circular trajectory in cluttered terrain. Additionally, we present an evaluation of the WPG against external perturbations and present a run-time analysis of the approach on the computational hardware of *HRP-2*.

Experimental Setup

The cost function gains are $\omega_1, \omega_2 = 2.5$, $\omega_3 = 10^3$ and $\omega_4 = 10^{-5}$, where experiments have shown that a different weight between linear and angular velocities, i.e., ω_1, ω_2, was not necessary at this stage. As specified in Section 3.2.3, ω_1, ω_2 are the reference tracking weight, ω_3 is a weight maintaining the ZMP close to the center of the foot, and ω_4 is the weight of the regularization term. They were chosen according to their experimental performance.

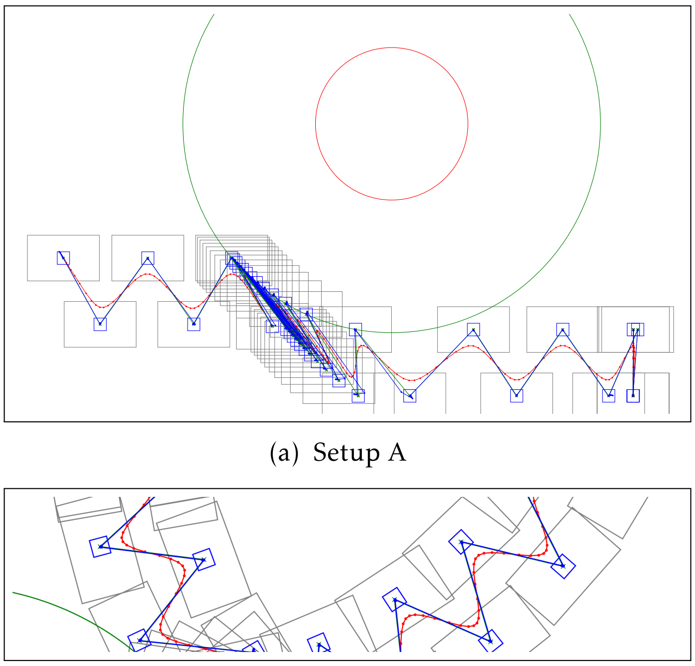
We define the duration of a single full step to be 0.8 s, including the single support time of 0.7 s and double support time of 0.1 s. The sampling time is chosen according to the double support time, i.e., $h = 0.1$ s. The preview horizon of the NMPC formulation is chosen such that it contains two full steps. The dynamic filter is applied to a sub-interval of the horizon containing only a single full step. In this way, we can achieve real-time feasibility of the composed approach on the internal hardware of *HRP-2*.

Motion Experiments with Obstacles

In Figure 3.14, the results in the form of the planned foot steps of the two experiments are depicted. The situation A is depicted in Figure 3.14a, where the output of the WPG is shown, while Figure 3.1a shows the robot *HRP-2* performing the task at CNRS-LAAS. Here, the reference velocity is set to $\boldsymbol{v}_{k+1}^{ref} = [0.2, 0, 0]$ and the obstacle to avoid is the red box, c.f. Figure 3.1a. The red box is represented by the inner red circle while the security margin is represented by the outer green circle in Figure 3.14a. According to the formulation of the collision avoidance constraints in Section 3.2.3, the robot is not allowed to step into the interior of the green. The security margin is chosen such that the upper body cannot collide with the obstacle.

The situation B is depicted in Figure 3.14b, where again the output of the WPG is shown, while the snapshot series of Figure 3.13 shows the real experiment of the robot. Here, a constant reference velocity is given by $\boldsymbol{v}_{k+1}^{ref} = [0.2, 0, 0.2]$ including a rotation around the vertical axis of the root body frame of the robot. Following this, the motion of the robot can be described by a circle. However, his path is blocked by the obstacle in his front and the robots gets stuck in front of it.

The constraints are linearized locally and the reference velocity is pointing towards the constraint, therefore the robot is blocked in translation. Thus, the robot stops moving forward, while the angular reference velocity results, which is not conflicting with the constraints, causes the robot to turn on spot. As soon as the robot has passed the obstacle, it can continue the forward motion and walk again freely.


(a) Setup A

(b) Setup B

Figure 3.14: Center of mass (red) and zero-moment point (blue) trajectories for obstacle avoidance and foot-step (gray boxes) orientation using our walking pattern generator based on nonlinear model predictive control from our article [133]. Setup A: Constant forward velocity. Setup B: Constant forward and angular velocity.

Robustness to External Perturbations

Here, the robustness of the WPG against external perturbations is evaluated in simulation. The external disturbance is a force applied to the CoM and we introduce it as acceleration increment to the actual initial value in the WPG formulation. The force is applied for a sampling period, i.e., $h = 0.1\,\mathrm{s}$. We consider two kind of perturbations, i.e., on the sagittal plane (both directions) and on the coronal plane (both directions). The robustness is evaluated in the walking situations, i.e., walking forward and on the spot.

The maximum lateral force on the coronal plane, that can be handled is 90 N, which is equivalent to −0.63 J, and −45 N or 0.675 J. The asymmetry comes from different walking situations during the push, i.e., the push may occur when the robot can perform a step without collision, or when it cannot. In the latter case, the magnitude of the force that can be rejected is smaller. We found roughly the same values for the two walking situations.

When walking on spot, the maximum forward and backward perturbation is ±115 N, equivalent to ±0.86 J, as the problem is symmetrical. When walking forward, the maximum disturbance is smaller in the forward direction. The interval found is $[-160, 70]$N, equivalent to $[-1.12, 1.54]$J.

Computational Time

The presented framework for motion generation runs online on the HRP2 CPU board ($Intel(R)\,Core2(TM)\,Duo\,E7500$, one core used, 2.8 GHz, 3 MB of cache size, on *Ubuntu 10.04 LTS*). The time measurement has been performed on the complete control architecture as depicted in Figure 3.2. The internal sampling time of the robot *HRP-2* is 5 ms.

Table 3.1: Measured run-time of the motion generation framework using the walking pattern generator based on nonlinear model predictive control from the article [133]. The time is measured in terms of CPU time on the internal computational hardware of the robot.

Measured run-time [ms]	Setup A	Setup B
Average	3.95	4.00
Standard deviation	0.14	0.18
Minimum	3.34	3.09
Maximum	4.34	5.19

Based on the timing statistics, there was only a single iteration violating the 5 ms real-time constraint, such that we can claim an overall real-time feasibility on the robotic hardware. This outlier was due to the stabilizer, which is expensive in terms of CPU time when the robot is in a configuration close to a kinematic singularity.

3.5.2 Results of the Walking Pattern Generator based on NMPC on *HeiCub*

In the following Section, we present experimental results from realizing different walking scenarios with the WPG from Section 3.2 on the humanoid robot *HeiCub*. The goal of the experiments was to achieve stable walking with the maximum possible velocity for the different scenarios. First, we run the WPG for experiments without feedback. The reference velocity and support times were set to the values previously achieved in [76] and then further increased and decreased, respectively, until the hardware could not perform the walking motion in a stable manner. In this way, forward walking speed was increased significantly and sideways, backwards and curved walking added to the capabilities of *HeiCub*.

Key Performance Indicators Definition

In order to assess the walking capabilities of the robot, we use the key performance indicators (KPIs) defined within the *KoroiBot* project [102]. Here, we focus on the group of technical KPIs measured in [76] for *HeiCub*. The measured quantities act as references to quantify the improvement of our WPG as well as installation of closed-loop feedback. The respective KPIs are given in the following listing.

- **Cost of Transport**

$$\mathbf{E}_{CT} = \frac{\sum_{m=1}^{M} \int_{t_0}^{t_f} I_m(t) V_m(t) dt}{m_{robot} g \cdot d}, \tag{3.106}$$

  where M is the total number of motors, I_m and V_m are the current and voltage measurements of the motor m, m_{robot} is the mass of the robot and d is the traveled distance. The Cost of Transport is unitless.
- ***Froude* Number**

$$Fr = \frac{v_{max}}{\sqrt{g \cdot h}}, \text{ for } h = l_{leg}, \tag{3.107}$$

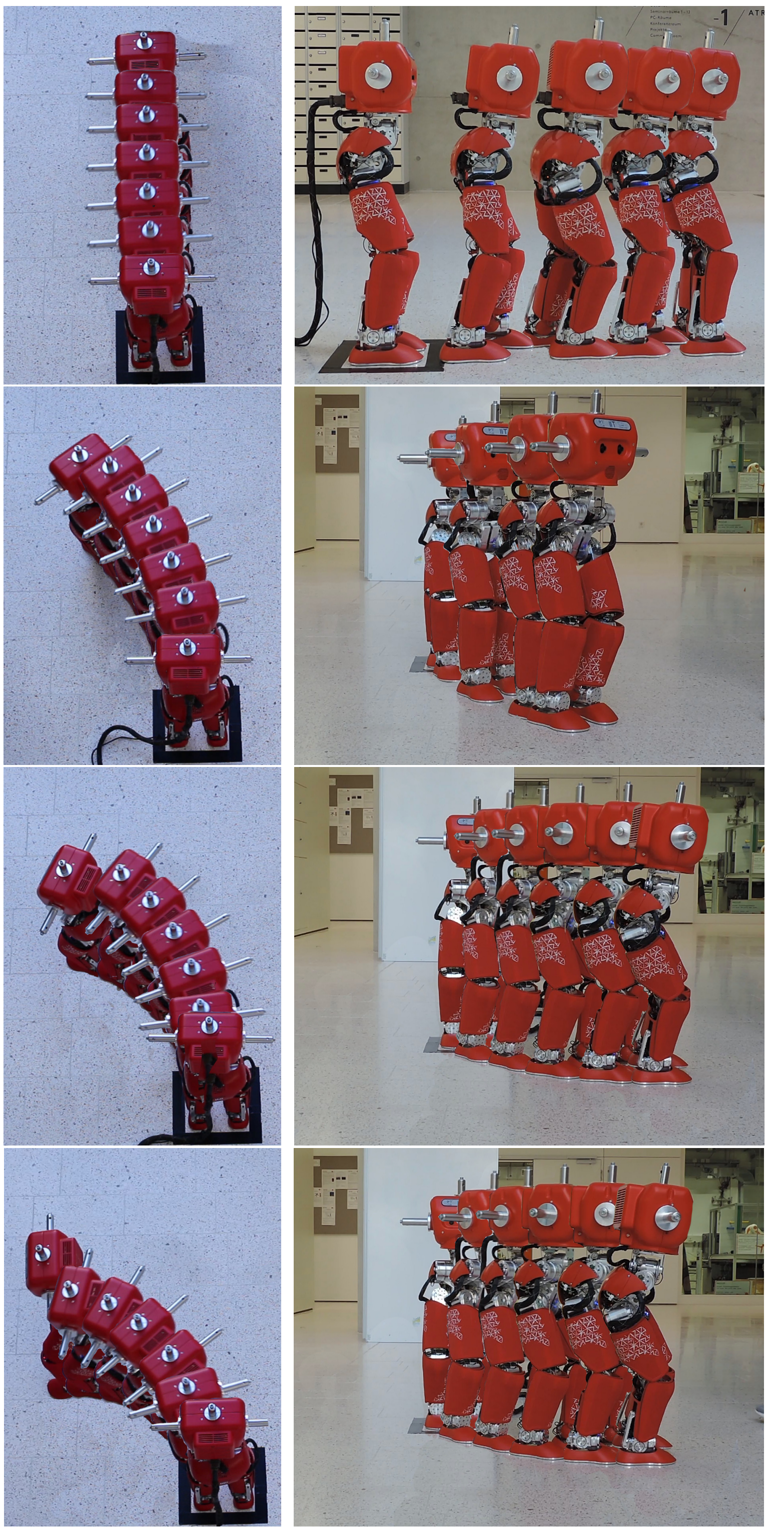


Figure 3.15: Snapshots taken from the video attached: Different walking sequences for straight walking and different angular velocities from top, side and front of *HeiCub*, from the article [163].

where l_{leg} is the robot's leg length. A given *Froude* number can be assigned to a certain walking style.

- **Precision of Task Execution**
  The root mean square error is computed by summing the squared difference between the measurement and the desired position over all points of the whole trajectory.

Feedback Elaboration and Evaluation

With active feedback the CoM position is provided as initial state to the optimization problem while otherwise the result of the last iteration is used. In order to compute the CoM position, the joint positions of the robot are measured using the *WholeBodyInterface* [30] of *iCub*. From the joint positions the CoM position in the local frame is computed from a unified robot description format model using *Rigid Body Dynamics Library* (RBDL) [49, 50]. The floating base is then required to transform the local CoM into the world frame.

There exists an online floating base estimation based on odometry in [30]. This was tested, but found to be currently too inaccurate for feedback purposes. Therefore the result of the inverse kinematics is added to the local CoM position to find its global position. This is published on a *YARP* port from which it is read by the WPG.

The stability of the walking motion is assessed by computing the actual global ZMP position $\mathbf{z}$ from the six-axis Force-Torque (F/T) sensor readings. From the torques $\boldsymbol{\tau} = [\tau_x, \tau_y, \tau_z]$ and forces $\mathbf{f} = [f_x, f_y, f_z]$ the local ZMP position can be derived according to *Kajita* [86] by

$$\begin{aligned} z_x &= \frac{(-\tau_{Ry} - f_{Rx}h) + (-\tau_{Ly} - f_{Lx}h)}{f_{Rz} + f_{Lz}} \\ z_y &= \frac{(\tau_{Rx} - f_{Ry}h) + (\tau_{Lx} - f_{Ly}h)}{f_{Rz} + f_{Lz}}, \end{aligned} \tag{3.108}$$

where R and L indicate the sensor of the right or left leg and h is the distance of the sensor to the ground.

Experimental Setup

The gain parameters of the WPG had to be adapted to the new robot platform, once. All motions without feedback were performed using the set of parameters

$$\omega_{1,2} = 0.02 \qquad \omega_3 = 1.0 \qquad \omega_4 = 10^{-5} \tag{3.109}$$

For motions using the CoM position feedback, the gain parameters were adjusted according to the behavior of the robot. It was mainly found, that for high values of the ZMP reference tracking gain ω_3, the WPG would overcompensate the ZMP error, by introducing sudden CoM movements. This leads to a diverging CoM position computation after a few iterations of the control loop. Therefore, ω_3 was reduced when observing this behavior.

For active feedback, the gain parameters were systematically changed in the range of $\omega_{1,2} \in [0.005, 0.1]$ and $\omega_3 \in [0.01, 1]$. Values that lead to unfeasible trajectories and motions that turned out to be unstable on the robot were discarded. Nonetheless, it was found that a high range of possible parameters leads to a stable motion.

The best results were observed for

$$\omega_{1,2} = 0.015 \qquad \omega_3 = 0.2 \qquad \omega_4 = 10^{-5}. \tag{3.110}$$

Additionally, straight walking on the soft soles of *HeiCub*, was achieved for the first time. Due to the required safety margin, the ZMP is maintained close enough to the center of the foot, such that only little deformation is occurring.

Forward Walking Experiments

Table 3.2: Key performance indicators (KPI) measured for forward walking comparing the cart-table WPG and the NMPC WPG with and without feedback from the article [163]. Measurements were taken for the fastest achievable velocity and minimal support times.

Parameters and KPI	cart-table	NMPC	NMPC + Feedback
v_{max} [m/s]	0.037	**0.065**	**0.065**
t^{ss}/t^{ds} [s]	1.5 / 1.0	0.7 / 0.7	**0.6 / 0.6**
step period [s]	2.5	1.4	**1.2**
Cost of Transport	4.27	**2.83**	2.99
Froude Number	0.017	0.029	**0.029**
Joint Error [deg]	1.45	1.22	**1.21**
CoM Error [cm]	0.61	0.50	**0.44**
ZMP Error [cm]	5.69	6.01	**3.02**

In Table 3.2, we show the obtained measurements for the maximum achieved forward walking velocity and the minimal step period that could be realized. The values obtained using the WPG based on the cart-table model is listed as reference. However, since for the original measurement, presented in [76], a floating base estimation was included in the error calculations, the local CoM and ZMP errors were recomputed from the original measurement data according to the definition in Section 3.5.2.

From the results in Table 3.2, a clear improvement from the cart-table to the NMPC WPG can be seen from the velocity of the robot which improved by 75 % from $0.037\,\mathrm{m\,s^{-1}}$ to $0.065\,\mathrm{m\,s^{-1}}$. With the feedback control the step period could be reduced by 50 %. Overall, the cost of transport decreased by 30 %, mainly due to the fact that a larger distance is covered in the same time. The tracking precision of the CoM is improved by 30 % and that of the ZMP by 50 %.

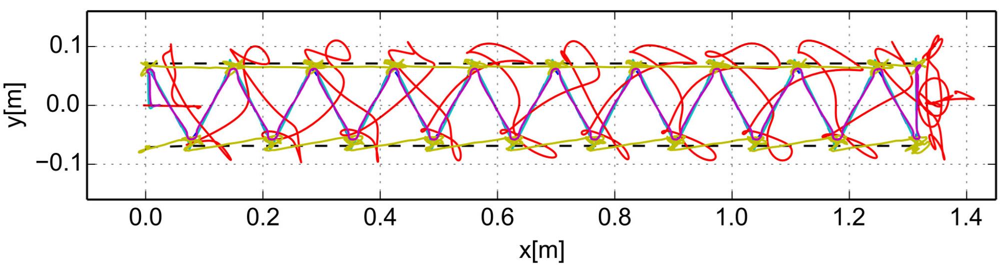
3.5.3 Feedback Analysis

In Table 3.2, we showed that it was possible to further decrease the support times while using feedback control. Figure 3.16 depicts the measurement for the parameters

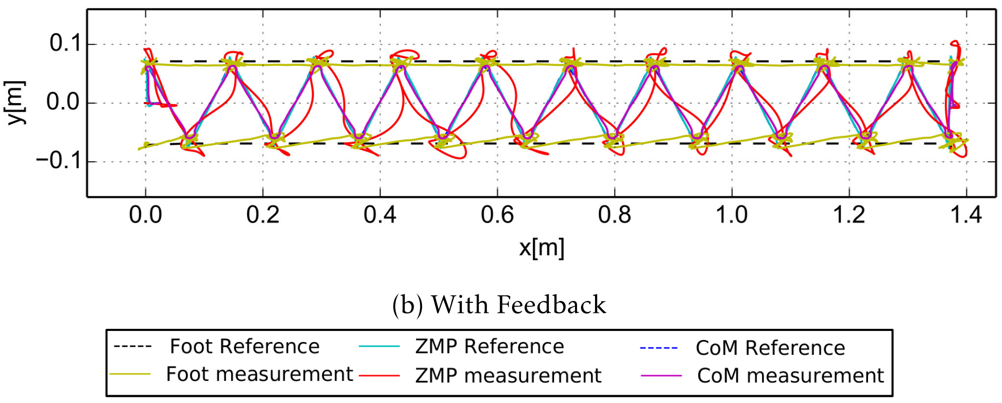
$$t_{ss} = 0.6\,\mathrm{s} \qquad t_{ds} = 0.6\,\mathrm{s} \qquad v_{max} = 0.065\,\mathrm{m\,s^{-1}} \tag{3.111}$$

with and without feedback.

Without feedback, the configuration is not stable, as it can be seen from the ZMP trajectory, which is far from the reference in the upper plot in Figure 3.16. With feedback, the robot follows the desired ZMP reference and the motion is stable. The ZMP tracking shows a reduction of the root mean square error by 60% from 6.99 cm to 3.02 cm.



(a) Without Feedback



(b) With Feedback

Figure 3.16: Comparison of an unstable motion that could be stabilized by the use of center of mass (CoM) position feedback from our article [163]. The measured zero-moment point (ZMP) reference tracking improved with activated feedback. Data was lowpass filtered for the ease of visualization.

Looking at the ZMP measurements in Figure 3.16 from the F/T sensors it can be seen that there are spikes in the measurement due to higher impact velocities and forces in the unstable motion without feedback. The CoM position feedback is able to compensate and smoothen the robots behavior. The measurement shows less spikes.

Additional Walking Scenarios

Table 3.3: Key performance indicator (KPI) measurements for sideways, backwards and curved walking from our article [163].

KPI	Backward	Sideways	Curved	Soft Soles
v_{max} [m/s]	0.065	0.02	0.06	0.06
v_{φ} [rad/s]	0	0	0.08	0
Cost of Transport	3.22	11.56	3.54	3.41
Froude Number	0.029	0.009	0.027	0.027
Joint Error [deg]	1.15	1.00	1.76	1.25
CoM Error [cm]	0.42	0.40	0.86	0.49
ZMP Error [cm]	7.90	6.63	7.34	6.58

No KPIs for *HeiCub* were yet available for sideways and curved walking, as well as for walking on soft soles as these motions were not achievable with the previous WPG. The measurements for the maximum achievable linear and in the case of curved walking, the angular velocity, and $t_{ss} = 0.7\,\text{s}$ and $t_{ds} = 0.7\,\text{s}$ are shown in Table 3.3.

Sideways walking was only performed at a low velocity, since no alternating steps can be taken, as legs cannot be crossed. The stepping area is very narrow in this direction and no steps bigger then 4 cm can be performed. Hence, the cost of transport is high, compared to other walking motions. The same is true for human walking, and therefore humans only walk sideways to very close goals where walking forward would result in a significantly longer path [126]. Curved walking is shown to be a more challenging motion, as every KPI has a less favorable value when comparing it to straight walking. This is also the case for human walking. Walking with the soft soles attached was achieved for the first time on an iCub. The KPIs have similar values as for forward walking without the soles. The ZMP error is higher as the soles get deformed as soon as the ZMP is not at the center of the foot.

Adapting Walking Directions

With the new WPG we are now able to adapt the walking direction online by modifying the reference velocity. During the measurement the velocity was changed from forward, to sideways and to curved walking. The transition between the different directions was performed during the double support phase. The reference velocity was set to zero at the beginning of double support and then changed to the new direction at the end of the double support. Figure 3.17 and 3.18 show a demonstration of this new feature on *HeiCub* in the lab and in simulation. It was found that in simulation the robot is more stable and higher velocities can be achieved.

3.5.4 Results of the Walking Pattern Generator based on MIQP on *HeiCub* in Simulation

In this section, we show the results obtained from applying the WPG based on MIQP, as presented in Section 3.3, on a model of the humanoid robot *HeiCub* for two walking scenarios in simulation. In the first experiment, we apply the WPG for a level-ground walking task in a scene with obstacles, which have to be avoided by the robot. The second scenario shows the ability of the WPG not only to avoid obstacle but also to plan a path in a space defined by non-connected step stones instead of a complete plane, such that the algorithm has to decide which step stones to use for actual steps.

Experimental Setup

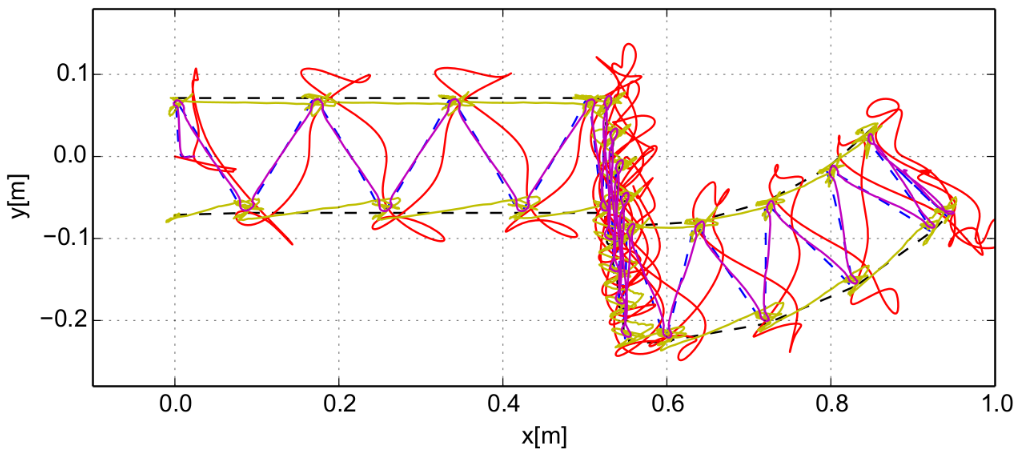
For both scenarios, we use the following weights for the different terms in the objective function

$$\omega_{1,2} = 100.0, \qquad \omega_3 = 0.01, \qquad \omega_4 = 10^{-5}, \tag{3.112}$$

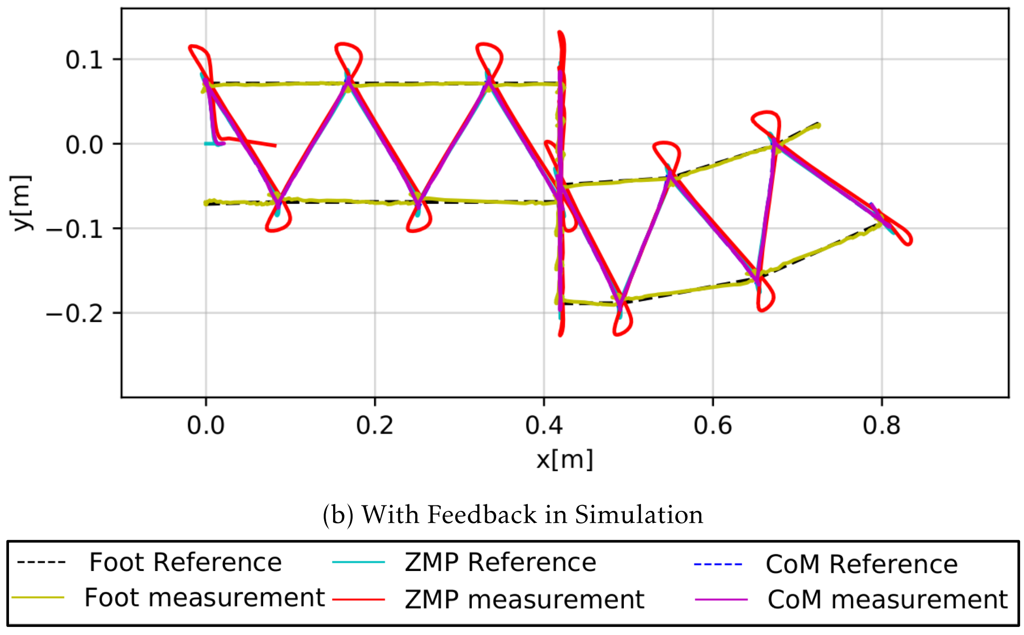
which control the reference velocity tracking, the distance of the ZMP to the center of the support foot and a regularization term on the CoM jerks. Additionally, the MIQP formulation of the WPG as presented in Section 3.3, requires additional weights in the form of

$$\omega_5 = 1.0, \qquad \omega_6 = 1.0, \qquad \omega_7 = 20.0, \qquad \omega_8 = 250.0, \tag{3.113}$$

where according to Section 3.3.3 ω_5 controls the penalty on the double support phase, ω_6 defines switch costs between support foot changes, ω_7 is the gain of centering the



(a) With Feedback in Lab



(b) With Feedback in Simulation

Figure 3.17: Sample trajectory of forward, sideways and curved walking from the article [163]. The measurement are taken with feedback for *HeiCub* (top) in the lab and *HeiCub* in simulation (bottom).

CoM between the support feet, and ω_8 is the weight for the ℓ_1-norm approximation of the support polygon according to (3.58). Here, the terms were chosen by experience of the experimenter according to their experimental performance.

In contrast to the WPG based on NMPC, the MIQP formulation does not require a distinction between single and double support time. This is due to the automatic foot placement formulation, which also handles double and single support phases respectively. However, we still have to define a sampling time, here, $h = 0.1\,\mathrm{s}$, as well as a horizon length. According to the choice of the NMPC-based WPG, we use a horizon length such that two consecutive steps are possible, i.e., $n = 10$.

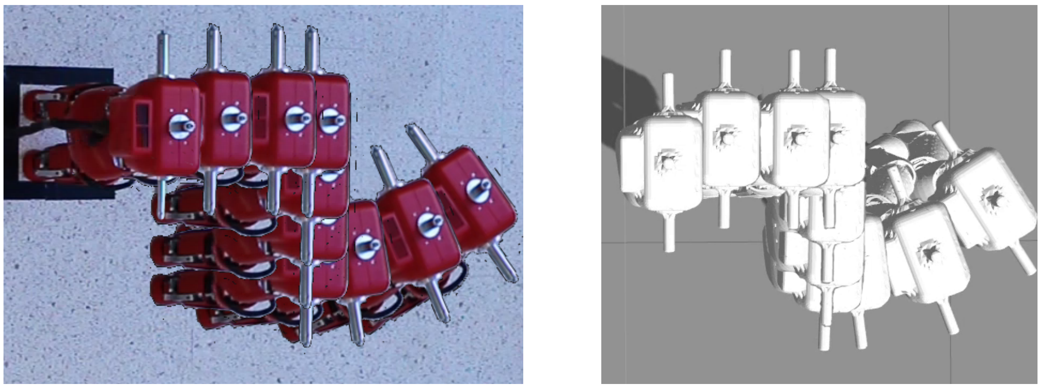


Figure 3.18: Snapshots of *HeiCub* walking in different directions in the lab and in simulation from the article [163].

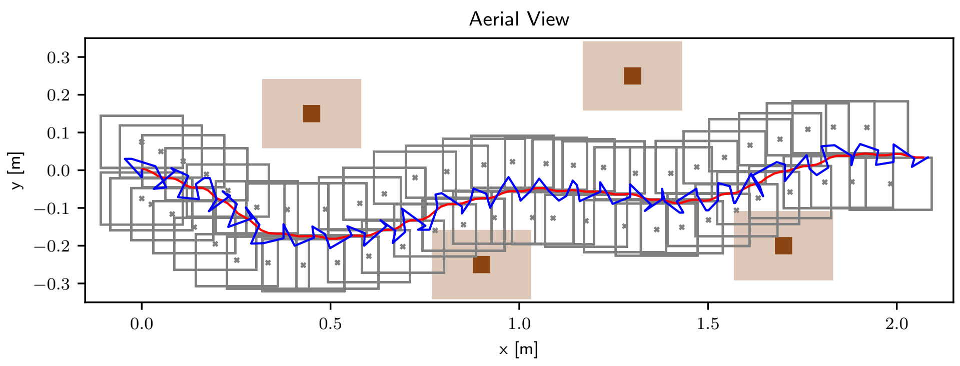


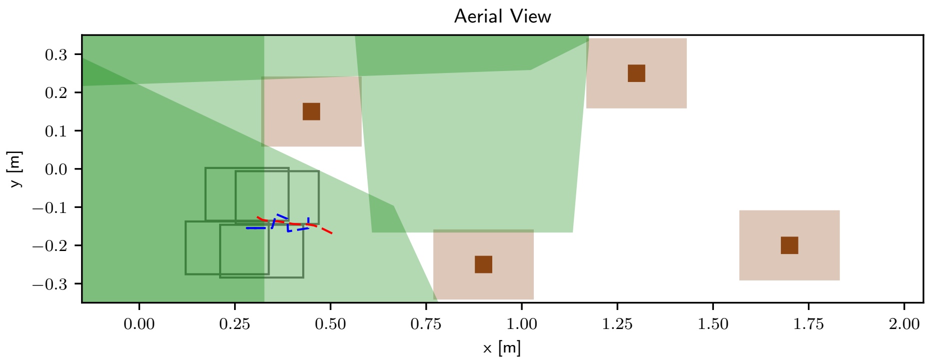
Figure 3.19: Resulting trajectories of the center of mass (CoM) (red) and the zero-moment point (ZMP) (blue) as well as the foot steps (gray boxes), where there centers are highlighted by gray $\times$ symbols, for the obstacle avoidance scenario. The obstacles are shown as brown boxes with the security margin required for a collision free foot placement in light brown.The path is collision free, when no foot step collides with one of the obstacles.

Table 3.4: Measured run-time of the preparation and solution phases of the walking pattern generator based on mixed-integer quadratic program for the obstacle avoidance scenario.

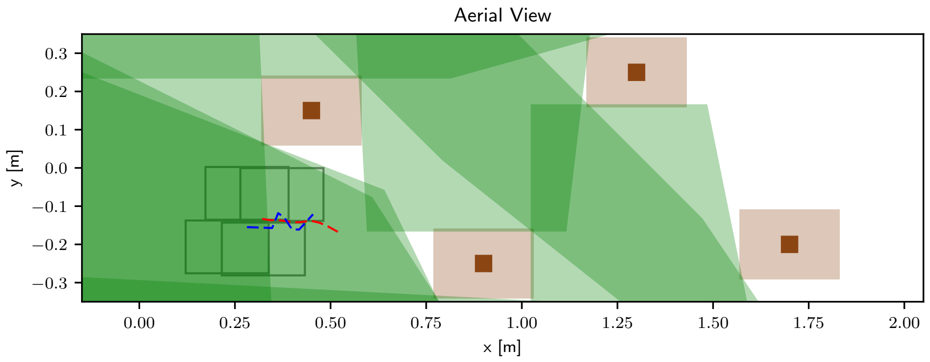
Phase	Iterations	Obstacles	Preparation	Solution
			mean ± std (max) [s]	mean ± std (max) [s]
1	[0, 24]	1	$0.032 \pm 0.036\,(0.203)$	$0.447 \pm 0.125\,(0.731)$
2	[25, 46]	2	$0.041 \pm 0.074\,(0.374)$	$1.044 \pm 0.224\,(1.625)$
3	[47, 67]	3	$0.058 \pm 0.148\,(0.702)$	$0.985 \pm 0.223\,(1.416)$
4	[68, 71]	2	$0.108 \pm 0.164\,(0.355)$	$1.093 \pm 0.199\,(1.287)$
5	[72, 94]	3	$0.048 \pm 0.104\,(0.527)$	$1.063 \pm 0.132\,(1.328)$
6	[95, 121]	2	$0.037 \pm 0.062\,(0.345)$	$1.006 \pm 0.217\,(1.534)$
7	[122, 122]	1	$0.185 \pm \quad (0.185)$	$0.904 \pm \quad (0.904)$

Obstacle Avoidance for Level-ground Walking

The first scenario is a level-ground walking task, i.e., following a reference velocity along the x-axis of $v^{x,ref} = 0.3\,\mathrm{m\,s^{-1}}$. We placed four obstacles into the direct path of the robot,



(a) Local prediction of the WPG on iteration 24 of the obstacle avoidance scenario.



(b) Local prediction of the WPG on iteration 25 of the obstacle avoidance scenario.

Figure 3.20: Visualization of the treatment of obstacle-free areas (OFAs) for collision avoidance for the walking pattern generator (WPG) based on mixed-integer quadratic program (MIQP). Showing the prediction of center of mass (CoM) (red, dashed), zero-moment point (ZMP) (blue, dashed) and foot placement (gray boxes) as well as actual considered OFAs (green areas) during two consecutive steps of the algorithm. A path is collision free when all planned foot steps are within the OFAs.

such that a straight path following the reference velocity is not possible without collision, as depicted by the brown boxes in Figure 3.19. The shape of the obstacles are poles with a quadratic ground projection with a width and height of 4 cm respectively. The actual positions of the poles are given by

$$\mathcal{O}_1 = \begin{bmatrix} 0.45 \\ 0.15 \end{bmatrix}, \quad \mathcal{O}_2 = \begin{bmatrix} 0.9 \\ -0.25 \end{bmatrix}, \quad \mathcal{O}_3 = \begin{bmatrix} 1.3 \\ 0.25 \end{bmatrix}, \quad \mathcal{O}_4 = \begin{bmatrix} 1.7 \\ -0.2 \end{bmatrix}.$$

The light brown boxes around the obstacle include an offset depending on the foot shape. In this way, the path of the robot is collision free, when there is no foot center in one of the light brown boxes or the gray boxes do not overlap with the obstacles.

The robot starts in double support phase, i.e., $\alpha_L = \alpha_R = 1$, with the initial CoM position of $\boldsymbol{c} = \boldsymbol{0}$, and zero initial velocity and acceleration $\dot{\boldsymbol{c}} = \ddot{\boldsymbol{c}} = \boldsymbol{0}$, and the left foot placed 7.5 cm above and the right foot below the CoM. We run the simulation for 12.0 s, then zero the reference velocity and stop the simulation as soon as a stable double support phase is reached.

Figure 3.19 shows the results obtained from simulation. The CoM trajectories are shown in red and the ZMP trajectories in blue. The CoM shows a slight sway motion such that it approaches the current support foot before lift-off. Due to proper discretization the result is still sufficiently smooth. The ZMP being only linear interpolated jumps between the interior of the support polygons according to their definition.

The actual foot placement is indicated by gray boxes, where the foot center is described by a gray $\times$. In Figure 3.19, one can see no overlap between obstacle boxes (brown) and foot steps, such that the overall path of the robot is collision free. Additionally, one can see the evasion motion undertaken due to the reference velocity deflection at the respective obstacles.

In Figure 3.20 , we show two consecutive iterations of the algorithm in order to show the collision avoidance based on OFAs as described in Section 3.3.2. In Figure 3.20a, we show iteration 24. Here, the bounding box of the OFAs search considers only $\mathcal{O}_1$, which is indicated by the four green boxes surrounding the obstacle in the upper left of the scene. The planned motion is shown by dashed lines, where the CoM is shown in red and the ZMP in blue. The foot placement is shown as gray boxes. The algorithm then considers the foot placement in the OFAs only to generate a collision-free path.

In Figure 3.20b, we show iteration 25. The main difference is that now both $\mathcal{O}_1$ as well as $\mathcal{O}_2$ are considered in order to find OFAs. $\mathcal{O}_2$ is the lower left box in the scene. The overlap of green boxes between $\mathcal{O}_1$ and $\mathcal{O}_2$ shows that there exists a feasible collision-free path around the obstacles.

In Table 3.4, we give an overview of the timing statistics of the preparation phase and the solution phase of the WPG. We grouped the results according to the different phases occurred during run-time, where each phase considers a different number of objects in the scene.

For the preparation phase, the mean run-time is much lower compared to the actual solution time by a factor of ≈ 10. However, at the maximum, the preparation time can be in the same order of magnitude as the solution phase.

The run-time seems to depend on the number of considered obstacles, however, a clear rule cannot be established. At first, the run-time is more than doubled for the phase 2 with two considered obstacles compared to phase 1. Additional obstacles do not seem to increase the run-time much. Considering the dependence on the considered obstacles, phase 7 is not representative as there is only a single iteration considered.

In contrast to the preparation and the solution phase, the computational time of the post-processing step is negligible with 0.922 ms at maximum and therefore not shown in the table explicitly.

The figures show that the algorithm properly works for this scenario, such that it generates a feasible and collision-free path through the scene for the robot. The OFAs are well identified such that a collision-free path can be found. The velocity deflection on the obstacles allows the robot to properly evade them during the motion, where fine-tuning the deflection strategy will avoid an over-emphasized evasive motion around the first obstacle in the future.

The CoM trajectories are smooth and show the required sway along the x-axis of the motion. However, the amplitude of the sway is quite small, which can be a result of the high stepping frequency as well as the small step size. The ZMP lies in the interior of the support polygon in both single as well as double support phase.

An issue that is very prominent in the motion, are the tiny steps proposed for the robot. Even though the kinematic capabilities of *HeiCub* are limited, larger steps are possible. While special care has taken to penalize the cost of changing the support foot and increases the minimum step size, the non-physical model and the configuration limits prevent a lower step frequency as well as a higher step-size.

All in all, the run-time statistics shown in Table 3.4 indicate that the presented MIQP formulation, while offering a solution to the path-planning and motion generation problem, is not real-time feasible at the moment.

3.6 Summary

In this Chapter, we have presented how the walking capabilities of two state-of-the-art humanoid robots could expanded by a dedicated walking control algorithm in the form of a walking pattern generator (WPG) based either on nonlinear model predictive control (NMPC) or mixed-integer programming within a whole-body motion generation framework as shown in Figure 3.2.

In this Chapter, we presented a real-time embedded nonlinear walking WPG. Nonlinear inequalities enabled us to choose the foot step automatically while considering orientation and local avoidance of convex obstacles. Its performance was demonstrated in two different experiments using the humanoid robot HRP2. The computational cost of the walking WPG is 2 ms on the robot such that the overall framework qualifies as real-time feasible on the robot. The proposed approach significantly enhanced the walking capabilities as shown by our partners from *Laboratory for Analysis and Architecture of Systems* (CNRS-LAAS) by investigating human-inspired power law trajectories, by letting *HRP-2* pull a fire hose or inside an adaptive motion synthesis framework [90, 130, 149].

Additionally, the online walking control framework was implemented on the robot *HeiCub*, where we additionally implemented feedback on the center of mass (CoM) position. The performance of the humanoid robot *HeiCub* has been analyzed and clear improvements due to the NMPC-based WPG and the CoM feedback were shown.

Finally, we showed an alternative formulation of the WPG based on a mixed-integer quadratic program (MIQP) formulation. Rather than avoiding obstacles, in this formulation we try to assess the free space in the direction of walking to find directly a walking pattern within this obstacle free area. We proposed an efficient algorithm to assess this free space and proposed a problem formulation in the form of a MIQP that lets a feasible walking motion of the robot emerge from the solution.

4 Motion Generation based on Centroidal Dynamics

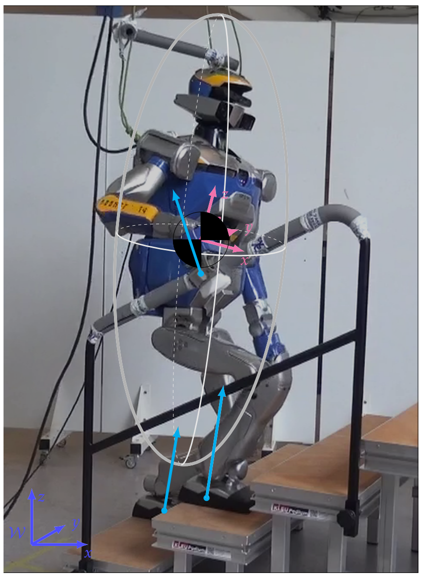


Figure 4.1: *HRP-2* climbing stairs with the support of a handrail from our article [106]. The overlay shows the employed reduced model. This includes the center of mass (CoM, black) at the waist, a CoM frame $\mathcal{C}$ (magenta), the current inertia ellipsoid, the current contact points (cyan dots) and contact forces (cyan arrows) as well as the world coordinate system $\mathcal{W}$ (blue). From optimizing the motion of the template model based on centroidal dynamics a whole-body motion for *HRP-2* is realized.

In this work, we propose an approach for motion generation of humanoids using centroidal dynamics. The centroidal dynamics extend the approaches based on CoM movement only as presented in Chapter 3 by not only considering the linear movement of the CoM but also considering the total angular momentum of the humanoid. Following this, reduced models based on centroidal dynamics allow to overcome the limitations resulting from simplified CoM dynamics such as the inverted pendulum models. Furthermore, centroidal dynamics enable the generation of whole-body multi-contact motions for humanoid robots as presented in the remainder of this chapter. We apply the approach to enable multi-contact motion generation for the humanoid robot *HRP-2* of the *Laboratory for Analysis and Architecture of Systems* (CNRS-LAAS). In contrast to the state of the art our approach for stair climbing is more robust and more efficient.

The contents of the following sections are based on our article [106].

The goal of motion generation for humanoids is to realize *human-like* tasks on the robot while considering the real-time constraints dictated by the robotic platform. Considering multiple contacts of the robot with its environment during motion generation generalizes bipedal locomotion and thus expands the functional range of humanoid robots. This enables a robot to climb ladders, perform crawling, evolve in cluttered environment and less impressively, but yet very useful, to climb stairs.

Here, we propose a complete solution to compute a fully-dynamic multi-contact motion of a humanoid robot. The actual motion generation is done via computation of dynamically-consistent CoM trajectories of the robot by employing a simplified dynamic model of the humanoid. From the whole-body centroidal dynamics of a humanoid we propose a model reduction in terms of an inertia ellipsoid model. This simplification is derived by decomposing the change of angular momentum into two parts, one depending on the non-actuated and the other depending on the actuated joint degrees of freedom (DoFs). Similar to the centroidal dynamics approach, we strive to find optimal contact forces as well as a kinematic feasible CoM trajectory from a predefined series of contacts in order to realize a given task. From this, realizing the whole-body motion is achieved by employing the instantaneous control framework of the robot to track the CoM and end-effector trajectories while keeping balance. We demonstrate the capabilities of the approach by making the humanoid robot platform *HRP-2* climb stairs with the use of a handrail.

The following contributions are made in this chapter:

- We propose a framework for the generation of whole-body motions of humanoids for predefined multi-contact supports based on a template model.
- We propose a mathematical formulation of reduced multi-contact centroidal dynamics of a humanoid by means of an ellipsoid model by separating the effects of actuated and non-actuated DoFs on the change of total momentum at CoM level.
- The reduced model incorporates only the DoFs of the floating-base as states, hence reducing the overall computational complexity of the approach and qualifying for being real-time feasible.
- The derived reduced model includes the major effects on the underactuated part and generalizes in the number of contacts such that the range of tasks is extended, e.g. level ground walking, walking non-flat floor, multi-contact like using the handrail during stair climbing.
- The approach is used to generate a whole-body motion for the humanoid robot platform *HRP-2* realizing the climbing of stairs with additional support of a handrail.
- For the first time, *HRP-2* realizes stair climbing with a height of 15 cm repeatedly without failure due to motor shutdown.
- The experimental study shows that handrail support reduces the overall motor power consumption by 25 %.

In Figure 4.2 the general idea of whole-body motion generation based on centroidal dynamics is visualized. First, a set of contact points has to be defined. For this task a possible way is to apply a contact planner as presented in [17, 24, 44] or the contact sequence can be hard-coded for a specific task. Second, the trajectories for the CoM are computed by means of solving an OCP by employing a reduced version of the full centroidal dynamics of the humanoid considering friction cone constraints as well as kinematic limits. Third, from the CoM trajectories and the specified end-effector trajectories, which are interpolated using B-splines to retrieve smooth trajectories from the discrete contact sequence, the actuated joint trajectories are computed by employing a generalized inverse kinematics approach. Finally, the processed set of joint trajectories $\boldsymbol{q}$ is provided to the low-level joint controls of the robot or into simulation.

The resulting trajectories can be directly provided as reference values to the position-controlled joints of the humanoid, e.g. the robot *HRP-2*. The computed trajectories are dynamically consistent and the contacts are realized according to the predefined schedule.

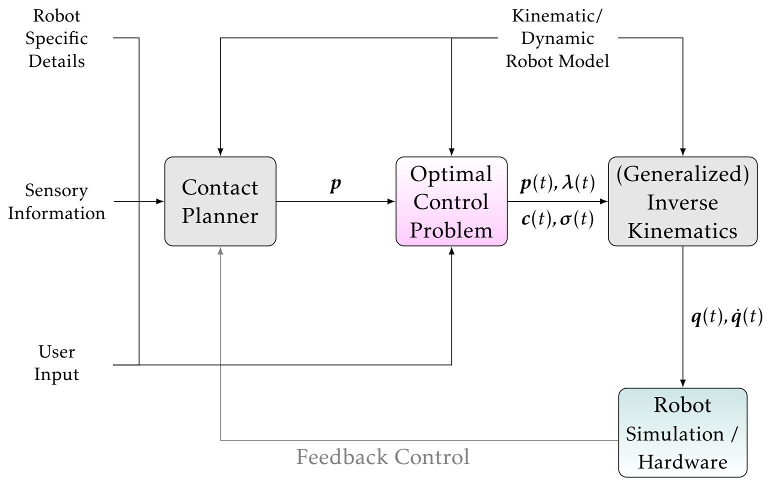


Figure 4.2: Visualization of the general framework based on reduced centroidal dynamics derived in this thesis. From robot specific details, sensory information and external user input a contact sequence $\boldsymbol{p}$ is defined either manually or automatically using a contact planner. The contact sequence enters an optimal control problem formulation that computes an optimal center of mass trajectory, end-effector trajectories $\boldsymbol{p}(t)$, contact forces $\boldsymbol{\lambda}(t)$ and the necessary change of angular momentum due to limb motions $\boldsymbol{\sigma}$, where discrete quantities are interpolated to achieve smooth trajectories. In order to realize a motion on the actual robot or in simulation the required joint trajectories are computed from the CoM and end-effector trajectories by means of generalized inverse kinematics. Feedback can be established on the current configuration of the robot.

The change of angular momentum term depending on the limb motion $\boldsymbol{\sigma}$ introducing a stabilizing effect on the motion, can be realized by methods of resolved momentum control using the free limbs during the execution of the motion.

In the following sections, we will go through the relevant building blocks of the scheme depicted in Figure 4.2 and explain their details. First, the reduced version of the centroidal dynamics is derived as well as a simplified force-contact complementarity is introduced in Section 4.1. Second, in Section 4.2 the OCP formulation is derived including the task-specific optimization criteria as well as the required constraints guaranteeing dynamic and kinematic consistency of the resulting CoM trajectories. Afterwards, the particular numerical scheme used to solve this problem is briefly discussed. In Section 4.3, we explain how to process the CoM and end-effector trajectories to create a feasible motion for the humanoid. Finally, we will show the results obtained from experiments on the robot *HRP-2* performing a stair climbing motion with handrail support in section 4.4.

4.1 Derivation of the Reduced Dynamic Model

In this Section, a reduced model of the whole-body dynamics of a humanoid is derived based on the idea of centroidal dynamics as introduced by *Orin* in [140] or used by other authors for motion generation [31, 137].

In contrast to the linear inverted pendulum model (LIPM) or other pendulum-like models, the idea of centroidal dynamics is to additionally include the total angular momentum at the CoM into the motion generation and therefore allowing to broaden the range of possible tasks. The application of centroidal dynamics for motion generation requires the following assumptions to be made.

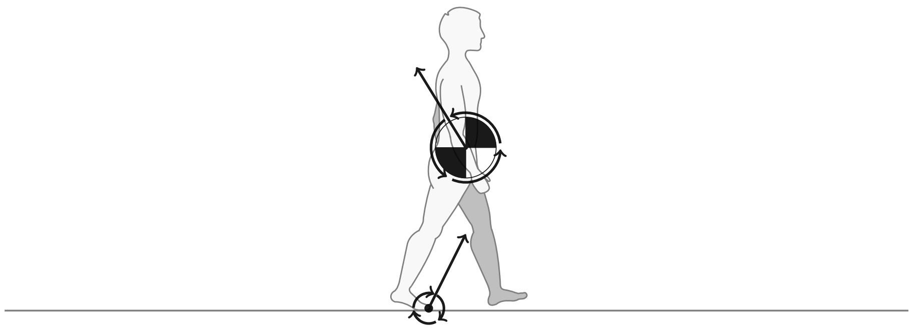


Figure 4.3: Visualization of centroidal dynamics derived from human motion during gait. The motion of the center of mass (CoM) is very characteristic for a humanoid and has attracted much attention in motion analysis. Beyond static postures, in order to judge the balance of the humanoid during motion researcher are interested in the forces (arrows) and moments (triple arrows in circle shape) introduced due to contacts because they determine the total linear and angular momentum at the CoM, i.e., the centroidal dynamics. Here, the actual torques applied at joint level are not explicitly considered.

Assumption (Centroidal Dynamics Assumptions) In order to approximate the whole-body dynamics of a multi-body system by means of its centroidal dynamics, we assume that the allowed joint actuation is sufficiently high, i.e., the respective joint limits $\underline{\boldsymbol{\tau}} \leqslant \boldsymbol{\tau} \leqslant \overline{\boldsymbol{\tau}}$ are never violated, such that the required contact forces can be realized. $\triangle$

Revisiting the derivation of the reduced model for the LIPM of Chapter 3, the projection of the whole-body dynamics of the underlying MBS in contact with its environment as given by Equation (3.1) or (1.4) on its CoM due to [181] yields the *Newton-Euler* equations for the CoM motion given by

$$\mathrm{m}_{tot}\,(\ddot{\boldsymbol{c}} + \boldsymbol{g}) = \sum_{i \in \mathrm{I}} \boldsymbol{Q}_i \boldsymbol{\lambda}_i, \tag{4.1a}$$

$$\mathrm{m}_{tot}\,\boldsymbol{c} \times (\ddot{\boldsymbol{c}} + \boldsymbol{g}) + \dot{\boldsymbol{l}}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}) = \sum_{i \in \mathrm{I}} \boldsymbol{p}_i \times \boldsymbol{Q}_i \boldsymbol{\lambda}_i, \tag{4.1b}$$

where m_{tot} denotes the total mass of the humanoid, $\boldsymbol{g} \in \mathbb{R}^3$ is the gravitational force acting on the system, $\boldsymbol{c} \in \mathbb{R}^3$ denotes the coordinate vector of the CoM, $\boldsymbol{l} \in \mathbb{R}^3$ denotes the angular momentum of the system, $\boldsymbol{p}_i \in \mathbb{R}^3$ denote the respective lever arms with respect to the contact points and contact forces $\boldsymbol{\lambda}_i \in \mathbb{R}^3$ given by the contact set $\mathrm{I} \subset \mathbb{N}$, and $\boldsymbol{Q}_i \in \mathcal{SO}(3)$ denote the rotational matrix mapping the contact forces defined in the contact frame into the world frame. In this way, the dynamics of the projected dynamics generalizes with the number of considered contact points $|\mathrm{I}|$. The contact force applied at $\boldsymbol{p}_i$ is given in a local coordinate system, with the z-axis normal to the contact surface at the contact point $\boldsymbol{p}_i$ with respect to the CoM. The component λ_i^z is the normal force applied at the contact point and $(\lambda_i^x, \lambda_i^y)$ denote the tangential components of the force. In order to motivate the idea of motion generation based on centroidal dynamics, we first introduce the underlying stability criterion and the required definitions for it, before the actual reduced model is derived.

Definition (Static Friction Cone, $\mathcal{K}^\mu$) For contact point $\boldsymbol{p} \in \mathbb{R}^3$ on a contact surface given by its normal vector $\boldsymbol{n} \in \mathbb{R}^3$ and a static friction parameter $\mu \geqslant 0$, the respective friction cone for a local contact force $\boldsymbol{\lambda} \in \mathbb{R}^3$, where x, y denote the tangential component and z the normal component of the contact force, is given by

$$\mathcal{K}^{\mu} = \{\boldsymbol{\lambda} \in \mathbb{R}^3 | (\lambda^x)^2 + (\lambda^y)^2 \leqslant (\mu\lambda^z)^2, \lambda^z \geqslant 0\}. \qquad \triangle$$

Definition (Sufficient Criterion for Dynamic Stability (based on Centroidal Dynamics))
Given a multi-body system that is in contact with its environment by means of a set of contact points $\mathrm{I} \subset \mathbb{R}^3$. If the total change of momentum $\dot{\boldsymbol{h}}_{tot}$ according to Definition 3.7 is compensated by the current contact forces $\boldsymbol{\lambda}_i$, $i \in \mathrm{I}$, i.e.,

$$\dot{\boldsymbol{h}}^0_{tot}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}) - \sum_{i \in \mathrm{I}} {}^0\boldsymbol{X}_i^{\star} \boldsymbol{\lambda}_i = 0, \tag{4.2}$$

where the momentum $\boldsymbol{h} = \begin{bmatrix} \boldsymbol{l} \\ \boldsymbol{k} \end{bmatrix}$ consists of a linear part $\boldsymbol{k} \in \mathbb{R}^3$ and angular part $\boldsymbol{l} \in \mathbb{R}^3$ and ${}^0\boldsymbol{X}_i^{\star}$ denotes the adjoint spatial transform that maps a force vector of the ith contact frame to the world frame indicated by index 0, and the contact forces lie within the interior of their respective friction cones $\mathcal{K}_i^{\mu_i}$, $i \in \mathrm{I}$ according to Definition 4.2, i.e.,

$$\boldsymbol{\lambda}_i \in \mathring{\mathcal{K}}_i^{\mu_i}, i \in \mathrm{I}, \tag{4.3}$$

and Assumption 4.1 holds, then the motion is *dynamically stable*. $\triangle$

Revisiting Equation (3.2) from Chapter 3, we decompose the DoFs $\boldsymbol{q}$ into linear and angular motion of the free-flyer $\boldsymbol{x}$, $\boldsymbol{\theta}_0$ and the actual joints of the robot denoted by $\hat{\boldsymbol{q}}$ given by

$$\boldsymbol{q} = \begin{bmatrix} \boldsymbol{x}_0 \\ \boldsymbol{\theta}_0 \\ \hat{\boldsymbol{q}} \end{bmatrix}. \tag{4.4}$$

Following this, we propose to separate the change of angular momentum $\dot{\boldsymbol{l}}$ into two terms

$$\dot{\boldsymbol{l}}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}) = \boldsymbol{H_c}(\hat{\boldsymbol{q}})\, \dot{\omega} - \sigma(\hat{\boldsymbol{q}}, \dot{\hat{\boldsymbol{q}}}, \ddot{\hat{\boldsymbol{q}}}), \tag{4.5}$$

where the first term describes the change of angular momentum due to angular acceleration of the whole robot body at the CoM $\dot{\omega} = \ddot{\boldsymbol{\theta}}$ described by the inertia matrix $\boldsymbol{H_c}$ of the whole humanoid considered as a single rigid body computed for the current robot configuration $\hat{\boldsymbol{q}}$, and the second term describes the change of angular momentum due to movements of the robot limbs such that $\boldsymbol{\sigma}$ is a function of the whole-body configuration, velocity and acceleration [140] that does not depend on $\boldsymbol{\theta}$ or $\boldsymbol{x}_0$. In this way, we can reformulate Equation 4.1 into

$$\boldsymbol{M}(\boldsymbol{c}) \begin{bmatrix} \ddot{\boldsymbol{c}} \\ \dot{\omega} \end{bmatrix} = \sum_{i \in \mathrm{I}} \boldsymbol{X}_i \boldsymbol{\lambda}_i + \boldsymbol{S}\boldsymbol{\sigma} - \boldsymbol{v}(\boldsymbol{c}), \tag{4.6}$$

with

$$\boldsymbol{M}(\boldsymbol{c}) = \begin{bmatrix} \mathrm{m}_{tot}\mathbf{1}_3 & \mathbf{0}_{3\times3} \\ \mathrm{m}_{tot}\boldsymbol{c}\times & \boldsymbol{H}_{\boldsymbol{c}} \end{bmatrix}, \tag{4.7}$$

$$\boldsymbol{X}_i = \begin{bmatrix} \boldsymbol{Q}_i \\ \boldsymbol{p}_i \times \boldsymbol{Q}_i \end{bmatrix}, \tag{4.8}$$

$$\boldsymbol{S} = \begin{bmatrix} \mathbf{0}_3 \\ \mathbf{1}_3 \end{bmatrix}, \tag{4.9}$$

$$\boldsymbol{v}(\boldsymbol{c}) = \begin{bmatrix} \mathrm{m}_{tot}\,\boldsymbol{g} \\ \mathrm{m}_{tot}\,\boldsymbol{c}\times\boldsymbol{g} \end{bmatrix}, \tag{4.10}$$

where $\boldsymbol{c}\times \in \mathbb{R}^{3\times3}$ is the skew symmetric matrix associated with vector $\boldsymbol{c}$ or $\boldsymbol{p}_i$ respectively.

The dynamic equilibrium constraints (4.6) are rewritten according to [18] as nonlinear first-order differential equation in the form of

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix} \boldsymbol{c} \\ \dot{\boldsymbol{\omega}} \\ \dot{\boldsymbol{c}} \\ \boldsymbol{\omega} \end{bmatrix} = \begin{bmatrix} \dot{\boldsymbol{c}} \\ \dot{\boldsymbol{\omega}} \\ (\boldsymbol{M}(\boldsymbol{c}))^{-1}\left(\sum_{i\in\mathrm{I}} \boldsymbol{X}_i\boldsymbol{\lambda}_i + \boldsymbol{S}\boldsymbol{\sigma} - \boldsymbol{v}(\boldsymbol{c})\right) \end{bmatrix}, \tag{4.11}$$

representing the ordinary differential equation (ODE) of the OCP as presented in Chapter 1 in Equation (1.9).

4.1.1 Simplified Force-Contact Complementarity

Revisiting the force-contact complementarity (1.6) of a MBS in contact with its environment described by kinematic constraints represents the fact that for unilateral constraints contact forces $\boldsymbol{\lambda}$ can only be applied when a contact is established. Transferring this to the simplified model, the complementarity can be reduced by implementing the contact part in terms of the motion of the contact described by $\|\dot{\boldsymbol{p}}_i\|_2 \geqslant 0$. We refer to this simplified contact complementarity to indicate if the end-effector is in contact or not given by

$$\|\dot{\boldsymbol{p}}_i\|_2 \cdot \|\boldsymbol{\lambda}_i\|_2 = 0. \tag{4.12}$$

4.2 Problem Formulation

We strive to formulate an OCP in a reduced form compared to Equation (1.9) of Chapter 1. We search for the best CoM trajectory respecting the dynamics derived in Section 4.1 and subject to constraints in terms of a combination of different optimization criteria. In the remainder of the section, we describe the cost function terms and the constraints and then state the motion generation of the CoM trajectories as an OCP.

4.2.1 Objective Function

Before giving the complete formulation of the OCP, we first define the cost terms used to formulate the objective function of the OCP. Therefore, we have to introduce the set of

Table 4.1: Objective weights for the optimal control problem based on centroidal dynamics.

i	0	1	2	3	4	5	6
ω_i	0.05	0.0005	1.0	1.0	1.0	0.1	1.0

contact points I used for the stair climbing task, i.e.,

$$\mathrm{I} = \{\mathrm{LH}, \mathrm{RH}, \mathrm{LF}, \mathrm{RF}\}, \tag{4.13}$$

where I specifies the respective robot end-effectors, i.e., LH, RH denote the left and right hand and LF, RF denote the left and right foot. The objective function is then given by a weighted sum of different terms,

$$l = \sum_{i=0}^{6} \omega_i l_i, \tag{4.14}$$

with weights $0 < \omega_i \leqslant 1$ given in Table 4.1 and each term l_i described in detail in the remainder of this section.

The first term l_0 keeps the projection of the CoM onto the ground close to the respective support foot contacts

$$l_0 = \|\boldsymbol{\lambda}_{\mathrm{LF}}\|_2^2 \, \|\boldsymbol{c}^{x,y} - \boldsymbol{p}_{\mathrm{LF}}^{x,y}\|_2^2 + \|\boldsymbol{\lambda}_{\mathrm{RF}}\|_2^2 \, \|\boldsymbol{c}^{x,y} - \boldsymbol{p}_{\mathrm{RF}}^{x,y}\|_2^2. \tag{4.15}$$

The second term l_1 uses the complementarity (4.12) to track a reference height $\bar{c}^z$ depending on the current foot contact height,

$$l_1 = \|(\lambda_{\mathrm{LF}}^z + \lambda_{\mathrm{RF}}^z)(c^z - \bar{c}^z) - \lambda_{\mathrm{LF}}^z \, p_{\mathrm{LF}}^z - \lambda_{\mathrm{RF}}^z \, p_{\mathrm{RF}}^z\|_2^2. \tag{4.16}$$

The four next terms l_2, l_3, l_4 are used to penalize a swaying motion of $\boldsymbol{c}$ in z direction.

$$l_2 = \|\dot{c}^z\|_2^2 \tag{4.17}$$

and stabilize the rotational DoFs by penalizing larges values through

$$l_3 = \|\omega^x\|_2^2, \quad l_4 = \|\omega^y\|_2^2, \quad l_5 = \|\omega^z\|_2^2. \tag{4.18}$$

The last term l_6 acts as another regularization term, i.e.,

$$l_6 = \|\ddot{\boldsymbol{c}}\|_2^2 + \|\dot{\boldsymbol{\omega}}\|_2^2. \tag{4.19}$$

4.2.2 Friction Cone Constraints

According to the stability criteria given by Definition 4.3 the friction cones play an important role for the stability of the resulting motion. Following this, the applied local contact forces $\boldsymbol{\lambda}$ have to satisfy friction cone constraints, which are given for each contact point by

$$\|(\lambda_i^x, \lambda_i^y)\|_2 = \sqrt{(\lambda_i^x)^2 + (\lambda_i^y)^2} \leqslant (\mu_i - \varepsilon_i)\lambda_i^z, \; i \in \mathrm{I}, \tag{4.20}$$

where $\mu_i > 0$ denotes the static friction coefficient at the contact point $\boldsymbol{p}_i$ and $\varepsilon_i > 0$ denotes a security margin. In this way, the resulting contact forces are guaranteed to stay strictly inside the respective friction cone $\mathcal{K}_i^{\mu_i}$ according to Definition 4.2 that reads

$$\boldsymbol{\lambda}_i \in \mathring{\mathcal{K}}_i^{\mu_i},\ i \in \mathrm{I}, \tag{4.21}$$

such that the contacts will not break during realization.

4.2.3 Kinematic Constraints

In order to render the resulting CoM motion to be kinematically feasible for a later execution on the robot, kinematic constraints have to be added to the OCP formulation. This is realized by defining simple bound constraints on the limb lengths relative to the CoM position $\boldsymbol{c}$ in global coordinates, defined by

$$\underline{\boldsymbol{p}}_i \leqslant \|\boldsymbol{c} - \boldsymbol{p}_i\|_2 \leqslant \overline{\boldsymbol{p}}_i,\ i \in \mathrm{I}. \tag{4.22}$$

We define the leg lengths for $\boldsymbol{p}_{\mathrm{LF}}, \boldsymbol{p}_{\mathrm{RF}}$ using $\underline{\boldsymbol{p}}_{\mathrm{LF/RF}} = 0.64\,\mathrm{m}$ and $\overline{\boldsymbol{p}}_{\mathrm{LF/RF}} = 0.8\,\mathrm{m}$ for the stair climbing motion, such that the robot does not bend the legs to far and also does not fully stretch its legs in order to stay away from kinematic singularities.

4.2.4 Path Constraints

In Section 4.1.1, we introduced the simplified force-contact complementarity. However, the complementarity is not directly resolved but manually formulated by means of appropriate bounds on the contact forces. The applied contact model treats both the contacts at the feet as well as at the hands as unilateral. This is achieved by defining bounds for the contact forces in the form of

$$\begin{aligned} \underline{\boldsymbol{\lambda}_{\mathrm{LF/RF}}} &\leqslant \boldsymbol{\lambda}_{\mathrm{LF/RF}} \leqslant \overline{\boldsymbol{\lambda}_{\mathrm{LF/RF}}}, \\ \underline{\boldsymbol{\lambda}_{\mathrm{RH}}} &\leqslant \boldsymbol{\lambda}_{\mathrm{RH}} \leqslant \overline{\boldsymbol{\lambda}_{\mathrm{RH}}}, \end{aligned} \tag{4.23}$$

where the exact values of the bounds are chosen according to the experience of the experimenters from CNRS-LAAS, i.e.,

$$\underline{\boldsymbol{\lambda}_{\mathrm{LF/RF}}} = \begin{bmatrix} -600 \\ -600 \\ 0 \end{bmatrix}, \qquad \overline{\boldsymbol{\lambda}}_{\mathrm{LF/RF}} = \begin{bmatrix} 600 \\ 600 \\ 600 \end{bmatrix}, \tag{4.24}$$

$$\underline{\boldsymbol{\lambda}}_{\mathrm{RH}} = \begin{bmatrix} -150 \\ -150 \\ 0 \end{bmatrix}, \qquad \overline{\boldsymbol{\lambda}}_{\mathrm{RH}} = \begin{bmatrix} 150 \\ 150 \\ 150 \end{bmatrix}, \tag{4.25}$$

$$\underline{\boldsymbol{\sigma}} = \begin{bmatrix} -600 \\ -600 \\ -600 \end{bmatrix}, \qquad \overline{\boldsymbol{\sigma}} = \begin{bmatrix} 600 \\ 600 \\ 600 \end{bmatrix}, \tag{4.26}$$

where the bounds on the forces are given in N, the bounds on the term $\boldsymbol{\sigma}$ are given in $\mathrm{N\,m\,s^{-1}}$.

4.2.5 Optimal Control Problem Formulation

The previously defined constraints and the dynamics of the reduced model allows us to formulate the whole-body motion generation problem for the humanoid given a pre-defined contact set as an OCP. The variables of interest are the states and the controls defined on the time horizon. The states $\boldsymbol{x} : \mathbb{R} \to \mathbb{R}^{n^{\mathrm{x}}}$ are composed of the CoM position and velocity as well as the orientation and the angular velocity and acceleration of root frame of the robot, i.e.,

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{c} \\ \dot{\boldsymbol{c}} \\ \boldsymbol{\theta} \\ \boldsymbol{\omega} \end{bmatrix}. \tag{4.27}$$

The controls $\boldsymbol{u} : \mathbb{R} \to \mathbb{R}^{n^{\mathrm{u}}}$ are composed of the contact forces of the active contacts at time t and the internal change of angular momentum, i.e.,

$$\boldsymbol{u} = \begin{bmatrix} \boldsymbol{\lambda}_1 \\ \boldsymbol{\lambda}_2 \\ \vdots \\ \boldsymbol{\lambda}_{|\mathbb{I}|} \\ \boldsymbol{\sigma} \end{bmatrix}. \tag{4.28}$$

In contrast to the multi-stage OCP presented in Chapter 1, we present here a reduced version of this OCP. We minimize an objective function of Lagrange type on a finite time horizon $t \in \mathcal{T} = [0, T]$ given by

$$\begin{aligned}
\min_{\boldsymbol{x}(\cdot),\boldsymbol{u}(\cdot)} \quad & \int_0^T l(\boldsymbol{x}(t), \boldsymbol{u}(t))\mathrm{d}t && (4.29\mathrm{a})\\
\text{s.t.} \quad & \dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t)), \quad t \in \mathcal{T}, && (4.29\mathrm{b})\\
& \boldsymbol{x}(0) = \boldsymbol{x}_0, && (4.29\mathrm{c})\\
& 0 \leqslant \boldsymbol{g}(\boldsymbol{x}(t), \boldsymbol{u}(t)), \quad t \in \mathcal{T}, && (4.29\mathrm{d})
\end{aligned}$$

where the objective function $l(\boldsymbol{x}, \boldsymbol{u})$ is defined as presented in Section 4.2.1, $\boldsymbol{f} : \mathbb{R}^{n^{\mathrm{x}}} \times \mathbb{R}^{n^{\mathrm{u}}} \to \mathbb{R}^{n^{\mathrm{x}}}$ is representing the dynamics of the system defined in (4.11), $\boldsymbol{x}_0$ is the initial (measured) state of the system, and $\boldsymbol{g}(\boldsymbol{x}, \boldsymbol{u}) : \mathbb{R}^{n^{\mathrm{x}}} \times \mathbb{R}^{n^{\mathrm{u}}} \to \mathbb{R}^{n^{\mathrm{c}}}$ are the mixed state-control path constraints defined by concatenating the friction cone constraints (4.20), the kinematic constraints (4.22) as well as the complementarity constraints (4.12), where the latter is defined via the contact sequence. In order to solve the OCP (4.29), we follow a direct multiple shooting approach. For the details, we refer the reader to Chapter 1.

4.3 Realization of the Motion on the Robot

In Figure 4.2, we visualize the work flow for motion generation using centroidal dynamics. So far, we have only discussed the OCP formulation in the scheme. We explain in the remainder of this section the last two building blocks, the contact planner and generalized inverse kinematics.

4.3.1 Contact Planner

For the formulation of the OCP the contact sequence has to be specified. This includes the definition of the contact state, active or not, for each contact on every time interval on the horizon. Additionally, it is required to specify which body is in contact including the position of the contact $\boldsymbol{p}$, the local friction coefficient μ, and the contact normal vector $\boldsymbol{n}$, which we describe via the respective transformation $\boldsymbol{Q}$. The contact sequence is then the set of all such contact specifications. Typically, the contact sequence is realized by employing a dedicated contact planner as presented in [6, 172]. Here, the contact sequence is predefined, but an extension to additionally optimizing the position of the contacts on a set of planar contact surfaces is possible.

4.3.2 Generalized Inverse Kinematics

The final whole-body motion is generated by applying the stack of tasks scheme implementing generalized inverse kinematics (GIK) as presented in [117]. Following this, the end-effector trajectories, i.e., the hand and the feet, as well as the CoM trajectories are retrieved from the results of the OCP by means of piecewise and continuous B-splines. Given the interpolated CoM, the root orientation and end-effector trajectories the SoT framework computes joint trajectories $\boldsymbol{q}, \dot{\boldsymbol{q}}$ for all the DoFs of the system. This is done by specifying distinct tasks for the SoT framework as already presented in Section 3.4.2.

The tasks are defined as a simple PD controller tracking a corresponding reference trajectory. Here, we define a task $\mathrm{T}_{\mathrm{CoM}}$ tracking the CoM trajectory along the x, y, z axes, a task $\mathrm{T}_{\mathrm{RH,LH,RF,LF}}$ for each end-effector position and orientation specification, a task T_{W} controlling the orientation of the waist, and a task $\mathrm{T}_{\boldsymbol{q}_0}$ regulating the posture of the robot around a nominal posture.

A hierarchy between the different tasks defined by means of the lexicographic order

$$\mathrm{T}_{\mathrm{CoM}} \prec \mathrm{T}_{\mathrm{RH,LH,RF,LF}} \prec \mathrm{T}_{\mathrm{W}} \prec \mathrm{T}_{\boldsymbol{q}_0}.$$

The dynamical consistency of the solution with respect to the robot model and the contact forces is implicitly given by the properties of the CoM trajectory computed by the OCP.

The corresponding contact forces and joint torques are then reconstructed in simulation employing a whole-body model of the humanoid under consideration. For each time step, the contact forces $\boldsymbol{\lambda}$ are computed as the minimal forces corresponding to the joint trajectory $\boldsymbol{q}$, $\dot{\boldsymbol{q}}$, $\ddot{\boldsymbol{q}}$ and respecting the contact model by

$$\min_{\boldsymbol{\lambda}_1,\dots,\boldsymbol{\lambda}_{|\mathrm{I}|}} \|\mathrm{RNEA}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}) - \sum_{i \in \mathrm{I}} \boldsymbol{J}_i^T \boldsymbol{\lambda}_i\|^2,$$

where $\boldsymbol{J}_i$ denotes the kinematic Jacobian mapping the contact force into the dynamics of the robot, such that $\boldsymbol{\lambda}_i \in \mathcal{K}_i^{\mu_i}$, $i \in \mathrm{I}$, where RNEA denotes the recursive *Newton-Euler* algorithm [114], which is presented in detail in Section 2.3.1. The motion can be checked to be dynamically consistent if the residual is zero for all time instants of the movement. For robots with a different weight distribution than *HRP-2*, the application of RNEA yields only an approximation of the change of total momentum, c.f. Remark 3.15 in Section 3.4.1.

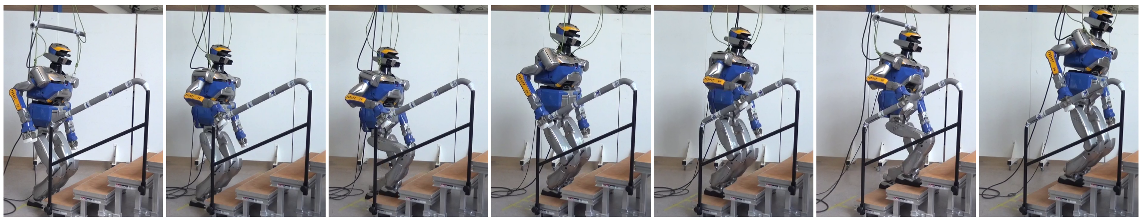


Figure 4.4: Set of contact stances realized by the humanoid robot *HRP-2* using the proposed method. The experiment has been realized five times in a row without any failure. The companion video of our article [106], shows the realization of the experiment.

4.4 Results

The approach described above enabled multi-contact stair climbing of the humanoid robot *HRP-2*. We realized a stair climbing task on stairs of a height of 15 cm with additional support of a handrail. So far, the focus of research only considered small step sizes, e.g. [74], or single overstepping motions, e.g. [111] shows a single step of 15 cm height and [100] an overstepping motion of a large obstacle. However, this is not comparable to stair climbing with repetitive steps in terms of stress on the hardware.

Trying to achieve the same stair climbing task of 15 cm height with state-of-the-art methods from [129], prior to this work, was problematic and could only be executed with fully charged batteries to compensate current peaks in the joint actuation. This leads to random shutdown of the robot during execution and prohibited multiple executions in a row.

The proposed method of this chapter easily succeeded in the stair climbing task fives times in a row without additional help of fully charged batteries as the current peaks could be compensated by additional handrail support and additionally making the resulting much more efficient in terms of overall power consumption.

4.4.1 Experimental Setup

We consider the experimental setup depicted in Figure 4.1 as proof of concept. The task is to make the humanoid robot *HRP-2* climb a set of stair treads with a handrail as additional support. The height, the depth, and the width of the steps are respectively 15 cm, 30 cm, and 1 m. The handrail is a cylinder with a diameter of 3 cm.

Starting and ending in a half-sitting configuration of the robot, the motion is divided into three phases that are executed twice:

P1 the right hand establishes contact with the handrail

P2 the right foot is set on top of the stair in front

P3 the left foot is lifted and placed next to the right one on the stair .

The robot is in double support phase during the movement of an end-effector, i.e., his hand or foot, and shortly in triple support phase during the CoM transition phase. We denote by CoM transition phase, the phase between phase **P2** and **P3** in which the CoM is transitioned from on stable configuration to the next. See Figure 4.4 for a snapshot series of the humanoid robot *HRP-2* performing the task.

The timing of the phase durations is crucial for the robot because they implicitly define the velocity of each limb. In comparison to ground-level walking, where the period of single support and double support are usually around 0.7 s and 0.1 s respectively, we adapted the timing schedule in this example, because the robot has to go through a larger

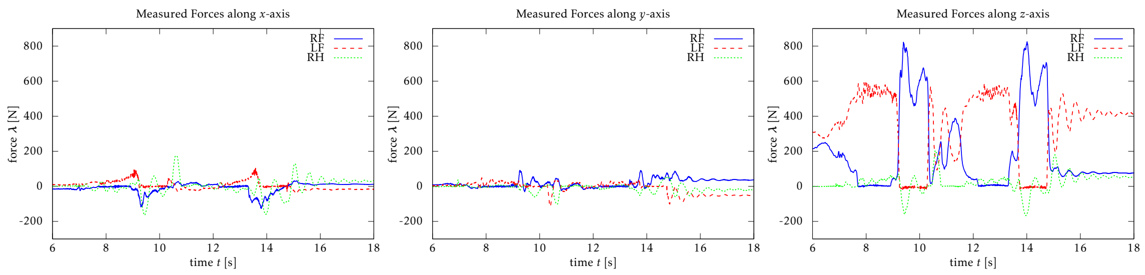


Figure 4.5: Measured contact forces during the motion of *HRP-2* during stair climbing with handrail support, as depicted in Figure 4.4. Forces along local x-axis in solid blue, along y-axis in dashed red, and along z-axis in dotted green.

distance at each phase. Keeping the same schedule as in ground-level walking makes the robot reaching its actuators limits in speed and current more likely. Here, the robot moves an end-effector in 1.4 s and moves its CoM position in 0.1 s during a transition phase.

The change of angular momentum term depending on the limb motion σ, c.f. Section 4.1, was of small effect for the stair climbing motion and was therefore neglected during motion generation. While possible in principle, we did not define a tracking task for σ in the hierarchy of the generalized inverse kinematics as presented in Section 4.3.2.

In the control chain, only the OCP part is not yet run in real time. In fact, the computation time for the motion in the video attachment is $\sim$ 30 min in order to solve the OCP until convergence. The large computational foot print is due to

- calculating the motion all-at-once on the whole preview-window of 18.4 s,
- an over parametrization of the problem (3003 DoFs),
- performing several sequential quadratic programming (SQP) iterations until convergence of the solution,
- and not exploiting the intrinsic sparsity resulting from the template model.

An analysis of the computational time suggests that the application of multi-level real-time iteration of nonlinear model predictive control (NMPC) as introduced together with a reduction of the planning horizon can easily lower the computational time to 50 ms considered as real-time feasible control on the robot, c.f. [24, 101]. Future work will include a tailored implementation of the algorithm considering these bottlenecks and allow a real-time execution on the robot. Despite this, the inverse dynamics run in 1 ms on the on-board CPU of *HRP-2* (*Intel(R) Core2*™ Duo *E*7500, frequency 2.8 GHz, 1 core, 3 MB of cache) under *Ubuntu 10.04 LTS*.

Forces During Contact Transition

Figure 4.5 shows the forces measured during one experiment, while Figure 4.6 shows a comparison between measured and planned forces. Almost no torque is applied at the level of the feet, therefore we do not show the respective plots. The propulsion of the robot is more visible in the tangential forces along the x- and y-axis. A comparison between the subfigures of Figure 4.5 shows that almost all the forces are acting along the z-axis. Therefore, we focusing the analysis on the z-axis components shown in Figure 4.6.

At the beginning of the motion the robot is stable on its feet and no other contact with its environment is established. This initial phase of 5 s is not shown in the graph. After the robot starts to move, the hand comes into contact with the handrail causing perturbations in the feet force distribution at 6 s. The next transition appears just before 8 s. The robot shifts its CoM to the left foot and puts the right foot on the first step. Then the robot has

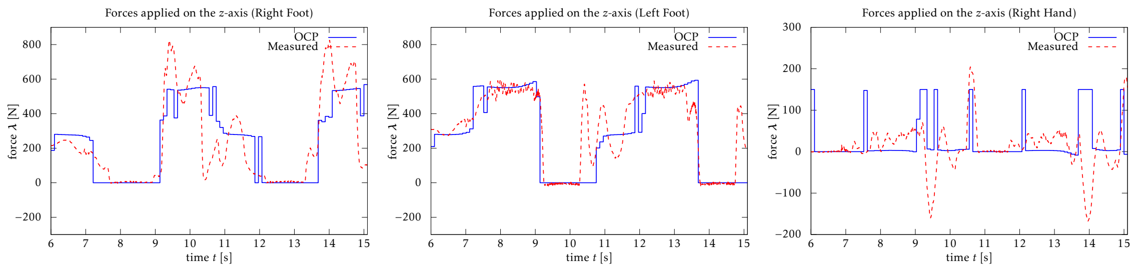


Figure 4.6: Comparison between the planned contact forces of optimal control problem solution (blue solid) and the measured contact forces (red dashed) during the motion of *HRP-2* depicted in Figure 4.4.

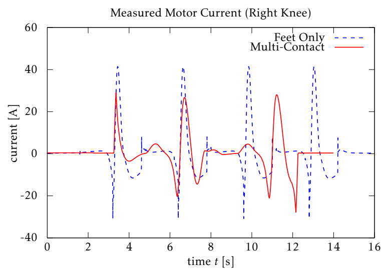


Figure 4.7: Comparison of applied current between motions employing feet only (blue dashed) and multiple contacts using feet and one hand (red solid) for climbing a 15 cm staircase.

three contacts with its environment and starts to utilize the hand contact. It pushes with the right leg and pulls with the hand to climb the stair. This particular motion excites a flexible part located under the ankle of *HRP-2*.

Between 10 s and 12 s the flexibility perturbs the system but the forces on the hand compensate for it. The hand contact stabilizes the robot to be able to move the hand towards the next grasping position at 12 s, where the robot is back to a stable state again. During the hand movement, the robot's CoM is effected by the flexibility exertion but not enough to fall down due the stabilizing influence of the grasping contact before and after the double support phase.

The motion is repeated once. The only difference is that the hand does not release contact with the handrail at the end of the motion. This helps the robot to stabilize and go back to an equilibrium state.

The right hand has an important role as it can realize forces up to 200 N and more remarkably also exerts negative forces at 9 s and 14 s, i.e., the robot also pulls itself utilizing the grip on the handrail.

4.4.2 Current Consumption

A severe limitation in climbing stairs with foot contacts only for human-sized humanoid robot is the current consumption. After performing a large number of experiments on a 15 cm staircase using a different algorithm from [129], it appears that the rate of success was highly dependent on the battery charge level. Based on this observation, and using a model of the robot actuator, the maximum amplitude of the current was detected to be 40 A on the right knee as depicted in Figure 4.7. We show the right knee current only, because this is the leg that bears the most stress during the motion. It is mostly due to the fact that the weight shifting is performed by one support leg. Using several contact points during weight shifting allows to distribute the load across several actuators. Therefore, the current asked for the right knee for the same motion using multiple contacts does not exceed 30 A. This allows performing the motion depicted in Figure 4.4 five times in a row even with low battery charge level.

4.5 Summary

In this Chapter, we have presented an approach to multi-contact motion generation for humanoids using our centroidal dynamics template model. The approach was successfully implemented on the humanoid robot *HRP-2* enabling a stair climbing task. For this case, we derived a reduced model based on the centroidal dynamics of the humanoid that generalizes in the number of contacts. This model qualifies the proposed approach for being real-time feasible, because it reduces the number of states to a minimum compared to other approaches considering all the degrees of freedom of the humanoid model. In order to generate whole-body motions fully leveraging the reduced model, a complete framework of motion generation was presented. Following this, from a pre-defined contact sequence of the end-effectors in contact with the environment an optimal centroidal motion, contact forces and end-effector trajectories are computed by solving a dedicated optimal control problem formulation.

The resulting trajectories are post-processed by a generalized inverse kinematics that defines a hierarchy of distinct tracking tasks for the optimal reference trajectories such that the resulting joint trajectories can be directly applied in simulation or the actual robot.

The feasibility of our approach was demonstrated by experiments of *HRP-2* climbing stairs. The generated motion was performed five times in a row on the robot, which was not possible before using the state of the art of motion generation prior to this work. Additionally, by utilizing multiple-contacts for motion generation the overall power consumption can be reduced and the distribution onto the whole-body makes the motion more efficient and robust.

5 Nonlinear Model Predictive Control for Humanoid Motion Generation

The following chapter proposes a novel nonlinear model predictive control (NMPC) strategy that combines different multi-level real-time iterations (MLRTIs) in order to provide fast feedback with minimal delay while improving the overall control rate even for complex multi-body system (MBS) models of humanoids. By using the whole-body dynamics of the humanoid our method enables a direct interaction with the low-level control of the robot without the need of intermediate step in contrast to reduced or template models. Additionally, the whole-body dynamics enable direct access to the quantities of interest as for example the respective contact forces and joint torques. This allows to easily model tasks as well as gain a better understanding of the actual motion of the robot. For the purpose of a proof of the concept, the approach was implemented on the robot *Leo* realizing a squatting task for the first time using NMPC running in real-time.

The goal of motion generation for humanoids is to realize *human-like* task performance on the robot while considering the real-time constraints dictated by the robotic platform. Today's humanoid robots are complex mechanical systems with many degrees of freedom (DoFs) that are built to achieve locomotion skills comparable to humans. In order to synthesize whole-body motions, real-time capable direct methods of optimal control are a subject of contemporary research. To this end, NMPC is the method of choice to realize motions on the physical robot using model-based optimal control, because a model based on rigid-body dynamics (RBD) comes closest to a first-principles model considering the mechanics of the structure of a humanoid as well as the principles affecting its motion. A representation of the humanoid as MBS and resolving its RBD includes all the relevant quantities like joint torques as well as contact forces that are also known from motion analysis. However, the complexity of the problem results in a high computational time that falls short of the expectations of robotic experimenters and control engineers. Therefore, while NMPC for motion generation is an active field of research in both engineering as well as computer animation, only few realizations on the actual robotic platform have been presented so far, for example [101].

An analysis of the different phases of state-of-the-art NMPC iterations reveals that the bottleneck of computation is the preparation phase, i.e., the forward simulation of the multi-body dynamics and the evaluation of derivative information, while the feedback is reasonably fast. Therefore, in this chapter, we propose a control strategy based on switching two controllers based on different MLRTI of NMPC. One controller provides feedback based on linear model predictive control for the last known linearization of the nonlinear control problem. The other controller is given actual initial values and computes a new linearization of the problem. When the problem is successfully linearized, a control feedback based on NMPC is provided and the linearization is communicated to the linear MPC controller. Subsequently, feedback is provided again using linear model predictive control (LMPC) based on this new linearization while the other controller is busy relinearizing the problem. The contribution of this thesis is this new and beneficial combination and parallelization of both linear and nonlinear methods for feedback control.

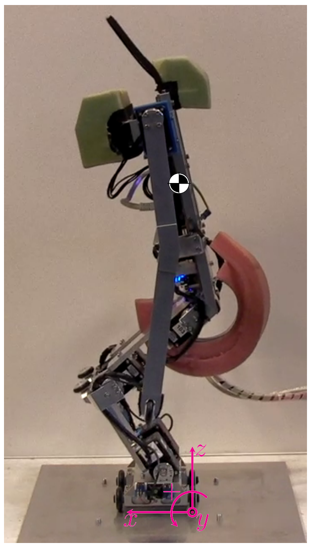


Figure 5.1: *Leo* robot of *Delft University of Technology* performing a squatting motion with annotations. Annotation shows the world frame (magenta) as well as the root frame origin (circle with black and white sectors).

The proposed strategy is a key to running NMPC on the robotic platform in real time and the performance of the proposed control strategy is evaluated on the 2D robot *Leo* of *Delft University of Technology*, c.f. [155]. The performed task, squatting of the robotic platform, is realized via tracking of switching setpoints of the NMPC controller.

The following contributions were made in this thesis:

- Development of a nonlinear model predictive control framework combining different levels multi-level real-time iteration in a thread-based parallelization of two controllers.
- By efficiently reusing control problem linearizations of the last iteration one controller provides fast feedback, while the other controller prepares the next nonlinear step.
- In this way, the control rate as well as the overall performance could be drastically improved.
- For the first time, the NMPC scheme enables a realization of a squatting task on the actual 2D-robot *Leo* of *Delft University of Technology*, which was not possible using a conventional nonlinear model predictive control scheme.
- During the experiment, an improvement of the control rate by a factor of 10–16 up to 190 Hz was achieved.

In Figure 5.2, the general idea of NMPC for whole-body motion generation of humanoids is depicted. The NMPC framework requires the formulation of an OCP that includes a dynamic model of the considered humanoid, robot specific details, e.g. joint limits, limits on the joint actuation, possible contact points, as well as task-specific user input, for example setpoints to be tracked by the controller.

The dynamic model allows to predict the behavior of the robot on a time horizon given measurements of the current state of the robot $\hat{\boldsymbol{q}}, \hat{\dot{\boldsymbol{q}}}$. The NMPC scheme then computes optimal joint angles, velocities and torques to realize the given in task in the best possible way considering the real-time constraints dictated by the sampling rate of the low-level control of the robot. The computed trajectories can then be directly passed to this low-level control as reference values for either position controlled, torque controlled motors

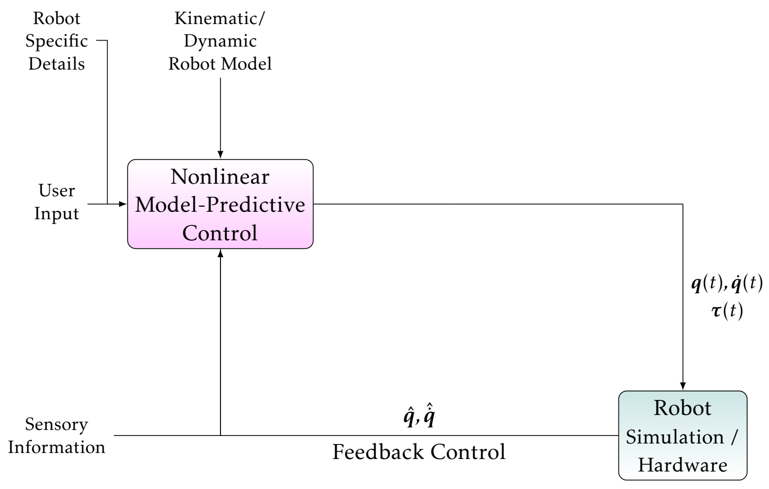


Figure 5.2: Visualization of the general approach of nonlinear model predictive control for motion generation as applied in this thesis. From robot specific details, external user input and a dynamic model of the humanoid an OCPs for a nonlinear model predictive controller is formulated. The output of NMPC, in the form of optimal joint angles, velocities and torques $\boldsymbol{q}$, $\dot{\boldsymbol{q}}$, $\boldsymbol{\tau}$ trajectories, can directly applied to the low-level control of the humanoid as reference quantities. Feedback is then established on the measured configuration of the robot $\hat{\boldsymbol{q}}$, $\hat{\dot{\boldsymbol{q}}}$ as well as additional external or internal sensory information directly.

of the robot or a simulation environment. The loop is then closed by providing the actual configuration $\hat{\boldsymbol{q}}$, $\hat{\dot{\boldsymbol{q}}}$ as well as additional sensory information, for examples sensors estimating the position and orientation of the humanoid with respect to its environment, to the NMPC scheme for the next iteration.

In the remainder of this chapter, we will revisit the relevant building blocks of the scheme depicted in Figure 5.2. Therefore, we will briefly revisit the whole-body dynamic model for motion generation. Afterwards, we focus on advanced methods of NMPC in the form of different multi-level real-time iteration that are applied in our control strategy. Following this, we propose a novel approach combining two multi-level real-time iteration in a concurrent way, such that the NMPC scheme is always responsive and able to provide feedback even while computing the next linearization of the internal optimal control problem. This is followed by the numerical and experimental results obtained from implementing a squatting task on the humanoid robot *Leo*. The chapter is concluded with a summary.

5.1 Whole-body Dynamic Model

In this section, we briefly revisit the whole-body dynamics of a humanoid based on the RBD of the respective MBS representation of the humanoid. As introduced in Chapter 1, the state of the art describes both humans and humanoid robots as MBS. The respective dynamics of the resulting MBS are then derived by RBD calculus as presented in Chapter 2. In principle, the application of the standard RBD approach for solving the dynamics of the MBS is equivalent to solving the equation of motion in descriptor form given by

$$\begin{bmatrix} \boldsymbol{H}(\boldsymbol{q}) & \boldsymbol{J}(\boldsymbol{q})^T \\ \boldsymbol{J}(\boldsymbol{q}) & 0 \end{bmatrix} \begin{bmatrix} \ddot{\boldsymbol{q}} \\ -\boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \boldsymbol{S}^T \boldsymbol{\tau} - \boldsymbol{c}(\boldsymbol{q}, \dot{\boldsymbol{q}}) \\ -\boldsymbol{\gamma}(\boldsymbol{q}, \dot{\boldsymbol{q}}) \end{bmatrix}. \tag{5.1}$$

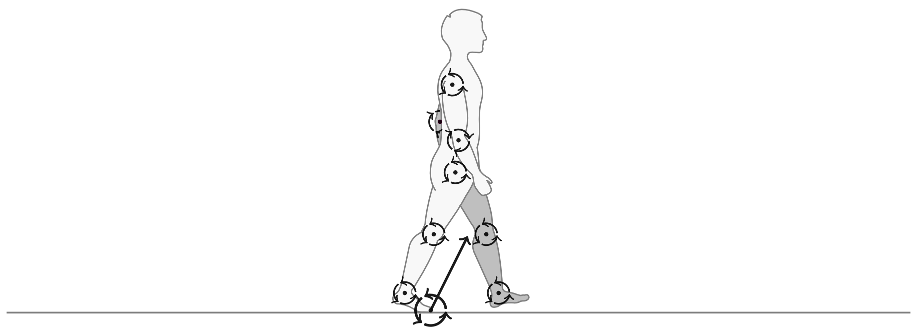


Figure 5.3: Visualization of a humanoid represented as multi-body system that considers each limb modeled as rigid body connected via joints. Contacts with the environment are modeled via contact points. The whole-body dynamics then examine the interplay between the locally applied torques at joint level with the resulting contact forces and moments in the contact points as well as the resulting whole-body motion.

Additionally, the force-contact complementarity, c.f. Equation (1.6), has to be respected as well as the discontinuities on establishment of a contact due to inellastic collision of a rigid body with its environment, c.f. Equation (1.7). We refer the reader to Chapter 1 for the details concerning the definitions, the implementation as well as embedding the dynamics in the NMPC or OCP formulation.

5.2 Multi-level Real-time Iterations of Nonlinear Model Predictive Control

In Section 1.5 of Chapter 1, we have already introduced a direct approach to optimal control, the concept of NMPC and the idea of real-time iterations. For more details, the reader is kindly referred to the respective sections of this thesis.

Here, we briefly revisit the most important aspects and extend the explanation to multi-level real-time iteration of NMPC that are required to derive the novel control strategy developed in this work. NMPC is a closed-loop control strategy in which the control action at the current sampling instant is computed by solving an open-loop OCP over a finite prediction horizon $\mathcal{T} := [0, T]$ online. The respective OCP formulation that we focus on in the remainder of this chapter is given by Equation (1.12) of Definition 1.4.

The differential states of the OCP are denoted by $\boldsymbol{x}(\cdot) \in \mathbb{R}^{n_x}$, and are implicitly defined by the chosen model dynamics in the form of an ordinary differential equation (ODE) as formulated in (1.12c). In order to find an optimal solution of the OCP, we strive to find the optimal inputs in the form of controls $\boldsymbol{u}(\cdot) \in \mathbb{R}^{n_u}$ and free parameters $\boldsymbol{p} \in \mathbb{R}^{n_p}$.

In order to solve the infinite dimensional OCP, a direct and all-at-once approach subdivides the time horizon $\mathcal{T}$ into N subintervals $[t_i, t_{i+1}], 0 \leqslant i \leqslant N$, and discretizes the control trajectory by means of constant control parameters $\boldsymbol{q} \in \mathbb{R}^{N n_u}$ on the time grid as presented in Section 1.5.1. According to [13], the direct multiple shooting approach further parametrizes the state trajectory by means of the solution of separate local initial value problems defined on the same time grid as the control discretization by introducing additional variables $\boldsymbol{s} \in \mathbb{R}^{(N+1)n_x}$ as described in Section 1.5.2. Continuity of the trajectories in the solution is established by matching conditions enforcing equality across the time nodes of the grid.

From this discretization and parametrization, a large but structured nonlinear programming problem is obtained that can be solved efficiently with tailored structure-exploiting sequential quadratic programming (SQP) methods, which was discussed in Section 1.5.3. Observing that the underlying OCP of the NMPC problem is linearly dependent on $\hat{\boldsymbol{x}}_0$, $\hat{\boldsymbol{p}}$ due to (1.12d),(1.12e), we choose to write the nonlinear program (NLP) in parametric form, in contrast to the formulation in Equation (1.19), as

$$\begin{aligned} \min_{\boldsymbol{w}} \quad & \varphi(\boldsymbol{w}) && (5.2a)\\ \text{s.t.} \quad & 0 = \boldsymbol{c}(\boldsymbol{w}) + \boldsymbol{P}(\hat{\boldsymbol{x}}_0, \hat{\boldsymbol{p}}), && (5.2b)\\ & 0 \leqslant \boldsymbol{d}(\boldsymbol{w}), && (5.2c) \end{aligned}$$

with a collection of the free variables from the discretized and parametrized state and control trajectories $\boldsymbol{s} := [\boldsymbol{s}_0, \ldots, \boldsymbol{s}_N, \boldsymbol{s}_{N+1}]$, $\boldsymbol{q} := [\boldsymbol{q}_0, \ldots, \boldsymbol{q}_N]$ as well as free parameters in the form of the iterate $\boldsymbol{w} := [\boldsymbol{s}, \boldsymbol{q}, \boldsymbol{p}]$. The projector $\boldsymbol{P}$ aligns the initial values and parameter with the rest of the equality constraints $\boldsymbol{c}$. Additional information on the exact formulation is given in Section 1.5.3.

In the following paragraph, we revisit the necessary details on the solution of NLP (5.2) by means of SQP method. A full-step SQP method employs, in every iteration k, a quadratic approximation of the NLP in the form of a quadratic program (QP) and, starting with an initial guess $(\boldsymbol{w}^0, \boldsymbol{\lambda}^0, \boldsymbol{\mu}^0)$ of the primal and dual variables of the NLP, performs a step $\boldsymbol{w}^{k+1} = \boldsymbol{w}^k + \Delta \boldsymbol{w}^k, \boldsymbol{\lambda}^{k+1} = \boldsymbol{\lambda}_{QP}, \boldsymbol{\mu}^{k+1} = \boldsymbol{\mu}_{QP}$ by using the solution $(\Delta \boldsymbol{w}^0, \boldsymbol{\lambda}_{QP}, \boldsymbol{\mu}_{QP})$ of the QP given by

$$\begin{aligned} \min_{\Delta \boldsymbol{w}} \quad & \tfrac{1}{2} \Delta \boldsymbol{w}^T \boldsymbol{B}(\boldsymbol{w}^k) \Delta \boldsymbol{w} + \Delta \boldsymbol{w}^T \boldsymbol{b}(\boldsymbol{w}^k) && (5.3a)\\ \text{s.t.} \quad & 0 = \boldsymbol{c}(\boldsymbol{w}^k) + \boldsymbol{C}(\boldsymbol{w}^k)\Delta \boldsymbol{w} + \boldsymbol{P}(\hat{\boldsymbol{x}}_0, \hat{\boldsymbol{p}}), && (5.3b)\\ & 0 \leqslant \boldsymbol{d}(\boldsymbol{w}^k) + \boldsymbol{D}(\boldsymbol{w}^k)\Delta \boldsymbol{w}, && (5.3c) \end{aligned}$$

where we denote the Jacobians of the inequality and equality constraints by $\boldsymbol{C} = \frac{\mathrm{d}\boldsymbol{c}}{\mathrm{d}\boldsymbol{w}}(\boldsymbol{w})$, $\boldsymbol{D} = \frac{\mathrm{d}\boldsymbol{d}}{\mathrm{d}\boldsymbol{w}}(\boldsymbol{w})$, respectively. The Hessian of the Lagrangian of the NLP is $\boldsymbol{B}(\boldsymbol{w})$, and the vector $\boldsymbol{b}(\boldsymbol{w})$ is the gradient of the objective.

State-of-the-art NMPC methods based on nonlinear programming rely on the real-time iteration scheme according to [35] to compute feedback in real-time. Subsequent problems only differ in the values of $\hat{\boldsymbol{x}}_0$, $\hat{\boldsymbol{p}}$ and this dependency is only linear in the QP (5.3). Due to the linearity, by means of the so-called *initial value embedding* methodology, i.e., a difference in $\boldsymbol{s}_0 \neq \hat{\boldsymbol{x}}_0$ and $\boldsymbol{p} \neq \hat{\boldsymbol{p}}$ due to initializing the problem with the state and control information of the last solution can be satisfied after a single full *Newton* step. The next iterate then represents a first-order tangential predictor of the solution, c.f. [38].

In this way, computationally expensive parts can be separated from time-critical ones and the computational delay of the feedback is reduced to the time required to solve a single QP, such that by careful initialization a separation into three phases of the SQP step is possible as follows:

- **Preparation:** Setup of QP (5.3), i.e., computation of $\boldsymbol{B}$, $\boldsymbol{b}$, $\boldsymbol{C}$, $\boldsymbol{c}$, $\boldsymbol{D}$ and $\boldsymbol{d}$ from last iterate $\boldsymbol{w}$.
- **Feedback:** Triggered by $\hat{\boldsymbol{x}}_0, \hat{\boldsymbol{p}}$, compute feedback control $\boldsymbol{u}^\star(t; \hat{\boldsymbol{x}}_0, \hat{\boldsymbol{p}}), t \in [t_0, t_1]$ by solving QP (5.3).
- **Transition:** perform *Newton* step, i.e., compute new iterate $\boldsymbol{w}^{k+1} = \boldsymbol{w}^k + \Delta \boldsymbol{w}^k$.

5.2.1 Multi-level Real-time Iterations

In the remainder of this section, the focus is on advanced multi-level real-time iterations, which further the above ideas by dividing the real-time iteration (Preparation, Feedback and Transition) into sub steps that can provide feedback even faster by evaluating only parts of the required Jacobian information, c.f. [3, 14, 56, 96]. Here, we revisit the levels proposed in [14] that are required for our implementation. In order to provide feedback even faster, the idea of the four different levels is to only update selected parts of the linearization during the preparation phase. This can be understood as updating the respective quantities in the QP (5.3), i.e., the evaluation of all, parts or none of $\boldsymbol{B}$, $\boldsymbol{b}$, $\boldsymbol{C}$, $\boldsymbol{c}$, $\boldsymbol{D}$ and $\boldsymbol{d}$. We focus on the two extremes of updating the full information and leaving out the preparation phase, i.e., the levels "D" and "A".

Level D

Multi-level real-time iterations of level D realize the preparation phase of a full SQP step, i.e., we evaluate the gradient of the objective $\boldsymbol{b}(\boldsymbol{w}^k)$, the constraint residuals $\boldsymbol{c}(\boldsymbol{w}^k)$, $\boldsymbol{d}(\boldsymbol{w}^k)$ as well as the respective Jacobians $\boldsymbol{C}(\boldsymbol{w}^k)$, $\boldsymbol{D}(\boldsymbol{w}^k)$ and computing a new Hessian (approximation) $\boldsymbol{B}(\boldsymbol{w}^k)$ using the latest iterate $(\boldsymbol{w}^k, \boldsymbol{\lambda}^k, \boldsymbol{\mu}^k)$. In this way, a full, new quadratic approximation of the NLP at the last iterate is computed, i.e., a new QP (5.3) is built from this information. After the solution of the QP is available, the feedback control is sent to the process and the SQP iteration is finalized.

Level A

Multi-level real-time iterations of level A realize LMPC using the most recent linearization provided by a completed D iteration. That is, the most recent set of matrices and vectors $\hat{\boldsymbol{B}}$, $\hat{\boldsymbol{b}}$, the constraints $\hat{\boldsymbol{C}}$, $\hat{\boldsymbol{c}}$, $\hat{\boldsymbol{D}}$, $\hat{\boldsymbol{d}}$ are kept fixed in the QP (5.3). From this, and given new estimates of the initial states $\hat{\boldsymbol{x}}_0$ as well as parameters $\hat{\boldsymbol{p}}$, a feedback control is computed by solving the QP. Level A iterations can be performed without any evaluation of the nominal or derivative information and consist of only a solution of the already prepared QP.

5.3 Combining Multi-Level Real-time Iterations

The application of NMPC for whole-body motion generation for humanoids includes the evaluation of the RBD of the MBS model of the humanoid as well as the evaluation of the respective sensitivities in every level D iteration. The complexity of the considered model representation of the humanoid as well as the inherent numerical ill-conditioning of RBD in general renders this part to be the bottleneck compared to the actual solution of the QP in the feedback phase of each SQP iteration. The feedback phase can be accelerated by common structure exploiting methods, c.f. [13, 56, 110]. However, this will increase the time required for the preparation phase. This ratio of computational time is already present in the case of small humanoids as presented later in Section 5.5.3 for robot *Leo*.

In sum, this means that feedback control, i.e., the solution of (5.3), can be computed fast and cheap while the re-computation of the quantities of the QP, i.e., $\boldsymbol{b}(\boldsymbol{w}^k)$, $\boldsymbol{c}(\boldsymbol{w}^k)$, $\boldsymbol{d}(\boldsymbol{w}^k)$ as well as $\boldsymbol{C}(\boldsymbol{w}^k)$, $\boldsymbol{D}(\boldsymbol{w}^k)$ and $\boldsymbol{B}(\boldsymbol{w}^k)$ form the latest available iterate $(\boldsymbol{w}^k, \boldsymbol{\lambda}^k, \boldsymbol{\mu}^k)$, is computationally expensive and known to be the reason for violating the real-time constraints dictated by the respective hardware.

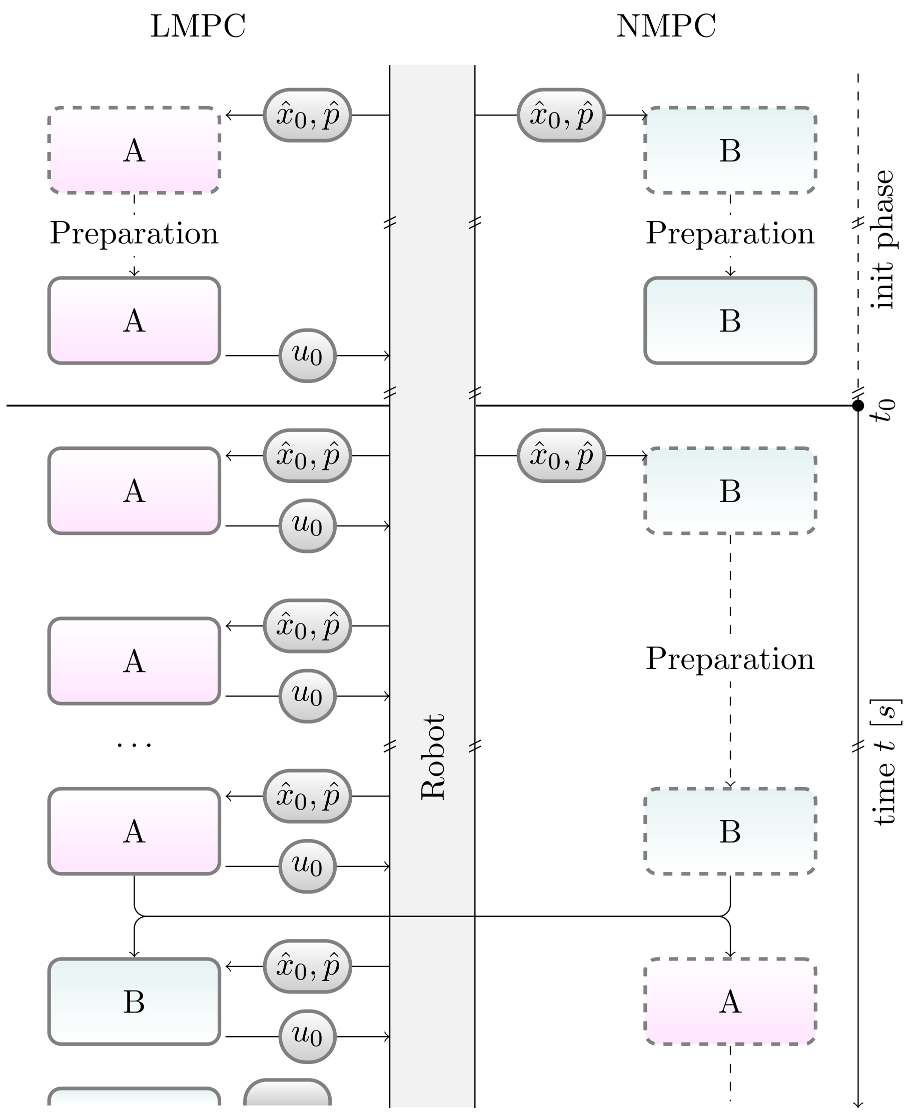


Figure 5.4: Schematic of the control approach based on multi-level real-time iteration as derived in this thesis. In an init(ialization) phase, both controllers (A,B) are prepared by estimates ($\hat{\boldsymbol{x}}_0$, $\hat{\boldsymbol{p}}$) from the robot. The concurrent controllers provide feedback $\boldsymbol{u}_0$ either with linear model predictive control (LMPC) or re-linearize and provide nonlinear feedback once, i.e., nonlinear model predictive control (NMPC). LMPC feedback is queried as long as NMPC thread is in preparation. As soon as the NMPC thread is ready again, roles are switched and the scheme is continued, which is indicated by a cropped scheme at the bottom.

Following the ideas proposed in [56], a huge benefit of the separation of feedback and preparation phase is that these phases can be implemented as independent processes with separate memory. In this way, it is possible to repeatably provide intermediate feedback using MLRTI of level A by reusing the latest linearization in the form of a solution of QP (5.3), while a new linearization is computed in the preparation phase of a level D iteration. In contrast to the theoretical scheme proposed by [56], we actually realize this idea by two concurrent threads either using level A or level D real-time iterations. The contribution of this work is an implementation of this control scheme as depicted in Figure 5.4.

First, both controllers are prepared in separate threads and receive initial states $\hat{\boldsymbol{x}}_0$ and parameters $\hat{\boldsymbol{p}}$ in a common initialization phase. While the threads run concurrently, each of the controllers has a distinct role in providing feedback $\boldsymbol{u}_0$ either using level A iterations and reusing the latest linearization (LMPC), or in using level D iterations with a full preparation phase (NMPC).

The level D iterations are subject to a non-negligible computational delay such that linear feedback can be provided multiple time using the latest linearization. Following this, as soon as the relinearization has been triggered by initial values, we provide linear feedback through queries to the level A iterations thread.

As soon as the level D thread is ready, i.e., several level A queries were served, we immediately switch roles of the two threads and provide full NMPC feedback by querying the level D thread. In the meantime, the former LMPC thread is queried again but this time the re-linearization at the current iterate is initiated. The level A thread then takes care of providing the feedback until the next level D iteration is ready. This scheme is then continued until the process is stopped. The proposed approach is implemented on top of OCP solver *MUSCOD-II*, c.f. [109, 110], and validated on the robotic platform *Leo*.

Our approach generalizes the application of LMPC and a relinearization of the current problem online. It is possible to exploit a task known beforehand, e.g. for squatting it is possible to linearize the optimal control problem at the setpoints and compose to level A controllers that can then be used to implement LMPC for the problem by switching both the setpoints and the controller. However, this will limit the proposed control approach to the considered task only and will therefore not be transferable to other tasks. In contrast to this, our approach is not subject to this problem and might also be used for more complex tasks such as walking.

5.4 Problem Formulation

The state of the art models a robot as a MBS and its RBD then define the respective mathematical model of the robot, as presented in Section 1.3. Here, we consider the robot *Leo* of TU Delft. The required dynamic properties of the links, i.e., masses, inertias and centers of mass (CoMs), were taken from [155]. We use the software library *Rigid Body Dynamics Library* (RBDL) [49, 50] to implement the model and to efficiently evaluate the forward dynamics of the model as required by the OCP formulation presented in Section 1.4.

In Figure 5.1, the world frame is depicted, where the x-axis exists along the 2D walking motion pointing forward, the z-axis point upward and, in order to have a Cartesian frame, the y-axis is pointing into the figure such that a positive rotation around the y-axis results in a counter-clock-wise motion.

The kinematic structure of the robot can be modeled in different ways. Here, we employ a fixed-base model that describes the kinematic chain from foot over ankle, knee and hip to the root and the arm of the robot. Furthermore, it is possible to exploit the symmetry for the task as in order to achieve squatting the legs and feet have to be aligned in parallel.

5.4.1 States and Controls

The dynamic model is implemented as ODE in the OCP (1.12c) of the NMPC problem (1.12) by order reduction, i.e., implemented on acceleration level by evaluating the forward dynamics (FD), $\ddot{\boldsymbol{q}} = \mathrm{FD}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{\tau})$, and integrated twice to obtain joint positions and velocities $\boldsymbol{q}$, $\dot{\boldsymbol{q}}$.

Following the formulation of the NMPC problem (1.12), the differential states $\boldsymbol{x}$ are in this case the joint positions $\boldsymbol{q}$ and velocities $\dot{\boldsymbol{q}}$, i.e.,

$$\boldsymbol{x}(t) = \begin{bmatrix} \boldsymbol{q}(t) \\ \dot{\boldsymbol{q}}(t) \end{bmatrix}, \quad \forall t \in \mathcal{T}.$$

The controllable variables of the system $\boldsymbol{u}$ are the input voltages of the servo motors $\boldsymbol{v}$, i.e.,

$$\boldsymbol{u}(t) = \boldsymbol{v}(t), \quad \forall t \in \mathcal{T}.$$

Control functions $\boldsymbol{u}$ are approximated as piecewise constant functions. The voltages $\boldsymbol{v}$ are setpoints provided to the motor internal controller. In order to receive joint torques from the motor input voltage $\boldsymbol{v}$ for the model, we apply a mapping depending on the angular velocity $\dot{\boldsymbol{q}}$, i.e., $\boldsymbol{\tau} \equiv \boldsymbol{\tau}(\boldsymbol{v}, \dot{\boldsymbol{q}})$. We refer to [155] for further details.

5.4.2 Objective Function

For the NMPC problem given by Definition 1.4 in Chapter 1, we employ an objective of *Lagrange* type, which minimizes the weighted norm $\|\boldsymbol{\ell} - \bar{\boldsymbol{\ell}}\|_{\boldsymbol{W}}^2$, where $\|\boldsymbol{v}\|_{\boldsymbol{W}}^2 = \boldsymbol{v}^T \boldsymbol{W} \boldsymbol{v}$.

As mentioned above, the squatting task is realized by means of the objective function

$$\boldsymbol{\ell}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{p}) = \begin{bmatrix} \boldsymbol{r}_z(\boldsymbol{q}) \\ c_x(\boldsymbol{q}) \\ pose(\boldsymbol{q}) \\ \dot{\boldsymbol{q}} \end{bmatrix}, \quad \bar{\boldsymbol{\ell}}(t, \boldsymbol{p}) = \begin{bmatrix} \boldsymbol{p} \\ \bar{c}_x \\ 0.30 \\ \boldsymbol{0} \end{bmatrix}, \tag{5.4}$$

where we denote the forward kinematic evaluations of the root frame as $\boldsymbol{r}(\boldsymbol{q})$ and the x position of the center of mass (CoM) as $c_x(\boldsymbol{q})$. The term $pose(\boldsymbol{q}) := \boldsymbol{q}_{\text{ankle}} + \boldsymbol{q}_{\text{knee}} + \boldsymbol{q}_{\text{hip}}$ denotes the evaluation of the torso angle with respect to the world frame computed from the angles of ankle, knee and hip, and the joint velocities $\dot{\boldsymbol{q}}$ are used as regularization terms improving the stability of the robot. The setpoint for the CoM term is the center of the support polygon denoted by $\bar{c}_x$ and given by the position of the tip and heel of the support foot. The height to track by the robot is provided via the model parameter $\boldsymbol{p}$ and the squatting is then realized by its modulation over time. We weight the different terms by the weighting matrix

$$\boldsymbol{W} = \operatorname{diag}\{50.0, 100.0, 50.0, 3.0, \ldots, 3.0\},$$

with a focus on stability, a trade-off between tracking and correct upright pose and a small weight on the joint velocity penalty.

5.4.3 Constraints

In order to guarantee stability of the robot, we additionally formulate static stability according to Definition 3.6 as constraint, which reads

$$\boldsymbol{g}(\boldsymbol{x},\boldsymbol{u}) = \begin{bmatrix} x_t - c_x(\boldsymbol{q}) \\ c_x(\boldsymbol{q}) - x_h \end{bmatrix} \geqslant \boldsymbol{0}, \tag{5.5}$$

where x_t, x_h denote the position of the tips and the heels of the robot feet. We exploit the symmetry and therefore one foot is enough to describe the polygon of support. The optimization is subject to the constraints

$$\begin{bmatrix} -1.57 \\ -2.53 \\ -0.61 \\ -3.00 \end{bmatrix} \leqslant \boldsymbol{q} \leqslant \begin{bmatrix} 1.45 \\ -0.02 \\ 2.53 \\ 0.36 \end{bmatrix} \tag{5.6}$$

given in rad and box constraints to limit the motor voltages by $-5\,\text{V} \leqslant \boldsymbol{v}_i \leqslant 5\,\text{V}$, $i \in \{\text{ankle}, \text{knee}, \text{hip}, \text{arm}\}$.

5.5 Results

In order to validate the proposed approach on combining different MLRTI of NMPC, we implement the approach on the robotic hardware *Leo* of TU Delft. The robot is depicted in Figure 5.1, where we indicate the used world coordinate frame by magenta and show the root frame origin by a circle with black and white sectors. In Section 1.2.3, we give an introduction to the robot and present its specifications.

In the remainder of this section, we first discuss the experimental setup, where we explain the chosen task as well as the friction compensation applied in order to realize the task on the actual robot. Second, we revisit the results obtained from a run in simulation, where the timing statistics are of special interest. Third, we show the results for the five separate runs of the NMPC control scheme on the robot. Additionally, we show the results of an experiment considering external perturbations.

5.5.1 Experimental Setup

For the experiments in this chapter, we focus on squatting, i.e., a posture or motion where the weight of the body is located above the feet but the knees and hips are bent such that the upper body is lowered. The realization of this task required to take the robot off the boom, which is used for walking experiments, and fix it to a ground plate below its feet in order to ready it for a squatting task. We performed two sets of experiments: a squatting task and an experiment evaluating the performance against external perturbations.

The experimental setup of the squatting task is shown in Figure 5.5. In order to bring the robot into a well-defined initial configuration, two phases precede each trial: a PID phase and a NMPC init(ialization) phase. In the PID phase a configuration is tracked by a low gain PID controller and in the following init phase the NMPC software is initialized. The actual trial starts as soon as NMPC is ready, which we indicate by $t_0 = 0.0\,\text{s}$ and ends

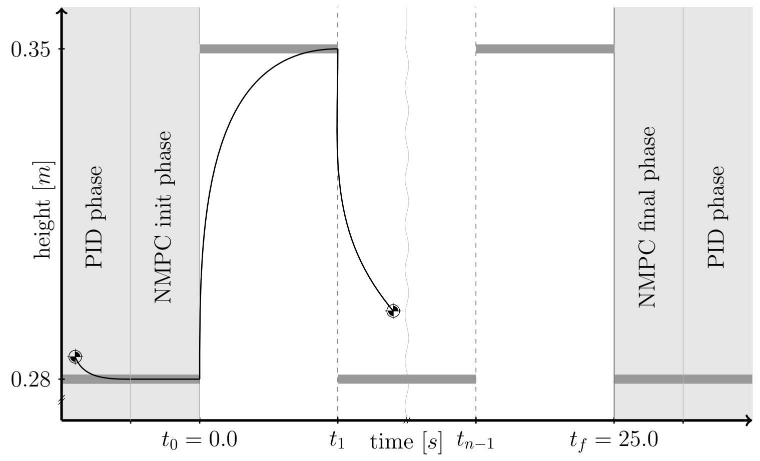


Figure 5.5: Scheme explaining the squatting task for the experiments with *Leo*. Before and after the experiment (white background, from t_0 to t_f), a setup and tear down phase (light gray background) brings the robot into a initial configuration. During the experiment the setpoint (dark gray bars) tracked by the root center (circle with black and white sectors, c.f. Figure 5.1) is switched at time points $t_i, 0 < i < f$.

after $t_f = 25.0\,\text{s}$. The actual task is implemented as a tracking of a setpoint at the center of the torso of the robot by NMPC run on an external computer.

The squatting motion is achieved by tracking two distinct setpoints, a lower $\boldsymbol{p}^{lo} = 0.28\,\text{m}$ and $\boldsymbol{p}^{up} = 0.35\,\text{m}$ one, such that the robot has to move its root by 7 cm (relative to the root height) from a crouching to almost fully stretched legs. These are repeatedly switched for the controller after periods of 1.5 s, such that 8 full squats have to be performed in total. After the experiment the robot is brought back into its initial configuration by a non-recorded NMPC phase followed by a final PID phase, during which the NMPC policy is teared down and finalized.

The setup to evaluate the performance against external perturbations follows that of the squatting task. Here, we track an intermediate setpoint $\boldsymbol{p}^{im} = 0.32\,\text{m}$, while the robot experiences external perturbations.

For the purpose of evaluation the following criteria are considered in the validation of each experiment:

- The value of the objective as indicator of performance.
- The trajectories of states, control and tracking task.
- The timing statistics of the respective NMPC policy.

Friction Compensation for Dynamixel Motors

The actuation of the robot is realized by *Dynamixel XM430* servo motors as described in Section 1.2.3. Here, we use voltage control mode of the motors, as proposed in [155]. The motors are subject to *Coulomb* as well as viscous frictions, and gearbox inefficiency. Without a compensation of the friction a realization of NMPC was not possible on the hardware. Therefore, we implemented an affine transformation of an input control signal. The actual applied voltage $\boldsymbol{v}_i = 0.75 \cdot \boldsymbol{v}_i^{nmpc} + \boldsymbol{v}_i^{comp}(\boldsymbol{p})$ is composed of the control signal from NMPC multiplied by the gearbox efficiency of 75 % and a compensatory term for the friction $\boldsymbol{v}_i^{comp}$, given by

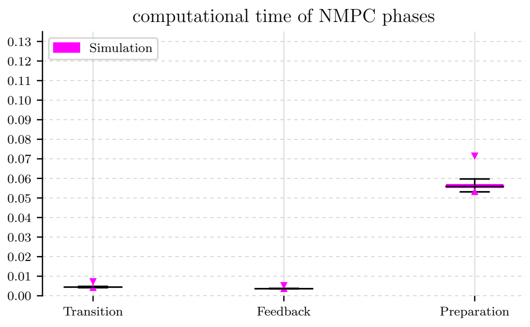
$$\boldsymbol{v}_i^{\text{comp}}(\boldsymbol{p}) = \begin{cases} 1.0\mu_C, & \text{if } \boldsymbol{p} = \boldsymbol{p}^{up} \\ -1.5\mu_C, & \text{if } \boldsymbol{p} = \boldsymbol{p}^{lo}, \end{cases}$$

where the *Coulomb* friction term was tuned to be $\mu_C = 0.86\,\text{V}$ for $i \in \{\text{ankle}, \text{knee}, \text{hip}\}$ and 0 V for arm.

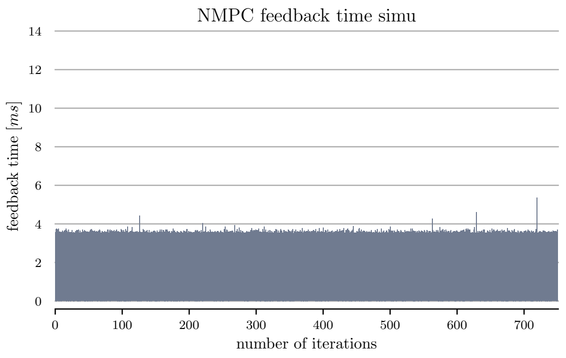
5.5.2 Numerical Results

Initially, we performed experiments in simulation to evaluate the timing statistics of a nominal NMPC run as depicted in Figure 5.6. From this, an analysis of the computational time of the different NMPC phases is visualized in Figure 5.6a by box plots and the timings of the feedback phase only of single NMPC run is shown in Figure 5.6b. The results show a relatively slow preparation phase of 71.61 ms, while transition (7.39 ms) and feedback (5.37 ms) phases are much faster in comparison. Following this, nominal NMPC with a total iteration time of 80.31 ms would end up with a minimum control rate of only $\sim$ 12 Hz , while a control rate of $\sim$ 186 Hz would be possible by leveraging the fast feedback time. This would improve control rate by a factor of 15. We only compared the maximum recorded run time data here and in the mean higher values are possible.

From this, we can conclude two things. On the one hand, using the novel strategy will enable us to run NMPC in real-time on the actual hardware because the actual feedback time is only a fifth of the fixed sampling time of the robot. On the other hand, the application of nominal NMPC to control the robot will be problematic at a control rate of $\sim$ 12 Hz, which was then verified during the experimental evaluation.



(a) Timing of different NMPC phases.



(b) Timing of feedback phase of nominal NMPC.

Figure 5.6: Run-time analysis of a squatting experiment under ideal conditions in simulation.

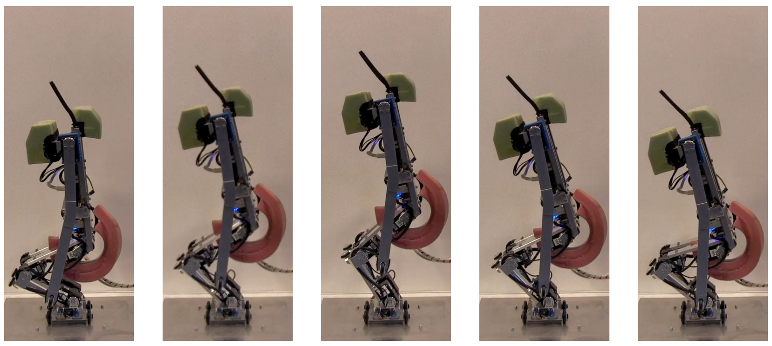


Figure 5.7: Snapshot series of *Leo* squatting during an experiment.

5.5.3 Experimental Results

Squatting Experiment In Figure 5.7, a scene from a squatting experiment shows *Leo* performing a single squat. The whole experiment was repeated five times in a row with breaks in between. The single runs are denoted by ts1, ts2, ts3, ts4 and ts5. Each trial was successful and no problems occurred. The recorded data from the experiments is visualized in Figures 5.8, 5.9. Due to the fact that the arm is not moving much and also has little effect due to its low weight, we have dropped the joint angles and the motor input voltages of the arm.

In Figure 5.8a, the actual height tracking is depicted as presented in Figure 5.5. The setpoints $p^{\mathrm{up}}, p^{\mathrm{lo}}$ (dark and light gray horizontal bars) attract the root origin correctly, which then slightly overshoots before the setpoint is switched again. During tracking of the lower setpoint the overshoot is stronger. We use the same colors as for the setpoints $p^{\mathrm{up}}, p^{\mathrm{lo}}$ (dark and light gray) to highlight ascending and descending phases in the other plots.

In Figure 5.8b, the reward, i.e., the instantaneous negative objective function value (5.4), is plotted against time. Here, optimal performance would show zero values. While this value is never reached exactly, it comes close to the optimal value at the end of reaching the lower setpoint. However during each ascending phase the value of the reward is significantly lower than during the respective get-down phase of a squat.

Figures 5.9 (a,c,e) show the angles of hip, knee and ankle joint. All three joints show both a high quantitative similarity of the different trials, with the only visible deviations near the extrema of the trajectories. Looking at the markers, slight deviations in the time profile are visible. While the trajectories are similar, the paths are not perfectly

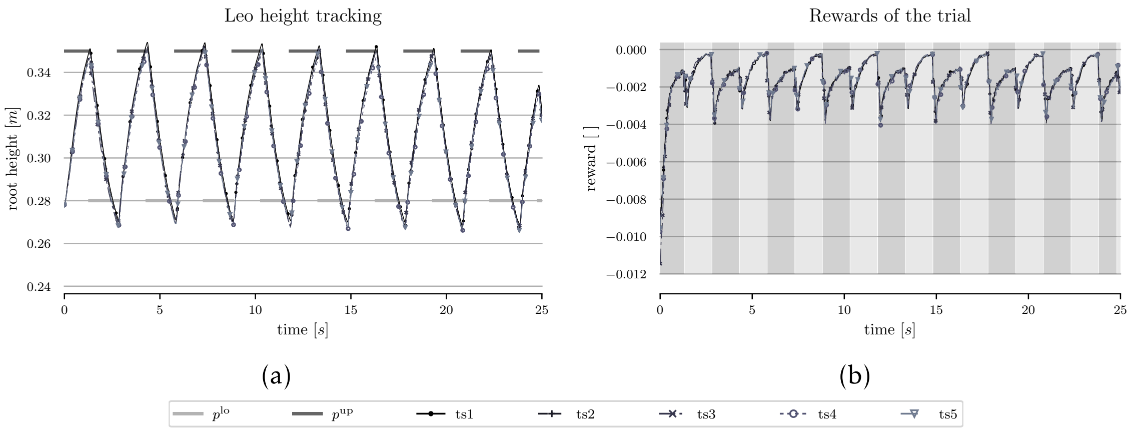


Figure 5.8: Recorded and processed data of the squatting experiment: height tracking (a) and instantaneous reward (b).

synchronized in time.

Figures 5.9 (b,d,f) show the motor input voltages at hip, knee, and ankle. While they show the same qualitative behavior, one can see the control effort due to the deviations during the tracking of the lower setpoint. Even though the joint angles show satisfactory behavior. The variability in control can be seen near the end of the experiment when, in the final phase, the lower setpoint is tracked. During ascending the knee voltage saturates to the limit of 5 V in the very beginning of the motion.

The timing statistics derived from the experiments are shown in Figure 5.10. Figure 5.10a reveals similar results for the timing statistics as the numerical experiments. Comparing the maximum values of each phase, we see again a dominant preparation phase of 132.04 ms, which stands out in comparison to the fast feedback and transition phases of 13.32 ms (and 15.15 ms). This shows again the benefit of the proposed multi-level approach as with this a speedup of the feedback time by a factor of 10–16 is easily possible and one would end up with a minimum control rate of ~ 75 Hz and up to ~ 190 Hz (mean). In contrast to this, a nominal scheme relying on the same timing statistics would result in a minimum control of not even 10 Hz.

In Figure 5.10b, the feedback times for the currently active LMPC thread are shown. While most of the feedback times are below 5 ms, outliers still show feedback times well beyond 10 ms.

Push Experiment In Figure 5.11, Leo recovers from the external push performed by an experimenter. The experiment was conducted once. Note that the trajectories of this experiment are not shown due to page limitations. The reader is referred to the supplementary video. The feedback was crisp and the robot directly reacted to the external perturbations by counteracting the force. However, due to the friction compensation the robot started to stretch his legs after some hard perturbations.

5.5.4 Discussion

In contrast to nominal NMPC, which was not applicable at all, the proposed multi-level NMPC scheme enabled real-time control on the robot. While the friction compensation made the scheme actually work on the hardware, the results reveal the erratic behavior due to model-plant mismatch resulting from our friction compensation, e.g. overshoot and

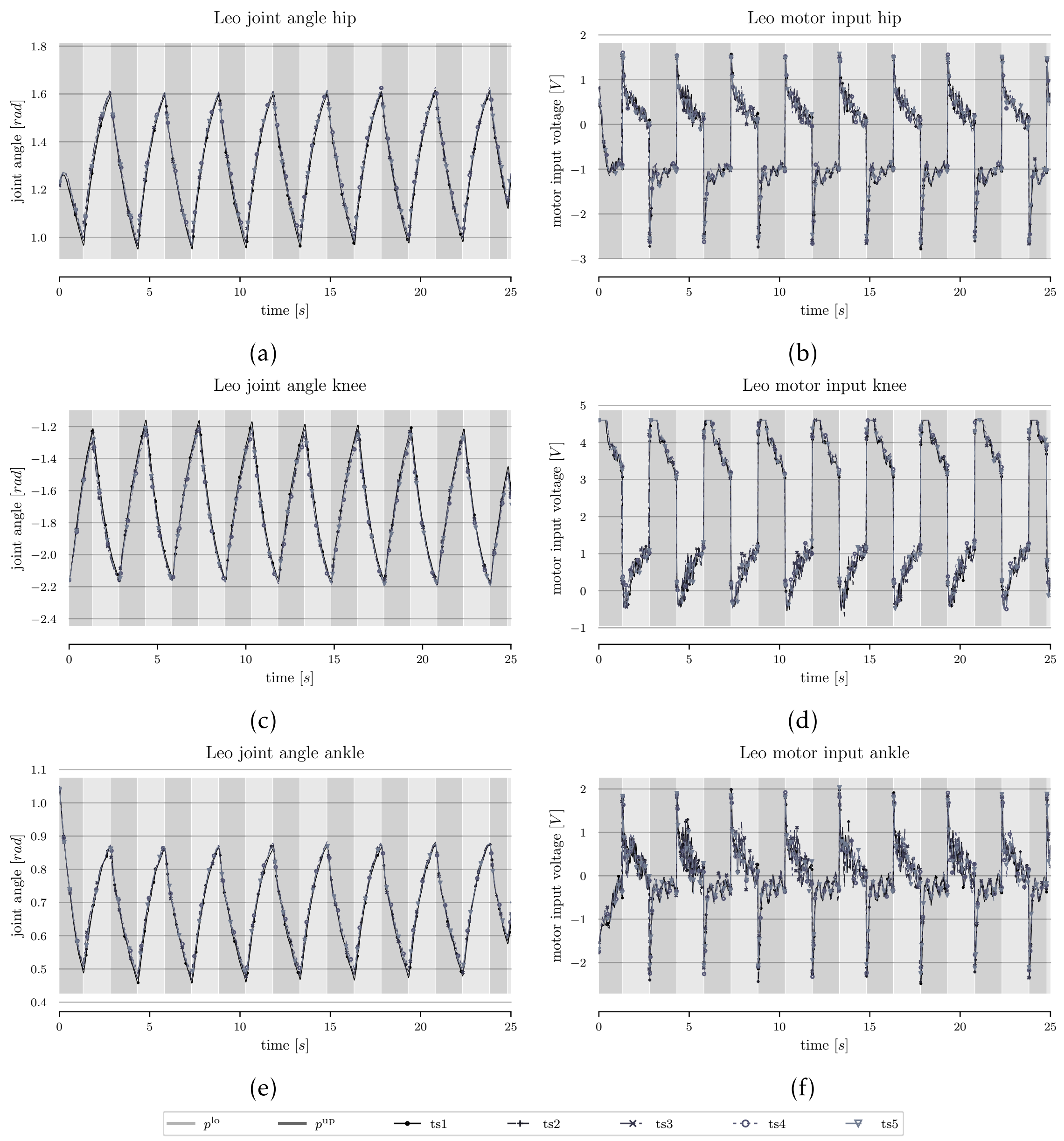
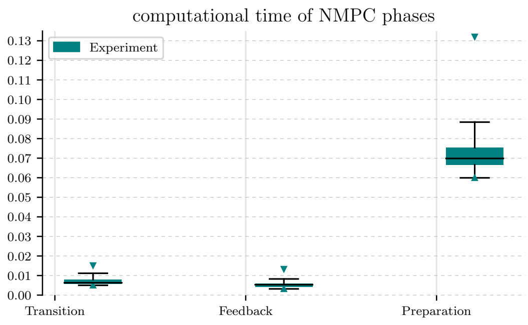
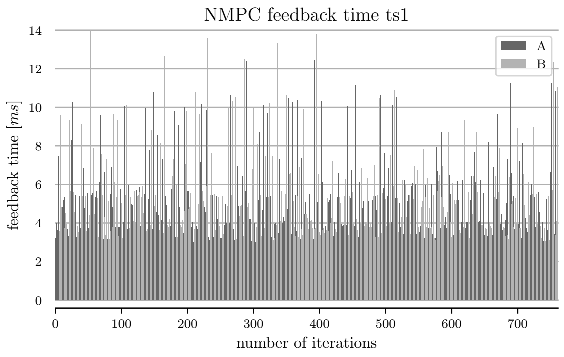


Figure 5.9: Recorded and processed data of the squatting experiment: joint angle trajectories (a, c, e) and motor input voltages (b, d, f).



(a) Timing of NMPC phases.



(b) Feedback times of threads A, B (dark, light gray).

Figure 5.10: Run-time analysis of the squatting experiment of *Leo* robot.

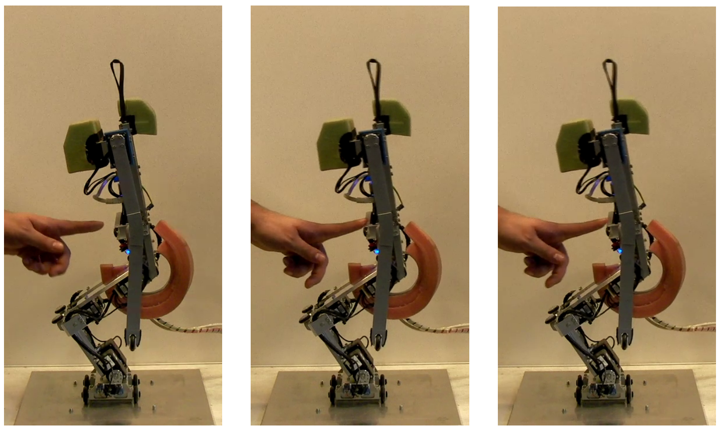


Figure 5.11: Snapshot series of external perturbation during an experiment.

voltage saturation during squatting and leg stretching in the push recovery experiment. We believe that this is not a problem for future research, as there are avenues to compensating for this either by improving the model using parameter identification methods, or by learning by means of reinforcement learning (RL), c.f. [103].

During the descending phase while squatting, or in the push experiment, we noticed a little jittering in the control. We offer two explanations for this. First, the friction compensation alters the control signal computed from the model. Especially during the down-phase or when the robot moves very slowly, we could see the importance of adjusting the compensatory term. Further improving it might help reducing the mismatch or compensating it on motor level will result in smoother motions also in regimes with slow speed. Second, switching the controller also means switching the current linearization point of the model, followed by up to four consecutive feedback phases using it. A change of the linearization has significant impact on the feedback, especially when the latest linearization is not up to date. By further improving the speed of the computations, especially during the preparation phase, both the feedback time as well as the ratio of A and D iterations can be improved.

Alternatively, the sampling time of the robot low-level control should be increased or changed to an event-driven approach to better cope with the NMPC framework which requires a more flexible timing scheme to provide fast feedback. Additionally, the preparation time can be reduced by parallelizing the integration steps on the shooting intervals using more than two threads or implementing less demanding levels of MLRTIs.

5.6 Summary

In this chapter, we presented a control approach of NMPC for feedback control employing a combination and parallelization of both linear and nonlinear methods that is based on two different levels of real-time iterations, named levels A and D. Furthermore, this chapter presented the first step on evaluating the combination of different multi-level real-time iteration of NMPC on the actual robotic hardware. This combination enabled us to achieve real-time execution of NMPC on the physical robotic hardware of Leo. Additionally, we proposed a problem specific model for compensation of the static friction hindering the robot to perform the task. We successfully performed experiments on the robot in different scenarios followed by the detailed analysis of the recorded results. In contrast to the nominal control scheme, only the new control scheme was able to realize closed-loop control on the actual hardware. In this way, the performance of the proposed approach is superior to nominal NMPC.

6 Towards a Combination of Model-free and Model-based Optimal Control

In robotics, one cannot expect to work with ideal models of the considered systems or their environments. Rather, we have to face unforeseen situations and unknown conditions, and aim for reactions that are feasible and, ideally, optimal with respect to given task performance criteria. A typical task is bipedal locomotion, where a robot needs to maintain stability and pace on an uneven floor with uncertain roughness and slope [155]. Robots are capable of executing a set of commands to achieve a task, however these commands are mostly encoded or tuned by hand.

Two common approaches to control dynamic systems are nonlinear model predictive control (NMPC) and reinforcement learning (RL). Both approaches can cope with uncertainties in the form of model-plant mismatch. Reinforcement learning has been proven suitable as a real-time closed-loop control concept in robotics [97]. NMPC is a closed-loop control concept already established in industrial practice [147]. However, the use of NMPC in robotic applications, especially humanoid robotics and bipedal walking, is still an open research field [42, 70, 108].

The following chapter presents our work on benchmarking and combining methods of model-based and model-free optimal control in the form of NMPC and RL. The contents of the following sections are based on our journal articles [103] and [104].

Usually, the dynamics of physical systems are known or can be modeled using first-principle models, but various uncertainties cannot be addressed so easily. Moving horizon estimation (MHE) techniques [107] can be employed for the identification of parametric uncertainties, however, for structural uncertainties, such as backlash, *Coulomb* friction or wear and tear, this is not easily possible. In these situations, methods of RL are capable of finding an optimal sequence of commands even without any prior assumption about the world. However, the applicability of learning to real systems is very limited due to intrinsically damaging exploratory policies. For example, when learning from scratch the robot *Leo* depicted in Figure 1.4 can withstand only five minutes of operation due to large and rapidly changing motor torques and frequent falls [155].

In model-based control, it is possible to enforce constraints such that the static stability is guaranteed for the evolution of the system on a short horizon, thus these constraints help preventing falls. In RL, it is possible to consider angle and velocity constraints by means of negative rewards. However, to learn avoiding such constraints, they need to be violated multiple times in different robot configurations. Random exploration exacerbates the problem and can lead to a very large number of falls. Therefore, we propose to combine RL and NMPC in one framework that allows RL to gather the required experience without damaging a many-degree-of-freedom system.

The crucial step to derive a beneficial combination of different methods is a proper analysis of strengths and weakness as well as determining the situation in which one method is superior to the other. We achieve this by proposing a computational study that benchmarks model-based against model-free methods of optimal control subject to structural uncertainties in [103]. The idea of the computational study is explained along a benchmark example.

Based on the knowledge of the performed computational study, we can propose schemes combining both methods. In [104], we propose two schemes addressing the compensation of model-plant mismatch due to structural uncertainties. An extensive comparison of the respective schemes in simulation is conducted. The simulation employs a model of *Leo* robot, the goal robotic platform, to perform a squatting task. Based on the analysis of the results, the superior combination scheme is chosen for validation on the robot. Therefore, the scheme is implemented to perform a squatting task on the actual hardware.

The following contributions are made in this chapter:

- A quantitative comparison of RL and NMPC subject to uncertainties is presented.
- The computational study is executed on a benchmark example of a cart-pendulum swing-up and balance task.
- Differences, trade-offs and pitfalls of the specific problem formulations are shown.
- The change of superiority of one method over the other at break-even points is shown.
- Two schemes are proposed to compensate model-plant mismatch induced by uncertainties, preserving the *Markov* property.
- Each scheme leverages a model-based controller, here NMPC.
- The combination schemes are compared in simulation on a squatting task using a model of *Leo*.
- The superior combination scheme is realized on the robot *Leo*.

In the following sections, we will first talk about the methods of model-free optimal control in the form of RL, while we refer the reader to Section 1.4 for the details on methods of model-based optimal control, nonlinear model predictive control and online parameter and system identification. In Section 6.2, we introduce our computational study allowing to benchmark model-based and model-free methods of optimal control in the presence of structural uncertainties as presented in [103]. In Section 6.3, we present our work on combining methods of model-based and model-free optimal control. This is followed by the results of the computational study in Section 6.4.1 for a cart-pendulum swing-up motion benchmarking NMPC and RL. Additionally, we present the results on applying the hybrid control strategies developed in Section 6.4.2 to a squatting task of robot *Leo* of *Delft University of Technology* in both numerical as well as hardware experiments. The chapter is concluded with a summary on the subject.

6.1 Reinforcement Learning

Reinforcement learning is an active research area in the field of artificial intelligence and machine learning with applications in control. The most important feature of RL is its ability to learn without prior knowledge about the system. The goal of the learning task is supplied externally in the form of a reward function. RL is a trial-and-error method, which generally takes many iterations before it finds an optimal solution. To reduce the number of interactions with the system, model-learning methods such as Dyna [21], learning from demonstration [1, 161], or optimized control policy parameterizations [98] can be applied. Because RL does not require an explicitly given model, it can naturally adapt to uncertainties of the real system. In this sense, RL can be viewed as a model-free adaptive optimal control approach [165], such that RL can naturally adapt to uncertainties in the real system [166].

A common approach in RL is to model the task as a *Markov* decision process (MDP).

The process is defined as a quadruple $\langle \mathcal{X}, \mathcal{U}, \mathcal{P}, \mathcal{R} \rangle$, where $\mathcal{X} \subset \mathbb{R}^{n_x}$ is a set of possible states, $\mathcal{U} \subset \mathbb{R}^{n_u}$ is a set of possible control actions, $\mathcal{P} : \mathcal{X} \times \mathcal{U} \times \mathcal{X} \to [0,1]$ is a transition function that defines the probability of ending up in state $\boldsymbol{x}_{k+1} \in \mathcal{X}$ after executing action $\boldsymbol{u}_k \in \mathcal{U}$ in state $\boldsymbol{x}_k \in \mathcal{X}$. The reward function $\mathcal{R} : \mathcal{X} \times \mathcal{U} \times \mathcal{X} \to \mathbb{R}$ gives a real-valued reward $r_{k+1} = \mathcal{R}(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{x}_{k+1})$ for the particular transition between states. A MDP satisfies the *Markov* property, which assumes that the next state $\boldsymbol{x}_{k+1}$ depends only on the current state $\boldsymbol{x}_k$ and action $\boldsymbol{u}_k$, but not on previous states or actions [166].

A deterministic control policy $\pi : \mathcal{X} \to \mathcal{U}$ defines an action $\boldsymbol{u}_k$ taken in a state $\boldsymbol{x}_k$. Assuming the system to be stochastic, the goal of the learning process is to find an optimal control policy π^* that maximizes the discounted return

$$\mathfrak{R}_k^{\gamma} = \mathbb{E}\left\{ \sum_{i=0}^{\infty} \gamma^i r(\boldsymbol{x}_{k+i}, \boldsymbol{u}_{k+i}, \boldsymbol{x}_{k+i+1}) \right\},$$

where the discount rate $\gamma \in [0,1)$ exponentially decays rewards the further they lie in the future and is required for the integrability of the infinite sum. The immediate rewards r are defined as

$$r(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{x}_{k+1}) = \begin{cases} -l(\boldsymbol{x}_{k+1}, \boldsymbol{u}_k) & \text{if } \boldsymbol{g}(\boldsymbol{x}_{k+1}, \boldsymbol{u}_k) \geqslant 0, \\ R_{\mathrm{a}} & \text{otherwise.} \end{cases} \tag{6.1}$$

Here $r(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{x}_{k+1})$ is the scalar reward given for a transition from $\boldsymbol{x}_k$ to $\boldsymbol{x}_{k+1}$ caused by the control signal $\boldsymbol{u}_k = \boldsymbol{\pi}(\boldsymbol{x}_k) + \boldsymbol{n}, \boldsymbol{n} \sim \boldsymbol{\mathcal{N}}$ chosen from some policy $\boldsymbol{\pi}$.

Constraints are established as soft constraints by means of the large negative reward R_{a}. Subsequently, the episode is terminated, and the system is restarted in state $\boldsymbol{x}_0$. Usually, RL requires at many repetitions to estimate the return (6.1) correctly. These repetitions are obtained by adding exploration noise $\boldsymbol{\mathcal{N}}$ to control signals $\boldsymbol{u}_k$ at every time step. The outcome of repetitions is not known in advance and therefore may be damaging to the system.

The value function $V^{\pi}(\boldsymbol{x})$ denotes the expected return assuming that the system starts in the state $\boldsymbol{x}$ and then follows a prescribed control policy π. The optimal control policy π^* maximizes the value for each state. Therefore, an optimization of the control policy is tightly coupled with the maximization of the value function in RL.

For real-world systems, continuous control is preferred. This requires a parametrization of the policy $\pi(\boldsymbol{x})$, e.g. using a set of basis functions and their associated weights. The weights are usually optimized by gradient-descent methods [11, 62], or by global gradient-free methods [16, 68]. In this case, we use a standard gradient-descent method because the latter methods require a substantial number of interactions with the system under consideration, which can be problematic for interaction with real systems. Since the estimation of policy gradients often results in a high variance, the policy update is usually coupled with an explicit estimation of a parametrized value function $V^{\pi}(\boldsymbol{x})$. This combination is known as the actor-critic method, where the policy is referred to as the actor, and the value function is referred to as the critic.

The method we use is the standard model-free temporal-difference-based method described in [62]. Since RL is a trial-and-error learning method, the quality of the policy as well as the learning speed depend on the exploration method. Exploration is commonly achieved either by perturbation of the so far optimal action, by optimistic initialization, or by both. Optimistic initialization is a method of initializing the value function with a value equal to or greater than the maximum possible value of a state. This causes the visited states to become less attractive than the states that have not been visited yet [118]. Optimistic initialization can speed up the learning in the absence of negative rewards. For the parametrization of the policy and value-function we use a tile coding approximator [166] that is linear in parameters. In [104], we solve the problem (6.1) using a deterministic policy gradient (DPG) with linear function approximation and compatible features, chosen for its ability to optimize continuous control policies and fast convergence.

6.2 Benchmarking Model-free and Model-based Optimal Control

In this section, we propose a computational study as well as an evaluation scheme to quantitatively assess control approaches of both model-based nonlinear model predictive control as well as model-free reinforcement learning. First, we present the general idea of the study and introduce the required notation. Second, we discuss the evaluation protocol and define the measures required for the key performance indicator. The computational study is applied to a benchmark problem in the form of a swing-up and balancing problem for a *cart-pendulum* system [7, 95]. The respective results are presented in Section 6.4.1.

6.2.1 Computational Study

In Figure 6.1, the proposed computational study is depicted. It is set up as follows. In the first step (I), we establish optimal control ("OC") solutions for the ideal benchmark problem. Then we consider the NMPC formulation and derive the corresponding RL formulation from it. We highlight the changes introduced in both formulations and discuss their effects.

Subsequently, we address the strengths and weaknesses of NMPC and RL in terms of their ability to adapt to *structural* and *parametric* uncertainties.

In the second step (II), we investigate NMPC and RL methods that are explicitly unable to adapt to uncertainties. We introduce the term *frozen* to refer to this inability.

In the third step (III), the effect of uncertainties and the ability to adapt to them is analyzed for NMPC methods that have explicitly been equipped with knowledge about the uncertainties and for RL that is allowed to interact with the real system for an additional 5 % of the learning time. We introduce the term *adaptive* to distinguish these from the *frozen* methods.

We use a single RL method denoted as "RL", and two NMPC versions denoted as "iNMPC" and "NMPC". "iNMPC" is an ideal NMPC controller that neglects computational time and returns an optimal control signal immediately. In turn, "NMPC" represents an actual NMPC implementation tuned for real-time feasible control.

6.2.2 Evaluation protocol

In order to guarantee comparability of the results of the computational study proposed above, we additionally propose a standardized evaluation scheme. Therefore, we first define some notations and introduce some methodology required for the evaluation. Second, we define the quantities of interest, derive a detailed workflow on how to apply the computational study.

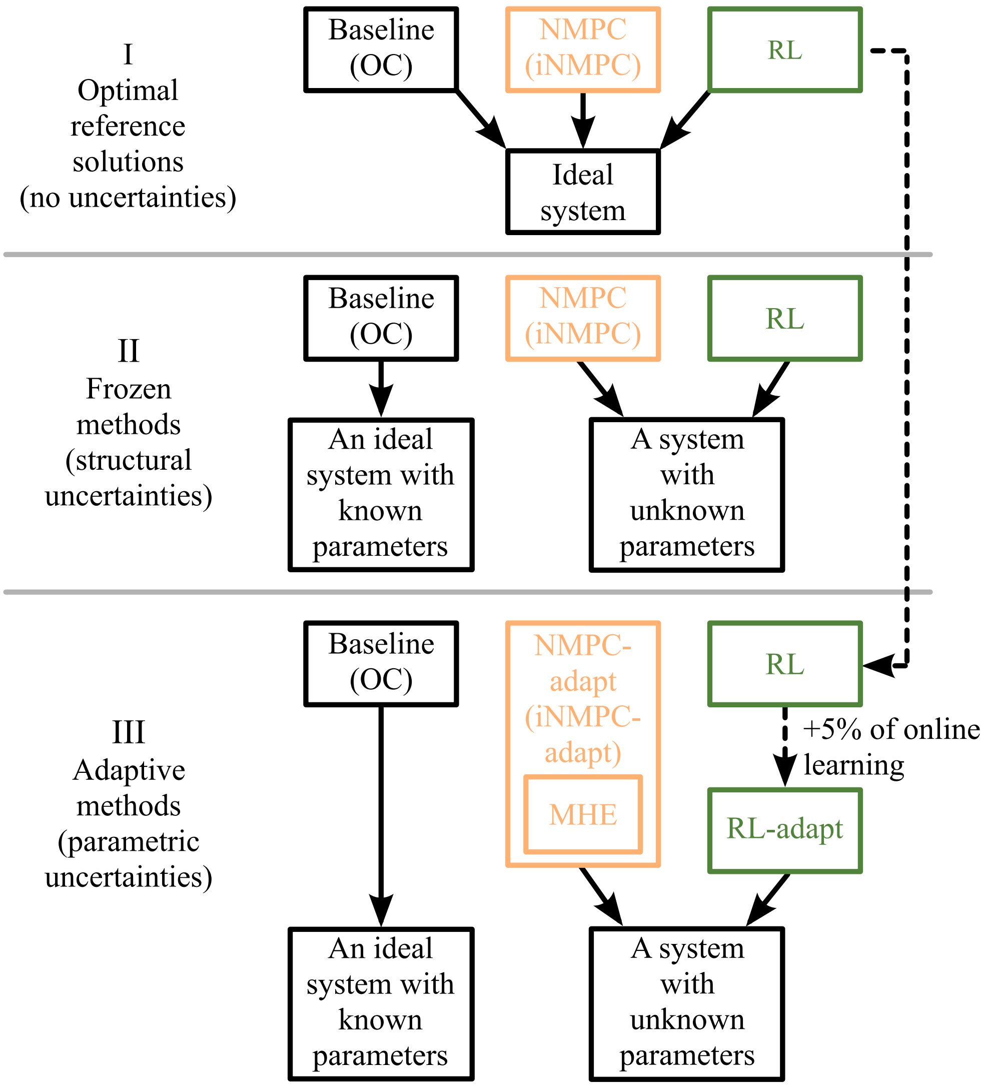

I
Optimal reference solutions (no uncertainties)
Baseline (OC)
NMPC (iNMPC)
RL
Ideal system
II
Frozen methods (structural uncertainties)
An ideal system with known parameters
A system with unknown parameters
III
Adaptive methods (parametric uncertainties)
NMPC-adapt (iNMPC-adapt)
MHE
+5% of online learning
RL-adapt

Figure 6.1: Overview of the computational study to assess the performance of NMPC and RL in the presence of structural uncertainty from our article [103]. Step I corresponds to a verification of state and control trajectories when problem formulations are equivalent for optimal control (OC), nonlinear model predictive control (NMPC) and reinforcement learning (RL). In steps II and III uncertainties of a varied magnitude are introduced. In the former case “NMPC”, “iNMPC” and “RL” are not equipped with an adaptation mechanism while in the latter case they are. In “NMPC-adapt” and “iNMPC-adapt” adaptation is accomplished by means of moving horizon estimation (MHE), and for “RL-adapt” we allow 5 % of additional interaction with the real system.

Notations and methodology

As summarized in Figure 6.1, we use the “OC” notation to denote the optimal solution obtained by offline optimal control. The cost of this solution serves as a baseline for all subsequent methods. As structural uncertainties, we consider uncertainties that originate from the lack of knowledge about the true physics of the underlying dynamic system. Examples in walking robots might include model-plant mismatch due to uneven floor, friction in joints, softness of the ground, etc. Being unaware of possible uncertainties in a system, the following three *frozen* methods are not explicitly equipped with an ability to adapt to the system:

- "iNMPC" denotes an ideal NMPC controller that neglects computational time constraints and returns an optimal control signal immediately.
- "NMPC" denotes an NMPC controller tuned to real-time performance for the specific task.
- "RL" denotes an RL controller that plays the optimal policy π^* after having learned on an ideal system.

Neither "iNMPC" nor "NMPC" apply MHE for state and parameter estimation.

As parametric uncertainties, we consider parameters which are included in the dynamic model of the system and whose values are not known *a priori*, but can be inferred from interactions with the real system, i.e., the parameters are observable. Accordingly, by the term *adaptive* we denote methods that have (in the case of NMPC) been explicitly equipped with knowledge about the uncertainties and an algorithm to adapt to them, or that are (in the case of RL) given additional time to interact with the system:

- "iNMPC-adapt" denotes a controller of a combination of both MHE and NMPC. The controller is able to estimate a specified unknown parameter of a model and adjust its control signal accordingly.
- "NMPC-adapt" denotes a combination of both MHE and NMPC tuned to real-time performance for the specific task.
- "RL-adapt" denotes the RL controller that is initialized using the optimal policy π^* learned by "RL" on the ideal system. To cope with uncertainties in the system, we allow "RL-adapt" to learn for an additional small number of episodes, c.f. Table 6.1.

Note that the NMPC approach requires explicit specification of the parameters to be estimated in the model, while RL can cope with them without explicit consideration.

Description of experiment and measures

For the benchmark problem, we provide a comprehensive comparison of the described methods for an ideal system, and for a system with structural and parametric uncertainties.

First, we investigate whether the three frozen methods produce similar trajectories on our benchmark system. For that we employ the coefficient of determination, R^2, as a similarity measure of trajectories. The measure quantifies the deviation of the trajectories obtained by "RL", "iNMPC", and "NMPC" from an optimal trajectory. Denoting a trajectory ζ as a sequence of states and controls, $\zeta = \{\zeta_k\}$, $0 \leqslant k \leqslant K$ and $\zeta_k = [\boldsymbol{x}_k^T, \boldsymbol{u}_k^T]^T$, we measure similarity between corresponding components by means of R^2. Formally, the R^2 measure is defined as

$$R^2 = 1 - \frac{\sum_i^K (\zeta_i^{\ddagger} - \zeta_i^{\mathrm{OC}})^2}{\sum_i^K (\zeta_i^{\ddagger} - \bar{\zeta})^2}, \quad \bar{\zeta} = \frac{1}{K}\sum_i^K \zeta_i^{\ddagger},$$

where ‡ is a wildcard for one of {"RL", "RL-adapt", "iNMPC", "iNMPC-adapt", "NMPC", "NMPC-adapt"}. For the "RL" method, which exhibits variability in trajectories, we compute both the R^2 measure of the mean trajectory, and the mean of R^2 values obtained across individual trajectories.

Second, after the similarity of the methods is verified, we employ regret as a measure to evaluate the performance of the methods against uncertainty $\Delta\Theta$. This measure is commonly used in evaluation of online machine learning and optimization methods [81, 159]. Regret quantifies the amount of additional cost which is incurred due to suboptimal actions taken by a controller with respect to the optimal control actions. Lower values of regret indicate a controller, whose behavior is closer to the optimal one. Since the optimal controller has zero regret, it becomes convenient to plot regrets of the methods instead of the direct costs.

We compute the regret $R^{\ddagger}(\zeta;\Delta\Theta)$, defined as the difference between $C^{\ddagger}(\zeta)$, the total cost of the method denoted by the wildcard ‡, and the baseline $C^{\mathrm{OC}}(\zeta;\Delta\Theta)$, i.e.,

$$R^{\ddagger}(\zeta;\Delta\Theta) = C^{\ddagger}(\zeta) - C^{\mathrm{OC}}(\zeta;\Delta\Theta),$$

where we use the NMPC cost (1.12a) for all methods directly,

$$C(\zeta) = \sum_{i=0}^{K} \gamma^i l(\boldsymbol{x}_i, \boldsymbol{u}_i)\Delta t_i. \tag{6.2}$$

By means of Δt_i, we take into account the different sampling periods of the methods. Note that all of the tested methods are unaware of the true extent of the uncertainty introduced. Here, we pursue a generic comparison across methods and benchmarks. Therefore, we do not include specific stability measures, but rather stick to a general notion of the cost of trajectories.

Due to the stochastic nature of RL, we plot a mean value of the regret averaged over 50 runs. If a run for a given uncertainty $\Delta\Theta$ is prematurely terminated due to the violation of constraints, then the corresponding value is not drawn.

6.3 Combining Model-free and Model-based Optimal Control

In this section, we present our efforts on combining RL and NMPC in one framework as shown in [104]. This combination enables RL to gather the required experience while using a model-based controller as support to avoid situations. Here, we apply NMPC as model-based nominal controller because of the complexity of the robot. The gathered experience is used by RL to compensate for the difference between the internal model of the system and the real one. For this, we have to introduce a different problem formulation that incorporates this uncertainty.

6.3.1 Problem Formulation

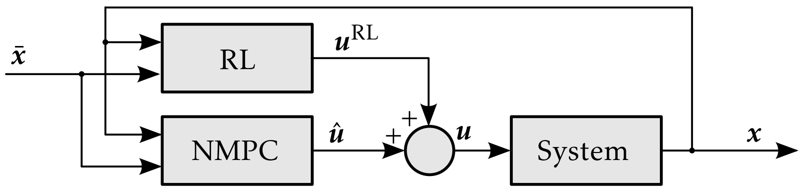
In contrast to the ordinary differential equation (ODE) introduced in Section 1, here, we consider the nonlinear time-invariant system in the form of

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{\rho}), \tag{6.3}$$

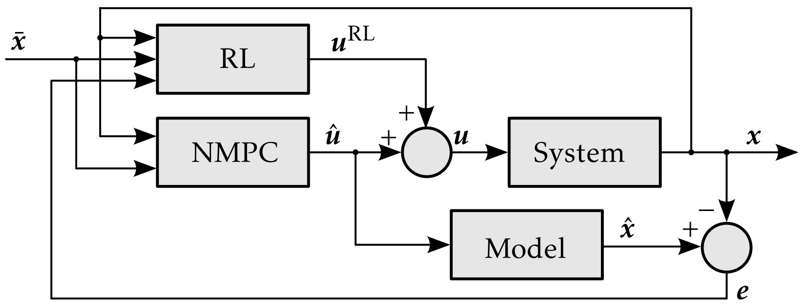
where $\boldsymbol{x}(t) \in \mathbb{R}^{n_x}$ is the system state at time t, $\boldsymbol{u}(t) \in \mathbb{R}^{n_u}$ is the control vector of joint motor voltages applied to the system at time t, and $\boldsymbol{\rho}$ is an unknown structural uncertainty. The presence of uncertainty causes the model-plant mismatch $\boldsymbol{e}$ which is formulated as the difference between the real system state $\boldsymbol{x}$ and the simulated state of the model $\hat{\boldsymbol{x}}$, see Figure 6.2b. We do not make any assumption on how the uncertainty enters the equations. Thus, it represents the general concept of model-plant mismatch.

6.3.2 Proposed Combination Schemes

In Figure 6.2, the two derived approaches are shown. The first approach learns a compensatory control action as depicted in Figure 6.2a, similar to [8]. We call this approach



(a) Compensatory action learning (CAL)



(b) Model-plant mismatch learning (MPML)

Figure 6.2: Visualization of two possible combination schemes of reinforcement learning (RL) and nonlinear model predictive control (NMPC) from our article [104].

compensatory action learning (CAL). Instead of a proportional-derivative (PD) controller, we use NMPC, which introduces an additional optimization problem into the scheme. Since both NMPC and RL optimize similar performance measures, the obtained policy is optimal with respect to the real system.

The second approach learns a compensatory signal from the difference of transitions predicted by the internal model and the actual transition, as depicted in Figure 6.2b. In this case, RL uses a different optimization goal, which does not divert NMPC from reaching its objective. As a result, the model-plant mismatch is eliminated by forcing the real system to behave as if it has no uncertainties. This approach is denoted by model-plant mismatch learning (MPML).

Compensatory Action Learning

In the proposed combination schemes shown in Figures 6.2a and 6.2b, we use $\hat{\boldsymbol{u}}$ notation for the output of the NMPC controller and $\boldsymbol{u}^{\mathrm{RL}}$ notation for the output of the RL controller.

The CAL approach presented in Figure 6.2a learns a compensatory control action added to the control input computed by nominal NMPC. For learning, we use the NMPC-inspired reward (6.1), which establishes similar optimization goals for both controllers. Due to small differences in the formulation and function approximations in RL, the obtained policy might be suboptimal compared to NMPC.

Model-plant Mismatch Learning

The MPML approach is presented in Figure 6.2b. The real system is actuated by the RL compensatory signal $\boldsymbol{u}^{\mathrm{RL}}$ which is added to the nominal controller signal. RL maximizes return (6.1) where the reward is given by

$$r(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{x}_{k+1}) = -\|\boldsymbol{e}_{k+1}\|_2 = -\|\boldsymbol{x}_{k+1} - \hat{\boldsymbol{x}}_{k+1}\|_2. \tag{6.4}$$

In the following, we prove two theorems. The first one explains the behavior of the system when the model-plant mismatch is minimized by RL. The subsequent corollary considers the case when the cumulative return ($\gamma > 0$) is useful for discovering a better control policy. The second theorem specifies conditions under which the system retains the *Markov* property. If the property is preserved, then RL will not diverge, and the mismatch can be minimized. In this case, the performance depends on the RL learning capability.

Theorem 6.1 The outcome of the control policy approaches the outcome of the optimal policy with respect to the idealized model *iff* the model-plant mismatch $\boldsymbol{e}_k \to 0$ when $k \to \infty$. △

Proof Writing the mismatch as $\boldsymbol{e}_k = \boldsymbol{x}_k - \hat{\boldsymbol{x}}_k \to 0$ results in $\boldsymbol{x}_k \to \hat{\boldsymbol{x}}_k$. Assuming the nominal controller can reach the setpoint on the model, $\hat{\boldsymbol{x}}_k \to \bar{\boldsymbol{x}}_k$, implies that the system state will also approach the setpoint, $\boldsymbol{x}_k \to \bar{\boldsymbol{x}}_k$. Since the mismatch $\boldsymbol{e}_k$ is minimized in every point of the reference trajectory $\bar{\boldsymbol{x}}_k$, we arrive at the proof of the theorem. The same logic holds for the reverse. □

Corollary If there is a time step k such that $\boldsymbol{e}_k \neq 0 \; \forall \boldsymbol{u}_k$, then $\min_{\pi^{\gamma=0}} \ell \geqslant \min_{\pi^{\gamma>0}} \ell$, where $\boldsymbol{\pi}^\gamma$ is the policy optimal with respect to $\mathcal{R}^\gamma$. Strict equality holds when the mismatch is eliminated along the reference trajectory. △

The corollary is based on the RL result that larger γ improves the quality of the policy [80]. However, if there exists a control action which achieves zero mismatch, then maximizing immediate rewards ($\gamma = 0$) is desirable because the problem becomes computationally easier.

In the following theorem, we assume that the system (6.3) can be discretized as $\boldsymbol{x}_{k+1} = \boldsymbol{f}(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{\rho})$ and the setpoint $\bar{\boldsymbol{x}}$ can be included into the state $\boldsymbol{x}$ for simplicity.

Theorem 6.3 The system controlled by the nominal controller is *Markov* with respect to RL if
(a) the system itself is *Markov* w.r.t. RL, $\boldsymbol{x}_{k+1} = \boldsymbol{f}(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{\rho})$;
(b) the internal model is *Markov* w.r.t. the nominal controller, $\hat{\boldsymbol{x}}_{k+1} = \boldsymbol{f}(\boldsymbol{x}_k, \hat{\boldsymbol{u}}_k, 0)$; and
(c) the nominal controller response $\hat{\boldsymbol{u}}_k \equiv \hat{\boldsymbol{u}}(\boldsymbol{x}_k) + \boldsymbol{m}$, $\boldsymbol{m} \sim \boldsymbol{\mathcal{M}}$ is stochastic with some stationary distribution $\boldsymbol{\mathcal{M}}$. △

Proof First, by looking at Figure 6.2b we write the condition (a) and show that the distribution of states $\boldsymbol{x}_{k+1}$ is defined by the current state $\boldsymbol{x}_k$

$$\begin{aligned}\boldsymbol{x}_{k+1} &= \boldsymbol{f}(\boldsymbol{x}_k, \hat{\boldsymbol{u}}_k + \boldsymbol{u}_k^{\mathrm{RL}}, \boldsymbol{\rho}) \\ &= \boldsymbol{f}(\boldsymbol{x}_k, \hat{\boldsymbol{u}}(\boldsymbol{x}_k) + \boldsymbol{\pi}(\boldsymbol{x}_k) + \boldsymbol{m} + \boldsymbol{n}, \boldsymbol{\rho}).\end{aligned} \tag{6.5}$$

Next, we show that the reward is also defined by $\boldsymbol{x}_k$.

$$\begin{aligned}r(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{x}_{k+1}) &= -\|\boldsymbol{x}_{k+1} - \hat{\boldsymbol{x}}_{k+1}\|_2 \\ &= -\|\boldsymbol{x}_{k+1} - \boldsymbol{f}(\boldsymbol{x}_k, \hat{\boldsymbol{u}}(\boldsymbol{x}_k) + \boldsymbol{m}, 0)\|_2\end{aligned}. \tag{6.6}$$

The expected return averages the sum of discounted rewards over the distribution of states and controls. Since both the dynamics and the return are predictable from the current state $\boldsymbol{x}_k$, we conclude that the system controlled by the nominal controller is *Markov* with respect to RL. □

Note that the real system does not have to be *Markov* with respect to NMPC, which means that any uncertainty $\boldsymbol{\rho}$ can be compensated. We use this observation in the real example where $\boldsymbol{\rho}$ depends on motor temperature which does not enter the model, but is used in RL as an extra state variable.

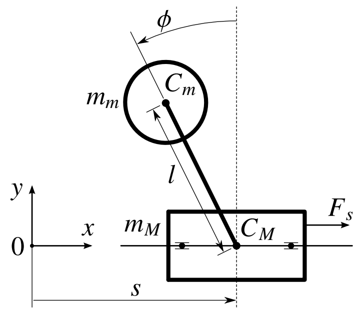


Figure 6.3: The inverted pendulum on a movable cart from our article [103].

In the special case of an affine system (6.3) with respect to controls, $\boldsymbol{x}_{k+1} = \boldsymbol{f}^{\boldsymbol{x}}(\boldsymbol{x}_k, \rho) + \boldsymbol{f}^{\boldsymbol{u}}(\boldsymbol{x}_k, \rho)\boldsymbol{u}_k$, a perfectly learned RL control, i.e., $\boldsymbol{e} = 0$, is explicitly given by

$$\begin{aligned} \boldsymbol{u}_k^{\mathrm{RL}} = &\left(\boldsymbol{f}^{\boldsymbol{u}}(\boldsymbol{x}_k, \rho)^\top \boldsymbol{f}^{\boldsymbol{u}}(\boldsymbol{x}_k, \rho)\right)^{-1} \boldsymbol{f}^{\boldsymbol{u}}(\boldsymbol{x}_k, \rho)^\top [\boldsymbol{f}^{\boldsymbol{x}}(\boldsymbol{x}_k, 0) \\ &- \boldsymbol{f}^{\boldsymbol{x}}(\boldsymbol{x}_k, \rho) + (\boldsymbol{f}^{\boldsymbol{u}}(\boldsymbol{x}_k, 0) - \boldsymbol{f}^{\boldsymbol{u}}(\boldsymbol{x}_k, \rho))\,\hat{\boldsymbol{u}}_k], \end{aligned} \tag{6.7}$$

where $\boldsymbol{f}^{\boldsymbol{x}}(\boldsymbol{x}_k, \rho) \in \mathbb{R}^{n_x \times 1}$ and $\boldsymbol{f}^{\boldsymbol{u}}(\boldsymbol{x}_k, \rho) \in \mathbb{R}^{n_u \times n_x}$ are terms independent of $\boldsymbol{u}_k$. As expected, the control $\boldsymbol{u}^{\mathrm{RL}}$ captures the model-plant mismatch caused by ρ.

6.4 Results

In the following sections, we show the results obtained from the investigation of the comparability of RL and NMPC in the form of benchmarking scenarios as well as their beneficial combination. In Section 6.4.1, we present the results obtained from benchmarking methods of NMPC and RL as presented in [103]. Here, the computational study proposed in Section 6.2 is applied on a benchmark example in the form of swing-up and balance task for a cart-pendulum.

In Section 6.4.2, we present the results of our work on combining methods of RL and NMPC. Here, we investigate the effect of the two proposed schemes of Section 6.3 on a squatting task of the robot *Leo* in simulation. Following this, the superior combination scheme is then additionally evaluated on the robotic platform.

6.4.1 Benchmarking Results on the Cart-pendulum Model

In the following section, we present the results obtained in our journal article [103]. We follow the proposed computational study and apply it to a common benchmark problem in the form of a swing-up and balancing problem for a *cart-pendulum* system [7, 95]. Our choice of this benchmark problem is motivated by the fact that main features of passive dynamic walking can be modeled by an inverted pendulum [183]. The same equivalence holds for the upper body of a more detailed model of a bipedal walker. The study presented in this sections highlights the differences in performance of NMPC and RL under structural and parametric uncertainties for this benchmark problem.

Cart-pendulum Model

The two-dimensional benchmark example studied in this article is a pendulum attached to a cart [7, 95], which is shown in Figure 6.3. The system consists of a cart with mass $\mathrm{m_M}$

and a pendulum that is attached to the cart's center of mass C_M.

The pendulum is a point mass m_m attached at the end of a massless rod of length l. The system has two degrees of freedom, namely the linear motion of the cart along the x-axis, described here by the coordinate $s \in \mathbb{R}$, and the rotary motion of the pendulum with respect to the cart, described by the angle $\varphi \in \mathbb{R}$. The only actuation is realized by a horizontal force $\boldsymbol{f}_s \in \mathbb{R}$ acting on the cart body.

The system's state is given as $\boldsymbol{x} = [s, \varphi, \dot{s}, \dot{\varphi},]^T \in \mathbb{R}^4$. Here, $s, \dot{s} \in \mathbb{R}$ denote the cart position and velocity, and $\varphi, \dot{\varphi} \in \mathbb{R}$ denote the pendulum's angle and angular velocity, respectively. The control $\boldsymbol{u} = \boldsymbol{f}_s$ is the force acting on the cart body.

In an ideal scenario, both the cart and the pendulum can move without friction along their respective degrees of freedom. For our second and third experiment, we employ uncertainty in the form of viscous friction at the rotary joint, i.e., in the pendulum joint bearing. This produces an internal torque $\tau_\varphi = -\kappa\mu\dot{\varphi}$ applied to the pendulum, where κ is a coefficient that depends on the configuration of the rotary joint, and μ is a viscous friction coefficient. Depending on whether or not this friction is included in the model, uncertainty in friction can be considered as a parametric or as a structural uncertainty.

By summarizing the positions, velocities and accelerations in $\boldsymbol{q} = [s, \varphi]^T$, $\dot{\boldsymbol{q}} = [\dot{s}, \dot{\varphi}]^T$ and $\ddot{\boldsymbol{q}} = [\ddot{s}, \ddot{\varphi}]^T$, the forward dynamics are given by $\ddot{\boldsymbol{q}} = (\boldsymbol{H}(\boldsymbol{q}))^{-1} (\boldsymbol{\tau} - \boldsymbol{c}(\boldsymbol{q}, \dot{\boldsymbol{q}}))$, where $\boldsymbol{H} \in \mathbb{R}^{2\times 2}$ is the system's mass matrix and $\boldsymbol{c}$ contain *Coriolis*, centrifugal, and gravitational terms and $\boldsymbol{\tau} = \left[\boldsymbol{f}_s, \tau_\varphi\right]^T \in \mathbb{R}^2$ denotes the actual actuation consisting of the one for the cart and for the pendulum.

We use the *Rigid Body Dynamics Library* [49, 50] for the efficient evaluation of the system's forward dynamics using the Articulated Body Algorithm (ABA) from [48]. The respective dynamic system in the form of an ordinary differential equation (1.12c) and initial values (1.12d) is then retrieved from the forward dynamics.

For simulations, we use the following parameters:

$$m_M = 10.0\,\text{kg}; \quad m_m = 1\,\text{kg}; \quad l = 0.5\,\text{m}; \quad 0\,\text{s} \leqslant t \leqslant 5\,\text{s}.$$

The cart and the pendulum are subject to simple constraints that enforce limits on the cart position and applicable force

$$-2.4\,\text{m} \leqslant s(t) \leqslant 2.4\,\text{m}, \quad -150\,\text{N} \leqslant \boldsymbol{f}_s(t) \leqslant 150\,\text{N}.$$

In the experiment with unknown viscous friction coefficient, the internal torque in the rotary joint of the pendulum is $\tau_\varphi = -\kappa\mu\dot{\varphi}$. We choose $\kappa = 1\,\text{m}^3$, and vary μ in the range

$$0.0\,\text{N}\,\text{s}\,\text{m}^{-2} \leqslant \mu \leqslant 0.2\,\text{N}\,\text{s}\,\text{m}^{-2}.$$

Problem Formulation

In this section, we provide formulations of the objective function used in the OC, NMPC and RL problems.

We investigate control scenarios for swing-up motions of the cart-pendulum system from the given initial state $\boldsymbol{x}(0) = [s(0), \varphi(0), \dot{s}(0), \dot{\varphi}(0)]^T = [0, \pi, 0, 0]^T$, which implies the system starts from rest, with the cart in the origin of the coordinate system and the pendulum pointing downwards. The goal of the task is to swing the pendulum up and to drive the cart back to the origin, i.e., to reach the final state $\boldsymbol{x}(T) = [s(T), \varphi(T), \dot{s}(T), \dot{\varphi}(T)]^T = [0, 0, 0, 0]^T$. This is realized for both the OC and the NMPC problem by the Lagrange term in the objective function

$$l(\boldsymbol{x}(t), \boldsymbol{u}(t)) = \|\boldsymbol{x}(t) - \bar{\boldsymbol{x}}(t)\|_{\boldsymbol{W}}^2 + \|\boldsymbol{u}(t)\|_{\boldsymbol{V}}^2, \tag{6.9}$$

as defined in (1.12a), where the weights $\boldsymbol{W} = \text{diag}(1, 0.5, 2, 0.2)$ and $\boldsymbol{V} = \text{diag}(0.0005)$ were chosen to scale the state elements to approximately the same range. Setpoint $\bar{\boldsymbol{x}} \equiv [0,0,0,0]^T$ is set according to the definition of the task. The benchmark constraints defined in Section 6.4.1 can be directly formulated as path constraints (1.12f) on both states and controls, while the prediction horizon of the NMPC controller is a subinterval of the problem horizon.

The discount rate γ, which is necessary for solving a continuing task in RL, affects the obtained RL solution. Therefore, to make the NMPC and RL results comparable, we include the same discount rate value into the objective function of OC and NMPC. The effect of the discount on the NMPC formulation is that it increases the focus on the beginning of the horizon by providing a weighting over time, i.e., the further the event lies in the future of the horizon, the less it will be considered for the computation of the optimal behavior.

To allow solving the control problem in real-time using NMPC alone and together with MHE, the problem formulation has to be adapted for the current algorithmic setup in the optimal control software package MUSCOD-II [109, 110]. Due to the nonlinear behavior of the cart-pendulum system, a control rate of at least 40 Hz has to be chosen to generate sufficient contraction in the real-time iteration scheme and to enable the standard structure exploitation for the sequential quadratic programming method. However, this increases the number of shooting nodes and the computational time. A sweet spot in performance is obtained by using a horizon of 3 s.

In RL, we construct the reward from the same *Lagrange* term (6.9), but we additionally add a negative reward and a shaping function $S(\boldsymbol{x}_k, \boldsymbol{x}_{k+1})$:

$$r(\boldsymbol{x}_k, \boldsymbol{x}_{k+1}) = \begin{cases} -1000 & \text{if } \boldsymbol{x}_{k+1} \in \mathcal{X}_\text{a}, \\ -l(\boldsymbol{x}_{k+1}, \boldsymbol{u}_k) + S(\boldsymbol{x}_k, \boldsymbol{x}_{k+1}) & \text{otherwise,} \end{cases} \tag{6.10}$$

where $\mathcal{X}_\text{a}$ is a set of absorbing states that lie outside of the cart's position constraints defined in Section 6.4.1.

The shaping function denoted by $S(\boldsymbol{x}_k, \boldsymbol{x}_{k+1})$ leaves the target objective unchanged but allows to reduce steady-state error. Due to the quadratic terms in the definition of $l(\boldsymbol{x}_{k+1}, \boldsymbol{u}_k)$, rewards become small for balancing states where all elements of the state are close to zero, except possibly the cart position s_k. This effect has previously been described in [41], where authors noticed that the quadratic reward, the L^2-norm, penalized large velocities much more than small steady-state errors. They solved the issue by showing that the absolute value reward, the L^1-norm, yielded a response with negligible errors. This solution is not directly applicable here, because our aim is to obtain results as similar to OC as possible, which uses a quadratic cost function. Instead, inspired by [41], we introduce a potential-based shaping function [135] encoded as $S(\boldsymbol{x}_k, \boldsymbol{x}_{k+1}) = \gamma\Psi(\boldsymbol{x}_{k+1}) - \Psi(\boldsymbol{x}_k)$, where $\Psi(\boldsymbol{x}_k) = \psi\|\boldsymbol{W}\boldsymbol{x}_k\|_1$ is the sum of absolute values of weighted state elements multiplied by a shaping weight ψ. The purpose of this shaping function is to provide a stronger guidance towards $\boldsymbol{x} = [0,0,0,0]^T$ in the region of the state space where the quadratic reward function fails to do so.

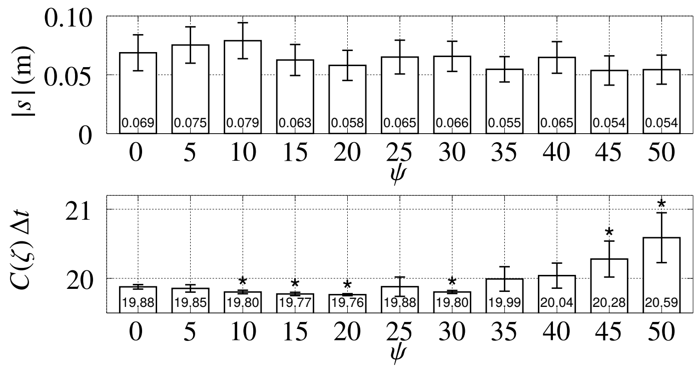


Figure 6.4: Influence of the shaping function on the results of the "RL" method from our article [103]. Top: absolute error in the cart position at the end of an episode depending on the weight ψ of the shaping function. Bottom: total cost of the trajectory. Shaping is not used for $\psi = 0$. Numbers inside of the bars show the mean value of the error averaged over 50 independent runs, while the error bars show the upper and lower 95% confidence limits. Statistically significant result, for which the p-value is less than 0.05, are marked with $\star$ above the bars.

Effect of shaping in Reinforcement Learning The results of the effect of the shaping weight ψ on the cost are presented in Figure 6.4. According to the graph, for a shaping weight of up to 20, the total cost reduces, and then starts increasing again. Note that the total cost is calculated using (6.2), which does not include the shaping weight ψ. The error in position is more volatile, but one may notice that it gets smaller for higher shaping weights. We selected $\psi = 20$, because our primary goal was to reduce the total cost and accept a moderate steady-state error.

It is possible to include hard constraints directly into the OC formulation by (1.12f). However, in the RL formulation, they have to be reformulated as soft constraints, which is done by including them directly in the reward function in the form of a negative reward as described in (6.10). This essentially changes the original optimization problem by introducing a trade-off between receiving positive rewards and avoiding negative ones. For example, once a very large negative reward is received, it will force the system to never violate this constraint again, even at a price of obtaining lower positive rewards. On the contrary, a small negative reward will allow infrequent violation of the constraint, which will slow down learning and may even damage a real-world system. In this benchmark example, the trade-off has a mild effect, because the cart position constraints are rather loose. While constraints are violated a few times in the process of learning, the final result is free of constraint violations.

For the cart-pendulum benchmark, the optimal combination of parameters can be found in Table 6.1. For NMPC, the tolerances were chosen according to best practices, the horizon length as well as the sampling period were chosen such that "iNMPC" uses the same formulation as "OC" and that "NMPC" computes feedback in less than 5 ms. The discount rate γ was chosen according to the RL formulation. For RL, we found the parameters using an exhaustive grid search. It generated tuples of candidate parameters by selecting them from a set of predefined parameter values commonly used in the actor-critic literature, c.f. [62].

For the RL policy and value function approximation, we used tile coding with 16 tilings, each of size $[2.5, 0.1\pi, 2.5, 0.5\pi]^T$. However, pendulum states close to the state $\boldsymbol{x} = [0,0,0,0]^T \in \mathbb{R}^{n_x}$ require a finer resolution of the function approximator. Therefore,

Table 6.1: Parameters of OC, NMPC and RL formulations for the cart-pendulum problem from our article [103]. The first group of parameters is relevant to OC and NMPC, where the value in brackets is given for real-time NMPC. The second group of parameters is relevant only to RL.

Parameter		Value
OC/NMPC:		
Horizon length	T	5 s (3 s)
Discount rate	γ	0.99
Sampling period	$T_s^{\text{OC/NMPC}}$	0.05 s
KKT-Tolerance		10^{-7}
Integration accuracy		10^{-6}
RL:		
Episode length	T	5 s
Discount rate	γ	0.99
Sampling period	T_s^{RL}	0.05 s
Number of learning episodes		$2.0 \cdot 10^5$
Additional learning episodes		5%
Eligibility discount rate		0.65
Exploration variance	$\Sigma_{\boldsymbol{u}}$	$0.004\,\boldsymbol{u}_{\max}^2$
Critic learning rate	α_{c}	0.10
Actor learning rate	α_{a}	0.01

before projecting a state on the tiles, we rescale each state element to the interval $[-1, 1]$, and then apply a squashing function with the parameter $\rho = 5$:

$$\Omega(\boldsymbol{x}^j, \rho) = \frac{(1+\rho)\boldsymbol{x}^j}{1+\rho|\boldsymbol{x}^j|}, \quad \forall j \in \{1, \ldots, n_{\boldsymbol{x}}\},$$

where $\boldsymbol{x}^j$ denotes a scaled element. This effectively controls resolution by a multiplier that varies continuously, from $(1+\rho)^{-1}$ in the downward position of the pendulum, to $1+\rho$ in the balancing state $\boldsymbol{x} = [0, 0, 0, 0]^T$.

Results on the Cart-Pendulum

In order to assess the similarity of the frozen methods, we analyze their performance for the cart-pendulum system without uncertainties. The resulting trajectories and the R^2 measure results are shown in Figure 6.5 and Table 6.2, respectively. All three methods show a qualitatively similar behavior and are successful in swinging the pendulum up and balancing it there. An overall similarity between "RL" and "iNMPC" of more than 90.3 % was achieved in terms of the R^2-measure. Comparing the R^2 values of a mean trajectory with the mean of R^2 values of the individual trajectories of "RL", we observe that the mean trajectory is closer to "iNMPC", while the individual trajectories exhibit some variability around their mean. The control trajectories $\boldsymbol{f}_s$ of "iNMPC" and "RL" differ in a small

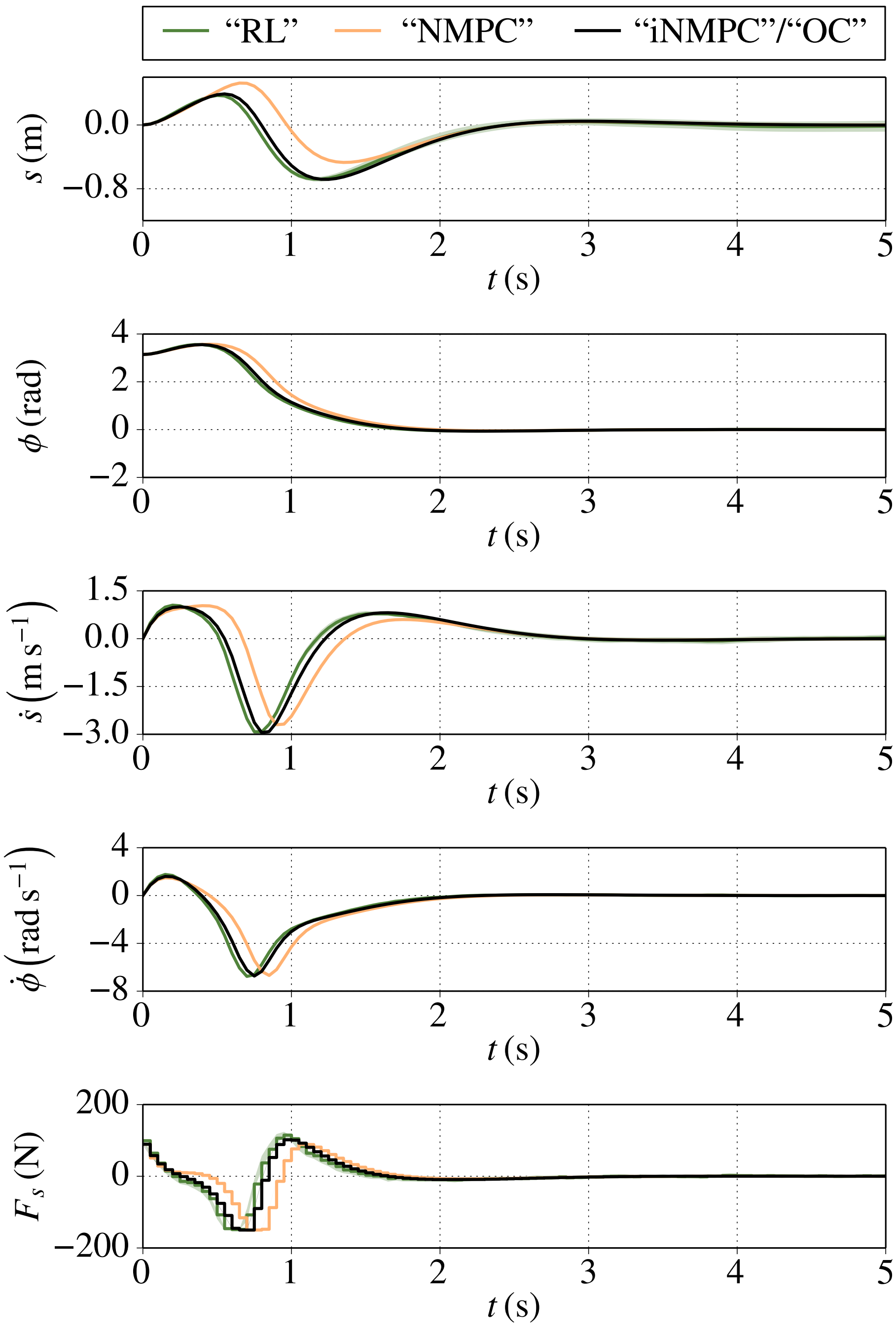


Figure 6.5: State and control trajectories obtained by the frozen methods for the cart-pendulum system without uncertainties from our article [103]. For "RL" the mean and standard deviation of 50 trajectories is shown. Y-axes variables $s, \dot{s}$ denote the cart position and velocity and $\varphi, \dot{\varphi}$ are the respective quantities of the pendulum. f_s is the force acting on the cart body.

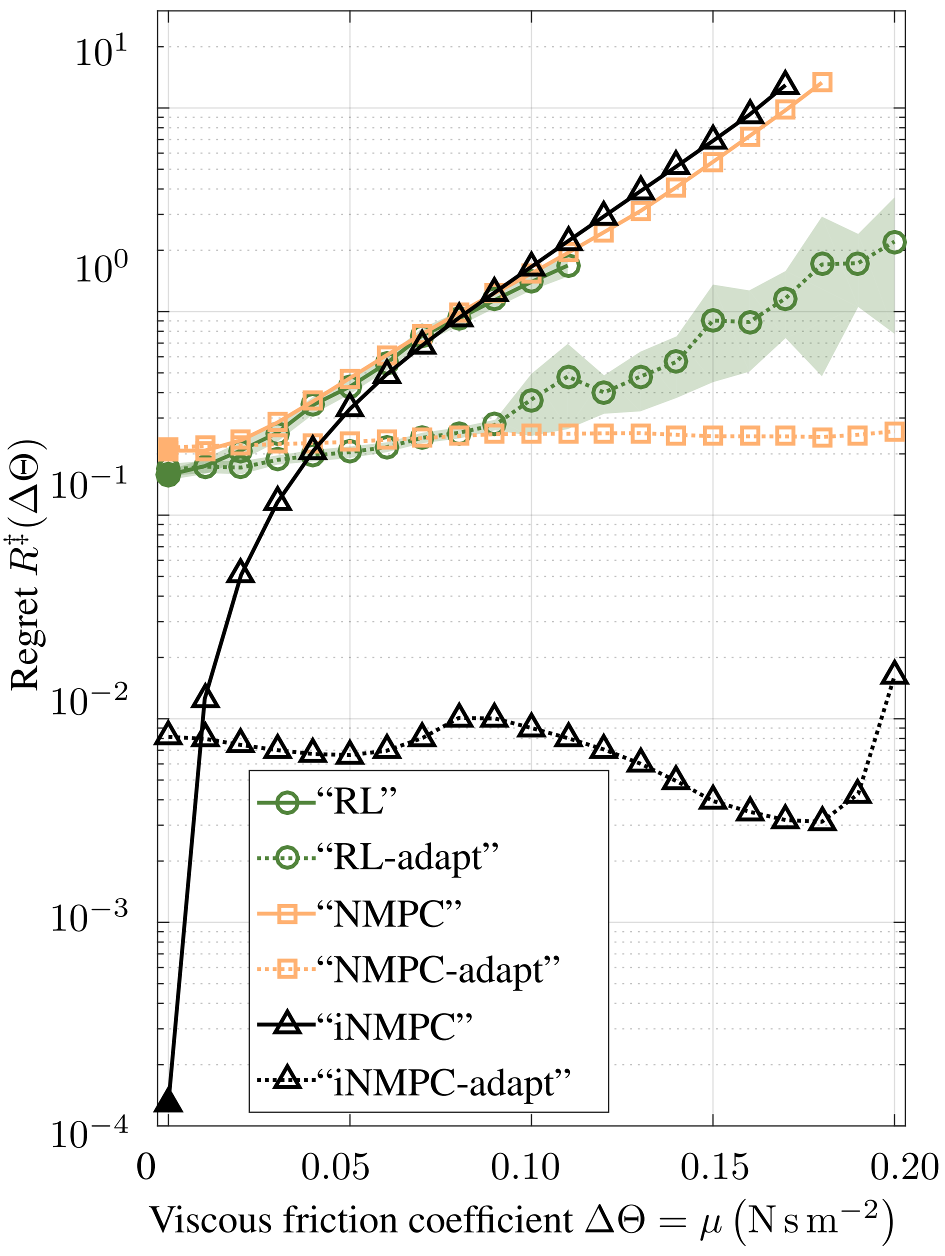


Figure 6.6: The graph of regrets, from our article [103], for the cart-pendulum system comparing performance of described methods against the optimal solution. The means with the upper and lower 95% confidence limits are shown for controllers with a stochastic component (50 samples per viscous friction coefficient were used). To indicate the scale of the plot, we employ three filled markers at $\mu = 0\,\mathrm{N\,s\,m^{-2}}$ that correspond to the trajectories. A further reference for interpretation of the scale: If the cart stood still and the pendulum hung down, then the regret of the solution would be equal to 11.73.

Table 6.2: Similarity of the cart-pendulum trajectories in terms of the coefficient of determination (R^2) from our article [103]. For RL we first report similarity of the mean trajectory, and second we report the mean of R^2 values.

Methods		s	φ	$\dot{s}$	$\dot{\varphi}$	f_s
"RL"- "OC", R^2 mean trajectory	(%)	98.1	99.8	96.4	98.4	92.9
"RL"- "OC", mean R^2	(%)	93.9	99.8	95.3	98.2	90.3
"NMPC"- "OC"	(%)	56.6	98.6	66.7	86.9	48.2
"iNMPC"- "OC"	(%)	100.0	100.0	100.0	100.0	100.0

time delay and are otherwise comparable. However, after approximately 1.0 s, we find that "RL" demonstrates variability between control trajectories compared to "iNMPC". Due to differences in control actions, the state trajectories start to differ slightly after approximately 0.5 s and recover from that after approximately 2.0 s; only the "RL" cart position s remains to show small steady-state errors. "NMPC" results deviate more from "iNMPC", and an overall similarity of approximately 48.2 % was achieved in terms of the R^2-measure.

Next, we show the behavior of both frozen and adaptive methods under the effect of uncertainties. Simulation results against variations of the friction parameter μ are shown in Figure 6.6. In the absence of friction, "iNMPC" does not reach zero regret due to numerical approximations. All of "RL", "iNMPC" and "NMPC" show a similar asymptotic behavior in reaction to the variation of the friction coefficient. "iNMPC" shows the lowest regret for low values of viscosity and "NMPC" regret is the largest. Larger values increase the regret, and for the value of $\mu = 0.09\,\mathrm{N\,s\,m^{-2}}$, "RL" yields a lower regret than "iNMPC". All three frozen methods violate the position constraints of the cart. For "RL", this happens at $\mu = 0.12\,\mathrm{N\,s\,m^{-2}}$ and for "iNMPC" and "NMPC" at $\mu = 0.18\,\mathrm{N\,s\,m^{-2}}$ and $\mu = 0.19\,\mathrm{N\,s\,m^{-2}}$, respectively.

The adaptive methods, "RL-adapt", "iNMPC-adapt", and "NMPC-adapt", show a different reaction to the variation of the friction coefficient. Both "iNMPC-adapt" and "NMPC-adapt" show a constant performance under the effect of the variation of friction. Note the logarithmic scale in Figure 6.6; the variances in performance of "iNMPC-adapt" and "NMPC-adapt" are similar, but appear differently due to the logarithmic scale. "RL-adapt" performs better than "NMPC-adapt" for smaller uncertainties of a value of up to $0.07\,\mathrm{N\,s\,m^{-2}}$ and is then outperformed by NMPC. "RL-adapt" regret is an order of magnitude higher than the regret of "iNMPC-adapt", and the RL performance deteriorates with higher friction. For friction coefficients larger than $0.09\,\mathrm{N\,s\,m^{-2}}$, "RL-adapt" shows a much higher variance in regret than for lower friction.

Comparing "RL-adapt" with the frozen method "iNMPC", the graphs show that "iNMPC" cannot compete with RL after the break-even point at $0.04\,\mathrm{N\,s\,m^{-2}}$. This viscous friction coefficient value corresponds to 6.1 % of the difference in energy consumed by the ideal ($0.00\,\mathrm{N\,s\,m^{-2}}$) and disturbed ($0.04\,\mathrm{N\,s\,m^{-2}}$) system. Compared to all three frozen methods, "RL-adapt" performs much better after the break-even point, and the gap grows with larger uncertainties.

A summary of the main results of comparison is presented in Table 6.3.

Discussion

Our results demonstrate that, with a proper formulation of the optimal control task, it is possible to obtain similar results for the three frozen methods on an ideal system. For the cart-pendulum benchmark example, a good similarity between “RL”, “OC” and “iNMPC” was achieved. However, for “NMPC”, the expected deviation from the optimal solution due to the tuning towards a real-time feasible controller is shown in Figure 6.6. This is a consequence of the implementation and not of the approach. A speed-up of the implementation by using a multi-level real-time iteration scheme [14, 96], by using a state-of-the-art sequential quadratic programming method tailored for multiple shooting [79] and replacing the quadratic program solver [154] could address the current time limitations. With a speed-up of the computations, a theoretical coverage of the area between the curves of “iNMPC” and “NMPC” is therefore possible. This will however not influence the already found break-even points, because the asymptotic behavior of the frozen NMPC methods is determined through “iNMPC” as the best possible outcome. Considering this, only an improvement for smaller variations of the friction coefficient is to be expected.

The adaptive methods successfully avoid constraint violations and substantially reduce regret compared to their frozen counterparts over the whole range of viscous friction coefficients. Interestingly, when the coefficient is not present in the system or has low values, “iNMPC-adapt” results in a higher regret than “iNMPC”, and the same holds for the fast versions of NMPC. The reason for this is that the combination of NMPC and MHE in the form of “iNMPC-adapt” and “NMPC-adapt” starts with an initial guess of the parameter that is adapted during the actual run of the system. Any mismatch between measurements and values predicted by the model will lead to adaption of the parameters in MHE. This is a crucial difference to RL, which adapts to the uncertainty through multiple trials prior to an assessment run, while NMPC is unaware of the mismatch at first. The cart-pendulum task is very sensitive to changes during movement initiation. Therefore, a wrongly identified parameter in the beginning can already cause substantial differences in terms of regret, which is seen in Figure 6.6. This effect is amplified for “NMPC-adapt” through the mentioned performance loss due to the real-time feasibility tuning.

In the absence of uncertainties, “iNMPC-adapt” performs superior to both “RL” and “RL-adapt”. This does not come as a surprise, as NMPC methods were running offline, they were using the model of the correct system and, moreover, the uncertain parameter was defined explicitly. However, “iNMPC” outperforms RL methods only for small values of uncertainties. In case of medium and large uncertainties, there exist break-even points after which “RL” and “RL-adapt” obtain lower regret. We remark that the estimation of the difference between ideal and uncertain systems in terms of energy is *ad-hoc*, and more generic measures for model uncertainties should be used in the future.

In the non-ideal setting, the performance of NMPC becomes comparable to RL. Nonetheless, one cannot directly report similarity of “NMPC-adapt” and “RL-adapt”; while regret of “NMPC-adapt” is almost constant for the whole range of uncertainties, the regret of “RL-adapt” significantly increases. The explanation for this effect is twofold. First, for any value of uncertainty, “RL-adapt” was learning for a fixed additional 5 % of time. The larger the value of uncertainty, the more time “RL-adapt” requires to adapt to a new parameter value. This can be supported by the fact of an increasing variance of RL regret, which indicates that the actor-critic algorithm simply did not have enough time to converge in areas of high uncertainties. Second, for large uncertainties, it might be necessary to

Table 6.3: Summary of results of the comparison of RL and NMPC from our article [103].

Category	Findings
Achieved similarity of "iNMPC" and "RL" methods on the ideal system	more than 90.3 %
Break-even point: the difference in energy consumed by ideal and noisy systems after which "RL-adapt" performance becomes better then "iNMPC"	6.1 %
Best performing algorithm under parametric uncertainties	"iNMPC-adapt"
Best performing algorithm under structural uncertainties	"iNMPC" before the break-even point and "RL-adapt" after the break-even point

significantly change the control strategy, i.e., to learn a new policy rather than adapt an ideal one. This will probably require more learning efforts, to first unlearn the initial policy, and then to learn a realistic one.

Several issues were encountered while formulating the benchmark problem with the aim of obtaining identical results. These issues are known to OC and RL communities, but, to the best of our knowledge, they were never explained in the same context before.

Issues related to Optimal Control

1. In contrast to RL, derivative-based methods of optimization to solve the discretized OC problem require a continuously differentiable formulation of the problem.
2. The performance of the ideal NMPC-MHE combination ("iNMPC-adapt"), for which computational time was neglected, is one order of magnitude better in terms of regret than the corresponding real-time version, mainly caused by a shortened prediction horizon used in the latter.

Issues related to Reinforcement Learning

1. In this work, we use a model-free RL method, which means that the transition model of a system is unknown *a priori*.
2. Learning a solution with a quality comparable to OC takes many episodes.
3. Constraints in the original OC problem are included into the RL formulation by means of negative rewards received for violating the constraints.
4. For the benchmark example, the OC objective function has been modified. Formulating a reward function by simply negating the OC objective results in a) a very slow learning in cases when no negative reward is used, b) an inability to learn or even a divergence of the value function if $\gamma = 1$.
5. For symmetrical problems, RL can use state space reduction techniques. For example, for the cart-pendulum example it is possible to wrap the pendulum angle to the $[-\pi, \pi)$ interval, which results in two equally possible optimal trajectories under our objective function. OC generally does not allow implementation of such techniques if they violate the smoothness assumptions.
6. When learning with a quadratic objective function, which is often used in OC, it is

useful to implement learning techniques that are able to reduce steady-state error while leaving the objective function unchanged, for example reward shaping.

The presented quantitative comparison is particularly important for our future plans of combining RL and NMPC to control a more complex system with a high number of degrees of freedom. One possible combination could be that RL learns a real model for NMPC, while NMPC provides a backup of a RL exploratory policy. Another scheme could be that RL receives a control signal from NMPC as a suggestion. Initially RL passes this suggestion to the actuators, but at a later stage it takes over in state space areas where it is confident. Independently of the chosen combination strategy, for value function-based RL it is important to retain the *Markov* property, which may impose restrictions on the NMPC controller as well. For example, such RL methods usually avoid time as a state, hence, the trajectory-tracking NMPC should not be used in the suggestion-based scheme.

6.4.2 Results on Model-plant Mismatch Learning

In the following Section, we present the results obtained in [104]. First, an in-depth comparison of the proposed combination schemes of RL and NMPC as presented in Section 6.3 is performed. Here, a dynamic model of *Leo* robot is employed to validate the hybrid control strategies to a squatting task in simulation. The simulation results are complemented with an experiment on the real robot demonstrating the advantage of our proposal in the presence of temperature- and torque-dependent *Coulomb* friction.

Experimental Setup

The robot under consideration is the robot *Leo* of TU Delft. We refer to Section 1.2.3 for the details concerning the robot. We only recall the fact that the gearboxes in the motors are subject to *Coulomb* friction dependent on motor temperature and torque. This friction represents the dominating factor causing model-plant mismatch.

The task of reaching upper and lower setpoints which together realize a squatting motion is the same as presented in Section 5.5.3. Therefore, we recapitulate the most important aspects.

The robot state $\boldsymbol{x} = (\boldsymbol{\varphi}, \dot{\boldsymbol{\varphi}}, \boldsymbol{p}, \tau_{\text{knee}})^\top$ is defined as a vector of all but the shoulder joint angles $\boldsymbol{\varphi}$, corresponding angular velocities $\dot{\boldsymbol{\varphi}}$, setpoint height $\boldsymbol{p} \in \{0.28\,\text{m}, 0.35\,\text{m}\}$ and mean temperature of knee motors τ_{knee}.

Exploiting the symmetry of *Leo*, we apply the same control voltages to both legs. The shoulder is actuated using a PD controller. The setpoints are switched over when the robot root point appears to be within $\pm 0.01\,\text{m}$ away from it. In the simulated experiment, we add shoulder angle and velocity to the state and the shoulder voltage is learned. The robot is initialized in a setpoint chosen randomly at the beginning of every episode. The idealized model is made such that no friction in joints is present. For the realistic model, we add *Coulomb* friction $\boldsymbol{u}_{\text{fr}} = -0.2\tanh(2000\dot{\boldsymbol{\varphi}})$ in all joints.

The control delay of $13.0 \pm 1.7\,\text{ms}$ comprises measurement, computation and actuation delays. A sampling period of $33.3\,\text{ms}$ is chosen to be larger than the control delay.

Objective Function and Constraints Here, the NMPC objective function (1.12a) is

$$\begin{aligned} l(\boldsymbol{x}, \boldsymbol{u}) = {} & 0.05\,(h(\boldsymbol{\varphi}) - \boldsymbol{p})^2 + 0.10\,(c(\boldsymbol{\varphi}) - \bar{\boldsymbol{x}}_c)^2 \\ & + 0.05\,(pose(\boldsymbol{\varphi}) - 0.3)^2 + 0.003\,\dot{\boldsymbol{\varphi}}^\top \dot{\boldsymbol{\varphi}}, \end{aligned} \tag{6.11}$$

where the first term accounts for the vertical distance $h(\boldsymbol{\varphi})$ to a setpoint $\boldsymbol{p}$, the second term maintains the horizontal position of the center of mass $c(\boldsymbol{\varphi})$ close to predefined value $\bar{x}_c$, the third term containing $pose(\boldsymbol{\varphi}) = \varphi_{\text{ankle}} + \varphi_{\text{knee}} + \varphi_{\text{hip}}$ is used as a regularization term improving the stability of the robot, and the last term favors small velocities.

The cost of some discretized trajectory $\boldsymbol{x}_0, \boldsymbol{u}_0, \boldsymbol{x}_1, \boldsymbol{u}_1, \dots$ obtained using policy $\boldsymbol{u}_k = \boldsymbol{\pi}(\boldsymbol{x}_k)$ is denoted as $\ell = \sum_k l(\boldsymbol{x}_k, \boldsymbol{u}_k)$.

We formulate static stability as a constraint $\boldsymbol{g}(\boldsymbol{x}, \boldsymbol{u}) = (\boldsymbol{x}_t - c(\boldsymbol{\varphi}), c(\boldsymbol{\varphi}) - \boldsymbol{x}_h)^\top$, where $\boldsymbol{x}_t$, $\boldsymbol{x}_h$ denote the position of the tip and the heel of robot feet. Additionally, robot angles and controls are subject to constraints

$$\begin{bmatrix} -1.57 \\ -2.53 \\ -0.61 \end{bmatrix} \leqslant \varphi_i \leqslant \begin{bmatrix} 1.45 \\ -0.02 \\ 2.53. \end{bmatrix} \qquad \begin{matrix} \{|\boldsymbol{u}_i|, |\hat{\boldsymbol{u}}_i|, |\boldsymbol{u}_i^{\text{RL}}|\} \leqslant 10.0\,\text{V} \\ \\ i \in \{\text{ankle}, \text{knee}, \text{hip}\} \end{matrix}$$

For the MPML approach the reward (6.4) is calculated based on the joint angles $r(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{x}_{k+1}) = -\|\boldsymbol{\varphi}_{k+1} - \hat{\boldsymbol{\varphi}}_{k+1}\|_2$.

Parameters The time horizon T for NMPC optimization is selected to be 1 s. One learning or testing episode lasts for 15 s. Advantage, critic and actor learning rates are chosen to be 0.01, 0.10 and 0.01, respectively. Additional parameters include discount rate $\gamma = 0.97$, and an eligibility trace decay rate of 0.65. We rely on NMPC to avoid falls of the robot, therefore negative reward R_{a} is not used.

Exploration is achieved by *Ornstein-Uhlenbeck* noise model $\Delta\boldsymbol{u}_{k+1} = 0.5\Delta\boldsymbol{u}_k + \boldsymbol{\mathcal{N}}(0, \sigma)$ with $\sigma = 0.005$. For the real experiment, we select a higher advantage learning rate and increased exploration.

Evaluation

For quantitative assessment, we evaluate objective (6.11) separately for reaching upper and lower setpoints $\ell^{\{\text{u,l}\}} = \sum l(\boldsymbol{x}, \boldsymbol{u})$. Second, we evaluate the minimization of model-plant mismatch (6.4) by computing the negative value of undiscounted return $\mathcal{E}^{\{\text{u,l}\}} = -\mathfrak{R}^{1,\{\text{u,l}\}} = \sum \|\boldsymbol{e}\|_2$ for reaching both setpoints separately. Third, we calculate root mean squared error (RMSE) between transitions obtained by both approaches and NMPC executed on the idealized model. Finally, to experimentally demonstrate safety barriers imposed by NMPC, we calculate the cumulative number of falls and violation of NMPC constraints at multiple levels of exploration noise σ for two proposed approaches and DPG.

For qualitative assessment, we calculate the number of squats the robot performs during the testing episode. This measure should be accounted only as a learning progress indicator since it is not included in the optimization objective.

Results in Simulation

At first, we show the performance of standalone NMPC on the idealized and realistic models. Results, shown in Figure 6.7 and Table 6.4, demonstrate a significant difference. On the idealized model, NMPC realizes three squats. On the realistic model, NMPC can reach neither upper nor lower switching points, which results in the inability to squat and high costs ℓ. This result motivates the need for an adaptive component in the controller. Next, we study the performance of the proposed approaches compared to the baseline performance of NMPC. Learning was performed for 10^6 time steps, which was enough

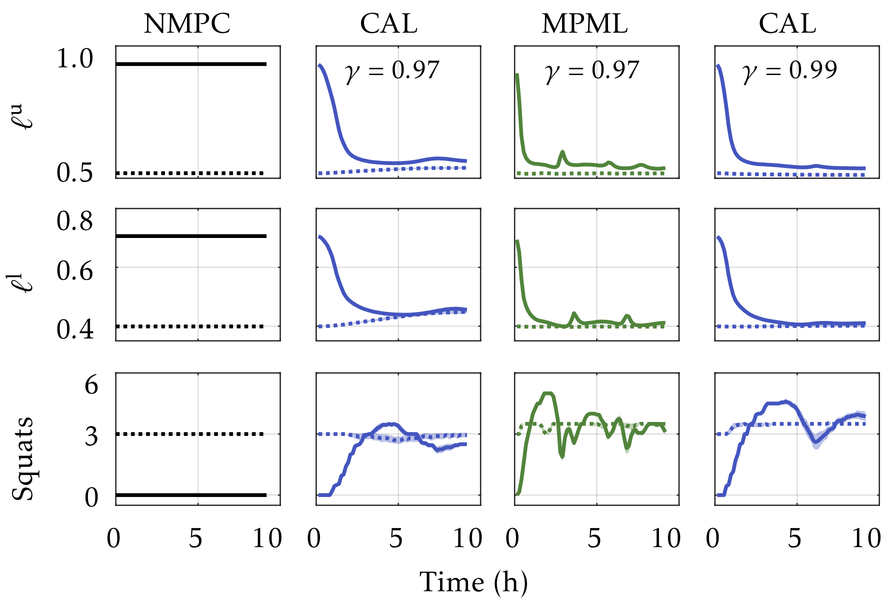


Figure 6.7: Learning to reach upper (*top row*) and lower setpoints (*middle row*) in simulation from our article [104]. Number of squats is given in the *bottom row*. *Dotted* and *solid* lines show learning on the idealized and realistic models, respectively. Means with upper and lower 95% confidence limits are shown for 10 runs.

for CAL and MPML to converge, while DPG required about hundred times more steps. Therefore, its results were excluded from the comparison.

We notice that on the idealized model the performance of both approaches becomes slightly worse than the baseline performance of NMPC. For CAL$^{\gamma = 0.97}$, we observe deviation from the optimal policy which is seen in the increase of ℓ cost. Yet, the approach is able to keep the number of squats close to the baseline value. For MPML, costs ℓ do not change, however the number of squats increases by 0.5 which indicates that the approach reaches the upper setpoint right before the episode ended. Deviation of the learned trajectory from the idealized one is captured by RMSE which is nonzero for both approaches. For the CAL and MPML approaches the mean RMSE is 68.6 % and 91.7 % below the reference of 20.4 ± 0.2 which is RMSE of NMPC trajectory obtained on the realistic model.

Results on the realistic model experiment show that both approaches improve the performance of NMPC. The decrease of the ℓ cost is at least 35.2 % and 41.9 % for CAL$^{\gamma = 0.97}$ and MPML approaches, respectively. Looking at the number of squats, we notice that both approaches overshoot the NMPC baseline of 3 squats and then continuously reduce the number towards the baseline. RMSE increases comparing to the idealized model experiment, but still remains significantly below the reference value.

To find the reason of CAL$^{\gamma = 0.97}$ performance decrease on the idealized model, we test the approach with a discount rate of $\gamma = 0.99$. Increasing γ leads to a longer planning horizon which makes RL return (6.1) more similar to NMPC objective (6.11). It turns out that CAL$^{\gamma = 0.99}$ obtains lower ℓ costs not only comparing to CAL$^{\gamma = 0.97}$ but also comparing to the baseline NMPC. We believe the reason of this is due to the early switching of setpoints described above. While CAL$^{\gamma = 0.99}$ can learn this fact, NMPC is not aware of it.

In Figure 6.8, we plot the number of falls and NMPC constraint violations accumulated

Table 6.4: Final performance of proposed combination approaches for RL and NMPC from our article [104]. Significant width of the confidence interval is shown in brackets.

Method	ℓ^{u} $\times 10$	ℓ^{l} $\times 10$	Number of squats	RMSE $\times 10^4$
Idealized model				
NMPC	5.2	**4.0**	**3.0**	**0.0**
CAL$^{\gamma = 0.97}$	5.4	4.4	3.0(0.1)	6.4(1.5)
MPML	5.2	**4.0**	3.5	1.7(0.1)
CAL$^{\gamma = 0.99}$	**5.1**	**4.0**	3.5	11.8(4.0)
Realistic model				
NMPC	9.3	7.1	0	20.1
CAL$^{\gamma = 0.97}$	5.6	4.6	2.5	10.4(0.5)
MPML	**5.4**	**4.1**	**3.2(0.1)**	**4.3(1.2)**
CAL$^{\gamma = 0.99}$	**5.4**	**4.1**	3.9(0.2)	13.2(1.2)
Real robot				
NMPC $^{38\,^\circ\mathrm{C}}$	22.0(2.1)	–	0	
MPML	**7.9(2.9)**	**4.4(1.8)**	**3.3(1.1)**	**94.9(26.5)**

over 10^6 time steps. Here, we prematurely stopped DPG for the sake of results comparability. The proposed approaches are almost identical. Both approaches prevent the robot from falling, while constraints get violated at $\sigma > 0.1$. A different picture is seen in DPG results. The smallest number of falls and constraint violations is achieved for the value of $\sigma = 0.02$. Smaller σ reduces the learning pace, while larger values increase chances of fall.

MPML learns almost twice as fast as CAL and does not exhibit deviating behavior, which are the main reasons for testing the approach on the real robot.

Results on the Real Robot

Results of standalone NMPC on *Leo* are shown in Figure 6.9. While on the idealized model NMPC successfully reaches switching points, on the real robot the controller is not able to do so. The reason is due to *Coulomb* friction in gearboxes, which depends on motor temperature and the applied torque. Modeling these effects is possible but requires a precise identification of the underlying physical processes.

To circumvent this problem, we apply the proposed MPML approach. In Figure 6.9, results of three independent runs are shown.

We observe that the approach successfully realizes squatting by learning the compensation signal. However, the variation of motor temperature leads to noticeable differences in the root point trajectories. The trajectory obtained in the 3rd run is noticeably less noisy and squatting itself is faster than the one achieved in 1st and 2nd runs. In particular, the gradient of the downwards motion in the 3rd run is very similar to NMPC executed on the idealized model. However, the slow approach of the 3rd run towards the setpoints diminishes compared to the idealized NMPC run.

Variation of motor temperature also leads to differences in the learning progress, see Figure 6.10. 1st and 2nd runs require substantially longer time before the squatting cycle is observed. This is due to the increase of motor temperature above 40.0 °C which requires

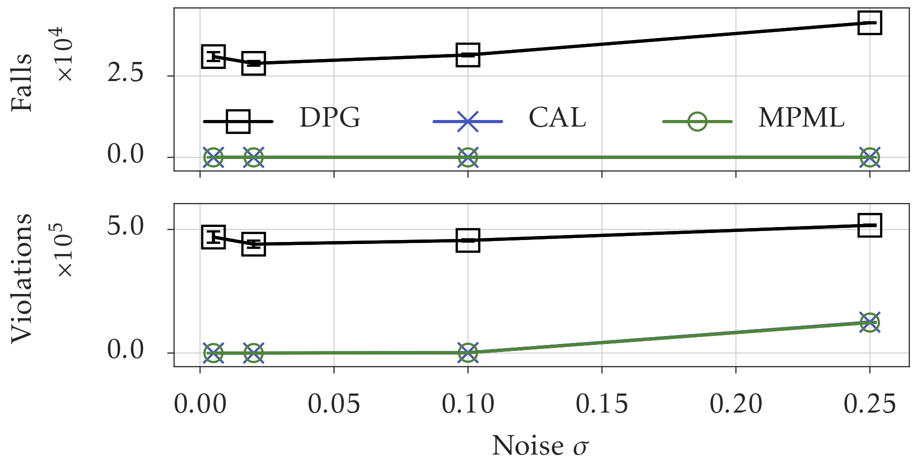


Figure 6.8: The number of falls (*top*) and NMPC constraint violations (*bottom*) as functions of σ from our article [104]. Means with upper and lower 95 % confidence limits are shown for 10 runs.

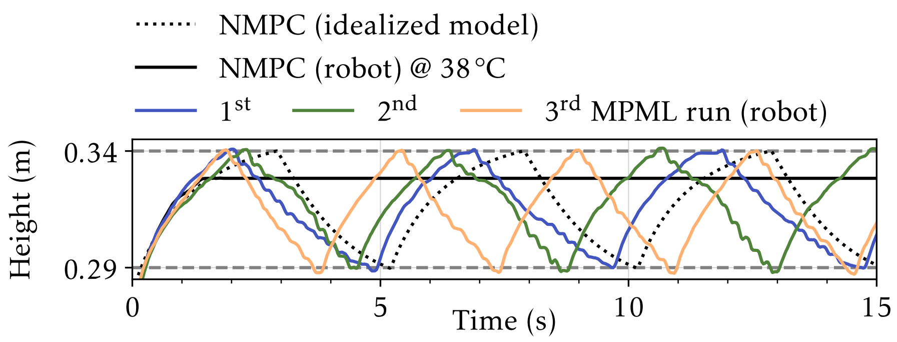


Figure 6.9: Robot root point trajectories obtained after learning from our article [104].

additional exploration of the state space. Nevertheless, all runs successfully attain a stable squatting cycle after 7.25 h.

Model-plant mismatch $\mathcal{E}^{u}$ and $\mathcal{E}^{l}$ is minimized to about 0.5 and 0.8 for reaching upper and lower setpoints, respectively. As it was expected, minimization of model-plant mismatch leads to minimization of the nominal controller objective (1.12a) shown by plots ℓ^{u} and ℓ^{l}. The smallest final costs are incurred by the 3rd run because it was stuck the least due to temperature fluctuations, while the largest costs are incurred by the 1st run which was stuck the most.

RMSE after learning is calculated in Table 6.4. RMSE of MPML trajectories is much higher than for the realistic model, and it also exhibits more variability. Unfortunately, it is not possible to obtain the reference RMSE value of the real robot.

Figure 6.11 shows the MPML knee control signal and the RL compensation component of it. For reference, NMPC control on the idealized model is also shown. MPML controls are very oscillatory comparing to NMPC. Nonetheless, the robot neither fell down, nor were its motors damaged, which is a significant result, c.f. [155]. As is expected with *Coulomb* friction compensation, RL learned to apply positive and negative controls for the upward and downward motions, respectively.

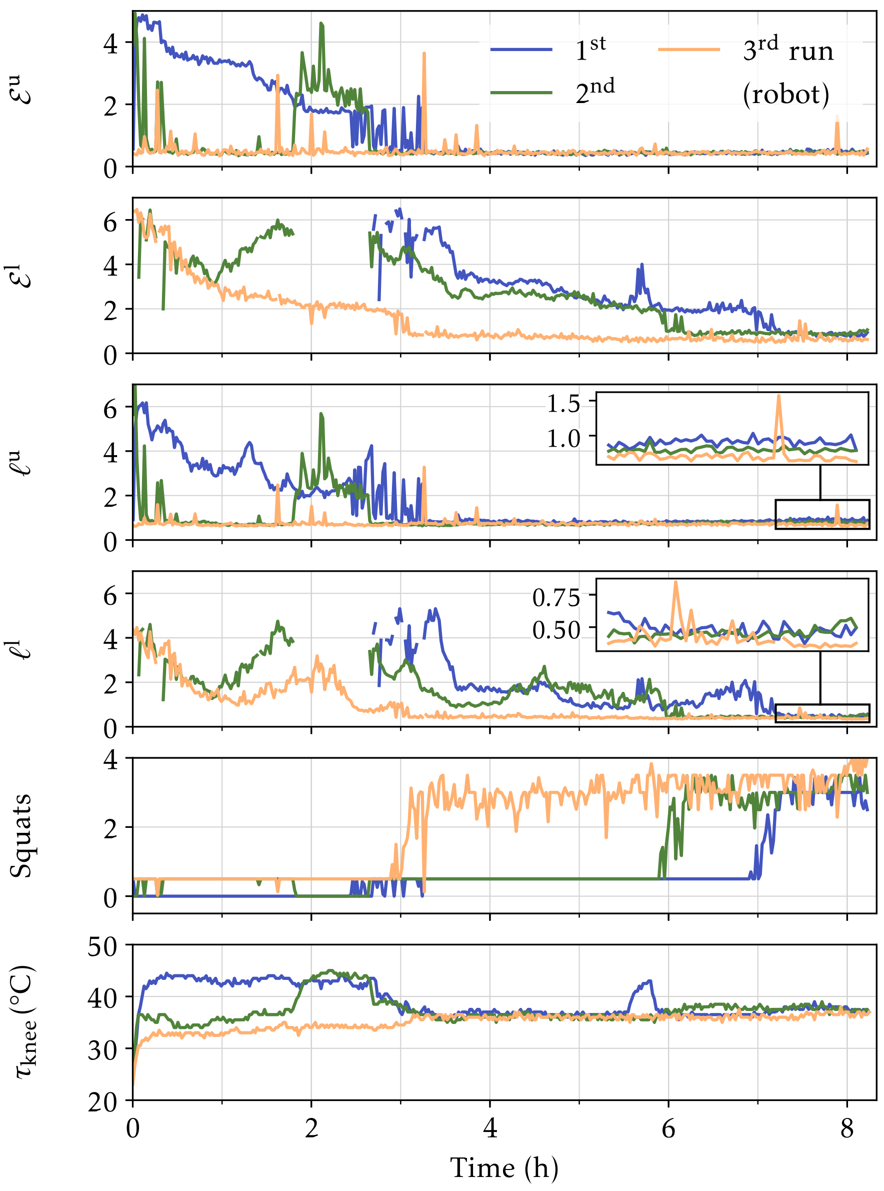


Figure 6.10: Model-plant mismatch (*upper two*), nominal controller objective (*middle two*), number of squats and the mean temperature of knee motors obtained during three real learning experiments from our article [104]. Measurements of $\mathcal{E}^l$ and ℓ^l in second and third plot are not depicted when upper switching point is not reached.

Discussion

Even though the final CAL policy is optimal with respect to the real system, the policy is suboptimal with respect to the objective of the nominal controller (1.12a). This result can be explained by the fact that RL and NMPC objectives are not exactly the same. While NMPC optimizes the undiscounted cost up to horizon T, RL optimizes γ-discounted reward on the infinite horizon. All in all, RL views the system and the nominal controller as a hybrid entity and the obtained policy becomes optimal with respect to the RL objective. This observation also explains the inability of CAL$^{\gamma=0.97}$ to reach the optimal performance on the realistic model, even though it can significantly improve the performance of the

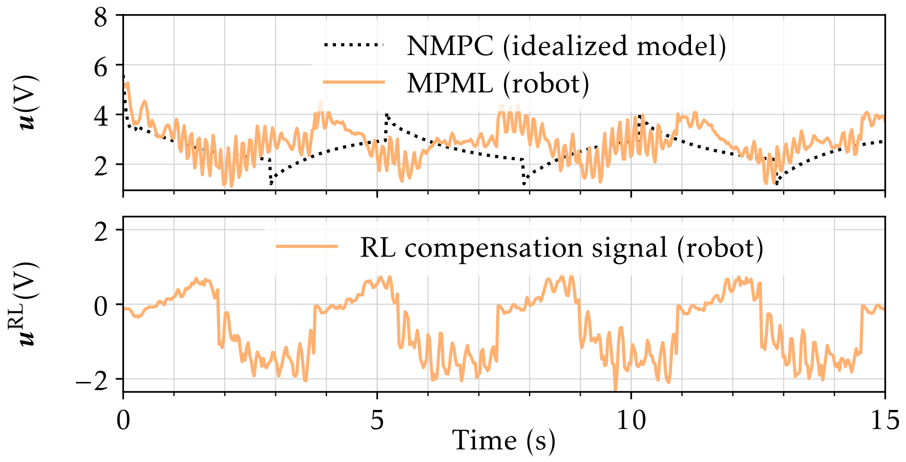


Figure 6.11: Knee control signal of NMPC applied to the idealized model and of MPML applied to the real robot after learning (*top*) from our article [104]. Compensation signal learned by MPML (*bottom*).

nominal controller. The longer prediction horizon used by CAL$^{\gamma=0.99}$ attains a better performance.

Another problem of CAL is the slow convergence which is caused by the fact that the reward constructed from the quadratic objective function of the nominal controller results in small gradients [103]. This hypothesis is supported by the fact that DPG with a quadratic cost function learns the task extremely slowly. To mitigate this, the RL cost function can be modified. The downside of this can be the difficulty of predicting the outcome of such modification, e.g. robot velocity may change drastically.

The MPML approach is free from these complications. However, it should be emphasized that MPML optimizes policy with respect to the internal model, that is RL forces the system to behave like the idealized model. In principle, a large mismatch may pose a problem because the obtained policy will be less optimal with respect to the real system and control constraints may prevent the necessary compensation to be applied. However, in our experiments, this is not a problem. Minimization of the model-plant mismatch $\mathcal{E}$ closely follows minimization of the nominal controller cost ℓ. MPML successfully learns to compensate the unknown *Coulomb* friction as well as its dependency on motor temperature and torque.

The MPML approach obtains the lowest RMSE. This does not come as a surprise, as MPML directly minimizes the mismatch by the specifically constructed reward function. CAL also minimizes RMSE, even though its primary goal is not defined in terms of such minimization. Arguably, in order for NMPC to successfully complete the task, the realistic model should resemble the idealized one which is achieved by learning with RL in both approaches.

For both proposed approaches, a little deviation caused by RL exploration leads to an immediate setback reaction from NMPC. Our simulated experiment reveals a wide range of admissible exploration noise σ for which the number of NMPC constraint violations and robot falls is zero. This result demonstrates the role of NMPC which provides safety barriers to constrain RL exploratory actions near dangerous state space regions. However, there are disadvantages. First, the formulated task demands deliberate control learning which is difficult in the presence of *Coulomb* friction. If the robot starts moving after a slight overshoot caused by RL exploration, this immediately causes the decrease of friction

(*Stribeck* effect), and at the next sampling moment, the system displacement appears to be too large. NMPC counteracts, so that resulting trajectories appear to be oscillatory. The other reason of oscillations is due to the large control delay. Given our results, it is hard to assess the role of NMPC counter-reaction in the oscillatory trajectories, but we expect that reduction of sampling time and control delay will reduce oscillations.

Second, NMPC can drive the ideal system very close to constraint boundaries. While for some systems violation of constraints can be very critical, such that exploration on top can cause damage. However, this was not true in our case.

We note that the success of model-plant mismatch compensation depends on the learning capabilities of RL on the hybrid system mentioned above. Whether there is a decrease or increase of computational complexity of that system against the original system remains an open problem.

It is a common practice to compensate for a steady-state error in a task completion by adding an integral term to the objective that is tuned by experimental data. However, learning the actual model-plant mismatch with MPML goes far beyond cost tuning for a certain task. It allows to predict the outcome of executed actions since the learned trajectory is expected to be optimal with respect to the idealized model.

6.5 Summary

In this chapter, we provided an extensive comparison of model-free RL and model-based NMPC methods. We began with presenting a proper formulation of NMPC and RL problems tackling the same task of a swing-up and balancing motion of a cart-pendulum system. The benchmark is standard and well-known in literature. To facilitate follow-up research, we provide the freely available source code of the benchmark online [22].

We showed that both methods were capable of solving the benchmark problem and that the resulting trajectories for states and controls and quantitatively assessed the the similarity of the results in terms of the coefficient of determination and regret.

In our experiments considering uncertainties, we showed that ideal NMPC with MHE is superior to RL for the whole range of uncertainties, but the realistic NMPC with MHE is comparable to RL. The major achievement is a quantification of a break-even point after which learning in a model-free setting becomes more beneficial than nonlinear model predictive control with an inaccurate model.

Furthermore, we proposed two learning approaches to compensate model-plant mismatch. These approaches realize a combination of methods of NMPC and RL in a hybrid control scheme. Our simulation results demonstrated the feasibility of both approaches. We implemented the superior one on a real robot affected by torque and temperature dependent friction. In the experimental setup the robot autonomously learned a squatting task. Trying to achieve a similar performance with the standalone nominal controller would require tedious identification of the law of such a dependency. The robot has not fallen during learning, and no motor has been damaged.

7 Conclusion and Outlook

In this thesis, we contributed to numerical methods of real-time feasible nonlinear model predictive control (NMPC) for the optimal closed-loop motion generation of humanoids modeled as multi-body systems (MBSs) realizable on today's humanoid robots.

The contributions to the field of rigid-body dynamics (RBD) based on recursive algorithms for MBS of tree topology subject to unilateral kinematic constraints and collisions concerns the efficient evaluation of first-order derivative information. The application of RBD in the context of direct optimal control requires not only the efficient evaluation of the nominal dynamic quantities but also requires the evaluation of their derivatives. Respective optimization methods greatly benefit from accurate gradients such that methods of numerical differentiation are neither sufficient in quality nor computation time despite their straight-forward implementation. In chapter 2, we showed that by applying the principle of algorithmic differentiation (AD) together with mathematical insights into the derivatives of spatial transformations in RBD, recursive algorithms can be augmented to outperform their numerical differentiation counterpart in both quality and computational time, which is thoroughly testing using benchmark examples. We found that our approach leads to faster convergence and smaller iteration counts in the optimal control context.

The contributions to whole-body motion generation and closed-loop control based on NMPC for humanoids consist of four distinct approaches and their realization on the respective robotic hardware. First, we proposed a generic framework for motion generation and closed-loop control combining high-level motion planning using reduced model variants together with online stabilizing control realizing the planned motion. Particularly, we proposed a real-time feasible, nonlinear control strategy to compute center of mass (CoM) trajectories as well as foot positions and orientations realizing a dynamically stable walking motion relying on linear inverted pendulum model dynamics as well as incorporating collision avoidance. We implemented this approach together with a tailored solver that enables real-time performance on the robotic hardware. Key performance indicators derived from benchmarks showed the improvement through our novel approach.

Second, we proposed a novel walking pattern generator formulated as mixed-integer optimization problem achieving collision-free motion planning with guaranteed dynamic feasibility of the planned motions while extending the range of operational scenarios.

Third, we proposed an adapted framework for whole-body motion generation relying on a high-level planner based on centroidal dynamics. In order to achieve real-time feasibility, we derived a further reduction in the form of an ellipsoid model. In this way, we've extended the range of possibilities by enabling multi-contact motions. The approach is implemented on the robotic hardware platform realizing a stair climbing motion.

Finally, we demonstrated real-time feasible NMPC using a whole-body dynamic model by implementing a thread-based controller separating the linearization, i.e., timely evolving the whole model, for the current time step from the fast feedback using the last linearization. In this way, we realized a practical implementation of the approach on the robotic hardware and show that a nominal NMPC controller cannot control the robot.

This thesis also contributed to the overlap of the fields of methods of model-based and model-free optimal control in two ways. First, we proposed a comprehensive performance comparison of model-free (RL) and model-based (NMPC) optimal control on a benchmark example that quantifies the performance of the methods subject to parametric and structural uncertainties in terms of different criteria. We demonstrated that NMPC has advantages over reinforcement learning if uncertainties can be eliminated through identification of the system parameters. Otherwise, there exists a break-even point after which RL performs better than NMPC.

Second, we found that a combination of both RL and NMPC can be beneficial for real systems being subject to uncertainties and therefore proposed two possible combinations to compensate model-plant mismatch. We proved the mathematical properties of the proposed schemes. After numerically benchmarking their performance, the superior scheme was then realized on the robot *Leo* for validation of the approach.

The research conducted in this thesis can be extended in several directions. The major challenge of motion generation using NMPC is the handling of the possible contacts of the system and how to chose/treat them correctly inside the problem formulation. This thesis already presented an approach that considers different feasible surfaces for motion generation in the form of a mixed-integer quadratic program as shown in Chapter 3. While the approach itself is feasible, the problem complexity resulted in an insufficient numerical performance and dictates that future research investigates dedicated solvers for such a problem formulation. Following this, and based on the different variants of model reduction, the goal would be to find a way to combine the high-level strategy based on the mixed-integer formulation, a reduced model variant, e.g. centroidal dynamics, and a suitable motion generation algorithm achieving real-time performance.

Minor challenges lie in improving the established concepts of this thesis in either functionality or further in performance. While, we could achieve the goal of reaching real-time feasible NMPC on the actual robotic hardware platform, the currently treatable models are still far from realizing every aspect of humanoid motions. In order to extend the models to being more realistic, the efficient derivative evaluation of Chapter 2 needs to be extended to support custom and multi-degree-of-freedom joints as well as torque-based muscle approximation models, e.g. from [122]. In the case that the current savings from our efficient algorithmic differentiation (ED) approach are not sufficient, further investigation of the sparsity exploitation or the evaluation of kinematic quantities to become even faster.

In both ways, further research on whole-body motion generation and closed-loop control will broaden the successful applications of the results of this thesis while further closing the gap to full exploitation of existing robot platforms and, in general, moving further towards the higher goal of achieving *human likeness* in locomotion, behavior, and intelligence of humanoid robots.

Bibliography

[1] P. Abbeel, A. Coates, and A. Y. Ng. “Autonomous Helicopter Aerobatics Through Apprenticeship Learning.” In: *International Journal of Robotics Research* 29.13 (2010), pp. 1608–1639.

[2] J. Albersmeyer. “Adjoint based algorithms and numerical methods for sensitivity generation and optimization of large scale dynamic systems.” PhD Thesis. Heidelberg University, 2010.

[3] J. Albersmeyer, D. Beigel, C. Kirches, L. Wirsching, H. G. Bock, and J. P. Schlöder. “Fast Nonlinear Model Predictive Control with an Application in Automotive Engineering.” In: *Nonlinear Model Predictive Control*. Ed. by L. Magni, D. Raimondo, and F. Allgöwer. Vol. 384. Lecture Notes in Control and Information Sciences. Springer Berlin Heidelberg, 2009, pp. 471–480.

[4] H. Amann and J. Escher. *Analysis II*. Grundstudium Mathematik. Birkhäuser Basel, 2008.

[5] U. M. Ascher, H. Chin, L. R. Petzold, and S. Reich. “Stabilization of Constrained Mechanical Systems with DAEs and Invariant Manifolds.” In: *Mechanics of Structures and Machines* 23.2 (1995), pp. 135–157.

[6] H. Audren, J. Vaillant, A. Kheddar, A. Escande, K. Kaneko, and E. Yoshida. “Model preview control in multi-contact motion-application to a humanoid robot.” In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2014, pp. 4030–4035.

[7] A. G. Barto, R. S. Sutton, and C. W. Anderson. “Neuronlike Adaptive Elements That Can Solve Difficult Learning Control Problems.” In: *IEEE Transactions on Systems, Man, and Cybernetics* SMC-13.5 (1983), pp. 834–846.

[8] Y. E. Bayiz and R. Babuska. “Nonlinear disturbance compensation and reference tracking via reinforcement learning with fuzzy approximators.” In: *IFAC Proceedings Volumes* 47.3 (2014), pp. 5393–5398.

[9] P. Beeson and B. Ames. “TRAC-IK: An Open-Source Library for Improved Solving of Generic Inverse Kinematics.” In: *Proceedings of the IEEE-RAS International Conference on Humanoid Robots*. 2015, pp. 928–935.

[10] M. Benzi, G. H. Golub, and J. Liesen. “Numerical solution of saddle point problems.” In: *Acta Numerica* (2005), pp. 1–137.

[11] S. Bhatnagar, R. S. Sutton, M. Ghavamzadeh, and M. Lee. *Natural Actor-Critic Algorithms*. TR09-10. Canada: University of Alberta, 2009.

[12] H. G. Bock. “Numerical treatment of inverse problems in chemical reaction kinetics.” In: *Modelling of Chemical Reaction Systems*. Ed. by K. H. Ebert, P. Deuflhard, and W. Jäger. Vol. 18. Springer Series in Chemical Physics. Heidelberg: Springer, 1981, pp. 102–125.

[13] H. G. Bock and K. J. Plitt. “A Multiple Shooting algorithm for direct solution of optimal control problems.” In: *Proceedings of the IFAC World Congress*. Budapest: Pergamon Press, 1984, pp. 242–247.

[14] H. G. Bock, M. Diehl, E. A. Kostina, and J. P. Schlöder. “Constrained Optimal Feedback Control of Systems Governed by Large Differential Algebraic Equations.” In: *Real-Time PDE-Constrained Optimization*. Ed. by L. Biegler, O. Ghattas, M. Heinkenschloss, D. Keyes, and B. van Bloemen Waanders. SIAM, 2007. Chap. 1, pp. 3–24.

[15] H. G. Bock, M. Diehl, P. Kühl, E. Kostina, J. Schlöder, and L. Wirsching. “Numerical Methods for Efficient and Fast Nonlinear Model Predictive Control.” In: *Assessment and Future Directions of Nonlinear Model Predictive Control*. Ed. by R. Findeisen, F. Allgöwer, and L. Biegler. Vol. 358. Lecture Notes in Control and Information Sciences. 2007, pp. 163–179.

[16] Z. I. Botev, D. P. Kroese, R. Y. Rubinstein, and P. L’Ecuyer. *The Cross-Entropy Method for Optimization*. Vol. 31. Elsevier, 2013, pp. 35–59.

[17] K. Bouyarmane and A. Kheddar. “Humanoid Robot Locomotion and Manipulation Step Planning.” In: *Advanced Robotics* 26.10 (2012), pp. 1099–1126.

[18] S.-P. Boyd and B. Wegbreit. “Fast Computation of Optimal Contact Forces.” In: *IEEE Transactions on Robotics* 23.6 (2007), pp. 1117–1132.

[19] B. Brogliato, A. A. ten Dam, L. Paoli, F. Genot, and M. Abadie. *Numerical simulation of finite dimensional multibody nonsmooth mechanical systems*. Tech. rep. Nationaal Lucht- en Ruimtevaartlaboratorium, 2001.

[20] S. Brossette, J. Vaillant, F. Keith, A. Escande, and A. Kheddar. “Point-cloud multi-contact planning for humanoids: Preliminary results.” In: *Procceedings of the IEEE Conference on Robotics, Automation and Mechatronics*. 2013, pp. 19–24.

[21] W. Caarls and E. Schuitema. “Parallel Online Temporal Difference Learning for Motor Control.” In: *IEEE Transactions on Neural Networks and Learning Systems* 27.7 (2016), pp. 1457–1468.

[22] W. Caarls. *Generic Reinforcement Learning Library*. URL: https://github.com/wcaarls/grl.

[23] J. Carpentier and N. Mansard. “Analytical Derivatives of Rigid Body Dynamics Algorithms.” In: *Proceedings of the International Conference on Robotics Science and Systems*. 2018.

[24] J. Carpentier, S. Tonneau, M. Naveau, O. Stasse, and N. Mansard. “A versatile and efficient pattern generator for generalized legged locomotion.” In: *Proceedings of the IEEE International Conference on Robotics and Automation*. 2016, pp. 3555–3561.

[25] J. Carpentier, F. Valenza, N. Mansard, et al. *Pinocchio: fast forward and inverse dynamics for poly-articulated systems*. 2015–2019. URL: https://stack-of-tasks.github.io/pinocchio.

[26] B. W. Char, K. O. Geddes, W. M. Gentleman, and G. H. Gonnet. “The design of Maple: A compact, portable, and powerful computer algebra system.” In: *Computer Algebra*. Ed. by J. A. van Hulzen. Berlin, Heidelberg: Springer Berlin Heidelberg, 1983, pp. 101–115.

[27] B. Chrétien, A. Escande, and A. Kheddar. “GPU Robot Motion Planning Using Semi-Infinite Nonlinear Programming.” In: *IEEE Transactions on Parallel and Distributed Systems* 27.10 (2016), pp. 2926–2939.

[28] P. Christiano, Z. Shah, I. Mordatch, J. Schneider, T. Blackwell, J. Tobin, P. Abbeel, and W. Zaremba. “Transfer from simulation to real world through learning deep inverse dynamics model.” In: *CoRR* abs/1610.03518 (2016).

[29] F. Clarke. *Functional Analysis, Calculus of Variations and Optimal Control*. Graduate Texts in Mathematics. Springer London, 2013.

[30] *CoDyCo Modules - Whole-body Compliant Dynamical Contacts in Cognitive Humanoids*. URL: https://github.com/robotology/codyco-modules.

[31] H. Dai, A. Valenzuela, and R. Tedrake. “Whole-body motion planning with centroidal dynamics and full kinematics.” In: *Proceedings of the IEEE-RAS International Conference on Humanoid Robots*. IEEE, 2014, pp. 295–302.

[32] A. Datta and L. Xing. “The theory and design of adaptive internal model control schemes.” In: *Proceedings of the American Control Conference*. Vol. 6. 1998, pp. 3677–3684.

[33] R. Deits and R. Tedrake. “Footstep planning on uneven terrain with mixed-integer convex optimization.” In: *Proceedings of the IEEE-RAS International Conference on Humanoid Robots*. 2014, pp. 279–286.

[34] R. Deits and R. Tedrake. “Computing large convex regions of obstacle-free space through semidefinite programming.” In: *Algorithmic Foundations of Robotics XI*. Springer, 2015, pp. 109–124.

[35] M. Diehl. *Real-Time Optimization for Large Scale Nonlinear Processes*. Vol. 920. Fortschritt-Berichte VDI Reihe 8, Meß-, Steuerungs- und Regelungstechnik. 2002.

[36] M. Diehl. “Real-Time Optimization for Large Scale Nonlinear Processes.” PhD Thesis. Heidelberg University, 2001.

[37] M. Diehl, H. G. Bock, and J. P. Schlöder. “A real-time iteration scheme for nonlinear optimization in optimal feedback control.” In: *SIAM Journal on Control and Optimization* 43.5 (2005), pp. 1714–1736.

[38] M. Diehl, H. J. Ferreau, and N. Haverbeke. "Efficient numerical methods for nonlinear MPC and moving horizon estimation." In: *Nonlinear model predictive control*. Springer, 2009, pp. 391–417.

[39] C. Dune, A. Herdt, E. Marchand, O. Stasse, P.-B. Wieber, and E. Yoshida. "Vision based control for Humanoid Robots." In: *Proceedings of the IEEE/RAS International Conference on Intelligent Robot and Systems, Workshop on Visual Control of Mobile Robots (ViCoMor)*. 2011.

[40] C. Dune, A. Herdt, O. Stasse, P.-B. Wieber, and K. Yokoi. "Visual Servoing of Dynamic Walking Motion by Ignoring the Sway Motion." In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2010, pp. 3175–3180.

[41] J.-M. Engel and R. Babuska. "On-line Reinforcement Learning for Nonlinear Motion Control: Quadratic and Non-Quadratic Reward Functions." In: *Proceedings of the IFAC World Congress*. Vol. 19. Cape Town, South Africa, 2014, pp. 7043–7048.

[42] T. Erez, K. Lowrey, Y. Tassa, V. Kumar, S. Kolev, and E. Todorov. "An integrated system for real-time model predictive control of humanoid robots." In: *Proceedings of the IEEE-RAS International Conference on Humanoid Robots*. 2013, pp. 292–299.

[43] D. Ernst, M. Glavic, F. Capitanescu, and L. Wehenkel. "Reinforcement Learning Versus Model Predictive Control: A Comparison on a Power System Problem." In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics* 39.2 (2009), pp. 517–529.

[44] A. Escande, A. Kheddar, and S. Miossec. "Planning contact points for humanoid robots." In: *Robotics and Autonomous Systems* 61.5 (2013), pp. 428–442.

[45] M. F. Fallon, P. Marion, R. Deits, T. Whelan, M. Antone, J. McDonald, and R. Tedrake. "Continuous humanoid locomotion over uneven terrain using stereo fusion." In: *Proceedings of the IEEE-RAS International Conference on Humanoid Robots*. 2015, pp. 881–888.

[46] F. Farshidian, M. Neunert, and J. Buchli. "Learning of closed-loop motion control." In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2014, pp. 1441–1446.

[47] R. Featherstone. "Plucker Basis Vectors." In: *Proceedings of the IEEE International Conference on Robotics and Automation*. 2006, pp. 1892–1897.

[48] R. Featherstone. *Rigid Body Dynamics Algorithms*. New York: Springer, 2008.

[49] M. Felis. *Rigid Body Dynamics Library (RBDL)*. 2012–2019. URL: https://bitbucket.org/MartinFelis/rbdl.

[50] M. L. Felis. "RBDL: an Efficient Rigid-Body Dynamics Library using Recursive Algorithms." In: *Autonomous Robots* 41.2 (2017), pp. 495–511.

[51] H. J. Ferreau. "Model Predictive Control Algorithms for Applications with Millisecond Timescales." PhD Thesis. K.U. Leuven, 2011.

[52] H. J. Ferreau. "qpOASES user's manual." In: *Optimization in Engineering Center (OPTEC) and Department of Electrical Engineering, KU Leuven* (2011).

[53] H. J. Ferreau, C. Kirches, A. Potschka, H. G. Bock, and M. Diehl. "qpOASES: A parametric active-set algorithm for quadratic programming." In: *Mathematical Programming Computation* (2014).

[54] J. F. Fisac, A. K. Akametalu, M. N. Zeilinger, S. Kaynama, J. H. Gillula, and C. J. Tomlin. "A General Safety Framework for Learning-Based Control in Uncertain Robotic Systems." In: *CoRR* abs/1705.01292 (2017). URL: http://arxiv.org/abs/1705.01292.

[55] P. Fitzpatrick, G. Metta, and L. Natale. "Towards Long- Lived Robot Genes." In: *Robotics and Autonomous Systems, Elsevier* 56.1 (2008), pp. 29–45.

[56] J. V. Frasch, L. Wirsching, S. Sager, and H. G. Bock. "Mixed–Level Iteration Schemes for Nonlinear Model Predictive Control." In: *IFAC Proceedings Volumes* 45.17 (2012), pp. 138–144.

[57] G. Garofalo, C. Ott, and A. Albu-Schäffer. "On the closed form computation of the dynamic matrices and their differentiations." In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2013, pp. 2364–2359.

[58] M. Giftthaler, M. Neunert, M. Stäuble, M. Frigerio, C. Semini, and J. Buchli. "Automatic Differentiation of Rigid Body Dynamics for Optimal Control and Estimation." In: *Advanced Robotics* 31.22 (2017), pp. 1225–1237.

[59] P. E. Gill, W. Murray, and M. A. Saunders. *User's guide for* QPOPT 1.0*: a FORTRAN package for quadratic programming*. Tech. rep. SOL 95-4. Department of Operations Research, Stanford University, 1995.

[60] P. E. Gill, W. Murray, M. A. Saunders, and M. H. Wright. *User's Guide for SOL/QPSOL: A Fortran Package for Quadratic Programming*. Tech. rep. Systems Optimization Laboratory, Department of Operations Research, Stanford University, 1982.

[61] A. Griewank and A. Walther. *Evaluating derivatives: principles and techniques of algorithmic differentiation*. Vol. 105. Society for Industrial and Applied Mathematics, 2008.

[62] I. Grondman, M. Vaandrager, L. Busoniu, R. Babuska, and E. Schuitema. "Efficient Model Learning Methods for Actor-Critic Control." In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics* 42.3 (2012), pp. 591–602.

[63] S. Gu, T. Lillicrap, I. Sutskever, and S. Levine. "Continuous deep Q-learning with model-based acceleration." In: *Proceedings of the International Conference on Machine Learning*. 2016, pp. 2829–2838.

[64] B. Guenter. "Efficient Symbolic Differentiation for Graphics Applications." In: *ACM Transactions on Graphics* 26.3 (2007).

[65] Gurobi Optimization, Inc. *Gurobi Optimizer Reference Manual*. 2018. URL: http://www.gurobi.com.

[66] S. Ha and K. Yamane. "Reducing hardware experiments for model learning and policy optimization." In: *Proceedings of the IEEE International Conference on Robotics and Automation*. 2015, pp. 2620–2626.

[67] A. Hans, D. Schneegaß, A. M. Schäfer, and S. Udluft. "Safe exploration for reinforcement learning." In: *European Symposium on Artificial Neural Networks*. 2008, pp. 143–148.

[68] N. Hansen and A. Ostermeier. "Completely derandomized self-adaptation in evolution strategies." In: *Evolutionary Computation* 9.2 (2001), pp. 159–195.

[69] K. Harada, S. Kajita, K. Kaneko, and H. Hirukawa. "An Analytical Method for Real-Time Gait Planning for Humanoid Robots." In: *International Journal of Humanoid Robotics* 3.1 (2006), pp. 1–19.

[70] A. Herdt, H. Diedam, P.-B. Wieber, D. Dimitrov, K. Mombaur, and M. Diehl. "Online Walking Motion Generation with Automatic Foot Step Placement." In: *Special Issue: Section Focused on Cutting Edge of Robotics in Japan 2010* 5–6 (2010), pp. 719–737.

[71] A. Herdt, N. Perrin, and P. Wieber. "Walking without thinking about it." In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2010, pp. 190–195.

[72] H. Herr and M. Popovic. "Angular momentum in human walking." In: *Journal of Experimental Biology* 211.4 (2008), pp. 467–481.

[73] A. Herzog, N. Rotella, S. Schaal, and L. Righetti. "Trajectory generation for multi-contact momentum control." In: *Proceedings of the IEEE-RAS International Conference on Humanoid Robots*. 2015, pp. 874–880.

[74] H. Hirukawa, S. Hattori, S. Kajita, K. Harada, K. Kaneko, F. Kanehiro, M. Morisawa, and S. Nakaoka. "A Pattern Generator of Humanoid Robots Walking on a Rough Terrain." In: *Proceedings of the IEEE International Conference on Robotics and Automation*. 2007, pp. 2181–2187.

[75] H. Hirukawa et al. "Humanoid Robotics Platforms developed in HRP." In: *Robotics and Autonomous Systems* 48 (2003), pp. 165–175.

[76] Y. Hu, J. Eljaik, K. Stein, F. Nori, and K. Mombaur. "Walking of the iCub humanoid robot in different scenarios: implementation and performance analysis." In: *Proceedings of the IEEE-RAS International Conference on Humanoid Robots* (2016), pp. 690–696.

[77] S. Hyon, J. G. Hale, and G. Cheng. "Full-Body Compliant Human-Humanoid Interaction: Balancing in the Presence of Unknown External Forces." In: 23.5 (2007), pp. 884–898.

[78] A. Ibanez, P. Bidaud, and V. Padois. "Automatic Optimal Biped Walking as a Mixed-Integer Quadratic Program." In: *Advances in Robot Kinematics*. Springer, 2014, pp. 505–516.

[79] D. Janka, C. Kirches, S. Sager, and A. Wächter. "An SR1/BFGS SQP algorithm for nonconvex nonlinear programs with block-diagonal Hessian matrix." In: *Mathematical Programming Computation* 8.4 (2016), pp. 435–459.

[80] N. Jiang, A. Kulesza, S. Singh, and R. Lewis. "The Dependence of Effective Planning Horizon on Model Accuracy." In: *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems*. Istanbul, Turkey: International Foundation for Autonomous Agents and Multiagent Systems, 2015, pp. 1181–1189.

[81] L. P. Kaelbling, M. L. Littman, and A. W. Moore. "Reinforcement Learning: A Survey." In: *Journal of Artificial Intelligence Research* 4.1 (1996), pp. 237–285.

[82] P. Kaiser, D. Kanoulas, M. Grotz, L. Muratore, A. Rocchi, E. M. Hoffman, N. G. Tsagarakis, and T. Asfour. "An affordance-based pilot interface for high-level control of humanoid robots in supervised autonomy." In: *Proceedings of the IEEE-RAS International Conference on Humanoid Robots*. 2016, pp. 621–628.

[83] S. Kajita, F. Kanehiro, K. Kaneko, K. Fujiwara, K. Harada, K. Yokoi, and H. Hirukawa. "Biped Walking pattern Generation by using Preview Control of Zero-Moment Point." In: *Proceedings of the IEEE International Conference on Robotics and Automation*. Vol. 2. 2003, pp. 1620–1626.

[84] S. Kajita, F. Kanehiro, K. Kaneko, K. Yokoi, and H. Hirukawa. "The 3D linear inverted pendulum mode: a simple modeling for a biped walking pattern generation." In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*. Vol. 1. 2001, pp. 239–246.

[85] S. Kajita, K. Yokoi, M. Saigo, and K. Tanie. "Balancing a Humanoid Robot Using Backdrive Concerned Torque Control and Direct Angular Momentum Feedback." In: *Proceedings of the IEEE International Conference on Robotics and Automation*. Vol. 4. 2001, pp. 3376–3382.

[86] S. Kajita, H. Hirukawa, K. Harada, and K. Yokoi. *Introduction to Humanoid Robotics*. Ed. by B. Siciliano and O. Khatib. Vol. 101. Springer Tracts in Advanced Robotics. Springer, 2014.

[87] S. M. Kakade. "A Natural Policy Gradient." In: *Advances in Neural Information Processing Systems*. Ed. by T. G. Dietterich, S. Becker, and Z. Ghahramani. MIT Press, 2002, pp. 1531–1538.

[88] S. Kamthe and M. P. Deisenroth. "Data-Efficient Reinforcement Learning with Probabilistic Model Predictive Control." In: *CoRR* abs/1706.06491 (2017). URL: http://arxiv.org/abs/1706.06491.

[89] K. Kaneko, F. Kanehiro, S. Kajita, H. Hirukawa, T. Kawasaki, M. Hirata, K. Akachi, and T. Isozumi. "Humanoid robot HRP-2." In: *Proceedings of the IEEE International Conference on Robotics and Automation*. Vol. 2. 2004, pp. 1083–1090.

[90] M. Karklinsky, M. Naveau, A. Mukovskiy, O. Stasse, T. Flash, and P. Soueres. "Robust human-inspired power law trajectories for humanoid HRP-2 robot." In: *Proceedings of the IEEE International Conference on Biomedical Robotics and Biomechatronics*. 2016, pp. 106–113.

[91] M. Kawato. "Feedback-error-learning neural network for supervised motor learning." In: *Advanced neural computers* 6.3 (1990), pp. 365–372.

[92] M. Khadiv, A. Herzog, S. A. A. Moosavian, and L. Righetti. "A robust walking controller based on online step location and duration optimization for bipedal locomotion." In: *CoRR* abs/1704.01271 (2017). URL: http://arxiv.org/abs/1704.01271.

[93] W. Khalil and E. Dombre. *Modeling, Identification & Control of Robots*. Kogan Page Sience, 2002.

[94] J. Kim. *Lie Group Formulation of Articulated Rigid Body Dynamics*. Tech. rep. 2012.

[95] H. Kimura and S. Kobayashi. "Stochastic real-valued reinforcement learning to solve a nonlinear control problem." In: *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*. Vol. 5. 1999, pp. 510–515.

[96] C. Kirches, L. Wirsching, H. G. Bock, and J. Schlöder. "Efficient Direct Multiple Shooting for Nonlinear Model Predictive Control on Long Horizons." In: *Journal of Process Control* 22.3 (2012), pp. 540–550.

[97] J. Kober, J. A. Bagnell, and J. Peters. "Reinforcement Learning in Robotics: A Survey." In: *International Journal of Robotics Research* 32.11 (2013), pp. 1238–1274.

[98] J. Kober and J. Peters. "Policy search for motor primitives in robotics." In: *Machine Learning* 84.1-2 (2011), pp. 171–203.

[99] K. H. Koch. “Using Model-based Optimal Control for Conceptional Motion Generation for the Humanoid Robot HRP-2 14 and Design Investigations for Exo-Skeletons.” PhD Thesis. Heidelberg University, 2015.

[100] K. H. Koch, K. Mombaur, P. Souères, and O. Stasse. “Optimization based exploitation of the ankle elasticity of HRP-2 for overstepping large obstacles.” In: *Procceedings of the IEEE/RAS International Conference on Humanoid Robots*. 2014, pp. 733–740.

[101] 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: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2015, pp. 3346–3351.

[102] *KoroiBot - Improving humanoid walking capabilities by human-inspired mathematical models, optimization and learning*. URL: http://orb.iwr.uni-heidelberg.de/koroibot/.

[103] I. Koryakovskiy, M. Kudruss, R. Babuška, W. Caarls, C. Kirches, K. Mombaur, J. P. Schlöder, and H. Vallery. “Benchmarking model-free and model-based optimal control.” In: *Robotics and Autonomous Systems* 92 (2017), pp. 81–90.

[104] I. Koryakovskiy, M. Kudruss, H. Vallery, R. Babuška, and W. Caarls. “Model-plant Mismatch Compensation Using Reinforcement Learning.” In: *IEEE Robotics and Automation Letters* 3.3 (2018), pp. 2471–2477.

[105] M. Kudruss, P. Manns, and C. Kirches. “Efficient Derivative Evaluation for Rigid-Body Dynamics Based on Recursive Algorithms Subject to Kinematic and Loop Constraints.” In: *IEEE Control Systems Letters* 3 (2019), pp. 619–624.

[106] M. Kudruss, M. Naveau, O. Stasse, N. Mansard, C. Kirches, P. Souères, and K. Mombaur. “Optimal Control for Whole-body Motion Generation using Center-of-mass Dynamics for Predefined Multi-contact Configurations.” In: *Proceedings of IEEE/RAS International Conference on Humanoid Robots*. 2015, pp. 684–689.

[107] P. Kühl, M. Diehl, T. Kraus, J. P. Schlöder, and H. G. Bock. “A real-time algorithm for moving horizon state and parameter estimation.” In: *Computers & Chemical Engineering* 35.1 (2011), pp. 71–83.

[108] S. Kuindersma, R. Deits, M. Fallon, A. Valenzuela, H. Dai, F. Permenter, T. Koolen, P. Marion, and R. Tedrake. “Optimization-based locomotion planning, estimation, and control design for the Atlas humanoid robot.” In: *Autonomous Robots* 40.3 (2016), pp. 429–455.

[109] D. B. Leineweber, I. Bauer, H. G. Bock, and J. P. Schlöder. “An efficient multiple shooting based reduced SQP strategy for large-scale dynamic process optimization. Part 1: Theoretical aspects.” In: *Computers & Chemical Engineering* 27.2 (2003), pp. 157–166.

[110] D. B. Leineweber, A. Schäfer, H. G. Bock, and J. P. Schlöder. “An efficient multiple shooting based reduced SQP strategy for large-scale dynamic process optimization: Part II: Software aspects and applications.” In: *Computers & chemical engineering* 27.2 (2003), pp. 167–174.

[111] S. Lengagne, J. Vaillant, E. Yoshida, and A. Kheddar. “Generation of whole-body optimal dynamic multi-contact motions.” In: *The International Journal of Robotics Research* 32.9-10 (2013), pp. 1104–1119.

[112] S. Levine and V. Koltun. “Guided Policy Search.” In: *Proceedings of the International Conference on Machine Learning*. 2013, pp. 1–9.

[113] M. Liu, Y. Tan, and V. Padois. “Generalized hierarchical control.” In: *Autonomous Robots* 40.1 (2016), pp. 17–31.

[114] J. Y. Luh, M. W. Walker, and R. P. Paul. “On-line computational scheme for mechanical manipulators.” In: *Journal of Dynamic Systems, Measurement, and Control* 102.2 (1980), pp. 69–76.

[115] P. Manns and K. Mombaur. “Towards Discrete Mechanics and Optimal Control for Complex Models.” In: *IFAC-PapersOnLine* 50.1 (2017), pp. 4812–4818.

[116] N. Mansard. “A Dedicated Solver for Fast Operational-Space Inverse Dynamics.” In: *Proceedings of the IEEE International Conference on Robotics and Automation*. 2012, pp. 4943–4949.

[117] N. Mansard, O. Stasse, P. Evrard, and A. Kheddar. “A Versatile Generalized Inverted Kinematics Implementation for Collaborative Working Humanoid Robots: The Stack of Tasks.” In: *Proceedings of the International Conference on Advanced Robotics*. 2009, pp. 1–6.

[118] L. Matignon, G. Laurent, and N. Le Fort-Piat. “Reward Function and Initial Values: Better Choices for Accelerated Goal-Directed Reinforcement Learning.” In: *Artificial Neural Networks – ICANN 2006*. Ed. by S. Kollias, A. Stafylopatis, W. Duch, and E. Oja. Vol. 4131. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2006, pp. 840–849.

[119] J. Mattingley and S. Boyd. “CVXGEN: A Code Generator for Embedded Convex Optimization.” In: *Optimization and Engineering* 12.1 (2012), pp. 1–27.

[120] G. Metta, L. Natale, F. Nori, et al. “The iCub humanoid robot: An open-systems platform for research in cognitive development.” In: *Neural Networks* 23.8 (2010), pp. 1125–1134.

[121] R. Michel. “Dynamic filter for walking motion corrections.” Bachelor Thesis. Interdisciplinary Center for Scientific Computing, Department of Physics and Astronomy, Heidelberg University, 2017.

[122] M. Millard, A. L. Emonds, M. Harant, et al. “A reduced muscle model and planar musculoskeletal model fit for the simulation of whole-body movements.” In: *Journal of Biomechanics* 89 (2019), pp. 11–20.

[123] T. M. Moldovan and P. Abbeel. “Safe Exploration in Markov Decision Processes.” In: *CoRR* abs/1205.4810 (2012).

[124] K. Mombaur, A. Kheddar, K. Harada, T. Buschmann, and C. Atkeson. “Model-based Optimization for Robotics [TC Spotlight].” In: *IEEE Robotics Automation Magazine* 21.3 (2014), pp. 24–161.

[125] K. Mombaur. “Stability Optimization of Open-loop Controlled Walking Robots.” PhD Thesis. Heidelberg University, 2001.

[126] K. Mombaur, A. Truong, and J.-P. Laumond. “From human to humanoid locomotion—an inverse optimal control approach.” In: *Autonomous Robots* 28.3 (2010), pp. 369–383.

[127] I. Mordatch, K. Lowrey, and E. Todorov. “Ensemble-CIO: Full-body dynamic motion planning that transfers to physical humanoids.” In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2015, pp. 5307–5314.

[128] I. Mordatch, E. Todorov, and Z. Popović. “Discovery of complex behaviors through contact-invariant optimization.” In: *ACM Transactions on Graphics* 31.4 (2012), p. 43.

[129] M. Morisawa, K. Harada, S. Kajita, S. Nakaoka, K. Fujiwara, F. Kanehiro, K. Kaneko, and H. Hirukawa. “Experimentation of Humanoid Walking Allowing Immediate Modification of Foot Place Based on Analytical Solution.” In: *Proceedings of the IEEE International Conference on Robotics and Automation*. 2007, pp. 3989–3994.

[130] A. Mukovskiy, C. Vassallo, M. Naveau, O. Stasse, P. Soueres, and M. A. Giese. “Adaptive synthesis of dynamically feasible full-body movements for the humanoid robot HRP-2 by flexible combination of learned dynamic movement primitives.” In: *Robotics and Autonomous Systems* 91 (2017), pp. 270–283.

[131] W. Nachtigall. *Bionik: Grundlagen und Beispiele für Ingenieure und Naturwissenschaftler (German Edition)*. Springer, 2002.

[132] S. Nakaoka, S. Hattori, F. Kanehiro, S. Kajita, and H. Hirukawa. “Constraint-based dynamics simulator for humanoid robots with shock absorbing mechanisms.” In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2007, pp. 3641–3647.

[133] M. Naveau, M. Kudruss, O. Stasse, C. Kirches, K. Mombaur, and P. Souères. “A Reactive Walking Pattern Generator Based on Nonlinear Model Predictive Control.” In: *IEEE Robotics and Automation Letters* 2.1 (2017), pp. 10–27.

[134] M. Neunert, M. Giftthaler, M. Frigerio, C. Semini, and J. Buchli. “Fast derivatives of rigid body dynamics for control, optimization and estimation.” In: *Proceedings of the IEEE International Conference on Simulation, Modeling, and Programming for Autonomous Robots*. 2016, pp. 91–97.

[135] A. Y. Ng, D. Harada, and S. J. Russell. “Policy Invariance Under Reward Transformations: Theory and Application to Reward Shaping.” In: *Proceedings of the International Conference*

on Machine Learning. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1999, pp. 278–287.

[136] K. Nishiwaki and S. Kagami. “Walking control on uneven terrain with short cycle pattern generation.” In: *Proceedings of the IEEE-RAS International Conference on Humanoid Robots*. 2007, pp. 447–453.

[137] K. Nishiwaki, J. E. Chestnutt, and S. Kagami. “Autonomous navigation of a humanoid robot over unknown rough terrain using a laser range sensor.” In: *International Journal of Robotics Research* 31.11 (2012), pp. 1251–1262.

[138] J. Nocedal and S. Wright. *Numerical Optimization*. Second. Berlin Heidelberg New York: Springer Verlag, 2006.

[139] D. E. Orin and A. Goswami. “Centroidal Momentum Matrix of a humanoid robot: Structure and properties.” In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2008, pp. 653–659.

[140] D. E. Orin, A. Goswami, and S.-H. Lee. “Centroidal dynamics of a humanoid robot.” In: *Autonomous Robots* 35.2-3 (2013), pp. 161–176.

[141] C. Ott, M. A. Roa, and G. Herzinger. “Posture and Balance Control for Biped Robots based on Contact Force Optimization.” In: *Proceedings of the IEEE-RAS International Conference on Humanoid Robotics*. 2011, pp. 26–33.

[142] J. Perry. *Gait Analysis: Normal and Pathological Function*. SLACK, 1992.

[143] J. Peters, K. Mülling, and Y. Altun. “Relative entropy policy search.” In: *Proceedings of the AAAI Conference on Artificial Intelligence*. 2010, pp. 1607–1612.

[144] B. Ponton, A. Herzog, S. Schaal, and L. Righetti. “A convex model of humanoid momentum dynamics for multi-contact motion generation.” In: *Proceedings of the IEEE-RAS International Conference on Humanoid Robots*. 2016, pp. 842–849.

[145] B. Ponton, A. Herzog, S. Schaal, and L. Righetti. “On Time Optimisation of Centroidal Momentum Dynamics.” In: *CoRR* abs/1709.09265 (2017). URL: http://arxiv.org/abs/1709.09265.

[146] J. Pratt, J. Carff, S. Drakunov, and A. Goswami. “Capture Point: A Step toward Humanoid Push Recovery.” In: *Proceedings of the IEEE-RAS International Conference on Humanoid Robots*. 2006, pp. 200–207.

[147] S. J. Qin and T. A. Badgwell. “A survey of industrial model predictive control technology.” In: *Control Engineering Practice* 11.7 (2003), pp. 733–764.

[148] A. Rajeswaran, S. Ghotra, S. Levine, and B. Ravindran. “EPOpt: Learning Robust Neural Network Policies Using Model Ensembles.” In: *CoRR* abs/1610.01283 (2016). URL: http://arxiv.org/abs/1610.01283.

[149] I. G. Ramirez-Alpizar, M. Naveau, C. Benazeth, O. Stasse, J.-P. Laumond, K. Harada, and E. Yoshida. “Motion generation for pulling a fire hose by a humanoid robot.” In: *Proceedings of the IEEE-RAS International Conference on Humanoid Robots*. 2016, pp. 1016–1021.

[150] O. E. Ramos, L. Saab, S. Hak, and N. Mansard. “Dynamic motion capture and edition using a stack of tasks.” In: *Proceedings of the IEEE-RAS International Conference on Humanoid Robots*. 2011, pp. 224–230.

[151] L. Saab, N. Mansard, F. Keith, J. Y. Fourquet, and P. Soueres. “Generation of dynamic motion for anthropomorphic systems under prioritized equality and inequality constraints.” In: *Proceedings of the IEEE International Conference on Robotics and Automation*. 2011, pp. 1091–1096.

[152] L. Saab, O. Ramos, N. Mansard, P. Souères, and J. Y. Fourquet. “Generic dynamic motion generation with multiple unilateral constraints.” In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2011, pp. 4127–4133.

[153] M. Saveriano, Y. Yin, P. Falco, and D. Lee. “Data-efficient control policy search using residual dynamics learning.” In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2017, pp. 4709–4715.

[154] L. Schork. “A parametric active set method for general quadratic programming.” Master Thesis. Heidelberg University, 2015.

[155] E. Schuitema. "Reinforcement Learning on autonomous humanoid robots." PhD Thesis. Netherlands: Delft University of Technology, 2012.

[156] J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel. "Trust Region Policy Optimization." In: *CoRR* abs/1502.05477 (2015).

[157] J. Schulman, P. Moritz, S. Levine, M. I. Jordan, and P. Abbeel. "High-Dimensional Continuous Control Using Generalized Advantage Estimation." In: *CoRR* abs/1506.02438 (2015). URL: http://arxiv.org/abs/1506.02438.

[158] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. "Proximal Policy Optimization Algorithms." In: *CoRR* abs/1707.06347 (2017). URL: http://arxiv.org/abs/1707.06347.

[159] S. Shalev-Shwartz. "Online Learning and Online Convex Optimization." In: *Foundations and Trends in Machine Learning* 4.2 (2012), pp. 107–194.

[160] B. Siciliano and O. Khatib, eds. *Springer Handbook of Robotics*. Berlin, Heidelberg: Springer, 2008.

[161] W. D. Smart and L. P. Kaelbling. "Practical Reinforcement Learning in Continuous Spaces." In: *Proceedings of the International Conference on Machine Learning*. ICML '00. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2000, pp. 903–910.

[162] O. Stasse. "Vision based motion generation for humanoid robots." Habilitation à diriger des recherches. Université Paul Sabatier - Toulouse III, 2013.

[163] K. Stein, Y. Hu, M. Kudruss, M. Naveau, and K. Mombaur. "Closed loop control of walking motions with adaptive choice of directions for the iCub humanoid robot." In: *Proceedings of the IEEE/RAS International Conference on Humanoid Robots*. Birmingham, 2017.

[164] T. Sugihara. "Solvability-unconcerned Inverse Kinematics based on Levenberg-Marquardt method with Robust Damping." In: *Proceedings of the IEEE-RAS International Conference on Humanoid Robots*. Vol. 27. 5. 2009, pp. 984–991.

[165] R. S. Sutton, A. G. Barto, and R. J. Williams. "Reinforcement learning is direct adaptive optimal control." In: *Control Systems, IEEE* 12.2 (1992), pp. 19–22.

[166] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[167] Y. Tassa, T. Erez, and E. Todorov. "Synthesis and stabilization of complex behaviors through online trajectory optimization." In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 4906–4913.

[168] Y. Tassa, N. Mansard, and E. Todorov. "Control-limited differential dynamic programming." In: *Proceedings of the IEEE International Conference on Robotics and Automation*. 2014, pp. 1168–1175.

[169] R. Tedrake and the Drake Development Team. *Drake: A planning, control, and analysis toolbox for nonlinear dynamical systems*. 2016. URL: http://drake.mit.edu.

[170] *The MOSEK optimization software*. http://www.mosek.com/. URL: http://www.mosek.com/.

[171] E. Todorov, T. Erez, and Y. Tassa. "MuJoCo: A physics engine for model-based control." In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 5026–5033.

[172] S. Tonneau, N. Mansard, C. Park, D. Manocha, F. Multon, and J. Pettré. "A Reachability-Based Planner for Sequences of Acyclic Contacts in Cluttered Environments." In: *Robotics Research: Volume 2*. Ed. by A. Bicchi and W. Burgard. Cham: Springer International Publishing, 2018, pp. 287–303.

[173] T. K. Uchida, M. A. Sherman, and S. L. Delp. "Making a meaningful impact: modelling simultaneous frictional collisions in spatial multibody systems." In: *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences* 471.2177 (2015).

[174] E. Vouga, B. Smith, D. M. Kaufman, R. Tamstorf, and E. Grinspun. "All's Well That Ends Well: Guaranteed Resolution of Simultaneous Rigid Body Impact." In: *ACM Transactions on Graphics* 36.4 (2017), 151:1–151:19.

[175] M. Vukobratović and D. Juricic. "Contribution to the Synthesis of Biped Gait." In: *IEEE Transactions on Biomedical Engineering* BME-16.1 (1969), pp. 1–6.

[176] M. Vukobratovic and J. Stephanenko. "On the stability of anthropomorphic systems." In: *Mathematical Biosciences* 15.1 (1972), pp. 1–37.

[177] M. Vukobratović and B. Borovac. “Zero-Moment Point - Thirty Five Years of its Life.” In: *Proceedings of the International Journal of Humanoid Robotics* 1.1 (2005), pp. 157–173.

[178] A. Wächter and L. T. Biegler. “On the Implementation of a Primal-Dual Interior Point Filter Line Search Algorithm for Large-Scale Nonlinear Programming.” In: *Mathematical Programming* 106.1 (2006), pp. 25–27.

[179] I. D. Walker. “The use of kinematic redundancy in reducing impact and contact effects in manipulation.” In: *Proceedings of the IEEE International Conference on Robotics and Automation*. Vol. 1. 1990, pp. 434–439.

[180] S. Walter. “Structured higher-order algorithmic differentiation in the forward and reverse mode with application in optimum experimental design.” PhD Thesis. Humboldt-Universität zu Berlin, Mathematisch-Naturwissenschaftliche Fakultät II, 2012.

[181] P.-B. Wieber. “On the stability of walking systems.” In: *Proceedings of the International Workshop on Humanoid and Human Friendly Robotics*. 2002.

[182] D. L. Wight, E. G. Kubica, and D. W. Wang. “Introduction of the Foot Placement Estimator: A Dynamic Measure of Balance for Bipedal Robotics.” In: *Journal of Computational and Nonlinear Dynamics* 3.1 (2007), pp. 011009–011009–10.

[183] M. Wisse. “Essentials of dynamic walking: Analysis and design of two-legged robots.” PhD Thesis. Netherlands: Delft University of Technology, 2004.

[184] J. Wojtusch, J. Kunz, and O. v. Stryk. “MBSlib – An Efficient Multibody Systems Library for Kinematics and Dynamics Simulation, Optimization and Sensitivity Analysis.” In: *IEEE Robotics and Automation Letters* 1.2 (2016), pp. 954–960.

[185] K. Yokoi, F. Kanehiro, K. Kaneko, K. Fujiwara, S. Kajita, and H. Hirukawa. “A Honda Humanoid Robot Controlled by AIST Software.” In: *Proceedings of the IEEE-RAS International Conference on Humanoid Robots*. 2001, pp. 259–264.

[186] T. Zhang, G. Kahn, S. Levine, and P. Abbeel. “Learning Deep Control Policies for Autonomous Aerial Vehicles with MPC-Guided Policy Search.” In: *CoRR* abs/1509.06791 (2015).

List of Figures

List of Tables

List of Acronyms

ABA Articulated Body Algorithm
AD algorithmic differentiation
CNRS-LAAS *Laboratory for Analysis and Architecture of Systems*
CoM center of mass
CoP center of pressure
CRBA composite rigid-body algorithm
DAE differential algebraic equation
DDP differential dynamic programming
DF dynamic filter
DoF degree of freedom
DPG deterministic policy gradient
ED efficient algorithmic differentiation
FD forward dynamics
FDCONTACTS Forward Dynamics Constraints Direct
GIK generalized inverse kinematics
ID inverse dynamics
IMC internal model control
IVP initial value problem
KPI key performance indicator
Leo Leo
LIPM linear inverted pendulum model
LMPC linear model predictive control
LQR linear-quadratic regulator
MBS multi-body system
MDP *Markov* decision process
MHE moving horizon estimation
MIQP mixed-integer quadratic program
MLRTI multi-level real-time iteration
MPML model-plant mismatch learning
FD numerical differentiation
NEFFECTS Nonlinear Effects based on recursive *Newton-Euler* algorithm (RNEA)
NLP nonlinear program
NMPC nonlinear model predictive control
OC optimal control
OCP optimal control problem
ODE ordinary differential equation
OFA obstacle-free area
PD proportional-derivative
PID proportional-integral-derivative
QP quadratic program
RBD rigid-body dynamics

RBDL	*Rigid Body Dynamics Library*
RL	reinforcement learning
RMSE	root mean squared error
RNEA	recursive *Newton-Euler* algorithm
SoT	stack of tasks
SQP	sequential quadratic programming
TU Delft	*Delft University of Technology*
UHEI	*Heidelberg University*
WPG	walking pattern generator
ZMP	zero-moment point