Technische Universität München
Fakultät für Elektrotechnik und Informationstechnik

Institute for Cognitive Systems (ICS)

Combining reduced dynamics models and whole-body control for agile humanoid locomotion

Dipl.-Ing. (Univ.) Johannes Adalbert Englsberger

Vollständiger Abdruck der von der Fakultät für Elektrotechnik und Informationstechnik der Technischen Universität München zur Erlangung des akademischen Grades eines Doktor-Ingenieurs (Dr.-Ing.) genehmigten Dissertation.

Vorsitzender: Univ.-Prof. Dr.-Ing. Klaus Diepold

Prüfer der Dissertation:

1. TUM Junior Fellow Dr.-Ing. Christian Ott
2. Univ.-Prof. Dr.-Ing. habil. Dirk Wollherr
3. Prof. Russell L. Tedrake, Ph.D., Massachusetts Institute of Technology (MIT)

Diese Dissertation wurde am 29.06.2016 bei der Technischen Universität München eingereicht und durch die Fakultät für Elektrotechnik und Informationstechnik am 27.10.2016 angenommen.

Preface

This dissertation reports the current state of my research, which I had - and still have - the chance and honor to pursue at the Institute for Robotics and Mechatronics of the German Aerospace Center (DLR) since April 2010. The work is embedded into the activities of the Helmholtz Young Investigator's Research Group for „Dynamic Control of Legged Humanoid Robots" (Grant no. VH-NG-808), lead by my supervisor and mentor Dr. Christian Ott, which is mainly centered around the humanoid robot Toro. I was involved in both the mechanical design of Toro and its control. This thesis focuses purely on the controls part of my work.

The contributions made in this work would not have been possible without the support and help of a great number of people, to who I want to express my thankfulness in the following:

First of all, I want to thank my supervisor Dr. Christian Ott. His personal support and scientific input were of the greatest value and shaped my personality and scientific career. Aside from his guidance, I also appreciate the great freedom he gave me to follow my own ideas and interests. Next, I want to thank Prof. Gerd Hirzinger, who sparked my interest in robotics during his lecture at Technical University of Munich (TUM). He and the current head of institute, Prof. Alin Albu-Schäffer, both contributed greatly to the atmosphere and spirit, but also the unique infrastructure, which make DLR's institute for Robotics and Mechatronics a most exciting place to work at. I want to cordially thank my team members Alexander Werner, Gianluca Garofalo, Bernd Henze and Dr. Maximo Roa. Their continuous support and the fruitful discussions greatly contributed to this work. A special thanks goes to my colleague and office mate Gianluca Garofalo, who developed the dynamics library which the whole-body control presented in this thesis is based on. The mechanical design of Toro was supported by my colleagues Erich Krämer and Jens Reinecke and my students Vikram Bangalore Nagendra and Arunkumar Rathinam. Great thanks also the my colleagues from the mechanical and electrical workshops who helped manufacture Toro and are involved in its maintenance. Special thanks to Robert Burger, Alexander Beyer and Martin Heumos for their efforts in software and electrical maintenance.

Next I want to thank Dr. Debora Clever, Dr. Moritz Maus and Dr. Christian Rode, who greatly contributed to this work with fruitful discussions and scientific data. My special thanks goes to Dr. Patrick Wensing, Dr. Michael Hopkins, Dr. Sylvain "Sylvester" Bertrand and Dr. Jerry Pratt, who I was in closest scientific contact with and who became my friends over the years. My three months research stay in 2013 at the robotics lab of Florida Institute for Human and Machine Cognition (IHMC), lead by Dr. Jerry Pratt, and the participation in the DARPA virtual robotics challenge (VRC) were a great inspiration for my work. I want to cordially thank team IHMC for inviting me to the DRC Trials and Finals and for the ongoing great exchange.

Last but not least, I want to thank my wife Anne for her patience, love and support.

Weßling, June 2016 Johannes Englsberger

Abstract

In the near future, mobile machines are expected to leave the labs and perform more and more complex tasks in environments designed for human beings. Systems based on legged and especially bipedal locomotion have the potential of outperforming wheel-based locomotion systems in certain scenarios. Yet, at the current point in time, it still seems to be a long way to go until legged systems can unfold their full potential.

Bipedal locomotion is a difficult controls problem due to issues such as underactuation, unilateral constraints and the hybrid nature of stepping. This work aims at first reducing the complexity of the considered problem as much as possible, while tackling the full problem in a second step. More explicitly: first, only the center of mass (CoM) dynamics of the robot is considered, which is a reduced but correct representation of its motion. Using this reduced model, analytical controllers are designed that include multi-step preview for both walking and running. The walking control method is based on the concept of Divergent Component of Motion (DCM), which extends the earlier concept of Capture Point to 3D. The proposed running control framework is called Biologically Inspired Dead-beat (BID) control. It is based on the encoding of CoM motions as polynomials during stance and explicitly solves for intuitively designed running boundary conditions. Both methods are powerful, purely analytical and very insightful.

In a second step, the locomotion controllers are embedded into a quadratic programming (QP) based whole-body control framework. The latter allows for instantaneous optimal control, which ensures good trade-offs between the different necessary tasks at hand.

Both control frameworks are tested extensively in simulation. Real experiments are successfully performed for DCM-based walking control and the whole-body control framework.

Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

1.1 Problem statement

Currently, we experience an increasing amount of automation of our everyday life. The digital revolution in form of computers, internet and smart phones is in full swing. While the central element of that mentioned revolution is flow and processing of information, it can be expected that this focus will widen towards automatic real-world operations, which involves an increased demand for mobile machines. These machines should be able to automatically perform tasks in our human environments. In some scenarios, unmanned aerial vehicles (UAV) or wheeled robots may perform sufficiently. In other cases, it can be advantageous if the machine has human-like proportions and capabilities, such that it can more naturally move through and make use of the human environment. Biological forms of locomotion - such as human walking and running - have evolved over millions of years. They are the product of relentless selection and can thus to some extent be regarded as optimal for traversing natural environments. The analysis and decoding of natural locomotion poses a complex yet exciting field of research for biomechanics researchers. Their results can serve as inspiration for roboticists. From an engineering point of view, gaited forms of locomotion - once fully understood - promise highly increased mobility of machines as compared to wheel-based locomotion. Overcoming a set of stepping stones, as shown in Fig. 6.1, is one possible example where a legged robot may have advantages over other machines of similar size and weight. Once a certain mobility level has been reached, humanoid robots may serve humans in many different scenarios including emergency relief, service robotics or space operations, to name but a few. The DARPA Robotics Challenge (DRC) [3, 4] provided an impressive insight in the current state of the art in robotics research. In just a few generations, humanoid robots may reach technology readiness levels that will allow real-world applications outside of controlled laboratory environments. One of the main challenges that is not yet solved satisfactorily is to find control algorithms that enable humanoid robots to locomote in a versatile, robust and agile way. From a controls perspective, bipedal locomotion poses several challenges such as underactuation, unilateral constraints and the hybrid nature of stepping. Till this day, bipedal locomotion has not been solved to a satisfactory level, yet.

The main goal of the research work presented in this thesis is thus clear:

To come up with new methods in the field of bipedal humanoid walking and running control that enhance the state of the art and thus contribute to the mentioned mobilization of machines.

1.2 Related work

Aside from robotics in general, the work presented in this thesis contributes to three fields of research: bipedal walking, bipedal running and humanoid whole-body control. In the following, a short overview of related works in the different fields is given. Note that these works cannot always be assigned to only one of the mentioned fields.

1.2.1 Bipedal walking

The first efforts in robotic bipedal locomotion have been put in the subdomain of bipedal walking. Over the decades, the field of bipedal walking control has made major progress. Alongside successes in passive dynamic walking [5] and walking based on the spring-loaded inverted pendulum (SLIP) [6–8], one of the major breakthroughs has been the introduction of Zero Moment Point (ZMP) control [9, 10] for bipedal walking. Many different works have used preplanning of feasible focus points (e.g. ZMPs) in combination with the linear inverted pendulum (LIP) model for bipedal gait generation and control: Stephens and Atkeson [11] present a Model Predictive Control (MPC) method for step adjustment and push recovery. Wieber [12] proposes a trajectory-free linear MPC scheme, allowing for compensation of strong perturbations. Herdt et al. [13] generate stable bipedal walking motions using an MPC formulation. Nishiwaki and Kagami [14] generate dynamically stable walking patterns by frequently updating a preview controller. Sugihara [15] introduces the Best COM-ZMP Regulator facilitating step adjustment of bipedal robots. Kajita et al. [16] demonstrate walking on uneven pavement. Urata et al. [17] present an online walking pattern generator that achieves fast changes in walking direction, high walking speed and strong push recovery on a real biped platform. Tedrake et al. [18] provide a closed-form solution for real-time ZMP gait generation and feedback control. They achieve dynamic walking on the humanoid robot Atlas by recomputing the optimal controller online.

Several previous works [19–27] propose to split the center of mass (CoM) dynamics into a stable and an unstable part. The state variable related to the unstable part of the dynamics has been referred to as '(instantaneous) Capture Point' by Pratt and Koolen et al. [20–22], 'Extrapolated Center of Mass' by Hof et al. [19] and 'Divergent Component of Motion' (DCM) by Takenaka et al. [23]. Motivated by the works of Pratt et al., in [24, 25] the term 'Capture Point' had been used for the DCM. Yet in [28], a significant difference between the Capture Point (defined as the point on the ground where the robot has to step to come to a stop asymptotically) and the Divergent Component of Motion was depicted, as the DCM is not restricted to the ground plane, but is a point in 3D. For 2D considerations (constant CoM height), Capture Point and DCM (projected to the floor) are equivalent, but not for the three-dimensional case.

The use of the LIP model for bipedal walking control is typically restricted to horizontal motions of the CoM ($z = const$). This motivates the derivation of methods that are not limited to constant CoM and floor height. Kajita et al. [10] introduce the 3D Linear Inverted Pendulum Mode, which constrains the CoM to a (not necessarily horizontal) plane. They present experiments for walking on spiral stairs. Zhao and Sentis [29] introduce the Prismatic Inverted Pendulum dynamics and solve it via numerical integration, allowing for three-dimensional foot placement planning on uneven ground surfaces. Yet, lateral foot-placement cannot be predefined, but depends on the sagittal dynamics. Also, the method is restricted to ground surfaces with laterally constant heights.

The walking control framework presented in this thesis provides certain advantages over the previously mentioned works. In [28], a method for three-dimensional bipedal gait planning and control on uneven terrain was presented that overcomes many of the aforementioned restrictions. Yet, that method lead to discontinuous desired leg forces and thus desired joint torques, which can cause perturbations in the actuation system. Therefore, in [30], a method for generating Continuous Double Support (CDS) trajectories is presented, which - similar to the “Multi-Contact Transitions” in [29] - results in smooth enhanced Centroidal Moment Pivot points (eCMP, 3D equivalent of ZMP) and related leg force profiles. The use of toe-off motion facilitates energy-efficient [31] and human-like [32] walking. Thus, the CDS trajectory generator from [30] was extended to a Heel-to-Toe (HT) trajectory generator in the same work, in which the eCMP is shifted from heel to toe during single support. This allows for early toe-off motions, increasing the maximum achievable step length and height. Additionally, [30] provides a robustness analysis of the DCM tracking controller with respect to CoM error, constant external perturbation force, mass estimation error and unknown force lag.

Hopkins et al. [33] enhance the concepts from [30] by introducing the time-varying DCM. By varying the natural frequency of the DCM, they achieve generic vertical CoM trajectories during walking. To smooth discontinuities in the corresponding DCM trajectories, Model Predictive Control (MPC) is applied. Also section 5 of this thesis improves the methods presented in [30] by deriving DCM reference trajectories that correspond to eCMP (3D equivalent of ZMP) reference trajectories that are explicitly designed to lie within the base of support of the robot. Thus, in contrast to [30], the nominal eCMPs are guaranteed to be feasible.

Aside from reduced dynamic models, such as the LIP, several authors propose the use of optimization techniques to either directly design walking motions or to derive movement primitives that can be applied online later. Werner et al. [34] generate bipedal walking gaits using nonlinear optimization and apply these gaits to the DLR Biped (primary stage of the humanoid robot Toro [1]) using a ZMP-based stabilizing controller. In [35], the same authors generate efficient walking trajectories for robots with series-elastic actuators, exploiting the full actuator capabilities. Clever and Mombaur [36] introduce a new template model for optimization studies of human walking and achieve three-dimensional CoM and foot trajectories for walking up and down stairs. Koch et al. [37] generate humanoid gaits based on movement primitives that are learned from optimal and dynamically feasible motion trajectories. Wittmann et al. [38] implement a real-time nonlinear model predictive footstep optimization for bipedal robots based on direct shooting and provide experimental results of their robot LOLA [39] walking under real-world conditions.

1.2.2 Bipedal running and other highly dynamic gaits

Running and hopping are challenging tasks because, during flight, some of the robot’s states are unavoidably non-actuated. Running provides a number of assets such as high achievable speed and efficiency. Back in 1985, Raibert [40] presented his controller that decomposes running into three parts: vertical hopping dynamics, forward velocity and attitude control. The controller design is rather heuristic, yet very powerful. Aside from few exceptions such as [41–45], most running algorithms are based on the spring-loaded inverted pendulum (SLIP) [6]. Dadashzadeh et al. [46] present a SLIP-based two-level controller for running simulations of the ATRIAS robot. Carver et al. [47] show that the number of required recovery steps depends on the goals of the control mechanism and present a SLIP-based controller for two-step recovery using synergies. Vejdani et al. [48] introduce bio-inspired swing leg control for running on ground with unexpected height disturbances. Wu et al. [49] present a deadbeat controller for the 3D SLIP model that can cope with unknown ground height variations of up to 30% of the leg length. Their method is based on multi-dimensional look-up tables and achieves deadbeat control of apex height and heading direction. Yet, since their model assumes energy conservation, the method cannot handle dissipative losses (e.g. during impact). Koepl and Hurst [50] control the stance phase impulse of a planar SLIP model and achieve robust running. Wensing and Orin [51, 52] compute periodic trajectories of the 3D-SLIP offline and apply a linearized control law to stabilize the virtual SLIP model around the periodic solutions. The desired leg forces are passed to a whole-body controller and bipedal running of a simulated humanoid robot is achieved. The method requires offline computation of each desired periodic SLIP gait (including particular turn rates) to obtain the required look-up tables and the linearized SLIP feedback controller is only capable of limited acceleration rates. Park et al. [53] present quadrupedal galloping with the MIT Cheetah 2 based on impulse control. They use third-order Bezier polynomials to encode the leg force profiles. Yet, their method is nominally unstable and designed for constant speeds, such that heuristic PD control laws have to be applied to achieve stability and speed control.

1.2.3 Whole-body control

Whole-body control is a wide field of research that includes control setups of different levels of complexity. One can distinguish controllers that take into account the whole robot motion over a time preview window from controllers that are based on instantaneous control. The first class of controllers typically uses offline optimization. Dai et al. [54] present a method for whole-body motion planning that includes centroidal dynamics and full kinematics and generate highly dynamic motion plans including bipedal walking and running gaits. Schultz and Mombaur [45] generate optimal gaits via an offline direct multiple shooting method.

The second class of *instantaneous* controllers optimizes only the current control inputs, while just few robot states (if any), such as the CoM dynamics, are previewed. This class of controllers typically works in real-time. Sentis and Khatib [55, 56] formulate a hierarchical whole-body control framework for humanoid robots and synthesize whole-body behaviors using behavioral primitives. Ramos et al. [57] integrate the Capture Point in an operational space inverse dynamics controller to achieve whole-body motions that prevent the robot from falling. Hopkins et al. [58] present balancing of a series elastic humanoid robots on unstable terrain using whole-body momentum control. Feng et al. [59] formulate a cascade of online optimizations that allows for compensation of modelling errors and external forces. Their controller was successfully implemented on the Atlas robot and used during the DARPA Robotics Challenge [3]. Wensing and Orin [60] generate dynamic humanoid behaviors through task space control with conic optimization. By controlling the centroidal momentum of the robot, they observe self-emergent whole-body behaviors such as arm swing.

Passivity-based whole-body controllers [61–65] are a subclass of instantaneous controllers. As compared to inverse-dynamics based approaches, these controllers promise higher robustness and more intuitive haptics, which are important for safe human-robot interaction.

1.3 Contributions and overview of the presented research

This thesis provides several contributions in the field of humanoid locomotion control. It extends the formerly two-dimensional concept of Divergent Component of Motion (DCM, also known as "(instantaneous) Capture Point" [19, 20, 23]) to a 3D version, which allows to treat all three spatial directions of the CoM dynamics equivalently and consistently. Also, this work introduces corresponding DCM trajectory generation methods and feedback controllers (see Chap. 5). These facilitate the process of bipedal gait generation and provide interesting insights into the dynamics of walking. When using the DCM as system coordinate, the second order CoM dynamics can be split into the instable DCM dynamics and a second naturally stable component. Leaving the stable component untouched, the original second order control problem turns into a first order one, which explains the mentioned simplicity and comprehensibility of the DCM control framework. The presented research also introduces two new points, the enhanced Centroidal Moment Pivot point (eCMP) and the Virtual Repellent Point (VRP), respectively, which encode external (e.g.

leg-) forces and total force acting on the robot’s CoM. These points can be seen as generalizations of the Zero Moment Point (ZMP) [] and the Centroidal Moment Pivot [66] point to 3D.

Another major contribution of this work is the introduction of Biologically Inspired Deadbeat (BID) control [67–69], which eliminates several drawbacks of state of the art running controllers. BID control is inspired by observations from human running experiments and uses polynomial splines to encode the robot’s CoM motion and leg forces during stance. It is real-time capable, enables versatile running motions and is very robust against external perturbations. The control design is very intuitive and purely analytical. This analyticity facilitates unique features such as online three-dimensional footstep targeting, which - to the author’s knowledge - is a unique feature of BID control, i.e. no other available online planning and control framework for running is able to accomplish that task. The next two upcoming foot aim points on the ground (i.e. the left and the right one) are predicted at all times, which facilitates the design of appropriate foot trajectories. Comparisons to human running gaits show major similarities, such that BID control may serve as tool for human running gait analysis.

The embedding of the DCM-based walking controller and BID-based running controller into the whole-body control (WBC) framework presented in chapter 4 is another contribution of this work. The implemented WBC framework uses a single weighted quadratic program (QP) to solve an inverse dynamics problem. It contains the walking or running task besides other tasks, such as foot tracking, torso orientation control and overall body posture control. The combination of these tasks allows to reproduce several different agile locomotion modes such as walking and running. All control components can be computed in real-time.

The general concept pursued in both the presented walking and running methods is the following: Instead of explicitly previewing future constraint violations, constraint-compatible reference trajectories for a multi-step preview are designed and tracked via feedback control. The references are not only compatible with the constraints but also maximize the corresponding distance in order to increase the margin of stability. That way in many cases the tracking controllers ask for control actions that do not violate the contraints, such that the nominal, stable controller behavior is actually achieved. The embedding of the CoM-based controllers into the presented QP-based whole-body control framework then guarantees feasibility even in case of strong perturbations, while stability cannot be guaranteed[1].

The thesis is organized as follows: Chapter 2 provides mathematical basics for robotics, which the derivations in this thesis are based on. These include transformations and velocities, robot kinematics and equations of motion. Chapter 3 is a collection of useful tools for robotics and motion design. These include polynomial interpolation, pole placement for PID controller parametrization, a constraint-compatible tracking method, a method for quaternion trajectory generation and tracking and an overview of task and null spaces. The whole-body control (WBC) framework used in this work is described in chapter 4. The methods, used in this work for walking and running are presented in chapters 5 and 6. Both the walking and running controllers use the reduced model of CoM dynamics for trajectory generation and feedback control and are then embedded into the WBC framework from chapter 4.

Parts of the research presented in this thesis have been published in conference and journal publications. Table 1.1 gives an overview of the author’s publications as first author, while his publications as coauthor are summarized in table 1.2.

[1]Note that, depending on the contact scenario and robot actuator limits, stability (for any possible perturbation) can never be guaranteed for free-floating robots.

Table 1.1: List of publications as first author

Reference	**Description**
Journal, [1]	J. Englsberger and Ch. Ott. Gangstabilisierung humanoider Roboter mittels Capture Point Regelung / Walking stabilization for humanoid robots based on control of the Capture Point. *at-Automatisierungstechnik, Oldenbourg Wissenschaftsverlag*, pages 692-703, 11/2012.
Journal, [2]	J. Englsberger, Ch. Ott, and A. Albu-Schäffer. Three-Dimensional Bipedal Walking Control Based on Divergent Component of Motion. *IEEE Transactions on Robotics (TRO)*, Vol. 31, No. 2, pp. 355-368, 2015.
Journal, [3]	J. Englsberger, P. Kozlowski, Ch. Ott, A. Albu-Schäffer. Biologically Inspired Deadbeat control for running: from human analysis to humanoid control and back. *IEEE Transactions on Robotics (TRO)*, Vol. 32, No. 4, pp. 854-867, 2016.
Conference: [4]	J. Englsberger, Ch. Ott, M. A. Roa, A. Albu-Schäffer, and G. Hirzinger. Bipedal walking control based on Capture Point dynamics. In *Int. Conf. on Intell. Robots and Systems*, pages 4420-4427, 2011.
Conference: [5]	J. Englsberger and Ch. Ott. Integration of vertical COM motion and angular momentum in an extended Capture Point tracking controller for bipedal walking. In *IEEE-RAS Int. Conf. on Humanoid Robots*, pages 183-189, 2012.
Conference: [6]	J. Englsberger, Ch. Ott and A. Albu-Schäffer. Three-dimensional bipedal walking control using Divergent Component of Motion. In *Int. Conf. on Intell. Robots and Systems*, pages 2600-2607, 2013.
Conference: [7]	J. Englsberger, T. Koolen, S. Bertrand, J. Pratt, Ch. Ott, and A. Albu-Schäffer. Trajectory generation for continuous leg forces during double support and heel-to-toe shift based on divergent component of motion. In *Int. Conf. on Intell. Robots and Systems*, pages 4022-4029, 2014.
Conference: [8]	J. Englsberger, A. Werner, Ch. Ott, B. Henze, M. A. Roa, G. Garofalo, R. Burger, A. Beyer, O. Eiberger, K. Schmid and A. Albu-Schäffer. Overview of the torque-controlled humanoid robot TORO. In *IEEE-RAS Int. Conf. on Humanoid Robots*, pages 916-923, 2014.
Conference: [9]	J. Englsberger, P. Kozlowski, and Ch. Ott. Biologically Inspired Dead-beat controller for bipedal running in 3D. In *IEEE/RSJ Int. Conf. on Intell. Robots and Systems*, pages 989-996, 2015.
Conference: [10]	J. Englsberger, P. Kozlowski, and Ch. Ott. Biologically Inspired Deadbeat control for running on 3D stepping stones. In *IEEE-RAS Int. Conf. on Humanoid Robots*, pages 1067-1074, 2015.

Table 1.2: List of publications as coauthor

Reference	**Description**
Journal, [11]	Ch. Ott, O. Eiberger, J. Englsberger, M. A. Roa, and A. Albu-Schäffer. Hardware and Control Concept for an Experimental Bipedal Robot with Joint Torque Sensors. *Journal of the Robotics Society of Japan*, Vol. 30, No. 4, pp. 378-382, May 2012.
Journal, [12]	Ch. Ott, A. Dietrich, D. Leidner, A. Werner, J. Englsberger, B. Henze, S. Wolf, M. Chalon, W. Friedl, A. Beyer, O. Eiberger, A. Albu-Schäffer. From torque-controlled to intrinsically compliant humanoid robots . *ASME Dynamic Systems and Control Magazine*, Vol. 3, No. 2, pp. 7-11, June 2015.
Journal, [13]	T. Koolen, S. Bertrand, G. Thomas, T. de Boer, T. Wu, J. Smith, J. Englsberger, J. Pratt. Design of a momentum-based control framework and application to the humanoid robot Atlas. *International Journal of Humanoid Robotics*, Vol. 13, No. 1, pp. 1650007 (34 pages) , March 2016
Journal, [14]	N. Perrin, Ch. Ott, J. Englsberger, O. Stasse, F. Lamiraux, D. G. Caldwell . Continuous Legged Locomotion Planning. *IEEE Transactions on Robotics (short paper, accepted)*
Conference: [15]	M. Krause, J. Englsberger, P.-B. Wieber, and Ch. Ott. Stabilization of the Capture Point Dynamics for Bipedal Walking based on Model Predictive Control. In *10th IFAC Symposium on Robot Control - SYROCO*, pages 165-171, 2012.
Conference: [16]	H. Kaminaga, J. Englsberger, and Ch. Ott. Kinematic optimization and online adaptation of swing foot trajectory for biped locomotion. In *IEEE-RAS Int. Conf. on Humanoid Robots*, pages 593-599, 2012.
Conference: [17]	T. Koolen, J. Smith, G. Thomas, S. Bertrand, J. Carff, N. Mertins, D. Stephen, P. Abeles, J. Englsberger, S. McCrory, J. van Egmond, M. Griffioen, M. Floyd , S. Kobus, N. Manor, S. Alsheikh, D. Duran, L. Bunch, E. Morphis, L. Colasanto, K.-L. Ho Hoang, B. Layton, P. Neuhaus, M. Johnson, and J. Pratt. Summary of team IHMC's virtual robotics challenge entry. In *IEEE-RAS Int. Conf. on Humanoid Robots*, pages 307-314, 2013.
Conference: [18]	B. Henze, A. Werner, M. A. Roa, G. Garofalo, J. Englsberger, and Ch. Ott. Control Applications of TORO - a Torque Controlled Humanoid Robot (**Best Video Award**). In *IEEE-RAS Int. Conf. on Humanoid Robots*, 2014.
Conference: [19]	G. Garofalo, B. Henze, J. Englsberger and Ch. Ott. On the inertially decoupled structure of the floating base robot dynamics . In *conference on mathematical modelling (MATHMOD)* , 2015.
Conference: [20]	G. Garofalo, J. Englsberger, Ch. Ott. On the regulation of the energy of elastic joint robots: excitation and damping of oscillations. In *American Control Conference (ACC)*, 2015.

CHAPTER 2

Mathematical basics for robotics

This chapter provides the notation and mathematical basics for the methods derived in the subsequent chapters of this thesis. For readers with little experience in robotics, it may serve as compact introduction to this exciting field of research.

2.1 Notations used in this work

In this chapter, the following notation for arbitrary points or vectors (denoted by $\boldsymbol{v}$ here) is used:

$$ {}^{\beta}\boldsymbol{v}_{r,k} \ . \tag{2.1}$$

The indices β, r and k denote the reference frame, the reference point (or link frame origin) and the point (or link frame origin) of interest, respectively. In other words, (2.1) defines a vector from the point (or link frame origin) indicated by index r to the point (or link frame origin) indicated by index k represented in the frame indicated by index β. The upper left index position is used for the latter to keep the upper right index position free for inversion or square terms.

The notation for rotation matrices and homogeneous transformations is

$$ {}^{\alpha}\boldsymbol{A}_{\beta} \ , \tag{2.2}$$

where ${}^{\alpha}\boldsymbol{A}_{\beta}$ denotes a rotation matrix ${}^{\alpha}\boldsymbol{R}_{\beta}$ or a homogeneous transformation matrix ${}^{\alpha}\boldsymbol{H}_{\beta}$. This notation represents the rotation or homogeneous transformation from a reference frame with index α to another frame with index β. For a more intuitive understanding: The rotation matrix ${}^{\alpha}\boldsymbol{R}_{\beta}$ denotes the base vectors of frame β represented in the frame with index α.

A wrench ${}^{\beta}\boldsymbol{w}_k$ combines a force ${}^{\beta}\boldsymbol{f}_k$ and a torque ${}^{\beta}\boldsymbol{\tau}_k$ in a single covector:

$$ {}^{\beta}\boldsymbol{w}_k = \begin{bmatrix} {}^{\beta}\boldsymbol{f}_k \\ {}^{\beta}\boldsymbol{\tau}_k \end{bmatrix} \ . \tag{2.3}$$

As above, the left upper index β denotes the base (or frame) that the wrench is represented in while the lower right index k denotes the link that the wrench is applied to.

2.2 Homogeneous transformations

A three-dimensional point with index k can be transformed from frame j to frame i via

$$ {}^{i}\boldsymbol{x}_{i,k} = {}^{i}\boldsymbol{x}_{i,j} + \underbrace{{}^{i}\boldsymbol{R}_j \, {}^{j}\boldsymbol{x}_{j,k}}_{{}^{i}\boldsymbol{x}_{j,k}} \ . \tag{2.4}$$

By augmenting the three-dimensional position (or vector) with an additional "1" (i.e. creating homogeneous coordinates), this correlation can be written in a more compact form:

$$ \begin{bmatrix} {}^{i}\boldsymbol{x}_{i,k} \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} {}^{i}\boldsymbol{R}_j & {}^{i}\boldsymbol{x}_{i,j} \\ \boldsymbol{0}_{1\times 3} & 1 \end{bmatrix}}_{{}^{i}\boldsymbol{H}_j} \begin{bmatrix} {}^{j}\boldsymbol{x}_{j,k} \\ 1 \end{bmatrix} \ . \tag{2.5}$$

Here, ${}^{i}\boldsymbol{H}_j$ denotes the homogeneous transformation from link i to link j. If the world frame is denoted by "0", the relative rotation matrix ${}^{i}\boldsymbol{R}_j$ can be derived from the global orientations ${}^{0}\boldsymbol{R}_i$ of link i and ${}^{0}\boldsymbol{R}_j$ of link j as ${}^{i}\boldsymbol{R}_j = {}^{0}\boldsymbol{R}_i^T \ {}^{0}\boldsymbol{R}_j$. The relative translation between the two links ${}^{i}\boldsymbol{x}_{i,j}$ can be computed from the corresponding global translations as ${}^{i}\boldsymbol{x}_{i,j} = {}^{i}\boldsymbol{x}_{0,j} - {}^{i}\boldsymbol{x}_{0,i}$ (still using the frame of link i as base).

Homogeneous transformations as in (2.5) can be concatenated as follows:

$$ {}^{i}\boldsymbol{H}_k = {}^{i}\boldsymbol{H}_j \ {}^{j}\boldsymbol{H}_k \ . \tag{2.6}$$

Using (2.6), arbitrary transformation chains can be constructed. Note that the rotations from one frame to another are automatically included in this notation.

The inverse of a homogeneous transformation can be conveniently computed as

$$ {}^{i}\boldsymbol{H}_j^{-1} = \begin{bmatrix} {}^{i}\boldsymbol{R}_j^T & -\,{}^{i}\boldsymbol{R}_j^T \, {}^{i}\boldsymbol{x}_{i,j} \\ \boldsymbol{0}_{1\times 3} & 1 \end{bmatrix} = {}^{j}\boldsymbol{H}_i = \begin{bmatrix} {}^{j}\boldsymbol{R}_i & -\,{}^{j}\boldsymbol{R}_i \, {}^{i}\boldsymbol{x}_{i,j} \\ \boldsymbol{0}_{1\times 3} & 1 \end{bmatrix} \ . \tag{2.7}$$

Remember that the inverse of a rotation matrix is its transpose, i.e. ${}^{i}\boldsymbol{R}_j^{-1} = {}^{i}\boldsymbol{R}_j^T = {}^{j}\boldsymbol{R}_i$, because rotation matrices are orthogonal and their determinant is 1.

2.3 The skew operator

One operation that is particularly convenient for many of the subsequent derivations is the following: for an arbitrary three-dimensional vector $\boldsymbol{v} = [v_1, v_2, v_3]^T$, its cross product with another vector $\boldsymbol{w}$ can be written as a matrix vector product, i.e. $\boldsymbol{v} \times \boldsymbol{w} = [\boldsymbol{v}\times]\, \boldsymbol{w}$ for any three-dimensional vector $\boldsymbol{w}$. It can be explicitly written as

$$ [\boldsymbol{v}\times] = \begin{bmatrix} 0 & -v_3 & v_2 \\ v_3 & 0 & -v_1 \\ -v_2 & v_1 & 0 \end{bmatrix} \tag{2.8}$$

Obviously, $[\boldsymbol{v}\times]$ is a skew-symmetric matrix. One example, where this skew operator is used, is the correlation between the spatial angular velocity ${}^{i}\boldsymbol{\omega}_{i,j}$ (see Sec. 2.5) and the derivative ${}^{i}\dot{\boldsymbol{R}}_j$ of a rotation matrix ${}^{i}\boldsymbol{R}_j$, which can be expressed as

$$ [{}^{i}\boldsymbol{\omega}_{i,j}\times] = {}^{i}\dot{\boldsymbol{R}}_j \ {}^{i}\boldsymbol{R}_j^T \ . \tag{2.9}$$

The equivalent correlation for the mapping that considers the body angular velocity ${}^{j}\boldsymbol{\omega}_{i,j}$ is

$$ [{}^{j}\boldsymbol{\omega}_{i,j}\times] = {}^{i}\boldsymbol{R}_j^T \ {}^{i}\dot{\boldsymbol{R}}_j \ . \tag{2.10}$$

2.4 Adjoint transformations

The transformation of a Jacobian or corresponding velocity from one frame to another is an operation which is widely used in robotics. This transformation is accomplished by the so called *adjoint transformation*. Any Jacobian (or corresponding velocity) that is represented in a frame j can be transformed to another frame i via

$$ {}^{i}\boldsymbol{J}_{k,l} \;=\; \boldsymbol{Ad}_{\,{}^{i}\boldsymbol{H}_j}\; {}^{j}\boldsymbol{J}_{k,l} \quad . \tag{2.11}$$

Here, i, j, k and l denote arbitrary frames (not necessarily different frames, see below). The adjoint transformation $\boldsymbol{Ad}_{\,{}^{i}\boldsymbol{H}_j}$ uses the homogeneous transformation ${}^{i}\boldsymbol{H}_j$ as input:

$$ \boldsymbol{Ad}_{\,{}^{i}\boldsymbol{H}_j} \;=\; \begin{bmatrix} {}^{i}\boldsymbol{R}_j & [{}^{i}\boldsymbol{x}_{i,j}\times]\; {}^{i}\boldsymbol{R}_j \\ \boldsymbol{0}_{3\times3} & {}^{i}\boldsymbol{R}_j \end{bmatrix} \tag{2.12}$$

In many applications, the inverse of the adjoint transformation is required. It can be computed as:

$$ \boldsymbol{Ad}^{-1}_{\,{}^{i}\boldsymbol{H}_j} \;=\; \begin{bmatrix} {}^{i}\boldsymbol{R}_j^{T} & -\;{}^{i}\boldsymbol{R}_j^{T}\; [{}^{i}\boldsymbol{x}_{i,j}\times] \\ \boldsymbol{0}_{3\times3} & {}^{i}\boldsymbol{R}_j^{T} \end{bmatrix} \;=\; \boldsymbol{Ad}_{\,{}^{i}\boldsymbol{H}_j^{-1}} \quad , \tag{2.13}$$

An additional useful operation is the *Lie bracket matrix* (presented in [70]) of a 6-dimensional velocity vector $\boldsymbol{\nu}$, which is defined as

$$ \boldsymbol{adj}(\boldsymbol{\nu}) = \begin{bmatrix} [\boldsymbol{\omega}\times] & [\dot{\boldsymbol{x}}\times] \\ 0 & [\boldsymbol{\omega}\times] \end{bmatrix} \quad . \tag{2.14}$$

Here, $\dot{\boldsymbol{x}}$ and $\boldsymbol{\omega}$ denote the linear and angular components of the six-dimensional (body or spatial) velocity vector $\boldsymbol{\nu} = [\dot{\boldsymbol{x}}^T, \boldsymbol{\omega}^T]^T$. This Lie bracket matrix can be used to compute the time derivative of the adjoint transformation, for example:

$$ \dot{\boldsymbol{Ad}}_{\,{}^{i}\boldsymbol{H}_j} = \boldsymbol{Ad}_{\,{}^{i}\boldsymbol{H}_j}\, \boldsymbol{adj}({}^{j}_{b}\boldsymbol{\nu}_{i,j}) \quad . \tag{2.15}$$

Here, ${}^{j}_{b}\boldsymbol{\nu}_{i,j}$ denotes the body velocity, which is introduced in the next section.

In addition to Jacobian transformations, the adjoints can also be used to *transform wrenches*. If a wrench represented in a given frame (upper left index, see notation (2.3)) has to be transformed to another base, the following correlations can be applied:

$$ {}^{j}\boldsymbol{w}_k = \boldsymbol{Ad}^{T}_{\,{}^{i}\boldsymbol{H}_j}\; {}^{i}\boldsymbol{w}_k \qquad \Leftrightarrow \qquad {}^{i}\boldsymbol{w}_k = \boldsymbol{Ad}^{-T}_{\,{}^{i}\boldsymbol{H}_j}\; {}^{j}\boldsymbol{w}_k \tag{2.16}$$

2.5 Six-dimensional velocity vectors

In [71], Murray et al. describe three different six-dimensional velocities that combine both linear and angular motion: the *hybrid velocity*, the *body velocity* and the *spatial velocity*. These velocities can be intuitively interpreted as follows:

The *hybrid velocity vector* combines the linear and angular velocity of a link j relative to another link i represented in the coordinate frame that is attached to link i.[1]

The *body velocity vector* combines the linear and angular velocity of a link j relative to another link i but now represented in the coordinate frame attached to link j. Intuitively, this would be the

[1]This representation correlates to the notation that is typically referred to, when engineers talk about “the” linear and angular velocity of a body.

relative velocity between the two links that a person sitting on link j would observe. For example: this velocity is typically used when processing data from an inertial measurement unit (IMU) that measures the velocity of a body w.r.t. the world, represented in the body frame.

The *spatial velocity vector* is less intuitive at first. While its angular component equals the hybrid counterpart (i.e. simply the spatial angular velocity), its linear component indicates the velocity of a point[2] that is attached to link j relative to the frame attached to link i with the peculiar feature that this point momentarily coincides with the origin of frame i. The spatial velocity vector is useful for a multitude of operations.

As compared to the notation presented in Sec. 2.1, an additional lower left index h (for hybrid), b (for body) and s (for spatial) is introduced to indicate the three different six-dimensional velocities. Using the mathematics introduced by Murray et al. [71] but utilizing the notation from Sec. 2.1, the correlations for the *hybrid velocity* can be formulated as

$$ {}^{i}_{h}\boldsymbol{\nu}_{i,j} = \begin{bmatrix} {}^{i}\dot{\boldsymbol{x}}_{i,j} \\ {}^{i}\boldsymbol{\omega}_{i,j} \end{bmatrix} = {}^{i}_{h}\boldsymbol{J}_{i,j}\,\dot{\boldsymbol{q}} = \begin{bmatrix} {}^{i}\boldsymbol{R}_{j} & \boldsymbol{0}_{3\times3} \\ \boldsymbol{0}_{3\times3} & {}^{i}\boldsymbol{R}_{j} \end{bmatrix} {}^{j}_{b}\boldsymbol{\nu}_{i,j} = \begin{bmatrix} \boldsymbol{I}_{3\times3} & -[{}^{i}\boldsymbol{x}_{i,j}\times] \\ \boldsymbol{0}_{3\times3} & \boldsymbol{I}_{3\times3} \end{bmatrix} {}^{i}_{s}\boldsymbol{\nu}_{i,j} \ , \tag{2.17} $$

the correlations for the *body velocity* as

$$ {}^{j}_{b}\boldsymbol{\nu}_{i,j} = \begin{bmatrix} {}^{j}\dot{\boldsymbol{x}}_{i,j} \\ {}^{j}\boldsymbol{\omega}_{i,j} \end{bmatrix} = {}^{j}_{b}\boldsymbol{J}_{i,j}\,\dot{\boldsymbol{q}} = \begin{bmatrix} {}^{i}\boldsymbol{R}_{j}^{T} & \boldsymbol{0}_{3\times3} \\ \boldsymbol{0}_{3\times3} & {}^{i}\boldsymbol{R}_{j}^{T} \end{bmatrix} {}^{i}_{h}\boldsymbol{\nu}_{i,j} = \boldsymbol{Ad}^{-1}_{{}^{i}\boldsymbol{H}_{j}}\ {}^{i}_{s}\boldsymbol{\nu}_{i,j} \ , \tag{2.18} $$

and the correlations for the *spatial velocity* as

$$ {}^{i}_{s}\boldsymbol{\nu}_{i,j} = \begin{bmatrix} {}^{i}\dot{\boldsymbol{x}}_{i,j} - {}^{i}\boldsymbol{\omega}_{i,j} \times {}^{i}\boldsymbol{x}_{i,j} \\ {}^{i}\boldsymbol{\omega}_{i,j} \end{bmatrix} = {}^{i}_{s}\boldsymbol{J}_{i,j}\,\dot{\boldsymbol{q}} = \begin{bmatrix} \boldsymbol{I}_{3\times3} & [{}^{i}\boldsymbol{x}_{i,j}\times] \\ \boldsymbol{0}_{3\times3} & \boldsymbol{I}_{3\times3} \end{bmatrix} {}^{i}_{h}\boldsymbol{\nu}_{i,j} = \boldsymbol{Ad}_{{}^{i}\boldsymbol{H}_{j}}\ {}^{j}_{b}\boldsymbol{\nu}_{i,j} \ . \tag{2.19} $$

The second terms (between the first equal signs) denote the actual definitions of the six-dimensional velocity vectors using the corresponding three-dimensional linear and angular velocity vectors $\dot{\boldsymbol{x}}$ and $\boldsymbol{\omega}$. The third terms demonstrate the velocity computations via the corresponding Jacobian matrices ($\dot{\boldsymbol{q}}$ being the generalized joint velocities of a robot (see Sec. 2.10)), and the subsequent terms represent the intercorrelations between hybrid, body and spatial velocity.

2.6 Propagation of Jacobians and their derivatives along a series of links

This section will provide equations for the propagation of Jacobians and their derivatives along a serial chain of links. This propagation is used to compute the Jacobian of a link further down the chain relating to the base frame.

The equations below are then typically evaluated iteratively to propagate from one link to the next until the link of interest is reached. Note that the correlations are also valid for propagation of hybrid, body or spatial velocities. The corresponding Jacobians simply have to be replaced by the appropriate six-dimensional velocity vectors. In the following, the propagation formulas will be provided without going into detail (more details can be found in [71]):

[2]Note that typically this is not a unique point on link j but more of a temporary construction.

2.7 Propagation of hybrid Jacobians

To obtain the hybrid Jacobian ${}^{i}_{h}\boldsymbol{J}_{i,k}$ of a link, the hybrid Jacobian of its parent ${}^{i}_{h}\boldsymbol{J}_{i,j}$ is propagated using the relative hybrid Jacobian between parent and link ${}^{j}_{h}\boldsymbol{J}_{j,k}$:

$$
{}^{i}_{h}\boldsymbol{J}_{i,k} = \underbrace{\begin{bmatrix} \boldsymbol{I}_{3\times3} & -\,{}^{i}\boldsymbol{R}_{j}\,[{}^{j}\boldsymbol{x}_{j,k}\times]\,{}^{i}\boldsymbol{R}_{j}^{T} \\ \boldsymbol{0}_{3\times3} & \boldsymbol{I}_{3\times3} \end{bmatrix}}_{\boldsymbol{A}_{\alpha}} {}^{i}_{h}\boldsymbol{J}_{i,j} + \underbrace{\begin{bmatrix} {}^{i}\boldsymbol{R}_{j} & \boldsymbol{0}_{3\times3} \\ \boldsymbol{0}_{3\times3} & {}^{i}\boldsymbol{R}_{j} \end{bmatrix}}_{\boldsymbol{A}_{\beta}} {}^{j}_{h}\boldsymbol{J}_{j,k} \tag{2.20}
$$

The derivative of the hybrid Jacobian is propagated via

$$
{}^{i}_{h}\dot{\boldsymbol{J}}_{i,k} = \begin{bmatrix} \boldsymbol{0}_{3\times3} & \boldsymbol{B} \\ \boldsymbol{0}_{3\times3} & \boldsymbol{0}_{3\times3} \end{bmatrix} {}^{i}_{h}\boldsymbol{J}_{i,j} + \boldsymbol{A}_{\alpha}\ {}^{i}_{h}\dot{\boldsymbol{J}}_{i,j} + \begin{bmatrix} {}^{i}\dot{\boldsymbol{R}}_{j} & \boldsymbol{0}_{3\times3} \\ \boldsymbol{0}_{3\times3} & {}^{i}\dot{\boldsymbol{R}}_{j} \end{bmatrix} {}^{j}_{h}\boldsymbol{J}_{j,k} + \boldsymbol{A}_{\beta}\ {}^{j}_{h}\dot{\boldsymbol{J}}_{j,k}\ , \tag{2.21}
$$

where

$$
\boldsymbol{B} = -\left({}^{i}\dot{\boldsymbol{R}}_{j}\,[{}^{j}\boldsymbol{x}_{j,k}\times]\ {}^{i}\boldsymbol{R}_{j}^{T} + {}^{i}\boldsymbol{R}_{j}\,[{}^{j}\dot{\boldsymbol{x}}_{j,k}\times]\ {}^{i}\boldsymbol{R}_{j}^{T} + {}^{i}\boldsymbol{R}_{j}\,[{}^{j}\boldsymbol{x}_{j,k}\times]\ {}^{i}\dot{\boldsymbol{R}}_{j}^{T} \right) \tag{2.22}
$$

and ${}^{i}\dot{\boldsymbol{R}}_{j} = [{}^{i}\boldsymbol{\omega}_{i,j}\times]\ {}^{i}\boldsymbol{R}_{j}$.

2.8 Propagation of body Jacobians

The propagation of body Jacobians and their derivatives is similar to the one for hybrid Jacobians. The corresponding propagation equations are

$$
{}^{k}_{b}\boldsymbol{J}_{i,k} = \boldsymbol{Ad}^{-1}_{{}^{j}\boldsymbol{H}_{k}}\ {}^{j}_{b}\boldsymbol{J}_{i,j} + {}^{k}_{b}\boldsymbol{J}_{j,k} \tag{2.23}
$$

for the body Jacobian, and

$$
{}^{k}_{b}\dot{\boldsymbol{J}}_{i,k} = \boldsymbol{Ad}^{-1}_{{}^{j}\boldsymbol{H}_{k}}\ {}^{j}_{b}\dot{\boldsymbol{J}}_{i,j} - \boldsymbol{adj}({}^{k}_{b}\boldsymbol{\nu}_{j,k})\ \boldsymbol{Ad}^{-1}_{{}^{j}\boldsymbol{H}_{k}}\ {}^{j}_{b}\boldsymbol{J}_{i,j} + {}^{k}_{b}\dot{\boldsymbol{J}}_{j,k} \tag{2.24}
$$

for the time derivative of the body Jacobian.

2.9 Propagation of spatial Jacobians

The propagation of spatial Jacobians and their derivatives again works similarly as for the other Jacobians. The corresponding propagation equations are

$$
{}^{i}_{s}\boldsymbol{J}_{i,k} = {}^{i}_{s}\boldsymbol{J}_{i,j} + \boldsymbol{Ad}_{{}^{i}\boldsymbol{H}_{j}}\ {}^{j}_{s}\boldsymbol{J}_{j,k} \tag{2.25}
$$

for the spatial Jacobian and

$$
{}^{i}_{s}\dot{\boldsymbol{J}}_{i,k} = {}^{i}_{s}\dot{\boldsymbol{J}}_{i,j} + \boldsymbol{Ad}_{{}^{i}\boldsymbol{H}_{j}}\ \boldsymbol{adj}(\boldsymbol{Ad}^{-1}_{{}^{i}\boldsymbol{H}_{j}}\ {}^{i}_{s}\boldsymbol{\nu}_{i,j})\ {}^{j}_{s}\boldsymbol{J}_{j,k} + \boldsymbol{Ad}_{{}^{i}\boldsymbol{H}_{j}}\ {}^{j}_{s}\dot{\boldsymbol{J}}_{j,k} \tag{2.26}
$$

for the time derivative of the spatial Jacobian.

The three different Jacobians (hybrid, body and spatial) and their propagation are presented in this chapter to provide a complete overview.

2.10 Robot kinematics

2.10.1 Kinematic robot description

The kinematic structure of a robot is typically described using a *default posture*, i.e. a posture in which all joint angles are defined to be zero. For all robot links k, the default transformation of the frame[3] is defined as

$$ {}^{0}_{def}\boldsymbol{H}_k = \begin{bmatrix} {}^{0}_{def}\boldsymbol{R}_k & {}^{0}_{def}\boldsymbol{x}_{0,k} \\ \boldsymbol{0}_{1\times3} & 1 \end{bmatrix} . \tag{2.27}$$

Here, ${}^{0}_{def}\boldsymbol{R}_k$ is the relative rotation between the global frame and the k-th link frame and ${}^{0}_{def}\boldsymbol{x}_{0,k}$ denotes its displacement, both represented w.r.t. the world frame “0”. The default relative transformation between a link k and its parent link p can be computed as

$$ {}^{p}_{def}\boldsymbol{H}_k = {}^{0}_{def}\boldsymbol{H}_p^{-1}\; {}^{0}_{def}\boldsymbol{H}_k . \tag{2.28}$$

In addition to the default link frames ${}^{0}_{def}\boldsymbol{H}_k$, information about the possible directions of motion of the robot links w.r.t. each other is required to describe the robot kinematics. One possible encoding of such motion constraints are the 6×1 relative spatial Jacobians ${}^{0}_{def}\boldsymbol{\jmath}_{p,k}$. They represent the possible motion of each link k w.r.t. its parent p in a six-dimensional vector that is represented in world frame “0” and *in the default posture*. These Jacobians will be used in (2.44) to formulate the exponential of twist formula ($\rightarrow$ forward kinematics).

For prismatic (i.e. linear) joints, these relative spatial Jacobians can be derived as

$$ {}^{0}_{def}\boldsymbol{\jmath}_{p,k} = \begin{bmatrix} {}^{0}_{def}\boldsymbol{u}_{lin,k} \\ \boldsymbol{0}_{3\times1} \end{bmatrix} . \tag{2.29}$$

Here, ${}^{0}_{def}\boldsymbol{u}_{lin,k} \in \mathbb{R}^{3\times1}$ denotes the unit vector pointing in the direction of the linear axis of the k-th link represented in world frame “0” and referring to the default pose of the robot.

For revolute (i.e. rotational) joints, the relative spatial Jacobians can be computed as

$$ {}^{0}_{def}\boldsymbol{\jmath}_{p,k} = \begin{bmatrix} [{}^{0}_{def}\boldsymbol{x}_{0,ax_k}\times] \\ \boldsymbol{I}_{3\times3} \end{bmatrix} {}^{0}_{def}\boldsymbol{u}_{rot,k} , \tag{2.30}$$

where ${}^{0}_{def}\boldsymbol{x}_{0,ax_k} \in \mathbb{R}^{3\times1}$ denotes a point on the axis of rotation represented w.r.t. the world frame “0” and ${}^{0}_{def}\boldsymbol{u}_{rot,k} \in \mathbb{R}^{3\times1}$ is the unit vector pointing in the direction of the rotational axis of the k-th link, also represented in world frame “0”. Both axis point and unit vector are represented in the default pose of the robot.

For screw joints[4] the relative spatial Jacobians are computed as:

$$ {}^{0}_{def}\boldsymbol{\jmath}_{p,k} = \begin{bmatrix} [{}^{0}_{def}\boldsymbol{x}_{0,ax_k}\times] + h_{screw}\boldsymbol{I}_{3\times3} \\ \boldsymbol{I}_{3\times3} \end{bmatrix} {}^{0}_{def}\boldsymbol{u}_{rot,k} \tag{2.31}$$

where ${}^{0}_{def}\boldsymbol{x}_{0,ax_k} \in \mathbb{R}^{3\times1}$ denotes a point on the screw axis represented w.r.t. the world frame “0” and ${}^{0}_{def}\boldsymbol{u}_{rot,k} \in \mathbb{R}^{3\times1}$ is the unit vector pointing in the direction of the screw axis of the k-th link, also represented in world frame “0”, and h_{screw} is the pitch of the screw. Again, both axis point and unit vector are represented in the default pose of the robot.

[3]Note: Link frames do not necessarily coincide with the corresponding joints.

[4]Note: Skrew joints can implement the most general one degree of freedom (DOF) motions. They include both revolute and prismatic joints as special cases.

The relative spatial Jacobians ${}^{0}_{def}\boldsymbol{j}_{p,k}$ can be transformed into the relative body Jacobians

$$\underbrace{{}^{k}_{b}\boldsymbol{j}_{p,k}}_{const} = \boldsymbol{Ad}^{-1}_{{}^{0}_{def}\boldsymbol{H}_k}\ {}^{0}_{def}\boldsymbol{j}_{p,k} \ . \tag{2.32}$$

Note that these relative body Jacobians are constant (i.e. ${}^{k}_{b}\boldsymbol{j}_{p,k} = {}_{def,b}{}^{k}\boldsymbol{j}_{p,k}$) because the motion constraints of the link w.r.t. its own frame do not change. They will be used in (2.54) for the forward propagation of global body Jacobians. Note that the 6×1 relative Jacobians ${}^{0}_{def}\boldsymbol{j}_{p,k}$ and ${}^{k}_{b}\boldsymbol{j}_{p,k}$ can be interpreted as twists coordinates (see next section) and thus, the methods from Sec. 2.10.2 can be applied.

To conclude, the default spatial link frames ${}^{0}_{def}\boldsymbol{H}_k$ and the relative spatial Jacobians ${}^{0}_{def}\boldsymbol{j}_{p,k}$ (or alternatively the relative body Jacobians ${}^{k}_{b}\boldsymbol{j}_{p,k}$) are sufficient to specify the kinematics of a robot.

2.10.2 Twists and their exponentials

A mathematical tool that is often used in the analysis of robot kinematics is the so called *twist*. It is a 4×4 matrix that can be written as

$$\widehat{\boldsymbol{\vartheta}} = \begin{bmatrix} [\boldsymbol{\vartheta}_{rot}\times] & \boldsymbol{\vartheta}_{lin} \\ \boldsymbol{0}_{1\times 3} & 0 \end{bmatrix} \ . \tag{2.33}$$

Its corresponding *twist coordinates* $\boldsymbol{\vartheta}$ consist of the 3×1 linear component $\boldsymbol{\vartheta}_{lin}$ and a 3×1 rotational component $\boldsymbol{\vartheta}_{rot}$ which are stacked as

$$\boldsymbol{\vartheta} = \begin{bmatrix} \boldsymbol{\vartheta}_{lin} \\ \boldsymbol{\vartheta}_{rot} \end{bmatrix} \ . \tag{2.34}$$

Any 6×1 Jacobian $\boldsymbol{j}$ that contains a 3×1 linear component $\boldsymbol{j}_{lin}$ stacked on a 3×1 rotational component $\boldsymbol{j}_{rot}$, i.e.

$$\boldsymbol{j} = \begin{bmatrix} \boldsymbol{j}_{lin} \\ \boldsymbol{j}_{rot} \end{bmatrix} \ , \tag{2.35}$$

can be interpreted as such twist coordinates. Thus, the same mathematical tools used for twists and twist coordinates can be applied to such Jacobians.

If the 6×1 Jacobian $\boldsymbol{j}_k$ of a link k describes a pure translation, i.e. $\boldsymbol{j}_{rot,k} = \boldsymbol{0}_{3\times 1}$, the twist exponential for that Jacobian can be computed as

$$e^{(\widehat{\boldsymbol{j}}_k\, q_k)} = \begin{bmatrix} \boldsymbol{I}_{3\times 3} & \boldsymbol{j}_{lin,k}\, q_k \\ \boldsymbol{0}_{1\times 3} & 1 \end{bmatrix} \ , \tag{2.36}$$

where q_k denotes the corresponding joint angle. Otherwise ($\boldsymbol{j}_{rot,k} \neq \boldsymbol{0}_{3\times 1}$), the twist exponential of a Jacobian $\boldsymbol{j}_k$ is

$$e^{(\widehat{\boldsymbol{j}}_k\, q_k)} = \begin{bmatrix} \boldsymbol{R}_e & (\boldsymbol{I}_{3\times 3} - \boldsymbol{R}_e)\,[\boldsymbol{j}_{rot,k}\times]\,\boldsymbol{j}_{lin,k} + \boldsymbol{j}_{rot,k}\,\boldsymbol{j}^T_{rot,k}\,\boldsymbol{j}_{lin,k}\, q_k \\ \boldsymbol{0}_{1\times 3} & 1 \end{bmatrix} \ . \tag{2.37}$$

Here, $\boldsymbol{R}_e$ is a rotation matrix that can be computed using *Rodrigues' formula* [72]:

$$\boldsymbol{R}_e = e^{(\widehat{\boldsymbol{j}}_{rot,k}\, q_k)} = \boldsymbol{I}_{3\times 3} + [\boldsymbol{j}_{rot,k}\times]\, sin(q_k) + [\boldsymbol{j}_{rot,k}\times]^2\,(1 - cos(q_k)) \ . \tag{2.38}$$

2.10.3 Forward kinematics: the product of exponentials formula

Ploen [73] summarizes different formulations for the forward kinematics of robots. To support the following derivations, the k-th reference frame ${}^{0}\boldsymbol{H}_{k_{ref}}$ is introduced, which is defined as

$$ {}^{0}\boldsymbol{H}_{k_{ref}} = {}^{0}\boldsymbol{H}_{p} \; {}^{p}_{def}\boldsymbol{H}_{k} \; , \tag{2.39}$$

i.e. it is fixed to the parent link frame ${}^{0}\boldsymbol{H}_{p}$ at a constant offset ${}^{p}_{def}\boldsymbol{H}_{k}$ (which is the default relative transformation between link k and its parent p from (2.28)). Reordering (2.39) yields

$$ {}^{p}\boldsymbol{H}_{k_{ref}} = {}^{p}_{def}\boldsymbol{H}_{k} \; . \tag{2.40}$$

Ploen shows that the relative transformation between a link k and its parent p can be expressed as

$$ {}^{p}\boldsymbol{H}_{k} = \underbrace{{}^{p}_{def}\boldsymbol{H}_{k}}_{{}^{p}\boldsymbol{H}_{k_{ref}}} \; \underbrace{e^{\left({}^{k}_{b}\widehat{\boldsymbol{\jmath}}_{p,k}\; q_k\right)}}_{{}^{k_{ref}}\boldsymbol{H}_{k}} \; . \tag{2.41}$$

Here, ${}^{k}_{b}\boldsymbol{\jmath}_{p,k}$ denotes the relative body Jacobian between the two links, $e^{\left({}^{k}_{b}\widehat{\boldsymbol{\jmath}}_{p,k}\; q_k\right)}$ is the exponential of the corresponding twist (computed via (2.36) or (2.37)) and q_k is the k-th joint angle. The left brace indicates the equality (2.40). The right brace clarifies that $e^{\left({}^{k}_{b}\widehat{\boldsymbol{\jmath}}_{p,k}\; q_k\right)}$ encodes the relative displacement between the k-th reference frame (as defined in (2.39)) and the current frame, i.e.

$$ {}^{k_{ref}}\boldsymbol{H}_{k} = e^{\left({}^{k}_{b}\widehat{\boldsymbol{\jmath}}_{p,k}\; q_k\right)} \; . \tag{2.42}$$

Concatenating the relative transformations of all involved joints, the transformation of the k-th link w.r.t. the world frame “0” can be expressed as

$$ {}^{0}\boldsymbol{H}_{k} = {}^{0}\boldsymbol{H}_{1} \cdot {}^{1}\boldsymbol{H}_{2} \cdots {}^{k-1}\boldsymbol{H}_{k} = \prod_{i=1}^{k} \left({}^{i-1}\boldsymbol{H}_{i} \right) \; . \tag{2.43}$$

Using the matrix identity ${}^{0}_{def}\boldsymbol{H}_{k}\; e^{\left({}^{k}_{b}\widehat{\boldsymbol{\jmath}}_{p,k}\; q_k\right)}\; {}^{0}_{def}\boldsymbol{H}_{k}^{-1} = e^{\left({}^{0}_{def}\boldsymbol{H}_{k}\, {}^{k}_{b}\widehat{\boldsymbol{\jmath}}_{p,k}\, {}^{0}_{def}\boldsymbol{H}_{k}^{-1}\; q_k\right)} = e^{\left({}^{0}_{def}\widehat{\boldsymbol{\jmath}}_{p,k}\; q_k\right)}$, equation (2.43) can be transformed into the so called *product of exponentials formula*:

$$ {}^{0}\boldsymbol{H}_{k} = e^{\left({}^{0}_{def}\widehat{\boldsymbol{\jmath}}_{0,1}\; q_1\right)} \cdot e^{\left({}^{0}_{def}\widehat{\boldsymbol{\jmath}}_{1,2}\; q_2\right)} \; \ldots \; e^{\left({}^{0}_{def}\widehat{\boldsymbol{\jmath}}_{k-1,k}\; q_k\right)}\; {}^{0}_{def}\boldsymbol{H}_{k} = \prod_{i=1}^{k} \left(e^{\left({}^{0}_{def}\widehat{\boldsymbol{\jmath}}_{i-1,i}\; q_i\right)} \right) {}^{0}_{def}\boldsymbol{H}_{k} \; . \tag{2.44}$$

Note that here the default spatial relative Jacobians ${}^{0}_{def}\widehat{\boldsymbol{\jmath}}_{i-1,i}$ (as introduced in 2.10.1) are used for the i-th link (or joint). This formula provides the direct correlation between the involved joint angles q_i and the global transformation of the k-th link.

2.11 Inertia computations and transformations

The kinematics of a robot, as presented in the previous section, has a major effect on its dynamics (see Sec. 2.12). The second important property that influences the dynamics is the robot’s inertia, i.e. its mass distribution. Using the notation introduced in Sec. 2.1, the inertia of a robot link k represented in its center of mass (denoted by k_{CoM}) can be written as

$$ {}^{k_{CoM}}\boldsymbol{M}_{k} = \begin{bmatrix} m_k\, \boldsymbol{I}_{3\times3} & \boldsymbol{0}_{3\times3} \\ \boldsymbol{0}_{3\times3} & \boldsymbol{I}_{k_{CoM}} \end{bmatrix} \tag{2.45}$$

Here, m_k is the mass of the link and $\boldsymbol{I}_{k_{CoM}}$ is its moment of inertia around its CoM.

Using the adjoint transformation, arbitrary inertias can be transformed via

$$ {}^{j}\boldsymbol{M}_k = \boldsymbol{Ad}^T_{{}^i\boldsymbol{H}_j}\ {}^{i}\boldsymbol{M}_k\ \boldsymbol{Ad}_{{}^i\boldsymbol{H}_j} \qquad \Leftrightarrow \qquad {}^{i}\boldsymbol{M}_k = \boldsymbol{Ad}^{-T}_{{}^i\boldsymbol{H}_j}\ {}^{j}\boldsymbol{M}_k\ \boldsymbol{Ad}^{-1}_{{}^i\boldsymbol{H}_j}\ . \tag{2.46}$$

Here, index k denotes the link, while the other two indices denote the bases that the inertias are represented in. When the link inertia represented in the link frame is required, (2.46) can be applied to (2.45) which yields

$$\begin{aligned} {}^{k}\boldsymbol{M}_k &= \boldsymbol{Ad}^{-T}_{{}^k\boldsymbol{H}_{k_{CoM}}}\ {}^{k_{CoM}}\boldsymbol{M}_k\ \boldsymbol{Ad}^{-1}_{{}^k\boldsymbol{H}_{k_{CoM}}} \\ &= \begin{bmatrix} m_k\,\boldsymbol{I}_{3\times3} & -m_k\,[{}^k\boldsymbol{x}_{k,k_{CoM}}\times] \\ m_k\,[{}^k\boldsymbol{x}_{k,k_{CoM}}\times] & \boldsymbol{I}_{k_{CoM}} - m_k\,[{}^k\boldsymbol{x}_{k,k_{CoM}}\times]\,[{}^k\boldsymbol{x}_{k,k_{CoM}}\times] \end{bmatrix}\ . \end{aligned} \tag{2.47}$$

Here, ${}^k\boldsymbol{H}_{k_{CoM}} = \begin{bmatrix} \boldsymbol{I}_{3\times3} & {}^k\boldsymbol{x}_{k,k_{CoM}} \\ \boldsymbol{0}_{1\times3} & 1 \end{bmatrix}$ and ${}^k\boldsymbol{x}_{k,k_{CoM}}$ denotes the offset of the link CoM from the link frame origin represented in link frame.

The matrix ${}^k\boldsymbol{M}_k$ in (2.47) is the inertia of link k represented in its own frame k, i.e. in body coordinates. To identify what effect the inertia of a single link has along the generalized system coordinates $\boldsymbol{q}$ (i.e. in the direction of the robot joints), it can be mapped onto joint space via

$$ {}^{q}\boldsymbol{M}_k = {}^k_b\boldsymbol{J}^T_{0,k}\ {}^k\boldsymbol{M}_k\ {}^k_b\boldsymbol{J}_{0,k}\ . \tag{2.48}$$

On the other hand, the total robot inertia ${}^q\boldsymbol{M}_{\boldsymbol{q}}$ (represented in joint space, i.e. along the generalized coordinates $\boldsymbol{q}$) can be mapped onto the k-th link space via

$$ {}^{k}\boldsymbol{M}_{\boldsymbol{q}} = ({}^k_b\boldsymbol{J}_{0,k}\ {}^q\boldsymbol{M}^{-1}_{\boldsymbol{q}}\ {}^k_b\boldsymbol{J}^T_{0,k})^{-1}\ . \tag{2.49}$$

Matrix ${}^k\boldsymbol{M}_{\boldsymbol{q}}$ indicates the projected inertia that the joint space inertia ${}^q\boldsymbol{M}^{-1}_{\boldsymbol{q}}$ opposes to an external wrench (expressed in body coordinates) that is applied to the k-th end effector. Note that for the sake of brevity, the joint space inertia matrix ${}^q\boldsymbol{M}_{\boldsymbol{q}}$ is denoted by $\boldsymbol{M}$ in (2.56).

2.12 Derivation of robotic equations of motion

Newton's 2^{nd} law relates the time derivative of the generalized momentum to the generalized force:

$$ \frac{d}{dt}(generalized\ momentum) = generalized\ force\ . \tag{2.50}$$

For a single body k, Newton's 2^{nd} law can be written in body coordinates (see [70]) as

$$ {}^k\boldsymbol{M}_k\,{}^k_b\dot{\boldsymbol{\nu}}_{0,k} + \Big(\underbrace{{}^k\boldsymbol{M}_k\,\boldsymbol{adj}({}^k_b\boldsymbol{\nu}_{0,k}) - \boldsymbol{adj}^T({}^k_b\boldsymbol{\nu}_{0,k})\ {}^k\boldsymbol{M}_k}_{{}^k\boldsymbol{C}_k}\Big)\,{}^k_b\boldsymbol{\nu}_{0,k} = \underbrace{{}^k\boldsymbol{w}_{k,grav} + {}^k\boldsymbol{w}_{k,c} + {}^k\boldsymbol{w}_{k,nc}}_{{}^k\boldsymbol{w}_k} \tag{2.51}$$

Here, the tensor ${}^k\boldsymbol{M}_k \in \mathbb{R}^{6\times6}$ denotes the inertia of the k-th link expressed in its own link coordinates (see (2.47)), ${}^k\boldsymbol{C}_k$ is the body Coriolis matrix, ${}^k_b\boldsymbol{\nu}_{0,k}$ and ${}^k_b\dot{\boldsymbol{\nu}}_{0,k}$ are the body velocity and acceleration of link k relative to the world frame "0" and ${}^k\boldsymbol{w}_k$ denotes the total wrench acting on link k expressed in its own coordinate frame (the "body wrench"). The single elements of ${}^k\boldsymbol{w}_k$ are the gravitational wrench ${}^k\boldsymbol{w}_{k,grav} = {}^k\boldsymbol{M}_k\ \boldsymbol{Ad}^{-1}_{{}^0\boldsymbol{H}_k}\ [0,0,-g,0,0,0]^T$ (g is the gravitational constant), the constraint reaction wrench ${}^k\boldsymbol{w}_{k,c}$ (if constraints are active) and a wrench ${}^k\boldsymbol{w}_{k,nc}$ acting along the non-constrained directions. The terms ${}^k_b\boldsymbol{\nu}_{0,k}$ and ${}^k_b\dot{\boldsymbol{\nu}}_{0,k}$ in (2.51) can be computed as

$$ {}^k_b\boldsymbol{\nu}_{0,k} = {}^k_b\boldsymbol{J}_{0,k}\,\dot{\boldsymbol{q}} \tag{2.52}$$

and

$$ {}^{k}_{b}\dot{\boldsymbol{\nu}}_{0,k} \;=\; {}^{k}_{b}\dot{\boldsymbol{J}}_{0,k}\,\dot{\boldsymbol{q}} + {}^{k}_{b}\boldsymbol{J}_{0,k}\,\ddot{\boldsymbol{q}} \;. \tag{2.53}$$

Here, ${}^{k}_{b}\boldsymbol{J}_{0,k}$ and ${}^{k}_{b}\dot{\boldsymbol{J}}_{0,k}$ denote the origin-related body Jacobian of link k and its time derivative, while $\dot{\boldsymbol{q}}$ and $\ddot{\boldsymbol{q}}$ are the robot's joint velocities and accelerations, respectively. Applying (2.23) the correlation for the body Jacobian ${}^{k}_{b}\boldsymbol{J}_{0,k}$ relative to the world frame "0" can be expressed as

$$ {}^{k}_{b}\boldsymbol{J}_{0,k} \;=\; \boldsymbol{Ad}^{-1}_{{}^{p}\boldsymbol{H}_k}\;{}^{p}_{b}\boldsymbol{J}_{0,p} + \underbrace{{}^{k}_{b}\boldsymbol{J}_{p,k}}_{=\,[\boldsymbol{0}_{6\times1},\;\dots\;,\;{}^{k}_{b}\boldsymbol{j}_{p,k},\;\dots\;,\;\boldsymbol{0}_{6\times1}]} \;. \tag{2.54}$$

It propagates the origin-related body Jacobian ${}^{p}_{b}\boldsymbol{J}_{0,p}$ of the parent link p to the k-th origin-related body Jacobian ${}^{k}_{b}\boldsymbol{J}_{0,k}$. Note: ${}^{k}_{b}\boldsymbol{J}_{0,k}$ serves as origin-related parent body Jacobian for the computation of the next link(s) down the kinematic chain. In the relative body Jacobian ${}^{k}_{b}\boldsymbol{J}_{p,k}$, the vector ${}^{k}_{b}\boldsymbol{j}_{p,k}$ (computed via (2.32)) denotes the "local body Jacobian" between link k and its parent p. It relates the k-th joint velocity $\dot{q}_k$ (the k-th joint connects link k to its parent p) to the relative body velocity between the two links, i.e. ${}^{k}_{b}\boldsymbol{\nu}_{p,k} \;=\; {}^{k}_{b}\boldsymbol{j}_{p,k}\,\dot{q}_k$.

Equivalent to (2.54) the corresponding relative body Jacobian time derivative ${}^{k}_{b}\dot{\boldsymbol{J}}_{0,k}$ is propagated

$$ {}^{k}_{b}\dot{\boldsymbol{J}}_{0,k} \;=\; \boldsymbol{Ad}^{-1}_{{}^{p}\boldsymbol{H}_k}\;{}^{p}_{b}\dot{\boldsymbol{J}}_{0,p} - \boldsymbol{adj}({}^{k}_{b}\boldsymbol{\nu}_{p,k})\;\boldsymbol{Ad}^{-1}_{{}^{p}\boldsymbol{H}_k}\;{}^{p}_{b}\boldsymbol{J}_{0,p} \;. \tag{2.55}$$

Now, the single link equations of motion (2.51) will be transformed into the robot equations of motion. It is well-known that wrenches can be transformed into joint space via the Jacobian transpose. Premultiplying both sides of (2.51) by ${}^{k}_{b}\boldsymbol{J}^{T}_{0,k}$, inserting ${}^{k}_{b}\boldsymbol{\nu}_{0,k}$ and ${}^{k}_{b}\dot{\boldsymbol{\nu}}_{0,k}$ from (2.52) and (2.53) and summing up all equations for $k \in \{1\,..\,n_{links}\}$ (n_{links} being the number of robot links[5]), finally the robot equations of motion are achieved

$$ \begin{aligned} &\underbrace{\sum_k \left({}^{k}_{b}\boldsymbol{J}^{T}_{0,k}\;{}^{k}\boldsymbol{M}_k\;{}^{k}_{b}\boldsymbol{J}_{0,k}\right)}_{\boldsymbol{M}}\ddot{\boldsymbol{q}} + \underbrace{\sum_k \left({}^{k}_{b}\boldsymbol{J}^{T}_{0,k}\;{}^{k}\boldsymbol{C}_k\;{}^{k}_{b}\boldsymbol{J}_{0,k} + {}^{k}_{b}\boldsymbol{J}^{T}_{0,k}\;{}^{k}\boldsymbol{M}_k\;{}^{k}_{b}\dot{\boldsymbol{J}}_{0,k}\right)}_{\boldsymbol{C}}\dot{\boldsymbol{q}} \;- \\ &\quad \underbrace{-\sum_k \left({}^{k}_{b}\boldsymbol{J}^{T}_{0,k}\;{}^{k}\boldsymbol{w}_{k,grav}\right)}_{\boldsymbol{\tau}_{grav}} \;=\; \sum_k \Big(\underbrace{{}^{k}_{b}\boldsymbol{J}^{T}_{0,k}\;{}^{k}\boldsymbol{w}_{k,c}}_{=\,\boldsymbol{0}}\Big) + \underbrace{\sum_k \left({}^{k}_{b}\boldsymbol{J}^{T}_{0,k}\;{}^{k}\boldsymbol{w}_{k,nc}\right)}_{\boldsymbol{\tau}} \;. \end{aligned} \tag{2.56}$$

Here, $\boldsymbol{M}$ and $\boldsymbol{C}$ denote the joint space inertia matrix and Coriolis matrix, respectively. Note that the constraint wrenches ${}^{k}\boldsymbol{w}_{k,c}$ vanish after the mapping with the Jacobian transposes. The generalized torques $\boldsymbol{\tau}$ are the torques along the non-constrained directions of the system, i.e. along the generalized coordinates $\boldsymbol{q}$. For a general free-floating robot with end effectors in contact with the environment, these torques can be composed by

$$ \boldsymbol{\tau} \;=\; \boldsymbol{S}_{act}\,\boldsymbol{\tau}_{act} + \boldsymbol{J}^{T}_{EE}\,\boldsymbol{w}_{EE} \tag{2.57}$$

where $\boldsymbol{S}_{act}$ is a mapping matrix that maps the actuated joint torques $\boldsymbol{\tau}_{act}$ to the generalized torques $\boldsymbol{\tau}$ (for a free-floating robot (with non-actuated base), typically $\boldsymbol{S}_{act} = [\boldsymbol{0}_{6\times6},\,\boldsymbol{I}_{n_{joints}\times n_{joints}}]^T$). The second term maps external end effector wrenches $\boldsymbol{w}_{EE}$ to the generalized torques via the Jacobian transpose $\boldsymbol{J}^{T}_{EE}$. Finally, the equations of motion of a free-floating robot can be written as

$$ \boldsymbol{M}\,\ddot{\boldsymbol{q}} + \boldsymbol{C}\,\dot{\boldsymbol{q}} + \boldsymbol{\tau}_{grav} \;=\; \boldsymbol{S}_{act}\,\boldsymbol{\tau}_{act} + \boldsymbol{J}^{T}_{EE}\,\boldsymbol{w}_{EE} \tag{2.58}$$

The end effector quantities $\boldsymbol{J}_{EE}$ and $\boldsymbol{w}_{EE}$ may be formulated in hybrid, body and spatial notation.

[5]Note: Depending on the chosen formulation, the free-floating DOF may be treated as single "links" as well.

CHAPTER 3

Useful tools for robotics and motion generation

This chapter describes several tools and methods for robotics and motion generation. Some of the methods are consistently used in robotics. To the author's knowledge, the constraint-compatible tracking method and the quaternion interpolation and tracking method, described here, contain a certain degree of novelty and may be of interest also for experienced robotics researchers.

3.1 Polynomial interpolation

A common task in motion generation is to find a trajectory that fulfills certain desired boundary conditions. Typical boundary conditions may include desired position, velocity and acceleration profiles. Polynomials are a class of functions that can satisfy such boundary conditions. Their advantage as compared to many other non-linear function classes is that polynomials and their derivatives can be written as *linear functions of the polynomial parameters*. The problem of boundary condition fulfillment thus is reduced to solving a system of linear equations.

A n-th order polynomial - comprising $n+1$ polynomial parameters and thus appropriate for fulfilling $n+1$ boundary conditions - can be written in the form

$$f(t) = \underbrace{[1, t, t^2, ..., t^n]}_{\boldsymbol{t}^T(t)} \boldsymbol{p} \ , \tag{3.1}$$

where t denotes the evaluation time, $f(t)$ is the function value at that time and $\boldsymbol{p} = [p_1, p_2, ..., p_{n+1}]^T$ is the polynomial parameter vector. The evaluation time vector $\boldsymbol{t}(t)$ combines all required time powers. Similar to (3.1), any d-th order derivative (indicated by the bracketed superscript "(d)") of the polynomial function can be computed as

$$f^{(d)}(t) = \boldsymbol{t}^{(d)T}(t)\, \boldsymbol{p} \ , \tag{3.2}$$

where $\boldsymbol{t}^{(d)T}(t)$ denotes the d-th derivative of $\boldsymbol{t}^T(t)$ w.r.t. time. Now, the $n+1$ boundary conditions can be combined in a single equation:

$$\underbrace{\begin{bmatrix} f^{(d_{c,1})}(t_{c,1}) \\ \vdots \\ f^{(d_{c,n+1})}(t_{c,n+1}) \end{bmatrix}}_{\boldsymbol{f}_c} = \underbrace{\begin{bmatrix} \boldsymbol{t}^{(d_{c,1})T}(t_{c,1}) \\ \vdots \\ \boldsymbol{t}^{(d_{c,n+1})T}(t_{c,n+1}) \end{bmatrix}}_{\boldsymbol{T}_c} \boldsymbol{p} \ . \tag{3.3}$$

Here, $d_{c,i}$ denotes the order of the i-th derivative and $t_{c,i}$ is the time at which the i-th boundary condition has to be fulfilled. The vector $\boldsymbol{f}_c$ combines all boundary conditions, while the matrix $\boldsymbol{T}_c \in \mathbb{R}^{(n+1)\times(n+1)}$ contains all corresponding time mapping vectors. Since $\boldsymbol{T}_c$ is a square matrix and full rank, (3.3) can be solved by inversion

$$\boldsymbol{p} = \boldsymbol{T}_c^{-1}\, \boldsymbol{f}_c \ , \tag{3.4}$$

which provides the solution to the boundary condition problem. Now, for any evaluation time t, the function value and its derivatives can be evaluated using (3.1) and (3.2).

3.2 Mastering P(I)D controllers: pole placement

A common task encountered by control engineers and roboticists is the control of linear (or linearized) systems. Often, for a n-th order system, only the n-th state derivative is controllable. In that case, the closed-loop system dynamics typically looks as follows:

$$\underbrace{\begin{bmatrix} e^{(1)} \\ \vdots \\ e^{(n)} \end{bmatrix}}_{\dot{\boldsymbol{e}}} = \underbrace{\begin{bmatrix} \boldsymbol{0}_{(n-1)\times 1} & \boldsymbol{I}_{(n-1)\times(n-1)} \\ [k_1, & \ldots, \quad k_n] \end{bmatrix}}_{\boldsymbol{A}_{cl}} \underbrace{\begin{bmatrix} e \\ \vdots \\ e^{(n-1)} \end{bmatrix}}_{\boldsymbol{e}} \ , \tag{3.5}$$

where $e^{(i)}$ denotes the i-th derivative of the state error e and $\boldsymbol{A}_{cl}$ is the closed-loop system matrix. The feedback gains $[k_1,\ldots,k_n]$ map the state error vector $\boldsymbol{e}$ to the n-th state error derivative, i.e. to the control input. The characteristic polynomial p_{char} of $\boldsymbol{A}_{cl}$ is computed as

$$p_{char} = det(\lambda\ \boldsymbol{I}_{n\times n} - \boldsymbol{A}_{cl}) \ , \tag{3.6}$$

its roots being the eigenvalues of the closed-loop system. Alternatively, the following formula is used for the n-th order characteristic polynomial:

$$p_{char} = (\lambda - \lambda_1) * \ldots * (\lambda - \lambda_n) \ , \tag{3.7}$$

whose roots obviously lie at λ_1 to λ_n. By expansion of (3.7) and comparison of parameters, the feedback gains $[k_1,\ldots,k_n]$ can be expressed as functions of the eigenvalues λ_1 to λ_n. The design of feedback gains using this technique is called *pole placement* [74]. As compared to manual gain tuning, this tool provides more intuitive access to the closed-loop system dynamics and is thus popular in control engineering. The following table provides the mapping from desired system eigenvalues λ_1 to λ_n to the feedback gains $[k_1,\ldots,k_n]$ of first to third order systems:

	k_1	k_2	k_3
1st order system	λ_1	—	—
2nd order system	$-\lambda_1\,\lambda_2$	$\lambda_1 + \lambda_2$	—
3rd order system	$\lambda_1\,\lambda_2\,\lambda_3$	$-(\lambda_1\,\lambda_2 + \lambda_1\,\lambda_2 + \lambda_1\,\lambda_2)$	$\lambda_1 + \lambda_2 + \lambda_3$

An interesting case are PID controllers, which are typically applied to second order systems. By introducing the error integral as additional system state, the second order system dynamics can be transformed into a third order error dynamics system. The control gains are then assigned as follows: integral gain $k_I \rightarrow k_1$, proportional gain $k_p \rightarrow k_2$, derivative gain $k_d \rightarrow k_3$, i.e. the third row of the table above is applied for corresponding pole placement.

3.3 Constraint-compatible tracking method

PID controllers, as presented in the previous section, are widely used to track reference trajectories. Here, a second order PID controller will be used as example:

$$\ddot{q}_{des} = PID(\Delta q, \Delta \dot{q}) \tag{3.8}$$

In this example, Δq and $\Delta \dot{q}$ stand for joint position and velocity errors, but can be replaced by arbitrary considered quantities (such as Cartesian positions/velocities etc.). This kind of controller returns a desired acceleration $\ddot{q}_{des}$ which typically doesn't consider constraints such as position or velocity limits. Thus, depending on the reference trajectory (e.g. in case of operator errors or bad trajectory design) the controller may exceed both position (e.g. hardware end stops) and velocity (e.g. maximum joint speed) limits.

The motivation of this section is thus to compute accelerations[1] that follow the desired accelerations from (3.8) as far as possible while avoiding constraint violations. It has to be noted that this constraint avoidance should be activated *before* the constraint is violated since otherwise the system dynamics may prohibit a successful constraint avoidance. For example, in case of accelerations as input, the velocity towards a position constraint should already be zero once the constraint is reached, otherwise an overshoot into the constraint is inevitable. To this end, the desired acceleration $\ddot{q}_{des}$ is compared to special reference dynamics (one for each constraint) that, if they are followed, make the system converge towards the constraints asymptotically. These reference dynamics can thus be called *asymptotic constraint attractors* (ACA). For a position limit, such an ACA can have the following form:

$$\ddot{q}_{pos,limit} = -\lambda_{pos,1}\lambda_{pos,2}\ (q - q_{limit}) + (\lambda_{pos,1} + \lambda_{pos,2})\ \dot{q}\ \ , \tag{3.9}$$

which, given the current state $[q, \dot{q}]$, leads to asymptotic convergence to the position limit q_{limit} for negative (and preferably non-imaginary to avoid overshooting) eigenvalues $\lambda_{pos,1}$ and $\lambda_{pos,2}$. Similarly, for a velocity limit, an ACA can be formulated as

$$\ddot{q}_{vel,limit} = \lambda_{vel}\ (\dot{q} - \dot{q}_{limit})\ \ , \tag{3.10}$$

which converges towards the velocity limit $\dot{q}_{limit}$ asymptotically for a negative (and non-imaginary) eigenvalue λ_{vel}. Now, combining all ACA reference dynamics, lower and upper limits for the feasible acceleration $\ddot{q}_{feas}$ can be formulated. The lower limit is simply the maximum of all lower limit ACAs (computed via (3.9) and (3.10) using $q_{limit} = q_{lowerlimit}$ and $\dot{q}_{limit} = \dot{q}_{lowerlimit}$):

$$\ddot{q}_{lowerlimit} = max(\ddot{q}_{pos,lowerlimit}, \ddot{q}_{vel,lowerlimit})\ \ . \tag{3.11}$$

Accordingly, the upper limit is the minimum of all upper limit ACAs (computed via (3.9) and (3.10) using $q_{limit} = q_{upperlimit}$ and $\dot{q}_{limit} = \dot{q}_{upperlimit}$):

$$\ddot{q}_{upperlimit} = min(\ddot{q}_{pos,upperlimit}, \ddot{q}_{vel,upperlimit})\ \ . \tag{3.12}$$

With the lower and upper acceleration limits, the feasible acceleration is finally computed as

$$\ddot{q}_{feas} = max(\ddot{q}_{lowerlimit}, min(\ddot{q}_{upperlimit}, \ddot{q}_{des}))\ \ , \tag{3.13}$$

i.e. the system dynamics is safely embedded into the ACA framework, which usually leaves the original controller untouched (i.e. $\ddot{q}_{feas} = \ddot{q}_{des}$) while converging to the constraint(s) if the original controller is in danger of violating the constraints.

[1]Note: The same method can be applied to higher derivatives as well.

The speed of convergence in (3.9) and (3.10) depends on the choices of $k_{p,pos}$, $k_{d,pos}$ and k_{vel}. Faster gains lead to a wider area in which the original controller (3.8) remains unaffected but require higher accelerations away from the constraint once an ACA is active. Thus, in practice these gains have to be chosen carefully.

An ACA-based safety framework as presented in this section could be implemented as constraints in a quadratic program (QP) based whole-body control framework. In the author's previous research work, the ACA framework was used for a safe joint interpolator, while its use in the presented whole-body control (WBC) framework (see Sec. 4) will be evaluated in the near future.

3.4 Controlling rotations: unit quaternions as powerful representation

Rotations can be represented in multiple ways, for example via Euler angles, angle-axis representation, rotation matrices or (unit) quaternions. In this work, both rotation matrices (since they are part of the homogeneous transformations that the forward kinematics presented in Sec. 2.10.3 works with) and quaternions are used to represent three-dimensional rotations. Rotation matrices have the advantage of being intuitively comprehensible, since their columns are simply the orthogonal basis vectors of the corresponding frame, while on the other side being redundant (9 elements for 3 rotational degrees of freedom). Both Euler and angle-axis representation suffer from (algorithmic, i.e. non-physical) singularities, which this work tries to avoid. With their four elements, quaternions are the most compact representation of three-dimensional rotations that doesn't suffer from singularities. For this reason quaternions are used for rotation reference generation and tracking control. The basic quaternion mathematics is taken from [75], while the presented quaternion trajectory generation and tracking control are the author's contribution.

3.4.1 Theoretical background on (unit) quaternions

This section will give a short overview on the theoretical background of unit quaternions. A quaternion is a four-dimensional vector that can be constructed as

$$\boldsymbol{\zeta} = \begin{bmatrix} \eta \\ \boldsymbol{\epsilon} \end{bmatrix} \quad . \tag{3.14}$$

Here, η denotes the scalar component of the quaternion and $\boldsymbol{\epsilon} = [\varepsilon_1, \varepsilon_2, \varepsilon_3]^T$ is its three-dimensional vector component[2]. Note that throughout this work, wherever not stated differently, when the term “quaternion” is used, actually *unit quaternions* are meant. In contrast to arbitrary quaternions, unit quaternions have a length of 1, i.e. the constraint

$$\boldsymbol{\zeta}^T \boldsymbol{\zeta} \;=\; 1 \tag{3.15}$$

is fulfilled. This equation defines a four-dimensional hypersphere of radius 1. Compared to rotation matrices, quaternions as four-dimensional vectors with scalar and vector elements are less comprehensible. Yet, unit quaternions can be derived from the angle-axis representation via

$$\boldsymbol{\zeta} = \begin{bmatrix} \eta \\ \boldsymbol{\epsilon} \end{bmatrix} = \begin{bmatrix} cos\left(\frac{\alpha_{ang.ax}}{2}\right) \\ sin\left(\frac{\alpha_{ang.ax}}{2}\right) \boldsymbol{u}_{ang.ax} \end{bmatrix} \quad , \tag{3.16}$$

[2]Note: In some textbooks the elements of $\boldsymbol{\epsilon}$ are interpreted as coordinates along three imaginary axes, while in this work vector and matrix calculus is used for all required quaternion operations.

where $\boldsymbol{u}_{ang.ax}$ denotes the unit vector indicating the axis of rotation and $\alpha_{ang.ax}$ is the rotation angle. Equation (3.16) facilitates the intuitive understanding of (unit) quaternions, especially when the rotation vector $\boldsymbol{u}_{ang.ax}$ and angle $\alpha_{ang.ax}$ are derived explicitly.

A useful tool in quaternion calculus is the quaternion mapping matrix defined as

$$\boldsymbol{Z} = \boldsymbol{Z}(\zeta) = \begin{bmatrix} \eta & -\varepsilon_1 & -\varepsilon_2 & -\varepsilon_3 \\ \varepsilon_1 & \eta & -\varepsilon_3 & \varepsilon_2 \\ \varepsilon_2 & \varepsilon_3 & \eta & -\varepsilon_1 \\ \varepsilon_3 & -\varepsilon_2 & \varepsilon_1 & \eta \end{bmatrix} . \tag{3.17}$$

It uses an arbitrary quaternion ζ as input. This quaternion mapping matrix $\boldsymbol{Z}$ is orthogonal and thus its inverse is equal to its transpose, i.e.

$$\boldsymbol{Z}^{-1} = \boldsymbol{Z}^T . \tag{3.18}$$

The quaternion mapping matrix $\boldsymbol{Z}$ allows relative quaternion operations. For example applying (3.17) to ${}^0\zeta_a$, i.e. ${}^0\boldsymbol{Z}_a = \boldsymbol{Z}({}^0\zeta_a)$, the relative quaternion ${}^a\zeta_b$ between a quaternion ${}^0\zeta_a$ and another quaternion ${}^0\zeta_b$ can be computed:

$${}^0\zeta_b = {}^0\boldsymbol{Z}_a\, {}^a\zeta_b \qquad \Leftrightarrow \qquad {}^a\zeta_b = {}^0\boldsymbol{Z}_a^T\, {}^0\zeta_b . \tag{3.19}$$

Note that instead of a simple subtraction (as used for Cartesian vectors) the transpose of the quaternion mapping matrix is used to compute the relative quaternion. The index “0” denotes spatial quantities and, for brevity, is sometimes omitted throughout this thesis. Equation (3.19) is equivalent to the following standard operations applied to corresponding rotation matrices in $\mathbb{R}^3$

$${}^0\boldsymbol{R}_b = {}^0\boldsymbol{R}_a\, {}^a\boldsymbol{R}_b \qquad \Leftrightarrow \qquad {}^a\boldsymbol{R}_b = {}^0\boldsymbol{R}_a^T\, {}^0\boldsymbol{R}_b , \tag{3.20}$$

which can be verified using the following mapping from quaternion to rotation matrix:

$$\boldsymbol{R}(\zeta) = \boldsymbol{R}(\eta, \boldsymbol{\epsilon}) = 2\, (\eta^2 \boldsymbol{I}_{3\times 3} + \boldsymbol{\epsilon}\boldsymbol{\epsilon}^T + \eta\, [\boldsymbol{\epsilon}\times]\,) - \boldsymbol{I}_{3\times 3} . \tag{3.21}$$

Here, $[\boldsymbol{\epsilon}\times]$ denotes the skew-symmetric matrix as defined in (2.8). By closer inspection of (3.21) it becomes obvious that the mapping from quaternions to rotation matrices is not unique: instead, a quaternion ζ correlates to the same rotation matrix as its antipode $-\zeta$, i.e. $\boldsymbol{R}(\zeta) = \boldsymbol{R}(-\zeta)$. This duality problem has to be considered in the design of quaternion trajectories and tracking controllers.

For completeness, the inverse mapping of (3.21), i.e. the mapping of a rotation matrix to a corresponding unit quaternion (its scalar and vector elements), will be presented here:

$$\eta = \frac{1}{2}\, \sqrt{1 + R_{1,1} + R_{2,2} + R_{3,3}} \tag{3.22}$$

$$\boldsymbol{\epsilon} = -\frac{1}{4\eta} \begin{bmatrix} R_{2,3} - R_{3,2} \\ R_{3,1} - R_{1,3} \\ R_{1,2} - R_{2,1} \end{bmatrix} , \tag{3.23}$$

where $R_{r,c}$ denotes the elements in row r and column c of the rotation matrix $\boldsymbol{R}$.

As discussed below (3.19), the computation of relative quaternions or quaternion errors is more complex than for Cartesian quantities. A possible candidate for quaternion error quantification is the arc length, i.e. the length of the shortest path connecting two quaternions on the hypersphere (see Fig. 3.2). For two quaternions ζ_i and ζ_j it is defined as

$$l_{arc} = acos(\zeta_i^T \zeta_j) . \tag{3.24}$$

For $\zeta_i = \zeta_j$, the arc length becomes zero. It may thus serve as error measurement.

3.4.2 Design of smooth trajectories in quaternion space

This section provides a method for the generation of smooth quaternion reference trajectories that may connect an arbitrary number of waypoint quaternions. To this end, first a set of desired quaternion waypoints is preprocessed to yield the shortest possible paths, the waypoints are interpolated via polynomials (yielding non-unit quaternion trajectories) and then the non-unit quaternion trajectories are consistently projected onto the unit quaternion constraint manifold to finally achieve smooth unit quaternion reference trajectories.

Quaternion waypoints

Given N_{quat} arbitrary desired quaternion waypoints $\zeta_{wp,des,i}$ (with $i \in \{1,..,N_{quat}\}$), the first step in the presented quaternion waypoint interpolation is to stepwise swap the desired quaternion waypoints $\zeta_{wp,des,i}$ to their antipodes (that represent the same rotation, see (3.21)), if the antipodes are closer (i.e. shorter arc length, see (3.24) and Fig. 3.1) to the previous quaternion waypoints:

$$\zeta_{wp,i} = \begin{cases} \zeta_{wp,des,i} & \zeta_{wp,i-1}^T \zeta_{wp,des,i} \geq 0 \\ -\zeta_{wp,des,i} & \zeta_{wp,i-1}^T \zeta_{wp,des,i} < 0 \end{cases} \tag{3.25}$$

Starting with $\zeta_{wp,1} = \zeta_{wp,des,1}$, equation (3.25) is iterated forward for $i = 2 \,..\, N_{quat}$. This ensures that each interpolation segment takes the shorter path, i.e. the path from the previous quaternion to the *closer* one of the two antipodal quaternions that represent the same waypoint orientation. In $\mathbb{R}^3$, the longer path (i.e. the one between the previous quaternion and the more distant one of the antipodal quaternion candidates) would correspond to a rotation about the same axis but by an angle of $2\pi - \alpha_{min} \geq \alpha_{min}$ (where $0 \leq \alpha_{min} \leq \pi$ denotes the smallest spatial angle which the rotation may be constructed with), which is undesirable in most cases. Additionally, this shorter path interpolation decreases the non-linear effects of the unit quaternion projections from equations (3.26), (3.28) and (3.30) that are presented in the next sections and ensures that the interpolated trajectories keep away from the origin $[0,0,0,0]^T$ of the hypersphere, which would lead to division by zero in the projections.

Euclidean interpolation

The next step in the quaternion reference trajectory design process is a Euclidean interpolation between the quaternion waypoints $\zeta_{wp,i}$. Any interpolation method (such as B-splines, for example) could be used. In this work, due to the typically limited number of quaternion waypoints, polynomial interpolation as presented in Sec. 3.1 was applied, starting from and ending with zero quaternion velocity and acceleration (at the initial and final quaternion waypoints) and interpolating through the quaternion waypoints without intermediate stops. This process yields a C2-continous (non-unit) quaternion reference trajectory $[\zeta_{ref,NU}, \dot{\zeta}_{ref,NU}, \ddot{\zeta}_{ref,NU}]$, where $\zeta_{ref,NU}$ is the (non-unit) quaternion reference position, $\dot{\zeta}_{ref,NU}$ is the (non-unit) quaternion reference velocity and $\ddot{\zeta}_{ref,NU}$ is the (non-unit) quaternion reference acceleration.

Consistent projection onto unit quaternion manifold

It is clear that, while providing a smooth interpolation between the quaternion waypoints in Euclidean space, the Euclidean interpolation introduced in the previous section (in almost all cases) violates the unit quaternion constraint (3.15) that is required for the rotation representation. Thus, a method to project the Euclidean (non-unit) quaternion reference to the more appropriate unit quaternion space (see Fig. 3.1) is provided.

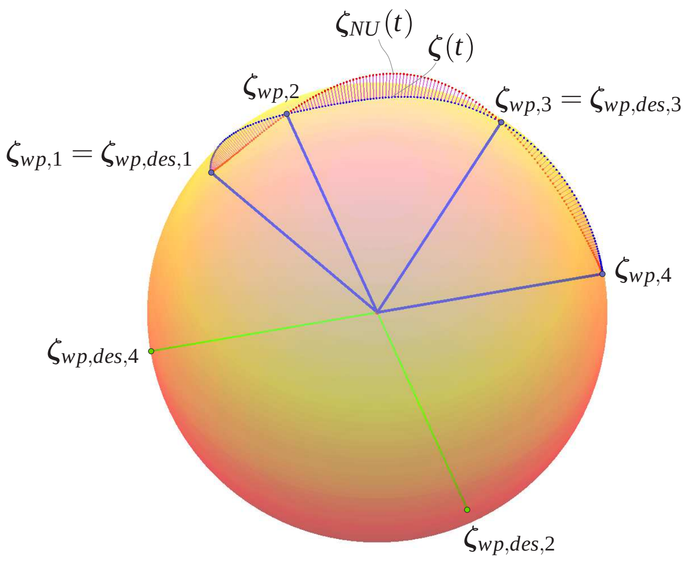

Figure 3.1: Projection of quaternion spline to hypersphere (3D illustration).

The following derivations will show how an arbitrary non-unit quaternion set $[\zeta_{NU}, \dot{\zeta}_{NU}, \ddot{\zeta}_{NU}]$ can be projected to the closest feasible unit quaternion set $[\zeta, \dot{\zeta}, \ddot{\zeta}]$. Unit quaternions have unit length, i.e. the constraint (3.15) holds. It can be shown that

$$\zeta = \frac{\zeta_{NU}}{\sqrt{\zeta_{NU}^T\,\zeta_{NU}}} \tag{3.26}$$

fulfills this constraint, while being closest to the original non-unit quaternion ζ_{NU}. Differentiation of (3.15) yields a constraint for the unit quaternion velocity $\dot{\zeta}$:

$$\zeta^T \dot{\zeta} \;=\; 0 \;. \tag{3.27}$$

This constraint is fulfilled by the quaternion velocity computed via differentiation of (3.26):

$$\dot{\zeta} = (\boldsymbol{I}_{4\times 4} - \zeta\,\zeta^T)\,\frac{\dot{\zeta}_{NU}}{\sqrt{\zeta_{NU}^T\,\zeta_{NU}}} \quad . \tag{3.28}$$

Differentiating (3.27) once more yields a constraint for the unit quaternion acceleration:

$$\zeta^T \ddot{\zeta} \;=\; -\,\dot{\zeta}^T \dot{\zeta} \;. \tag{3.29}$$

This constraint is fulfilled by the quaternion acceleration

$$\ddot{\zeta} = \underbrace{-\,\dot{\zeta}^T\dot{\zeta}\;\zeta}_{\ddot{\zeta}_{radial}} + \underbrace{(\boldsymbol{I}_{4\times 4} - \zeta\,\zeta^T)\,\frac{\ddot{\zeta}_{NU}}{\sqrt{\zeta_{NU}^T\,\zeta_{NU}}}}_{\ddot{\zeta}_{tangent}} \quad , \tag{3.30}$$

which is achieved by differentiating (3.28). Here, $\ddot{\zeta}_{radial}$ can be interpreted as centripetal acceleration that ensures that the quaternion remains on the unit hypersphere, while $\ddot{\zeta}_{tangent}$ is the quaternion acceleration in the hypersphere's tangent space constructed in the current quaternion ζ.

Now, using (3.26), (3.28) and (3.30), the (non-unit) quaternion reference trajectory $[\boldsymbol{\zeta}_{ref,NU}, \dot{\boldsymbol{\zeta}}_{ref,NU}, \ddot{\boldsymbol{\zeta}}_{ref,NU}]$ from the previous section is projected to a unit quaternion reference trajectory $[\boldsymbol{\zeta}_{ref}, \dot{\boldsymbol{\zeta}}_{ref}, \ddot{\boldsymbol{\zeta}}_{ref}]$ that is closest to the original reference trajectory. This unit quaternion reference is smooth (typically C2-continuous, depending on the chosen interpolation method), fulfills the unit quaternion constraints (3.15), (3.27) and (3.29) and is *consistent*.

3.4.3 Tracking of quaternion trajectories

This section presents a method for quaternion tracking control. To this end, first a general quaternion regulation controller is presented which is then applied to the case of relative quaternion regulation to finally yield a quaternion tracking controller.

For the *regulation case* (i.e. $\dot{\boldsymbol{\zeta}}_{ref} = \mathbf{0}$, $\ddot{\boldsymbol{\zeta}}_{ref} = \mathbf{0}$), the following suitable control law was found

$$\ddot{\boldsymbol{\zeta}}_{des} = \underbrace{-\dot{\boldsymbol{\zeta}}^T \dot{\boldsymbol{\zeta}}\, \boldsymbol{\zeta}}_{\text{radial constraint}} \underbrace{+k_{\zeta,p} \overbrace{acos(\boldsymbol{\zeta}_{ref,c}^T \boldsymbol{\zeta})}^{\text{arc length } l_{arc}} \overbrace{\frac{(\boldsymbol{I}_{4\times4} - \boldsymbol{\zeta}\,\boldsymbol{\zeta}^T)\, \boldsymbol{\zeta}_{ref,c}}{\sqrt{\boldsymbol{\zeta}_{ref,c}^T (\boldsymbol{I}_{4\times4} - \boldsymbol{\zeta}\,\boldsymbol{\zeta}^T)\, \boldsymbol{\zeta}_{ref,c} + \lambda^2}}}^{\text{tangential unit vector } \boldsymbol{u}_{tang}} \overbrace{- k_{\zeta,d}\, \dot{\boldsymbol{\zeta}}}^{\text{damping term}}}_{\text{tangential space control}} , \qquad (3.31)$$

which for $\ddot{\boldsymbol{\zeta}} = \ddot{\boldsymbol{\zeta}}_{des}$ makes the the quaternion $\boldsymbol{\zeta}$ converge to the reference quaternion $\boldsymbol{\zeta}_{ref,c}$. The main components of the controller are the radial constraint (as in (3.30)) and the tangential space control, which consists of a proportional feedback and a damping term. The proportional feedback component unreels the arc length l_{arc} between the commanded reference quaternion $\boldsymbol{\zeta}_{ref,c}$ and the current quaternion $\boldsymbol{\zeta}$ onto the tangential space of the hypersphere via the tangential unit vector $\boldsymbol{u}_{tang}$ (see Fig. 3.2). The tangential unit vector $\boldsymbol{u}_{tang}$ points from the current quaternion $\boldsymbol{\zeta}$ towards the commanded reference quaternion $\boldsymbol{\zeta}_{ref,c}$ as good as possible while being constrained to the tangential space. A small damping term $0 < \lambda^2 \ll 1$ is used that keeps the fractional term in (3.31) from exploding in case $\boldsymbol{\zeta} = \boldsymbol{\zeta}_{ref,c}$. Similar to (3.25), the commanded reference quaternion $\boldsymbol{\zeta}_{ref,c}$ is either chosen to be $\boldsymbol{\zeta}_{ref}$ or $-\boldsymbol{\zeta}_{ref}$, depending on which of the two antipodes is closer to $\boldsymbol{\zeta}$:

$$\boldsymbol{\zeta}_{ref,c} = \begin{cases} \boldsymbol{\zeta}_{ref} & \boldsymbol{\zeta}_{ref}^T \boldsymbol{\zeta} \geq 0 \\ -\boldsymbol{\zeta}_{ref} & \boldsymbol{\zeta}_{ref}^T \boldsymbol{\zeta} < 0 \end{cases} \qquad (3.32)$$

Equivalently, the commanded reference quaternion velocity $\dot{\boldsymbol{\zeta}}_{ref,c}$ is chosen to be

$$\dot{\boldsymbol{\zeta}}_{ref,c} = \begin{cases} \dot{\boldsymbol{\zeta}}_{ref} & \boldsymbol{\zeta}_{ref}^T \boldsymbol{\zeta} \geq 0 \\ -\dot{\boldsymbol{\zeta}}_{ref} & \boldsymbol{\zeta}_{ref}^T \boldsymbol{\zeta} < 0 \end{cases} \qquad (3.33)$$

For a positive proportional gain $k_{\zeta,p} > 0$ and damping gain $k_{\zeta,d} > 0$ (pole placement as presented in Sec. 3.2 can be applied alternatively), the regulation control law (3.31) was found to be stable.

The regulation controller (3.31) can also be applied to the regulation of relative quaternions:

$$\ddot{\boldsymbol{\zeta}}_{rel,des} = -\dot{\boldsymbol{\zeta}}_{rel}^T \dot{\boldsymbol{\zeta}}_{rel}\, \boldsymbol{\zeta}_{rel} + k_{\zeta,p} \overbrace{acos(\boldsymbol{\zeta}_{ref,rel}^T \boldsymbol{\zeta}_{rel})}^{\text{rel. arc length } l_{arc,rel}} \overbrace{\frac{(\boldsymbol{I}_{4\times4} - \boldsymbol{\zeta}_{rel}\, \boldsymbol{\zeta}_{rel}^T)\, \boldsymbol{\zeta}_{ref,rel}}{\sqrt{\boldsymbol{\zeta}_{ref,rel}^T (\boldsymbol{I}_{4\times4} - \boldsymbol{\zeta}_{rel}\, \boldsymbol{\zeta}_{rel}^T)\, \boldsymbol{\zeta}_{ref,rel} + \lambda^2}}}^{\text{rel. tangential unit vector } \boldsymbol{u}_{tang}} - k_{\zeta,d}\, \dot{\boldsymbol{\zeta}}_{rel} , \qquad (3.34)$$

Here, $\boldsymbol{\zeta}_{ref,rel} = [1,0,0,0]^T$ is the reference relative quaternion, while $\boldsymbol{\zeta}_{rel}$ denotes the relative quaternion between a commanded reference quaternion $\boldsymbol{\zeta}_{ref,c}$ and the current quaternion $\boldsymbol{\zeta}$, which can be computed as

$$\boldsymbol{\zeta}_{rel} = \boldsymbol{Z}_{ref,c}^T\, \boldsymbol{\zeta} \ . \qquad (3.35)$$

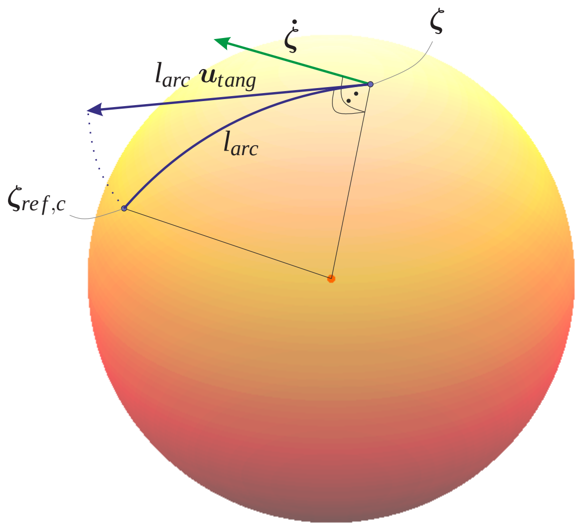

Figure 3.2: Unreeling of arc length for quaternion tracking (3D illustration).

Here, $\boldsymbol{Z}_{ref,c}$ is derived from $\zeta_{ref,c}$ via (3.19). Note that a matrix operation is used instead of a Euclidean difference (which might have the form $\zeta_{rel} = \zeta - \zeta_{ref,c}$) to define the relative quaternion. Also note that for $\zeta = \zeta_{ref,c}$, the relative quaternion becomes $[1,0,0,0]^T$ (instead of $\mathbf{0}_{4\times1}$). This means that by regulating the relative quaternion ζ_{rel} to the reference relative quaternion $\zeta_{ref,rel} = [1,0,0,0]^T$ (or its antipode, respectively), ζ converges to $\zeta_{ref,c}$, i.e. the quaternion tracking error vanishes. Differentiating (3.35), the relative quaternion velocity $\dot{\zeta}_{rel}$ is computed as

$$\dot{\zeta}_{rel} = \dot{\boldsymbol{Z}}^T_{ref,c}\,\zeta + \boldsymbol{Z}^T_{ref,c}\,\dot{\zeta} \ . \tag{3.36}$$

Differentiating (3.36) once more yields the relative quaternion acceleration

$$\ddot{\zeta}_{rel} = \ddot{\boldsymbol{Z}}^T_{ref,c}\,\zeta + 2\,\dot{\boldsymbol{Z}}^T_{ref,c}\,\dot{\zeta} + \boldsymbol{Z}^T_{ref,c}\,\ddot{\zeta} \ . \tag{3.37}$$

Solving (3.37) for $\ddot{\zeta}$, setting $\ddot{\zeta} = \ddot{\zeta}_{des}$ and using the desired relative quaternion acceleration from (3.34) (i.e. setting $\ddot{\zeta}_{rel} = \ddot{\zeta}_{rel,des}$), the desired quaternion acceleration is found as

$$\ddot{\zeta}_{des} = \boldsymbol{Z}_{ref,c}\,(\ddot{\zeta}_{rel,des} - \ddot{\boldsymbol{Z}}^T_{ref,c}\,\zeta - 2\,\dot{\boldsymbol{Z}}^T_{ref,c}\,\dot{\zeta}) \ . \tag{3.38}$$

Finally, the combination of (3.34) and (3.38) forms the quaternion tracking control which asymptotically tracks the quaternion reference trajectory $[\zeta_{ref}, \dot{\zeta}_{ref}, \ddot{\zeta}_{ref}]$. In combination with the planning method presented in the previous sections, this control provides a powerful tool for multi-waypoint quaternion reference trajectory generation and tracking.

3.4.4 Transformations between quaternion and angular space

Orientation control is typically performed in angular space, i.e. using angular velocities and accelerations. Therefore, the quaternion velocities and accelerations derived in this section need to be transformed into angular space. Given a quaternion velocity $\dot{\zeta}$, the corresponding angular velocity $\boldsymbol{\omega}$ can be computed as

$$\boldsymbol{\omega} \;=\; 4\,\boldsymbol{E}^T\dot{\zeta} \ . \tag{3.39}$$

The matrix $\boldsymbol{E}$ is defined as

$$\boldsymbol{E} \;=\; -\frac{1}{2}\begin{bmatrix} \boldsymbol{\epsilon}^T \\ [\boldsymbol{\epsilon}\times] - \eta\,\boldsymbol{I}_{3\times3} \end{bmatrix} \in \mathbb{R}^{4\times3} \ . \tag{3.40}$$

Note that $\boldsymbol{E}$ is not square and thus cannot be inverted directly. The inverse operation of (3.39) - i.e. the mapping of an angular velocity $\boldsymbol{\omega}$ to the corresponding quaternion velocity $\dot{\boldsymbol{\zeta}}$ - is

$$\dot{\boldsymbol{\zeta}} = \boldsymbol{E}\,\boldsymbol{\omega} \ . \tag{3.41}$$

Differentiating (3.39), quaternion accelerations $\ddot{\boldsymbol{\zeta}}$ can be mapped to angular accelerations $\dot{\boldsymbol{\omega}}$:

$$\dot{\boldsymbol{\omega}} = 4\,(\boldsymbol{E}^T\ddot{\boldsymbol{\zeta}} + \dot{\boldsymbol{E}}^T\dot{\boldsymbol{\zeta}}) = 4\,\boldsymbol{E}^T(\ddot{\boldsymbol{\zeta}} - \dot{\boldsymbol{E}}\boldsymbol{\omega}) \ , \tag{3.42}$$

where $4\boldsymbol{E}^T\boldsymbol{E} = \boldsymbol{I}_{3\times3}$ and $\dot{\boldsymbol{E}}^T = -4\,\boldsymbol{E}^T\dot{\boldsymbol{E}}\boldsymbol{E}^T$ was used. The derivative of (3.40) is

$$\dot{\boldsymbol{E}} = -\frac{1}{2}\begin{bmatrix} \dot{\boldsymbol{\epsilon}}^T \\ [\dot{\boldsymbol{\epsilon}}\times] - \dot{\eta}\,\boldsymbol{I}_{3\times3} \end{bmatrix} \in \mathbb{R}^{4\times3} \ . \tag{3.43}$$

The inverse mapping of (3.40) can be computed via differentiation of (3.41) as

$$\ddot{\boldsymbol{\zeta}} = \dot{\boldsymbol{E}}\,\boldsymbol{\omega} + \boldsymbol{E}\,\dot{\boldsymbol{\omega}} = 4\,\dot{\boldsymbol{E}}\boldsymbol{E}^T\dot{\boldsymbol{\zeta}} + \boldsymbol{E}\,\dot{\boldsymbol{\omega}} \ . \tag{3.44}$$

3.5 Task and null spaces

This section gives a short overview of different types of pseudoinverses and null space projectors, which can be used to find control inputs that satisfy given tasks as good as possible.

In many cases, a linear (or linearized) correlation

$$\boldsymbol{d}_{task} = \boldsymbol{D}_{task}\,\boldsymbol{u} \tag{3.45}$$

can be found, which maps the input vector $\boldsymbol{u} \in \mathbb{R}^{n\times1}$ to the task vector $\boldsymbol{d}_{task} \in \mathbb{R}^{m\times1}$ via the task space mapping matrix $\boldsymbol{D}_{task} \in \mathbb{R}^{m\times n}$, i.e. there are n control inputs available for m tasks. Now, the aim is to find an input vector that either perfectly fulfills the desired tasks, i.e.

$$\boldsymbol{d}_{task} = \boldsymbol{d}_{task,des} \tag{3.46}$$

or minimizes a corresponding cost function. Here, three different cases have to be distinguished: $m = n$, $m > n$ and $m < n$. These cases are examined in the following sections.

3.5.1 Fully determined case

The task mapping (3.45) is fully determined if the number of tasks equals the number of control inputs available (i.e. $m = n$) and the robot is in a non-singular configuration. In that case, $\boldsymbol{D}_{task}$ is square and, if $\boldsymbol{D}_{task}$ is full rank, (3.45) can be solved via inversion

$$\boldsymbol{u}^* = \boldsymbol{D}_{task}^{-1}\,\boldsymbol{d}_{task,des} \ , \tag{3.47}$$

which fully determines all available control inputs and fulfills the desired tasks (3.46).

3.5.2 Over-determined case

If the number of tasks is bigger than the number of control inputs available (i.e. $m > n$), the task mapping (3.45) is over-determined, i.e. there are not enough control inputs available to completely fulfill all tasks at hand. One possibility to handle this problem is to split the over-determined task into several under-determined tasks (as presented in the next section) to build up a hierarchical control policy that ensures that higher-priority tasks are perfectly fulfilled, while lower-priority tasks may be violated. Another possible approach is to define a cost function that leads to a good trade-off between the subtasks. Such a scalar cost function may have the following form:

$$G = \frac{1}{2}\ \boldsymbol{\Delta d}^T\ \boldsymbol{W}_{task}\ \boldsymbol{\Delta d} + \frac{1}{2}\ \boldsymbol{u}^T \boldsymbol{\Lambda}\ \boldsymbol{u}\ . \tag{3.48}$$

Here, $\boldsymbol{\Delta d} = \boldsymbol{d}_{task} - \boldsymbol{d}_{task,des} = \boldsymbol{D}_{task}\ \boldsymbol{u} - \boldsymbol{d}_{task,des}$, $\boldsymbol{W}_{task} \in \mathbb{R}^{m\times m}$ is the (typically diagonal) task weighting matrix and $\boldsymbol{\Lambda}$ is a damping matrix that minimizes the input vector $\boldsymbol{u}$. By differentiating (3.48) by $\boldsymbol{u}$ and setting the result to zero, the optimal input vector $\boldsymbol{u}^*$ is found as

$$\boldsymbol{u}^* = \underbrace{\underbrace{(\boldsymbol{D}_{task}^T\, W_{task}\ \boldsymbol{D}_{task} + \boldsymbol{\Lambda})}_{\boldsymbol{\Theta}}{}^{-1} \boldsymbol{D}_{task}^T \boldsymbol{W}_{task}}_{\boldsymbol{D}_{task}^{\#}}\ \ \boldsymbol{d}_{task,des}\ , \tag{3.49}$$

which minimizes G and thus optimally trades off the desired tasks. This optimal solution is highly dependent on the chosen weighting matrix $\boldsymbol{W}_{task}$. The pseudoinverse $\boldsymbol{D}_{task}^{\#} \in \mathbb{R}^{n\times m}$ maps the desired task vector $\boldsymbol{d}_{task,des}$ to the optimal control input vector. Due to the minimization of the input vector $\boldsymbol{u}$ via the (typically positive definite and diagonal) matrix $\boldsymbol{\Lambda}$ in (3.48) and the corresponding element in $\boldsymbol{\Theta}$, the latter is guaranteed to be invertible, even when some of the tasks become singular. The corresponding solution from (3.49) behaves well, such that $\boldsymbol{D}_{task}^{\#}$ (including the damping term $\boldsymbol{\Lambda}$) is also called damped pseudoinverse.

3.5.3 Under-determined case

If the number of tasks is smaller than the number of control inputs available (i.e. $m < n$), the task mapping (3.45) is under-determined. This means that all original tasks can be fulfilled, while in their null space additional DOF remain available for other tasks. The solution for that case is

$$\boldsymbol{u}^* = \boldsymbol{D}_{task}^{+}\, \boldsymbol{d}_{task,des} + \boldsymbol{N}_D\, \tilde{\boldsymbol{u}}\ . \tag{3.50}$$

Here, $\boldsymbol{D}_{task}^{+} \in \mathbb{R}^{n\times m}$ is the pseudoinverse of $\boldsymbol{D}_{task}$ which is defined as

$$\boldsymbol{D}_{task}^{+} = \boldsymbol{D}_{task}^{T}(\boldsymbol{D}_{task}\boldsymbol{D}_{task}^{T})^{-1} \tag{3.51}$$

To distinguish this pseudoinverse from the one in (3.49), the superscript "+" is used instead of "#". For the inversion to work, $\boldsymbol{D}_{task}$ needs to be of full row rank.

The matrix $\boldsymbol{N}_D \in \mathbb{R}^{n\times n}$ in (3.50) is the corresponding null space mapping:

$$\boldsymbol{N}_D = (\boldsymbol{I}_{n\times n} - \boldsymbol{D}_{task}^{+}\boldsymbol{D}_{task})\ . \tag{3.52}$$

Inserting (3.50) in (3.45) fulfills the desired correlation (3.46) since $\boldsymbol{D}_{task}\boldsymbol{D}_{task}^{+} = \boldsymbol{I}_{m\times m}$ and $\boldsymbol{D}_{task}\boldsymbol{N}_D = \boldsymbol{0}_{m\times n}$. The vector $\tilde{\boldsymbol{u}} \in \mathbb{R}^{n\times 1}$ in (3.50) can be used to pursue additional tasks. This vector, just like $\boldsymbol{u}$, is n-dimensional, yet via the projection through $\boldsymbol{N}_D$, m DOF become unavailable for the additional tasks. The redundancy in the vector $\tilde{\boldsymbol{u}}$ can be undesirable. Thus, the next section will present a non-redundant solution to this problem.

3.5.4 Reduced null space matrix

Several works such as [76] and [77] present approaches that avoid the redundancy problem mentioned in the previous section. To this end, similar to the correlation $\boldsymbol{D}_{task}\boldsymbol{N}_D = \boldsymbol{0}_{m\times n}$, a reduced null space matrix $\boldsymbol{N}_{D,red} \in \mathbb{R}^{n\times(n-m)}$ is designed such that $\boldsymbol{D}_{task}\,\boldsymbol{N}_{D,red} = \boldsymbol{0}_{m\times(n-m)}$, i.e.

$$\underbrace{\left[\boldsymbol{D}_{task,square},\,\boldsymbol{D}_{task,rem}\right]}_{\boldsymbol{D}_{task}}\underbrace{\begin{bmatrix}\boldsymbol{N}_{D,red,upper}\\ \boldsymbol{I}_{(n-m)\times(n-m)}\end{bmatrix}}_{\boldsymbol{N}_{D,red}} = \boldsymbol{0}_{m\times(n-m)} = \boldsymbol{D}_{task,square}\,\boldsymbol{N}_{D,red,upper} + \boldsymbol{D}_{task,rem} \;. \tag{3.53}$$

Here, $\boldsymbol{D}_{task,square} \in \mathbb{R}^{m\times m}$ is the square submatrix combining the first m rows and columns of $\boldsymbol{D}_{task}$, and $\boldsymbol{D}_{task,rem} \in \mathbb{R}^{m\times(n-m)}$ combines the remaining columns. The appropriate upper matrix $\boldsymbol{N}_{D,red,upper} \in \mathbb{R}^{m\times(n-m)}$ is found via inversion of $\boldsymbol{D}_{task,square}$ such that $\boldsymbol{N}_{D,red}$ becomes

$$\boldsymbol{N}_{D,red} = \begin{bmatrix}-\boldsymbol{D}_{task,square}^{-1}\,\boldsymbol{D}_{task,rem}\\ \boldsymbol{I}_{(n-m)\times(n-m)}\end{bmatrix} \;. \tag{3.54}$$

Obviously, this only works if $\boldsymbol{D}_{task,square}$ is invertible. Otherwise, a collection of m independent columns of $\boldsymbol{D}_{task}$ may be used. The advantage of $\boldsymbol{N}_{D,red}$ over the original null space mapping matrix $\boldsymbol{N}_D$ is that it has only $(n-m)$ columns. It matches the dimensions of the $(n-m)$-dimensional input vector $\tilde{\boldsymbol{u}}_{red}$ that encodes the whole remaining null space such that (3.50) can replaced by

$$\boldsymbol{u}^* = \boldsymbol{D}_{task}^{+}\,\boldsymbol{d}_{task,des} + \boldsymbol{N}_{D,red}\,\tilde{\boldsymbol{u}}_{red} \;. \tag{3.55}$$

This way, the redundancy problem mentioned above is solved. Due to the lower dimensionality of the reduced null space vector $\tilde{\boldsymbol{u}}_{red}$ the complexity of derivations may be reduced.

CHAPTER 4

Whole-body control framework

The previous chapters provided some basic mathematical introduction to robotics and a collection of useful tools for robotics and motion design. Many of the methods presented in this work are based on these tools. The previous chapters also form the basis for the whole-body control framework presented in this section.

Humanoid robots can be expected to take over a multitude of tasks in the near future. Their typically high number of joints and thus high number of degrees of freedom (DOF) have the potential of fulfilling a multitude of tasks. Often, the number of desired tasks is even higher than the number of available DOF. For a legged humanoid robot with its free-floating base, some of the most important tasks are balancing and locomotion. Often, these tasks are carried out by the legs of the robot, while the arms are mostly dedicated to manipulation tasks. Yet, situations where the arms are used for balance (multi-contact) or a leg is used for manipulation (e.g. when kicking a ball) are not uncommon either. The main challenge in whole-body control (WBC) is to find the optimal control outputs, which best fulfill the desired tasks (e.g. contact force control) while fulfilling the given constraints.

The particular choice of the control architecture has a great effect on the achievable robot performance. Prominent examples for whole-body control architectures include Virtual Model Control [78, 79], passivity-based whole-body controllers [61–65] and inverse dynamics base approaches such as presented in [56, 80–83]. Most approaches work with local optimizers (optimizing for the current time step only) with preview typically limited to CoM dynamics, while whole-body model predictive control approaches, as presented in [84], due to the highly increased complexity, are still in their beginnings. This work uses an inverse dynamics architecture which, as in many other recent works, is embedded into a quadratic program (QP) based optimization framework. In this work, the actuated joint torques and linear contact forces are used as *optimization variables* (or control inputs) and the optimization problem is formulated as a quadratic program (QP). This chapter presents the used optimization framework.

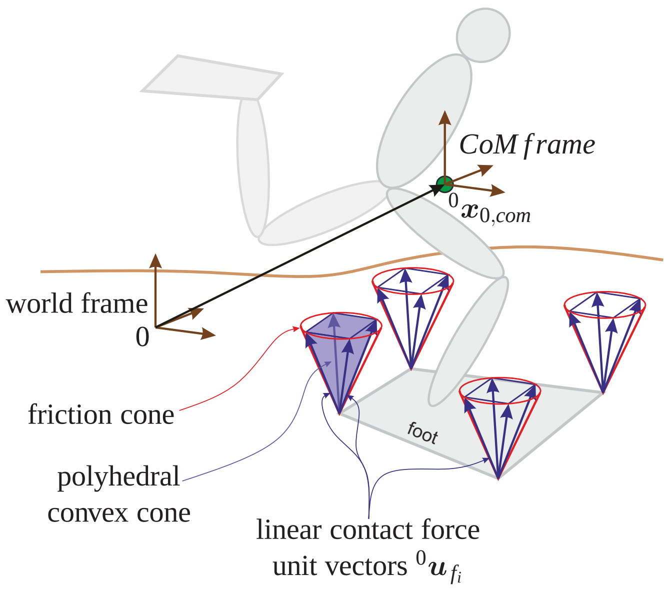
CoM frame
$^{0}\boldsymbol{x}_{0,com}$
world frame
0
friction cone
polyhedral convex cone
foot
linear contact force unit vectors $^{0}\boldsymbol{u}_{f_i}$

Figure 4.1: Outline of CoM frame and polyhedral convex cone approximation.

4.1 Contact representation and adapted equations of motion

One of the challenges in the control of humanoid or other free-floating[1] robots is how the different contact scenarios and the corresponding feasible wrenches can be encoded. One basic idea is to approximate continuous contact surfaces (such as a robot foot) via contact points and make assumptions about the feasible contact forces that can be applied at these contact points. For a robot foot, the constraints for the feasible contact forces in a contact point are typically formulated as friction cone constraints: only forces that lie within this friction cone (centered around the ground normal at the contact point) are feasible. This cone representation is non-linear. Although there are solvers that can handle conic constraints (as for example presented by Wensing et al. [60]), often (e.g. in [80–83]) these constraints are approximated via polyhedral convex cones (see Fig. 4.1) that span the original friction cone as best as possible, thereby turning the constraints into linear ones and thus simplifying the problem. In this work, this approximation via polyhedral convex cones is used.

In the polyhedral convex cone framework, the feasible contact wrenches are encoded via linear contact forces $\boldsymbol{f}_i \in \mathbb{R}^{3\times1}$ that are composed of a unit vector $^{0}\boldsymbol{u}_{f_i}$ indicating the force direction in space and a scalar parameter ρ_i denoting the contact force magnitude, i.e.

$$\boldsymbol{f}_i = {}^{0}\boldsymbol{u}_{f_i}\ \rho_i \ . \tag{4.1}$$

That way, all friction cones considered for a given contact scenario are represented by a total number of n_ρ linear contact forces. The feasible directions $^{0}\boldsymbol{u}_{f_i}$ are typically determined by the given contact configuration. The contact force magnitudes are typically (at least when friction cones are approximated) constraint to be positive, i.e. $\rho_i \geq 0$. The contact force magnitude vector

[1]The term “free-floating” is not necessarily related to space robots but to robots whose base link is not rigidly connected to the environment.

$\boldsymbol{\rho}$ combines all n_ρ contact force magnitudes as

$$\boldsymbol{\rho} = \begin{bmatrix} \rho_1 \\ \vdots \\ \rho_{n_\rho} \end{bmatrix} . \tag{4.2}$$

The contact force magnitudes $\boldsymbol{\rho}$ can be mapped to six-dimensional spatial (i.e. represented in world frame “0”) wrenches ${}^0\boldsymbol{w}_{EE_k}$ via

$${}^0\boldsymbol{w}_{EE_k} = {}^0\boldsymbol{A}_{\rho,k}\,\boldsymbol{\rho} , \tag{4.3}$$

where k is the index of end effector EE_k. Each ${}^0\boldsymbol{A}_{\rho,k}$ denotes a special adjoint matrix that maps the vector of contact force magnitudes $\boldsymbol{\rho}$ to the k-th spatial wrench. If the i-th ($i \in 1..n_\rho$) contact force is associated with the k-th end effector, the corresponding column in ${}^0\boldsymbol{A}_{\rho,k}$ is

$${}^0\boldsymbol{a}_{\rho,k} = \begin{bmatrix} {}^0\boldsymbol{u}_{f_i} \\ [{}^0\boldsymbol{x}_{0,c_i}\times]\,{}^0\boldsymbol{u}_{f_i} \end{bmatrix} . \tag{4.4}$$

Otherwise, the corresponding column is ${}^0\boldsymbol{a}_{\rho,k} = \boldsymbol{0}_{6\times1}$. Here, ${}^0\boldsymbol{u}_{f_i}$ is the contact force direction vector as used in (4.1), while ${}^0\boldsymbol{x}_{0,c_i}$ denotes the spatial position of the contact point c_i that the contact force $\boldsymbol{f}_i$ is associated with. In this work, four contact points in the corners of each foot and four contact forces at each contact point are used that approximate the corresponding friction cone. This leads to a total of $n_\rho = 16$ contact forces per end effector/foot such that (4.3) involves a 10-dimensional null space in the mapping from contact force magnitudes $\boldsymbol{\rho}$ to foot wrenches. The control input regularization as presented in Sec. 4.2.5 assures stability of that null space.

It has to be noted that the contact forces $\boldsymbol{f}_i$ are only assured to actually be feasible if the estimated friction coefficients and ground normals are estimated correctly and if the assumption that a certain contact point is in contact at all is correct. Otherwise the computed contact forces/wrenches are invalid which may lead to major stability issues.

Using the contact wrench encoding (4.3), the general robot equations of motion (2.58) become

$$\begin{aligned} \boldsymbol{M}\ddot{\boldsymbol{q}} + \underbrace{\boldsymbol{C}\dot{\boldsymbol{q}} + \boldsymbol{\tau}_{grav}}_{\boldsymbol{n}} &= \boldsymbol{S}_{act}\boldsymbol{\tau}_{act} + \sum_{k=1}^{n_{EE}} ({}^0_s\boldsymbol{J}^T_{0,EE_k}\,\underbrace{{}^0\boldsymbol{A}_{\rho,k}\,\boldsymbol{\rho}}_{{}^0\boldsymbol{w}_{EE_k}}) \\ &= \underbrace{\begin{bmatrix} \boldsymbol{S}_{act}, & \sum_{k=1}^{n_{EE}} ({}^0_s\boldsymbol{J}^T_{0,EE_k}\,{}^0\boldsymbol{A}_{\rho,k}) \end{bmatrix}}_{{}^\tau\boldsymbol{A}_u} \underbrace{\begin{bmatrix} \boldsymbol{\tau}_{act} \\ \boldsymbol{\rho} \end{bmatrix}}_{\boldsymbol{u}} . \end{aligned} \tag{4.5}$$

Here, n_{EE} denotes the number of considered end effectors and ${}^0_s\boldsymbol{J}_{0,EE_k}$ is the corresponding spatial end effector Jacobian that corresponds to the k-th spatial end effector wrench ${}^0\boldsymbol{w}_{EE_k}$. In this work, the actuated joint torques $\boldsymbol{\tau}_{act}$ and the contact force magnitudes $\boldsymbol{\rho}$ are chosen as *control inputs*. They are stacked in the control input vector $\boldsymbol{u}$ as

$$\boldsymbol{u} = \begin{bmatrix} \boldsymbol{\tau}_{act} \\ \boldsymbol{\rho} \end{bmatrix} . \tag{4.6}$$

The matrix ${}^\tau\boldsymbol{A}_u$ in (4.5) combines both the actuated torque mapping matrix $\boldsymbol{S}_{act}$ and the contact force mapping matrices into a single operator that maps the input vector $\boldsymbol{u}$ to generalized joint torques. In the subsequent task derivations, often the correlation between control inputs $\boldsymbol{u}$ and generalized joint accelerations $\ddot{\boldsymbol{q}}$ is required. It can be derived from (4.5) via inversion of $\boldsymbol{M}$:

$$\ddot{\boldsymbol{q}} = \underbrace{-\boldsymbol{M}^{-1}\boldsymbol{n}}_{\ddot{\boldsymbol{q}}_{MB}} + \underbrace{\boldsymbol{M}^{-1}\ {}^{\tau}\!\boldsymbol{A}_u}_{\boldsymbol{Q}}\ \boldsymbol{u} \tag{4.7}$$

Here, $\boldsymbol{Q}$ maps the control inputs $\boldsymbol{u}$ to generalized accelerations $\ddot{\boldsymbol{q}}$, while $\ddot{\boldsymbol{q}}_{MB}$ are the accelerations due to multi-body effects, such as gravitational and Coriolis accelerations.

4.2 Tasks used for whole-body control in this work

Whole-body control of humanoid robots is an exciting field of research which comprises challenges regarding stability, robustness and safety, to name but a few. A humanoid robot has the physical ability to perform a multitude of useful tasks simultaneously. The issues of stability and robustness have to be tackled with highest priority. Once these rather physics-related tasks are solved, humanoid robots can use their additional available DOF to perform actually useful tasks. This work focuses on robot stability. Most of the tasks described in the following subsections are at least to some extent related to the stability of the robot. All whole-body control tasks will be formulated using the control input vector $\boldsymbol{u} = \begin{bmatrix}\boldsymbol{\tau}_{act} \\ \boldsymbol{\rho}\end{bmatrix}$ in the following form:

$$\boldsymbol{d}_{i,des} = \boldsymbol{D}_i\,\boldsymbol{u}\ . \tag{4.8}$$

Here, $\boldsymbol{d}_{i,des}$ denotes the desired task space vector of the i-th task and $\boldsymbol{D}_i$ is the corresponding task mapping matrix.

4.2.1 Balance and locomotion: linear and angular momentum control

As mentioned above, balance is one of the most critical issues in the control of humanoid robots, which due to their underactuation is a difficult control problem. Locomotion can be seen as dynamic balance which makes proper trajectory generation and feedback control even more important as compared to static balancing. The centroidal dynamics [85] of a robot (including linear and angular momentum dynamics) is strongly connected to its balance. This motivates the derivation of momentum-based control strategies in this work (see sections 5 and 6). The two following subsections present how these tasks are preprocessed for the whole-body control framework.

Embedding linear CoM dynamics for walking and running

Newton's second law of motion provides a fundamental correlation for the CoM dynamics that can be applied to any multi-body system such as a humanoid robot. It is important to note that the CoM dynamics is a *reduced* model rather than a *simplified* model, since it captures the most prominent part of the overall robot dynamics while not making any simplifications or introducing errors. In sections 5 and 6, two frameworks for humanoid walking and running control will be introduced that are based on CoM dynamics. For walking, the corresponding desired linear external force on the CoM $\boldsymbol{F}_{ext,des}$ from (5.63) is used while for running (6.45) is used. The use of force control for the linear CoM balance and locomotion task is directly compatible with the torque-based control framework presented in this work, since the contact force magnitudes $\boldsymbol{\rho}$ can be directly mapped to the desired external forces (see (4.11)).

Regulation of angular momentum

The angular momentum $\boldsymbol{l}$ of a robot can be computed from its generalized joint velocities via

$$\boldsymbol{l} = \boldsymbol{A}_{G,l} \ \dot{\boldsymbol{q}} \ . \tag{4.9}$$

Here, $\boldsymbol{A}_{G,l}$ denotes the part of the centroidal momentum matrix $\boldsymbol{A}_G$, introduced by Orin et al. [85], that relates to the angular momentum $\boldsymbol{l}$. If a reference trajectory for the angular momentum of the robot $[\boldsymbol{l}_{ref}, \dot{\boldsymbol{l}}_{ref}]$ is given, it can be tracked via the following controller

$$\underbrace{\dot{\boldsymbol{l}}_{des}}_{\boldsymbol{\tau}_{com,des}} = \dot{\boldsymbol{l}}_{ref} - k_{h,l}\ (\boldsymbol{l} - \boldsymbol{l}_{ref}) \ , \tag{4.10}$$

which is stable for $k_{h,l} > 0$. In this work, both the reference angular momentum $\boldsymbol{l}_{ref}$ and its rate of change $\dot{\boldsymbol{l}}_{ref}$ are chosen to be zero ($\rightarrow$ *regulation case*), which, for moderate walking and bipedal running, yields satisfactory results. Yet, for other forms of locomotion such as galloping or faster locomotion (e.g. high speed running) a non-zero angular momentum reference seems to be highly desirable or even necessary. The interplay of the angular momentum task with the overall posture task (see Sec 4.2.4) is a particularly important, yet challenging issue. In many cases these two tasks seem to compete and thus trade-offs have to be accepted. The design of appropriate angular momentum references (as for example presented by Herzog et al. [86]) that fit well the requirements of a particular locomotion form while being well feasible regarding the robot hardware limitations and other tasks at hand is a challenge that has to be investigated further. In contrast to offline optimizations, online generation of such angular momentum references would support arbitrary and sudden changes in the locomotion patterns and thus facilitate more agile and versatile humanoid locomotion. Methods for online generation of nominal references that reconcile both tasks, to the author's knowledge, do not exist so far. Such considerations will be part of future research.

Embedding the linear and angular momentum task into whole-body control

The desired linear external force on the CoM $\boldsymbol{F}_{ext,des}$ from (5.63) or (6.45) and the desired torque around the CoM $\boldsymbol{\tau}_{com,des}$ from (4.10) are related to the chosen optimization variables as follows

$$\underbrace{\begin{bmatrix} \boldsymbol{F}_{ext,des} \\ \boldsymbol{\tau}_{com,des} \end{bmatrix}}_{\boldsymbol{d}_{com,des}} = \underbrace{\left[\boldsymbol{0}_{6\times n_{act}},\ {}^{com}\boldsymbol{A}_{\rho}\right]}_{\boldsymbol{D}_{com}} \ \boldsymbol{u} \ . \tag{4.11}$$

Here, n_{act} denotes the number of actuated joints. The matrix ${}^{com}\boldsymbol{A}_{\rho}$ can be computed as

$${}^{com}\boldsymbol{A}_{\rho} \ = \ \boldsymbol{Ad}^T_{{}^0\boldsymbol{H}_{com}} \sum_{k=1}^{n_{EE}} ({}^0\boldsymbol{A}_{\rho,k}) \ . \tag{4.12}$$

It maps the linear contact forces $\boldsymbol{\rho}$ to a combined external wrench in the CoM frame: ${}^{com}\boldsymbol{w}_{ext} = {}^{com}\boldsymbol{w}_{\rho} = {}^{com}\boldsymbol{A}_{\rho}\ \boldsymbol{\rho}$. The matrices ${}^0\boldsymbol{A}_{\rho,k}$ map the contact forces $\boldsymbol{\rho}$ to spatial wrenches via (4.3). The adjoint transpose $\boldsymbol{Ad}^T_{{}^0\boldsymbol{H}_{com}}$ (computed via (2.16)) acts as transformer from world frame (index "0") to CoM frame. Here, the CoM frame ${}^0\boldsymbol{H}_{com}$ (see Fig. 4.1) is defined as:

$${}^0\boldsymbol{H}_{com} \ = \ \begin{bmatrix} \boldsymbol{I}_{3\times 3} & {}^0\boldsymbol{x}_{0,com} \\ \boldsymbol{0}_{1\times 3} & 1 \end{bmatrix} \ , \tag{4.13}$$

where ${}^{0}\boldsymbol{x}_{0,com}$ is the spatial position of the CoM. The corresponding rotation matrix is a unit matrix, i.e. the CoM frame is not rotated w.r.t. the world frame "0". The centroidal momentum task mapping matrix $\boldsymbol{D}_{com}$ and the corresponding desired task vector $\boldsymbol{d}_{com,des}$ in (4.11) compactly represent the CoM wrench task and will be merged with the other tasks in Sec. 4.3.

4.2.2 Foot acceleration tasks

Unlike wheeled robots, legged robots have to execute a discrete series of footsteps in order to move forward. The robot feet alternate between phases of stance and swing. During the stance phase, a foot supports the weight of the robot, transmitting the required forces to the ground. During swing phases, the swing foot moves from its previous footstep location to the next one. This foot relocation has to be thoroughly coordinated with the previewed CoM motion, i.e. the foot needs to target the previewed footstep location precisely enough and with the right timing. This section presents the six-dimensional foot trajectory generation and tracking control method used in this work and its embedding into the whole-body control framework. Although the section is about foot control, the more general term "end effector" is used instead of "foot" to indicate that other end effectors can be controlled using the very same methods.

Design and tracking of translational swing leg trajectories

In this section, the polynomial interpolation method from Sec. 3.1 is used. The linear foot trajectory generation presented in this work typically uses fifth order polynomials (i.e. with 6 unknowns) for the horizontal directions $\chi \in x, y$ and a sixth order polynomial (with 7 DOF) for the vertical direction:

$$\boldsymbol{p}_{\chi,EE} = \underbrace{\begin{bmatrix} \boldsymbol{t}_{\chi}^{T}(0) \\ \dot{\boldsymbol{t}}_{\chi}^{T}(0) \\ \ddot{\boldsymbol{t}}_{\chi}^{T}(0) \\ \boldsymbol{t}_{\chi}^{T}(T_{sw}) \\ \dot{\boldsymbol{t}}_{\chi}^{T}(T_{sw}) \\ \ddot{\boldsymbol{t}}_{\chi}^{T}(T_{sw}) \end{bmatrix}^{-1}}_{\boldsymbol{T}_{c,\chi}} \underbrace{\begin{bmatrix} \chi_{EE}(0) \\ \dot{\chi}_{EE}(0) \\ \ddot{\chi}_{EE}(0) \\ \chi_{EE}(T_{sw}) \\ \dot{\chi}_{EE}(T_{sw}) \\ \ddot{\chi}_{EE}(T_{sw}) \end{bmatrix}}_{\boldsymbol{f}_{c,\chi}} \qquad \boldsymbol{p}_{z,EE} = \underbrace{\begin{bmatrix} \boldsymbol{t}_{z}^{T}(0) \\ \dot{\boldsymbol{t}}_{z}^{T}(0) \\ \ddot{\boldsymbol{t}}_{z}^{T}(0) \\ \boldsymbol{t}_{z}^{T}(\frac{T_{sw}}{2}) \\ \boldsymbol{t}_{z}^{T}(T_{sw}) \\ \dot{\boldsymbol{t}}_{z}^{T}(T_{sw}) \\ \ddot{\boldsymbol{t}}_{z}^{T}(T_{sw}) \end{bmatrix}^{-1}}_{\boldsymbol{T}_{c,z}} \underbrace{\begin{bmatrix} z_{EE}(0) \\ \dot{z}_{EE}(0) \\ \ddot{z}_{EE}(0) \\ z_{EE}(\frac{T_{sw}}{2}) \\ z_{EE}(T_{sw}) \\ \dot{z}_{EE}(T_{sw}) \\ \ddot{z}_{EE}(T_{sw}) \end{bmatrix}}_{\boldsymbol{f}_{c,z}} \tag{4.14}$$

Here, equation (3.4) from Sec. 3.1 is applied. The time in swing phase is denoted by $t_{sw} \in [0, T_{sw}]$, where T_{sw} is the total swing time. Applying (3.1) and (3.2), the time mapping vectors can be computed as $\boldsymbol{t}_{\chi}^{T}(t_{sw}) = [1, t_{sw}, t_{sw}^2, t_{sw}^3, t_{sw}^4, t_{sw}^5]$, $\dot{\boldsymbol{t}}_{\chi}^{T}(t_{sw}) = [0, 1, 2t_{sw}, 3t_{sw}^2, 4t_{sw}^3, 5t_{sw}^4]$, $\ddot{\boldsymbol{t}}_{\chi}^{T}(t_{sw}) = [0, 0, 2, 6t_{sw}, 12t_{sw}^2, 20t_{sw}^3]$, $\boldsymbol{t}_{z}^{T}(t_{sw}) = [1, t_{sw}, t_{sw}^2, t_{sw}^3, t_{sw}^4, t_{sw}^5, t_{sw}^6]$, $\dot{\boldsymbol{t}}_{z}^{T}(t_{sw}) = [0, 1, 2t_{sw}, 3t_{sw}^2, 4t_{sw}^3, 5t_{sw}^4, 6t_{sw}^5]$ and $\ddot{\boldsymbol{t}}_{z}^{T}(t_{sw}) = [0, 0, 2, 6t_{sw}, 12t_{sw}^2, 20t_{sw}^3, 30t_{sw}^2]$, where $t_{sw} = 0$ and $t_{sw} = T_{sw}$, respectively, is used in (4.14) to compute the time mapping matrices $\boldsymbol{T}_{c,\chi}$ and $\boldsymbol{T}_{c,z}$. The latter are square and full rank, and thus invertible. Equation (4.14) provides the polynomial parameter vectors $\boldsymbol{p}_{\chi,EE}$ and $\boldsymbol{p}_{z,EE}$ that comply with the boundary condition vectors $\boldsymbol{f}_{c,\chi}$ and $\boldsymbol{f}_{c,z}$. Typically, zero initial and final velocities and accelerations are used as boundary condition, i.e. $\dot{\chi}_{EE}(0) = 0$, $\ddot{\chi}_{EE}(0) = 0$, $\dot{\chi}_{EE}(T_{sw}) = 0$, $\ddot{\chi}_{EE}(T_{sw}) = 0$, $\dot{z}_{EE}(0) = 0$, $\ddot{z}_{EE}(0) = 0$, $\dot{z}_{EE}(T_{sw}) = 0$ and $\ddot{z}_{EE}(T_{sw}) = 0$. As initial and final position boundary conditions the previous stance foot position and the foot target position (which may be adjusted online, see Sec. 5.3.3) are used. As seventh vertical boundary condition, a vertical foot position at mid swing $z_{EE}(\frac{T_{sw}}{2})$ is used, which yields the required ground clearance during foot swing.

Now, with the solution for the polynomial parameter vectors $\boldsymbol{p}_{\chi,EE}$ and $\boldsymbol{p}_{z,EE}$, the foot position, velocity and acceleration reference for the current time in swing t_{sw} is found using

$$\begin{bmatrix}\chi_{EE,ref}\\ \dot{\chi}_{EE,ref}\\ \ddot{\chi}_{EE,ref}\end{bmatrix}(t_{sw}) = \begin{bmatrix}\boldsymbol{t}_{\chi}^{T}(t_{sw})\\ \dot{\boldsymbol{t}}_{\chi}^{T}(t_{sw})\\ \ddot{\boldsymbol{t}}_{\chi}^{T}(t_{sw})\end{bmatrix} \boldsymbol{p}_{\chi} \qquad \begin{bmatrix}z_{EE,ref}\\ \dot{z}_{EE,ref}\\ \ddot{z}_{EE,ref}\end{bmatrix}(t_{sw}) = \begin{bmatrix}\boldsymbol{t}_{z}^{T}(t_{sw})\\ \dot{\boldsymbol{t}}_{z}^{T}(t_{sw})\\ \ddot{\boldsymbol{t}}_{z}^{T}(t_{sw})\end{bmatrix} \boldsymbol{p}_{z} \ . \tag{4.15}$$

All spatial directions (x, y, z) are now combined in the end effector reference coordinates $\boldsymbol{x}_{EE,ref} = [x_{EE,ref}, y_{EE,ref}, z_{EE,ref}]^T$. The corresponding reference $[\boldsymbol{x}_{EE,ref}, \dot{\boldsymbol{x}}_{EE,ref}, \ddot{\boldsymbol{x}}_{EE,ref}]$ is tracked by applying a second order tracking control law as described in Sec. (3.2):

$$\ddot{\boldsymbol{x}}_{EE,des} = \ddot{\boldsymbol{x}}_{EE,ref} - \lambda_{EE,1}\, \lambda_{EE,2}(\boldsymbol{x}_{EE} - \boldsymbol{x}_{EE,ref}) + (\lambda_{EE,1} + \lambda_{EE,2})(\dot{\boldsymbol{x}}_{EE} - \dot{\boldsymbol{x}}_{EE,ref}) \ . \tag{4.16}$$

Here, $\lambda_{EE,1}$ and $\lambda_{EE,2}$ denote the desired eigenvalues of the end effector tracking dynamics. Equation (4.16) returns the desired linear end effector (here: foot) acceleration.

Design and tracking of rotational swing leg trajectories

For quaternion trajectory generation, the methods presented in Sec. 3.4.2 are used. The initial quaternion ${}^0\zeta_{EE,prev.stance}$ computed from the previous stance foot rotation, the final quaternion target ${}^0\zeta_{EE,target}$ as terminal rotation and optionally several quaternion waypoints distributed over the swing time are used as desired waypoints for the quaternion reference. The latter may be used for example for features such as toe-off motions and heel touchdown preparation (i.e. heel lower than toes right before touchdown to lower the impacts in case of premature landings). These desired quaternion waypoints are then preprocessed via (3.25) and interpolated via polynomials (timings and equations similar to (4.14) and (4.15)) using zero initial and final quaternion velocities and accelerations as additional boundary conditions (to achieve smooth interpolation). The resulting non-unit quaternion trajectory is then consistently projected to the unit quaternion manifold via (3.26), (3.28) and (3.30) (see Sec. 3.4.2). The gained unit quaternion reference trajectory is then tracked via the quaternion tracking controller (3.34) and (3.38), which yields the desired end effector quaternion acceleration $\ddot{\boldsymbol{\zeta}}_{EE,des}$. Setting $\ddot{\boldsymbol{\zeta}} = \ddot{\boldsymbol{\zeta}}_{EE,des}$ in (3.42) yields the desired angular acceleration of the end effector

$$\dot{\boldsymbol{\omega}}_{EE,des} = 4\,(\boldsymbol{E}_{EE}^{T}\, \ddot{\boldsymbol{\zeta}}_{EE,des} + \dot{\boldsymbol{E}}_{EE}^{T}\, \dot{\boldsymbol{\zeta}}_{EE}) \ . \tag{4.17}$$

Implementation of 6-DOF foot acceleration tasks

In [71], Murray et al. introduced three types of different velocities: spatial velocity, body velocity and hybrid velocity (see also Sec. 2.5). The hybrid velocity combines the translational velocity[2] and the spatial angular velocity of a link in a single six-dimensional vector: ${}^0_h\boldsymbol{\nu}_{0,link} = [{}^0\dot{\boldsymbol{x}}_{0,link}^{T}, {}^0\boldsymbol{\omega}_{0,link}^{T}]^T$. The concept of hybrid velocity suits the end effector tracking tasks (4.16) and (4.17) very well. The hybrid velocity (see Sec. 2.5) of an end effector is computed as

$${}^0_h\boldsymbol{\nu}_{0,EE} = {}^0_h\boldsymbol{J}_{0,EE}\, \dot{\boldsymbol{q}} \ . \tag{4.18}$$

Here, ${}^0_h\boldsymbol{J}_{0,EE}$ denotes the hybrid Jacobian of the end effector and $\dot{\boldsymbol{q}}$ are the generalized joint velocities. Differentiation of (4.18) yields the hybrid end effector acceleration:

$${}^0_h\dot{\boldsymbol{\nu}}_{0,EE} = {}^0_h\dot{\boldsymbol{J}}_{0,EE}\, \dot{\boldsymbol{q}} + {}^0_h\boldsymbol{J}_{0,EE}\, \ddot{\boldsymbol{q}} \ . \tag{4.19}$$

[2]in spatial frame, however not to be confused with spatial velocity.

During foot swing, the desired hybrid end effector acceleration combines (4.16) and (4.17) into:

$$ {}_h^0\dot{\boldsymbol{\nu}}_{0,EE,des} = \begin{bmatrix} \ddot{\boldsymbol{x}}_{EE,des} \\ \dot{\boldsymbol{\omega}}_{EE,des} \end{bmatrix} \ , \tag{4.20}$$

while during the stance phase it is assumed that the end effector (e.g. foot) is not accelerating since it is in firm contact, i.e. ${}_h^0\dot{\boldsymbol{\nu}}_{0,EE,des} = \mathbf{0}_{6\times1}$. This is just an assumption, of course, which may cause problems regarding robustness in case it is not true (e.g. robot standing on soft or unstructured grounds). Now, using (4.7) and setting ${}_h^0\dot{\boldsymbol{\nu}}_{0,EE} = {}_h^0\dot{\boldsymbol{\nu}}_{0,EE,des}$ in (4.19), the end effector acceleration task can be expressed as function of the control variables $\boldsymbol{u}$:

$$\underbrace{{}_h^0\dot{\boldsymbol{\nu}}_{0,EE,des} - {}_h^0\dot{\boldsymbol{J}}_{0,EE}\,\dot{\boldsymbol{q}} - {}_h^0\boldsymbol{J}_{0,EE}\,\ddot{\boldsymbol{q}}_{MB}}_{\boldsymbol{d}_{EE,des}} = \underbrace{{}_h^0\boldsymbol{J}_{0,EE}\,\boldsymbol{Q}}_{\boldsymbol{D}_{EE}}\ \boldsymbol{u}\ . \tag{4.21}$$

This equation provides the desired end effector (or foot) acceleration task vector $\boldsymbol{d}_{EE,des}$ and the corresponding task mapping matrix $\boldsymbol{D}_{EE}$, which is embedded into the WBC framework in Sec. 4.3.

4.2.3 Torso orientation control

The torso orientation tracking control implemented in this work is equivalent to the angular component of the end effector tracking task in the previous section. The spatial angular velocity can be computed as

$$ {}^0\boldsymbol{\omega}_{0,torso} = {}_\omega^0\boldsymbol{J}_{0,torso}\,\dot{\boldsymbol{q}}\ , \tag{4.22}$$

where ${}_\omega^0\boldsymbol{J}_{0,torso}$ denotes the angular Jacobian of the torso and $\dot{\boldsymbol{q}}$ are the generalized joint velocities (including the free-floating joints).

Differentiation of (4.22) yields the angular acceleration of the torso:

$$ {}^0\dot{\boldsymbol{\omega}}_{0,torso} = {}_\omega^0\dot{\boldsymbol{J}}_{0,torso}\,\dot{\boldsymbol{q}} + {}_\omega^0\boldsymbol{J}_{0,torso}\,\ddot{\boldsymbol{q}}\ . \tag{4.23}$$

The angular Jacobian ${}_\omega^0\boldsymbol{J}_{0,torso}$ and its time derivative ${}_\omega^0\dot{\boldsymbol{J}}_{0,torso}$ correspond to the angular components of the hybrid Jacobian of the torso ${}_h^0\boldsymbol{J}_{0,torso}$ and its time derivative ${}_h^0\dot{\boldsymbol{J}}_{0,torso}$, respectively. Inserting (4.7) and setting ${}^0\dot{\boldsymbol{\omega}}_{0,torso} = {}^0\dot{\boldsymbol{\omega}}_{0,torso,des}$, the following desired task mapping for the torso orientation is found:

$$\underbrace{{}^0\dot{\boldsymbol{\omega}}_{0,torso,des} - {}_h^0\dot{\boldsymbol{J}}_{0,torso}\,\dot{\boldsymbol{q}} - {}_\omega^0\boldsymbol{J}_{0,torso}\,\ddot{\boldsymbol{q}}_{MB}}_{\boldsymbol{d}_{torso,des}} = \underbrace{{}_\omega^0\boldsymbol{J}_{0,torso}\,\boldsymbol{Q}}_{\boldsymbol{D}_{torso}}\ \boldsymbol{u}\ . \tag{4.24}$$

Here, ${}^0\dot{\boldsymbol{\omega}}_{0,torso,des}$ is the desired angular torso acceleration which can be computed via (3.42) as

$$ {}^0\dot{\boldsymbol{\omega}}_{0,torso,des} = 4\,(\boldsymbol{E}_{torso}^T\,{}^0\ddot{\zeta}_{torso,des} + \dot{\boldsymbol{E}}_{torso}^T\,{}^0\dot{\zeta}_{torso})\ , \tag{4.25}$$

where ${}^0\ddot{\zeta}_{torso,des}$ denotes the desired torso quaternion acceleration which is used to track the torso quaternion reference $[{}^0\zeta_{torso,ref}, {}^0\dot{\zeta}_{torso,ref}, {}^0\ddot{\zeta}_{torso,ref}]$. The latter can be designed by averaging the vertical components of the left and right foot reference quaternion trajectories (the detailed derivation is omitted here). The quaternion transformation matrices $\boldsymbol{E}_{torso}$ and $\dot{\boldsymbol{E}}_{torso}$ in (4.25) are computed from the torso quaternion ${}^0\zeta_{torso}$ and its derivative ${}^0\dot{\zeta}_{torso}$ via (3.40) and (3.43).

Equation (4.24) provides the desired torso acceleration task vector $\boldsymbol{d}_{torso,des}$ and the corresponding task mapping matrix $\boldsymbol{D}_{torso}$, which will be embedded into the WBC framework in Sec. 4.3.

4.2.4 Joint posture reference

The tasks introduced in the previous subsections are not sufficient to determine all desired robot DOF (at least for high-DOF humanoid robots). In other words, the null space of these tasks in non-empty and needs to be controlled. One convenient option is to add a posture task, which typically involves all actuated robot joints. A possible desired task space dynamics for the joint posture task can be defined as follows:

$$\ddot{\boldsymbol{q}}_{post,des} \;=\; \ddot{\boldsymbol{q}}_{post,ref} - \lambda_{post,1}\lambda_{post,2}\,(\boldsymbol{S}_{post}\,\boldsymbol{q} - \boldsymbol{q}_{post,ref}) + (\lambda_{post,1}+\lambda_{post,2})(\boldsymbol{S}_{post}\,\dot{\boldsymbol{q}} - \dot{\boldsymbol{q}}_{post,ref})\ , \tag{4.26}$$

where $\boldsymbol{S}_{post}$ maps the generalized joint coordinates $\boldsymbol{q}$ to the actuated ones $\boldsymbol{q}_{act}$ (i.e. $\boldsymbol{q}_{act} = \boldsymbol{S}_{post}\,\boldsymbol{q}$, in this work $\boldsymbol{S}_{post} = \boldsymbol{S}_{act}^T$), $[\boldsymbol{q}_{post,ref}, \dot{\boldsymbol{q}}_{post,ref}, \ddot{\boldsymbol{q}}_{post,ref}]$ is the postural reference and $\lambda_{post,1}$ and $\lambda_{post,2}$ are the corresponding desired eigenvalues (see Sec. 3.2). While such a dynamic postural reference including corresponding joint velocities and accelerations would be conceivable, this work uses a constant reference posture $\boldsymbol{q}_{post,ref}$ for all actuated joints (i.e. $\dot{\boldsymbol{q}}_{post,ref} = \boldsymbol{0}$ and $\ddot{\boldsymbol{q}}_{post,ref} = \boldsymbol{0}$) such that (4.26) turns into the regulation case.

Reordering (4.7), premultiplying with $\boldsymbol{S}_{post}$ and setting $\boldsymbol{S}_{post}\,\ddot{\boldsymbol{q}} = \ddot{\boldsymbol{q}}_{post,des}$ yields

$$\underbrace{\ddot{\boldsymbol{q}}_{post,des} - \boldsymbol{S}_{post}\,\ddot{\boldsymbol{q}}_{MB}}_{\boldsymbol{d}_{post,des}} \;=\; \underbrace{\boldsymbol{S}_{post}\,\boldsymbol{Q}}_{\boldsymbol{D}_{post}}\,\boldsymbol{u}\ , \tag{4.27}$$

which provides the desired torso acceleration task vector $\boldsymbol{d}_{post,des}$ and the corresponding task mapping matrix $\boldsymbol{D}_{post}$. These will be embedded into the WBC framework in Sec. 4.3.

4.2.5 Control input regularization

In the section about over-determined pseudo inverses, a damping term $\frac{1}{2}\,\boldsymbol{u}^T\boldsymbol{\Lambda}\,\boldsymbol{u}$ was introduced in the cost function (3.48) which ensures good behavior of the resulting input vector $\boldsymbol{u}$ in case of singularities. The latter typically occur when the (undamped) matrix that would have to be inverted is not full rank. Here, a "damping" task is introduced, which fulfills the same function as in (3.48). Its main task is to prevent the control input vector $\boldsymbol{u}$ from growing to infinity when singularities occur. To this end, $\boldsymbol{u}_{des} = \boldsymbol{0}$ will be used as desired input vector. By considering this task in the subsequent optimization this instable behavior is successfully avoided.

For control inputs chosen in this work, as mentioned in Sec. 4.1, the mapping of linear contact forces to contact wrenches (4.3) is *by default under-determined*. Thus, at least the null space that correlates to that mapping has to be damped (or regularized). Additionally, depending on the tasks and robot configuration at hand, further singularities may occur such that the damping of all elements of the input vector $\boldsymbol{u}$ is advisable. That way situations such as fully stretched knees (as often observed in humanoid robots, e.g. in case of perturbations or when taking very long steps while walking) can be handled well, while without damping they can lead to total loss of control. The desired input vector from above can be mapped to the actual input vector via

$$\underbrace{\boldsymbol{u}_{des}}_{\boldsymbol{d}_{damp,des}} \;=\; \underbrace{\boldsymbol{I}_{n_u\times n_u}}_{\boldsymbol{D}_{damp}}\,\boldsymbol{u}\ , \tag{4.28}$$

where, as proposed above, $\boldsymbol{u}_{des} = \boldsymbol{0}$ is chosen as desired damping task vector $\boldsymbol{d}_{damp,des}$ and the damping task mapping matrix $\boldsymbol{D}_{damp}$ is simply $\boldsymbol{I}_{n_u\times n_u}$ (n_u denotes the total number of elements in the control input vector $\boldsymbol{u}$). Via $\boldsymbol{d}_{damp,des}$ and $\boldsymbol{D}_{damp,des}$, the damping task will be embedded into the WBC framework in Sec. 4.3.

4.3 Optimizing the task objectives via quadratic programming

In humanoid robotics the number of desired tasks (as derived in the previous sections) is typically higher than the number of available robot DOF, i.e. the problem is over-determined (as in Sec. 3.5.2). Additionally, the robot hardware limitations have to be met and ground contact constraints have to be fulfilled. Therefore, finding the optimal[3] solution, which is the best trade-off between the desired tasks while fulfilling the given constraints and limitations typically involves optimization. One possible candidate for such an optimization is the use of a *quadratic program* (QP). Such a QP exhibits a quadratic cost function and linear constraints which yields a convex and thus comparably simple problem with only one single global optimum. A QP solves the same optimization problem as an over-determined pseudo inverse in Sec. 3.5.2, only that it additionally considers constraints. The over-determined pseudo inverse as presented in (3.49) is the analytical solution for a quadratic program without constraints.

Inspired by the works of Koolen [81], Righetti [80, 82] and Wensing [51, 52] (besides others), this work formulates a QP, which takes the tasks as presented in the previous sections, certain robot hardware limitations and contact constraints into account. All tasks are treated at the same level, their priorities solely determined by a weighting matrix (single-weighted QP). Hierarchical QP formulations, as for example presented in [52, 87, 88], strictly ensure certain higher-priority tasks to be fulfilled, while lower-priority tasks are overruled, if necessary. This may be viewed as advantage. Yet, most of such hierarchical QPs are not robust against singularities. This is a drawback as compared to single-weighted QPs, which may contain damping terms that drastically reduce such singularity issues. This was the main motivation of using a single-weighted QP in this work.

The previous sections introduced the desired task vectors $\boldsymbol{d}_{i,des}$ and task mappings matrices $\boldsymbol{D}_i$ (here, "i" stands for the different tasks from the previous sections, i.e. "CoM", "left foot", "right foot", "torso","posture" and "damping"), which encode the linear and direct mapping from the control input vector $\boldsymbol{u} = \begin{bmatrix} \boldsymbol{\tau}_{act} \\ \boldsymbol{\rho} \end{bmatrix}$ to the desired tasks. These direct linear mappings are used to express the task errors as

$$\Delta \boldsymbol{d}_i = \boldsymbol{d}_i - \boldsymbol{d}_{i,des} = \boldsymbol{D}_i\, \boldsymbol{u} - \boldsymbol{d}_{i,des} \ . \tag{4.29}$$

Now, a quadratic cost function can be formulated as

$$\begin{aligned} G &= \frac{1}{2} \sum_i \left(\boldsymbol{\Delta d}_i^T\, \boldsymbol{W}_{task,i}\, \boldsymbol{\Delta d}_i \right) \qquad (4.30) \\ &= \frac{1}{2} \boldsymbol{u}^T \underbrace{\sum_i \Big(\underbrace{\boldsymbol{D}_i^T\, \boldsymbol{W}_{task,i}\, \boldsymbol{D}_i}_{\boldsymbol{H}_{QP,i}} \Big)}_{\boldsymbol{H}_{QP}} \boldsymbol{u} - \underbrace{\sum_i \Big(\underbrace{\boldsymbol{d}_{i,des}^T\, \boldsymbol{W}_{task,i}\, \boldsymbol{D}_i}_{\boldsymbol{f}_{QP,i}^T} \Big)}_{\boldsymbol{f}_{QP}^T} \boldsymbol{u} + \underbrace{\frac{1}{2} \sum_i \left(\boldsymbol{d}_{i,des}^T\, \boldsymbol{W}_{task,i}\, \boldsymbol{d}_{i,des} \right)}_{const.} \end{aligned}$$

where the index i stands for all task indices "CoM", "left foot", "right foot", "torso","posture" and "damping" and the sum in the first line adds up the weighted error squares for all tasks. The task weighting matrices $\boldsymbol{W}_{task,i}$ (one for each task i) are typically chosen to be diagonal to independently weight the different task elements. The second line of (4.30) shows that, for each task, a square Hessian matrix $\boldsymbol{H}_{QP,i} \in \mathbb{R}^{n_u \times n_u}$ and a row vector $\boldsymbol{f}_{QP,i}^T \in \mathbb{R}^{1 \times n_u}$ can be independently computed (n_u is the total number of elements of $\boldsymbol{u}$). The final Hessian matrix $\boldsymbol{H}_{QP}$ and QP-defining row vector $\boldsymbol{f}_{QP}^T$ are then simply the sums of these quantities over all tasks. One advantage

[3]optimal w.r.t. a chosen cost function

over task stack formulations (where all task mapping matrices and desired task vectors are stacked in one big matrix and vector, respectively) is that the size of the Hessian matrices and QP-defining vectors is equal for each of the subtasks. This is particularly convenient if the exact structure of the optimization problem is changed, e.g. when new tasks are added. The combined Hessian matrix $\boldsymbol{H}_{QP}$ and vector $\boldsymbol{f}_{QP}^T$ are forwarded to a QP solver. Note that the right term in the second line of (4.30) is constant and can thus be omitted.

Finally, the considered optimization problem in this work is formulated as

$$\begin{aligned} \min_{\boldsymbol{u}}. \quad & G = \frac{1}{2}\boldsymbol{u}^T \boldsymbol{H}_{QP}\boldsymbol{u} - \boldsymbol{f}_{QP}^T \boldsymbol{u} \\ \text{s.t.} \quad & \underline{\boldsymbol{\tau}}_{act} \leq \boldsymbol{\tau}_{act} \leq \overline{\boldsymbol{\tau}}_{act} \\ & \boldsymbol{0} \leq \boldsymbol{\rho} \leq \overline{\boldsymbol{\rho}} \end{aligned} \quad . \tag{4.31}$$

The operators $\underline{\bullet}$ and $\overline{\bullet}$ in the inequality constraints denote lower and upper limits. The contact force magnitudes $\boldsymbol{\rho}$ are constraint to be positive, such that the contact constraints are ensured. Note that by the use of (4.7) in the previous subsections, the robotic equations of motion are implicitly considered. When a foot is in contact with the environment, the motion constraints have to be modeled which is typically accomplished by setting the corresponding accelerations to zero. A peculiarity in the presented QP formulation is that the contact constraints (i.e. zero foot acceleration during stance) are not implemented as hard equality constraint but as standard tasks. The corresponding task weights are chosen to be very high such that the constraints are typically fulfilled well[4]. As compared to the hard equality constraint, the presented formulation is less sensitive against ill-posed situations such as singular configurations. This may help the QP solver find a solution, which otherwise may fail to converge. Of course, choosing to implement these constraints as hard constraints could be considered as well. This choice of soft or hard contact constraints is subordinate as compared to the following issue: any possible contact scenario, unless the environment surrounding the robot including its compliance parameters is perfectly known to the programmer, is always just a collection of *assumptions*. In case of interaction with natural environments, the exact motion constraints and interaction forces can never be predicted perfectly. The mismatch between assumed and actual contact can have severe effects on the achieved robot dynamics and stability. Contact estimation and robustification is a research field that requires further investigation, but out of the scope of this work.

The solution to (4.31) is the optimal control input vector $\boldsymbol{u}^* = \begin{bmatrix} \boldsymbol{\tau}_{act}^* \\ \boldsymbol{\rho}^* \end{bmatrix}$ that trades off between the different tasks according to the cost function (4.30), while fulfilling maximum joint torque and contact constraints. In this work, to achieve this optimal solution the qpOASES software [89] is used, which is based on an online active set strategy described in [90]. The optimal joint torques $\boldsymbol{\tau}^*$ resulting from (4.31) are commanded to the robot.

[4]Note: during stance, very high task weights for the foot accelerations are used, while during swing these weights are lowered by a factor of about 100.

CHAPTER 5

Bipedal walking control based on Divergent Component of Motion

The previous section gave an overview of the whole-body control framework used in this work, while, with regard to robust walking and running (see chapter 6) control, referring to this section. This section will present the walking trajectory generation and feedback control framework used in this work, which is based on the CoM dynamics. The latter can be analyzed and controlled in a particularly elegant and comprehensible way by using the so called Divergent Component of Motion (DCM), also known as "(instantaneous) Capture Point", as system coordinate.

Note that, due to the linearity of the CoM dynamics of a general robot, the equations and control laws, derived in this chapter[1] and in chapter 6 on running, hold for general free-floating robot models and not only for simplified models, such as the telescopic or linear inverted pendulum.

The contributions of this chapter are (i) the extension of the concept of Divergent Component of Motion (DCM) to 3D, (ii) the introduction of Enhanced Centroidal Moment Pivot point (eCMP) and Virtual Repellent Point (VRP) to encode external and total force acting on the CoM, (iii) the design of smooth and constraint-compatible eCMP and DCM trajectories and their tracking, (iv) the introduction of an analytical step adjustment method and (v) the introduction of a CoM disturbance observer to compensate for unknown perturbations. The Linear Inverted Pendulum-based derivations in Sec. 5.1 provide a comprehensible introduction to walking reference generation and tracking, while later the general CoM dynamics is used as model instead of the LIP.

5.1 Controlling the Linear Inverted Pendulum using the concept of Capture Point

The following sections will recapitulate the Linear Inverted Pendulum (LIP) model and present simple Capture Point (CP) based methods for basic planning and feedback control. The purpose of these sections is to familiarize the reader with the concept of LIP and CP. Section 5.3 will then present the author's current more advanced state of research which is based on the three-dimensional Divergent Component of Motion (DCM) and allows for smooth and robust planning and control of bipedal walking motions.

[1]aside from the introductory sections on LIP/ZMP-based Capture Point control

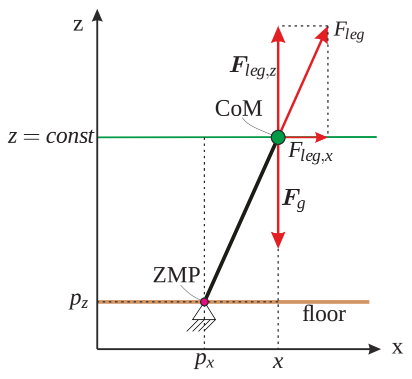

Figure 5.1: Linear inverted pendulum (LIP) model.

5.1.1 Linear Inverted Pendulum model as starting point

The basic challenge in robotic locomotion research is to maneuver the robot (or more precisely: its CoM) from one point A to another point B in space. The CoM dynamics are inertially decoupled from the rest of the robot dynamics [91] and are only affected by gravity and the external (e.g. leg) forces (Newton's second law on momentum conservation):

$$\ddot{\boldsymbol{x}} = \frac{1}{m}\boldsymbol{F}_{com} = \frac{1}{m}\left(\boldsymbol{F}_g + \boldsymbol{F}_{ext}\right) \tag{5.1}$$

Here, $\ddot{\boldsymbol{x}} = [\ddot{x}\;\ddot{y}\;\ddot{z}]^T$ denotes the CoM acceleration, $\boldsymbol{F}_{ext}$ is the sum of all external forces, $\boldsymbol{F}_g = m\;[0\;0\;-g]^T$ is the gravitational force and g is the gravitational constant. As the CoM dynamics is a second order dynamics, the CoM position at a given point t in time can be derived by double integration of the CoM acceleration. One difficulty arises from the fact, that there exists an infinite number of different time-transient total forces $\boldsymbol{F}_{com}$ for which the CoM motion would fulfill the constraints $\boldsymbol{x}_0 = \boldsymbol{x}_A$ and $\boldsymbol{x}_{end} = \boldsymbol{x}_B$. This poses the problem that no unique solution to the problem exists. Additionally, in order to be physically feasible, the line of action of the sum of external forces acting on the robot has to pass through the base of support (e.g. convex hull of robot's stance feet). The use of arbitrary CoM reference trajectories $[\boldsymbol{x}_{ref}, \dot{\boldsymbol{x}}_{ref}, \ddot{\boldsymbol{x}}_{ref}]$ combined with standard tracking controllers of the form $\boldsymbol{F}_{com} = k_p\,(\boldsymbol{x}_{ref} - \boldsymbol{x}) + k_d\,(\dot{\boldsymbol{x}}_{ref} - \dot{\boldsymbol{x}}) + m\,\ddot{\boldsymbol{x}}_{ref}$ typically disregards this physical constraint and is therefore not well suited for bipedal walking (and running, see chapter 6) control. One idea to solve this feasibility issue for the problem of bipedal walking is the following: Instead of *checking* if the lines of action of the leg forces intersect the base of support, a point of intersection (focus of all lines of action of external forces) is *designed* and related to a corresponding force via an appropriate force law. The *Linear Inverted Pendulum* (LIP) model [92] follows this idea. Its torque-free base joint represents the mentioned focus point. This torque-free focus point is also known as *zero moment point* (ZMP), as introduced by Vukobratovic et. al [9], or *center of pressure* (COP). Note: these two terms denote the same quantity, namely a point on the foot (or base of support) of the robot that is most appropriate to support the external forces (or more exactly: wrenches), while the corresponding torque in that point is minimized. The term "ZMP" is traditionally used for flat floor walking and the corresponding control, while the CoP can be applied to the 3D case (see Sec. 5.2.2) as well. For that reason, both terms are used in this thesis.

The LIP model uses the following assumptions:

- no horizontal torque around ZMP (by definition) $\rightarrow$ leg force passes through ZMP
- no torque around CoM $\rightarrow$ leg force passes through CoM
- constant vertical height $\rightarrow$ vertical component of leg force compensates for gravity

Using these assumptions, the relation between leg force and ZMP position becomes particularly simple. By inspecting Fig. 5.1, the following relation between the force acting on the CoM, the CoM position and the ZMP position is found:

$$\frac{F_\chi}{F_z} = \frac{m\,\ddot{\chi}}{m\,g} = \frac{\chi - p_\chi}{z - p_z} \quad . \tag{5.2}$$

The variable χ represents both horizontal directions of the CoM position, i.e., $\chi \in \{x,y\}$. The ZMP height p_z equals the floor height. Reordering (5.2) yields the second order CoM dynamics

$$\ddot{\chi} = \omega^2\,(\chi - p_\chi) \ , \tag{5.3}$$

with

$$\omega = \sqrt{\frac{g}{z - p_z}} \quad . \tag{5.4}$$

The complete horizontal system dynamics of the LIP model is given by

$$\begin{bmatrix} \dot{\chi} \\ \ddot{\chi} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ \omega^2 & 0 \end{bmatrix} \begin{bmatrix} \chi \\ \dot{\chi} \end{bmatrix} + \begin{bmatrix} 0 \\ -\omega^2 \end{bmatrix} p_\chi \ . \tag{5.5}$$

Using (5.1) and (5.3), the three-dimensional force acting on the CoM can be written as

$$\boldsymbol{F}_{com,LIP} = \underbrace{\begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix}}_{\boldsymbol{F}_g} + \underbrace{\frac{m\,g}{z - p_z}(\boldsymbol{x} - \boldsymbol{p})}_{\boldsymbol{F}_{leg}} = m\,\omega^2 \begin{bmatrix} x - p_x \\ y - p_y \\ 0 \end{bmatrix} , \tag{5.6}$$

where $\boldsymbol{F}_g$ and $\boldsymbol{F}_{leg}$ are the gravitational and leg force, $\boldsymbol{x} = [x\,y\,z]^T$ denotes the position of the CoM and $\boldsymbol{p} = [p_x\ p_y\ p_z]^T$ are the LIP base joint coordinates. The vertical components of $\boldsymbol{F}_g$ and $\boldsymbol{F}_{leg}$ cancel each other, so that $\ddot{z} = 0$ holds. Note that the term $\frac{mg}{z-p_z} = m\,\omega^2$ is constant and might even be combined into one single constant. Due to the torque-free base and the point mass assumption, all forces pass through the CoM and the LIP base joint, which therefore can be seen as a focus point of all lines of action of leg forces. When the LIP is used for bipedal walking control, the ZMP [9] is assumed to be equivalent to the LIP base joint $\boldsymbol{p}$ and is designed to be within the base of support. In a real robot, the resulting ZMPs generally deviate from the planned focus points due to the model inaccuracies, but often not far enough to make the robot tilt and fall. This way, preplanning a feasible set of focus points (e.g. ZMP's) is a successful method for bipedal gait generation and control: Stephens and Atkeson [11] present a Model Predictive Control (MPC) method for step adjustment and push recovery. Wieber [12] proposes a trajectory free linear MPC scheme, allowing for compensation of strong perturbations. Nishiwaki and Kagami [14] generate dynamically stable walking patterns by frequently updating a preview controller. Sugihara [15] introduces the Best CoM-ZMP Regulator facilitating step adjustment of bipedal robots. Kajita et al. [16] demonstrate walking on uneven pavement.

5.1.2 Derivation of the two-dimensional Capture Point

Pratt et al. [20–22] introduced the (instantaneous) *Capture Point* (CP) as the point on the floor, where a robot (modeled as a LIP) has to step (more precisely, where the ZMP of the robot has to be) to come to a complete rest, which means that the CoM is exactly located over the ZMP and its velocity is zero. Pratt et al. derive the CP from the linear inverted pendulum orbital energy. This work provides an alternative derivation of the CP. To this end, the explicit solution of (5.5) for a constant ZMP position p_χ is computed as

$$\begin{bmatrix} \chi(t) \\ \dot{\chi}(t) \end{bmatrix} = \begin{bmatrix} \cosh(\omega t) & \frac{1}{\omega}\sinh(\omega t) \\ \omega \sinh(\omega t) & \cosh(\omega t) \end{bmatrix} \begin{bmatrix} \chi_0 \\ \dot{\chi}_0 \end{bmatrix} + \begin{bmatrix} 1-\cosh(\omega t) \\ -\omega \sinh(\omega t) \end{bmatrix} p_\chi \ , \tag{5.7}$$

where χ_0 and $\dot{\chi}_0$ are the initial CoM position and velocity, respectively. As announced above, the particular ZMP position p_χ towards which - if applied to the robot - the CoM position converges when time goes to infinity is sought-after. Using this information in the first row of (5.7) yields

$$\chi|_{t\to\infty} = p_\chi = \chi_0 \ \cosh(\omega t) + \frac{\dot{\chi}_0}{\omega} \ \sinh(\omega t) \ + \ p_\chi - p_\chi \cosh(\omega t) \ , \tag{5.8}$$

which can be solved for p_χ as

$$p_\chi = \chi_0 + \frac{\dot{\chi}_0}{\omega} \ \cancelto{1}{\tanh(\omega t)|_{t\to\infty}} \ . \tag{5.9}$$

By inserting (5.9) into the second row of (5.7) it can be shown that for this choice of p_χ the horizontal velocity of the CoM tends to zero for $t \to \infty$. The derived point p_χ is equivalent to the definition of the Capture Point ξ. For arbitrary CoM positions χ and velocities $\dot{\chi}$ it is defined as

$$\xi_\chi = \chi + \frac{\dot{\chi}}{\omega} \ . \tag{5.10}$$

Note: As proposed by several previous works [19–27], the overall CoM dynamics (5.5) can be split into a stable and an unstable part via diagonalization. Interestingly, the unstable part yields exactly the same definition as in (5.10). This unstable component has been referred to as '(instantaneous) Capture Point' (iCP or CP) by Pratt et al. [20–22], as 'extrapolated Center of Mass' (eCOM) by Hof et al. [19] and, due to its instability, as 'Divergent Component of Motion' (DCM) by Takenaka et al. [23]. Note that in some of the author's previous works [24, 25], motivated by the works of Pratt et al., the term 'Capture Point' was used for the DCM. Therefore, in this section the expression Capture Point is utilized, while starting from Sec. 5.2.1 the term (three-dimensional) Divergent Component of Motion (DCM) will be used, since it gives a better impression of its diverging characteristics. Throughout this work, the symbol $\boldsymbol{\xi}$ is used to represent both the two-dimensional Capture Point (or 2D DCM) and the three-dimensional Divergent Component of Motion (DCM). Combining both horizontal directions x and y of (5.10), the definition of the two-dimensional Capture Point (or 2D DCM) becomes

$$\boldsymbol{\xi} = \boldsymbol{x} + \frac{1}{\omega} \ \dot{\boldsymbol{x}} \ , \tag{5.11}$$

i.e., the Capture Point is composed of the horizontal CoM position $\boldsymbol{x} = [x,y]^T$ and velocity $\dot{\boldsymbol{x}} = [\dot{x},\dot{y}]^T$ scaled by the LIP time-constant $\frac{1}{\omega}$. Reordering (5.11) yields the CoM dynamics

$$\dot{\boldsymbol{x}} = -\omega \ (\boldsymbol{x} - \boldsymbol{\xi}) \ , \tag{5.12}$$

which for $\omega > 0$ is a stable first order dynamics, i.e., the CoM naturally follows the Capture Point. Differentiating (5.11) and inserting (5.3) yields the Capture Point dynamics

$$\dot{\boldsymbol{\xi}} = \omega \ (\boldsymbol{\xi} - \boldsymbol{p}) \ , \tag{5.13}$$

Here, $\boldsymbol{p} = [p_x, p_y]^T$ denotes the horizontal components of the ZMP position. For $\omega > 0$, (5.13) is an unstable first-order dynamics, i.e., the Capture Point is pushed away from the ZMP. Note that here the second order overall CoM dynamics (5.5) was decoupled into the naturally stable first order CoM dynamics (5.12) and the instable first order Capture Point dynamics (5.13). As shown later, this simplifies the design of planning and control algorithms which is the main motivation for the use of the concept of Capture Point (or DCM) in this work.

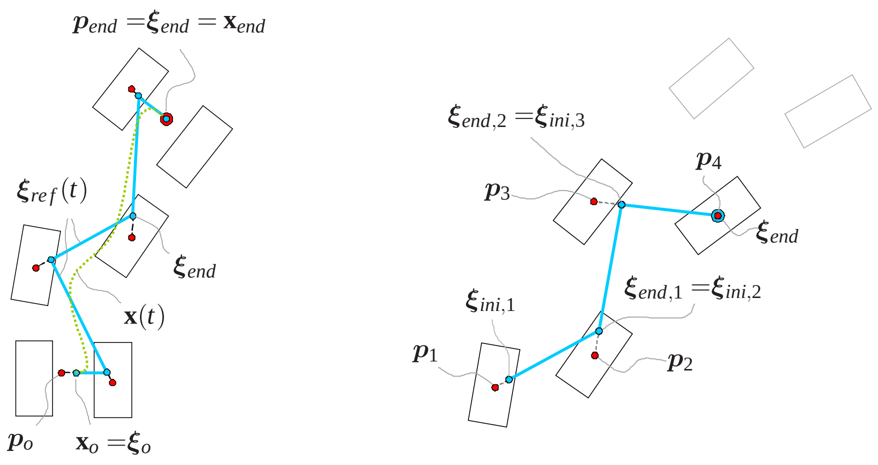

Figure 5.2: Design of two-dimensional Capture Point reference trajectory:
(a) Foot to foot shift (b) Preview of next three steps for CP reference
Note: the reference trajectory is not affected by the current/measured Capture Point.

5.1.3 Planning Capture Point reference trajectories for constant ZMPs

In this section, Capture Point (CP) reference trajectories for nominally constant ZMPs will be designed that have instantaneous transitions (no double support) between them. To this end, the CP dynamics (5.13) are solved for a constant ZMP $\boldsymbol{p}$ which yields

$$\boldsymbol{\xi}(t) = \boldsymbol{p} + e^{\omega\, t}\, (\boldsymbol{\xi}_{ini} - \boldsymbol{p}) \ . \tag{5.14}$$

This equation shows, that, starting from its initial position $\boldsymbol{\xi}_{ini}$, the CP diverges exponentially. The main idea of the CP reference trajectory planning in this work is to generate a “controlled divergence” of the CP, i.e., to align the natural divergence of the CP with the locomotion task at hand. To this end, a set of $(N-1)$ Capture Point reference trajectories is designed that are consistent with the N given ZMP positions (see Fig. 5.2). Therefore, starting with $\boldsymbol{\xi}_{ref,end,N-1} = \boldsymbol{p}_{ref,N}$ (corresponding to the *terminal constraint* $\dot{\boldsymbol{\xi}} = 0$), the following backwards iteration is used:

$$\boldsymbol{\xi}_{ref,end,i-1} = \boldsymbol{\xi}_{ref,ini,i} = \boldsymbol{p}_{ref,i} + e^{-\omega\, T_i}\, (\boldsymbol{\xi}_{ref,end,i} - \boldsymbol{p}_{ref,i}) \ , \tag{5.15}$$

which tells what position $\boldsymbol{\xi}_{ref,ini,i}$ the CP reference should start from in order to be pushed by the constant reference ZMP position $\boldsymbol{p}_{ref,i}$ to the final desired CP reference position $\boldsymbol{\xi}_{ref,end,i}$ of that step during the i-th step time T_i. In order to have continuous transitions between the sub-trajectories, the final CP reference points $\boldsymbol{\xi}_{ref,end,i-1}$ of the $(i-1)$-th step has to equal the initial CP $\boldsymbol{\xi}_{ref,ini,i}$ of the i-th step. That way all initial CP reference positions $\boldsymbol{\xi}_{ref,ini,i}$ are computed and can be combined with (5.14) to provide CP reference trajectories

$$\boldsymbol{\xi}_{ref,i}(t_i) = \boldsymbol{p}_{ref,i} + e^{\omega\, t_i}\, (\boldsymbol{\xi}_{ref,ini,i} - \boldsymbol{p}_{ref,i}) \ . \tag{5.16}$$

Here, $t_i \in [0, T_i]$ is the i-th time in step (T_i being the i-th step duration). Inserting $\boldsymbol{p}_{ref,i}$ and $\boldsymbol{\xi}_{ref,i}(t_i)$ into (5.13) yields the same solution for the CP reference velocities as differentiation of (5.16):

$$\dot{\boldsymbol{\xi}}_{ref,i}(t_i) = \omega\, (\boldsymbol{\xi}_{ref,i}(t_i) - \boldsymbol{p}_{ref,i}) = \omega\, e^{\omega\, t_i}\, (\boldsymbol{\xi}_{ref,ini,i} - \boldsymbol{p}_{ref,i}) \ . \tag{5.17}$$

The current CP reference position $\boldsymbol{\xi}_{ref,1}(t_1)$ and velocity $\dot{\boldsymbol{\xi}}_{ref,1}(t_1)$ will be used for CP tracking control in the next section. Note: in this work, index $i = 1$ always relates to the first/current step.

5.1.4 Capture Point tracking control

In section 5.1.2 the CoM was found to have a stable first order dynamics (5.12) that naturally follows the DCM. This section relies on the natural stability of the CoM dynamics while focusing on the stabilization of the naturally unstable Capture Point dynamics (5.13). In order to locally track the desired CP reference trajectory from the previous section, the following stable first-order behavior for the CP dynamics is eligible:

$$\underbrace{\dot{\boldsymbol{\xi}} - \dot{\boldsymbol{\xi}}_{ref,1}}_{\Delta\dot{\boldsymbol{\xi}}} = -k_\xi \, (\underbrace{\boldsymbol{\xi} - \boldsymbol{\xi}_{ref,1}}_{\Delta\boldsymbol{\xi}}) \ , \tag{5.18}$$

i.e., the CP error $\Delta\xi$ asymptotically converges towards zero. This behavior is achieved by the following control law for the desired ZMP:

$$\boldsymbol{p}_{des} = \boldsymbol{p}_{ref,1} + \left(1 + \frac{k_\xi}{\omega}\right) (\underbrace{\boldsymbol{\xi} \, - \, \boldsymbol{\xi}_{ref,1}}_{\Delta\boldsymbol{\xi}}) \ , \tag{5.19}$$

which can be verified by inserting (5.19) back into (5.13) and using (5.17). Note that the first (i.e. current) CP reference position $\boldsymbol{\xi}_{ref,1}$ and velocity $\dot{\boldsymbol{\xi}}_{ref,1}$ are used here, since the tracking controller acts on a purely local level. The CP reference velocity $\dot{\boldsymbol{\xi}}_{ref,1}$ does not appear in (5.19) explicitely, since the relation (5.17) was applied. It turns out that the controller - for $k_\xi > 0$ - asks for a ZMP position $\boldsymbol{p}_{des}$ that is located at a certain deviation from the current ZMP reference position $\boldsymbol{p}_{ref,1}$. This deviation is bigger than the current CP error $\Delta\xi$ (since $(1 + \frac{k_\xi}{\omega}) > 1$), which yields the desired convergence behavior with respect to the CP reference trajectory.

5.2 Extension of the Capture Point to 3D

The use of the LIP model for bipedal walking control is restricted to horizontal motions of the CoM ($z = const$). This motivates new methods, which are not limited to constant CoM and floor height. Zhao and Sentis [29] introduce the Prismatic Inverted Pendulum dynamics and solve it via numerical integration, allowing for three-dimensional foot placement planning on uneven ground surfaces. Yet, the lateral foot-placement cannot be predefined, but is dependent on the sagittal dynamics and on the desired CoM Surface. Additionally, the method is restricted to ground surfaces with laterally constant heights (“roughness” of terrain only in forward direction).

In [30] - motivated by the capabilities of CP control [25] - a method for bipedal gait planning and control on uneven terrain is derived, facilitated by the use of the linear properties of the Divergent Component of Motion (DCM) dynamics and suffering from none of the afore mentioned restrictions. In the following sections, the methods from [30] will be further improved: The planning method now guarantees smooth and safe point to point interpolation (replacing unsafe interpolation heuristics from [30]) of enhanced Centroidal Moment Pivot points (eCMPs) (replacing ZMPs, see below). Also, new methods for reactive step adjustment and increased robustness against strong and continuous perturbations are provided.

5.2.1 Three-dimensional Divergent Component of Motion (DCM)

As shown in [30], there exists a significant difference between the Capture Point (defined as the point on the ground where the robot has to step to come to a stop asymptotically) and the (three-dimensional) *Divergent Component of Motion* (DCM), as the 3D DCM is not restricted to the ground plane, but can be interpreted as three-dimensional state. For 2D considerations (constant CoM height), Capture Point and DCM (projected to the floor) are equivalent, but this is not true in 3D. Motivated by the performance of Capture Point control in [24, 25] and by the definition of the Capture Point (or two-dimensional DCM) in [20, 23], Englsberger et al. introduced the three-dimensional Divergent Component of Motion (DCM) in [30] as

$$\boldsymbol{\xi} = \boldsymbol{x} + b\,\dot{\boldsymbol{x}}, \tag{5.20}$$

where $\boldsymbol{\xi} = [\xi_x, \xi_y, \xi_z]^T$ is the DCM, $\boldsymbol{x} = [x, y, z]^T$ and $\dot{\boldsymbol{x}} = [\dot{x}, \dot{y}, \dot{z}]^T$ are CoM position and velocity and $b > 0$ is the time-constant of the DCM dynamics. Note the similarity of (5.20) to the state transformation proposed by Slotine and Li [93] used for adaptive control. In contrast to the definition of the Capture Point in [19–25], the DCM as defined in equation (5.20) lies at a certain distance in front of the CoM (w.r.t. its current moving direction), i.e., it is generally not located on the ground but somewhere in space. By reordering (5.20), the CoM dynamics can be derived

$$\dot{\boldsymbol{x}} = -\frac{1}{b}\,(\boldsymbol{x} - \boldsymbol{\xi}). \tag{5.21}$$

This shows that the CoM has a stable first order dynamics for $b > 0$ ($\rightarrow$ it follows the DCM). By differentiating (5.20) and inserting (5.21) and (5.1), the DCM dynamics is found

$$\dot{\boldsymbol{\xi}} = -\frac{1}{b}\,\boldsymbol{x} + \frac{1}{b}\,\boldsymbol{\xi} + \frac{b}{m}\,\boldsymbol{F}_{com}\ . \tag{5.22}$$

This shows that $\boldsymbol{F}_{com}$ directly influences the DCM dynamics.

5.2.2 Enhanced Centroidal Moment Pivot point (eCMP)

Generally, a robot is subject to gravity and external forces. As proposed further above, external forces are *designed* to be appropriate for the locomotion task while fulfilling the feasibility constraint (center of pressure (COP) in base of support). To simplify this design process, a force-to-point transformation similar as in the LIP model is used. Remember that the term $\frac{mg}{z-p_z}$ in (5.6) is constant. This motivates the encoding of external (e.g. leg-) forces in a linear repelling force law based on the difference of the CoM and the so called *Enhanced Centroidal Moment Pivot* point (eCMP), denoted by $\boldsymbol{r}_{ecmp}$:

$$\boldsymbol{F}_{ext} = s\,(\boldsymbol{x} - \boldsymbol{r}_{ecmp})\ . \tag{5.23}$$

Here, $s > 0$ is a constant that will be determined later. The eCMP is closely related to the CMP [66], but not restricted to the ground surface. This allows to encode not only the direction of the external force, but also its magnitude. The CMP in contrast is located at the intersection of the line between CoM and eCMP with the ground (see fig. 5.3). The total force acting on the CoM is

$$\boldsymbol{F}_{com} = \boldsymbol{F}_{ext} + \boldsymbol{F}_g = s\,(\boldsymbol{x} - \boldsymbol{r}_{ecmp}) + m\boldsymbol{g}\ , \tag{5.24}$$

with the gravity vector $\boldsymbol{g} = [0\ 0\ -g]^T$. Inserting (5.24) into (5.22) yields the DCM dynamics

$$\dot{\boldsymbol{\xi}} = \left(\frac{bs}{m} - \frac{1}{b}\right)\boldsymbol{x} + \frac{1}{b}\,\boldsymbol{\xi} - \frac{bs}{m}\,\boldsymbol{r}_{ecmp} + b\,\boldsymbol{g}\ . \tag{5.25}$$

This shows that the states $\boldsymbol{x}$ and $\boldsymbol{\xi}$ are coupled in general. Though, by the choice $s = \frac{m}{b^2}$, the DCM dynamics is decoupled from the CoM dynamics:

$$\dot{\boldsymbol{\xi}} = \frac{1}{b}\,\boldsymbol{\xi} - \frac{1}{b}\,\boldsymbol{r}_{ecmp} + b\,\boldsymbol{g}\ . \tag{5.26}$$

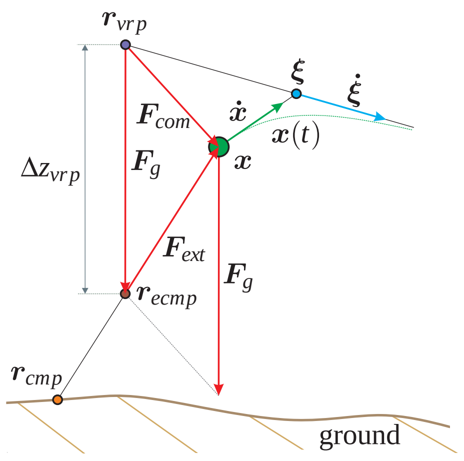

Figure 5.3: Force encoding via eCMP and VRP.

This decoupling of the DCM dynamics from the CoM dynamics clearly facilitates planning and control for bipedal walking. Here, in contrast to methods that yield decoupling via feedback control, the decoupling is achieved via the appropriate choice of the constant s. The equation for the sum of external forces becomes

$$\boldsymbol{F}_{ext} = \frac{m}{b^2}\left(\boldsymbol{x} - \boldsymbol{r}_{ecmp}\right) = \frac{mg}{\Delta z_{vrp}}\left(\boldsymbol{x} - \boldsymbol{r}_{ecmp}\right) \ . \tag{5.27}$$

Here, the identity $b = \sqrt{\Delta z_{vrp}/g}$ was already used, which will be derived in the next section.

5.2.3 Virtual Repellent Point (VRP)

To further simplify (5.26), the *Virtual Repellent Point* (VRP) is introduced, which is defined as

$$\boldsymbol{r}_{vrp} = \boldsymbol{r}_{ecmp} + \left[0\ 0\ b^2 g\right]^T = \boldsymbol{r}_{ecmp} + \left[0\ 0\ \Delta z_{vrp}\right]^T \ . \tag{5.28}$$

The $x-$ and $y-$ components of the eCMP and the VRP are equal. Their vertical components z_{ecmp} and z_{vrp} differ by $\Delta z_{vrp} = b^2 g$, which is a constant that is used as design variable in this work. Note: Δz_{vrp} can be interpreted as “average height” of the CoM over the ground surface. With (5.28), the time-constant of the DCM dynamics is found as

$$b = \sqrt{\frac{\Delta z_{vrp}}{g}} \ . \tag{5.29}$$

In this work, b can always be equated with $\sqrt{\frac{\Delta z_{vrp}}{g}}$. With (5.28) the DCM dynamics (5.27) is rewritten as

$$\dot{\boldsymbol{\xi}} = \frac{1}{b}\left(\boldsymbol{\xi} - \boldsymbol{r}_{vrp}\right) \ . \tag{5.30}$$

This shows that the DCM has an unstable first order dynamics (if the VRP stays at a constant location, the DCM is “pushed” away from it on a straight line), whereas the CoM follows the DCM with the stable first order dynamics (5.21) (see fig. 5.3). The overall open-loop dynamics is

$$\begin{bmatrix}\dot{\boldsymbol{x}}\\ \dot{\boldsymbol{\xi}}\end{bmatrix} = \begin{bmatrix}-\frac{1}{b} & \frac{1}{b}\\ 0 & \frac{1}{b}\end{bmatrix}\begin{bmatrix}\boldsymbol{x}\\ \boldsymbol{\xi}\end{bmatrix} + \begin{bmatrix}0\\ -\frac{1}{b}\end{bmatrix}\boldsymbol{r}_{vrp} \ . \tag{5.31}$$

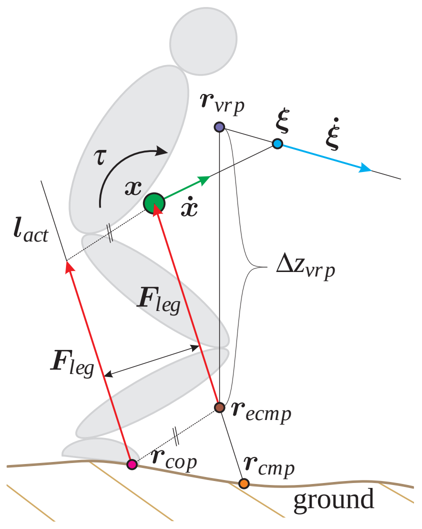

Figure 5.4: Point relations for general robot dynamics.

The VRP encodes gravity and external forces in one single point. With (5.24), (5.27), (5.28) and $\Delta z_{vrp} = b^2 g$, the relation between the total force $\boldsymbol{F}_{com}$ acting on the CoM and the VRP is found as

$$\boldsymbol{F}_{com} = \frac{m}{b^2}(\boldsymbol{x} - \boldsymbol{r}_{vrp}) = \frac{mg}{\Delta z_{vrp}}(\boldsymbol{x} - \boldsymbol{r}_{vrp}) \ . \tag{5.32}$$

If required, a desired VRP $\boldsymbol{r}_{vrp,des}$ can be translated into the required external (e.g. leg-) force via

$$\boldsymbol{F}_{ext,des} = \frac{m}{b^2}(\boldsymbol{x} - \underbrace{(\boldsymbol{r}_{vrp,des} - \left[0 \ 0 \ \Delta z_{vrp}\right]^T)}_{\boldsymbol{r}_{ecmp,des}}) \ . \tag{5.33}$$

Geometrical interpretation of eCMP and VRP

Figures 5.3 and 5.4 graphically delineate the relations between the introduced points, eCMP and VRP, and the corresponding forces. Figure 5.3 shows that the VRP encodes the sum of all forces (gravity and external forces) $\boldsymbol{F}_{com}$ via (5.32). The Enhanced Centroidal Moment Pivot (eCMP) encodes the external forces $\boldsymbol{F}_{ext}$ via (5.27) and is located at a vertical offset Δz_{vrp} below the VRP. Note that by means of appropriate scaling (length of force arrow equals $\Delta z_{vrp}/(mg)$ times force magnitude) the force vectors can be geometrically related to correspondent point distances (comp. to (5.32) and (5.27)). Figure 5.4 depicts the relations between eCMP, CMP and CoP in a general bipedal robot. The CMP is found at the intersection of the line CoM-to-eCMP with the ground. The line of action $\boldsymbol{l}_{act}$ of the leg force can be shifted by means of a torque τ around the CoM, so that the CoP does generally not coincide with the CMP.

5.3 Mature method for DCM planning and control

In the previous sections, the LIP model was recapitulated and basic methods for Capture Point based trajectory generation and feedback control were presented. Then, the three-dimensional Divergent Component of Motion (DCM) and a useful force-to-point transformation was introduced, which resulted in the definitions of the Enhanced Centroidal Moment Pivot point (eCMP) and the

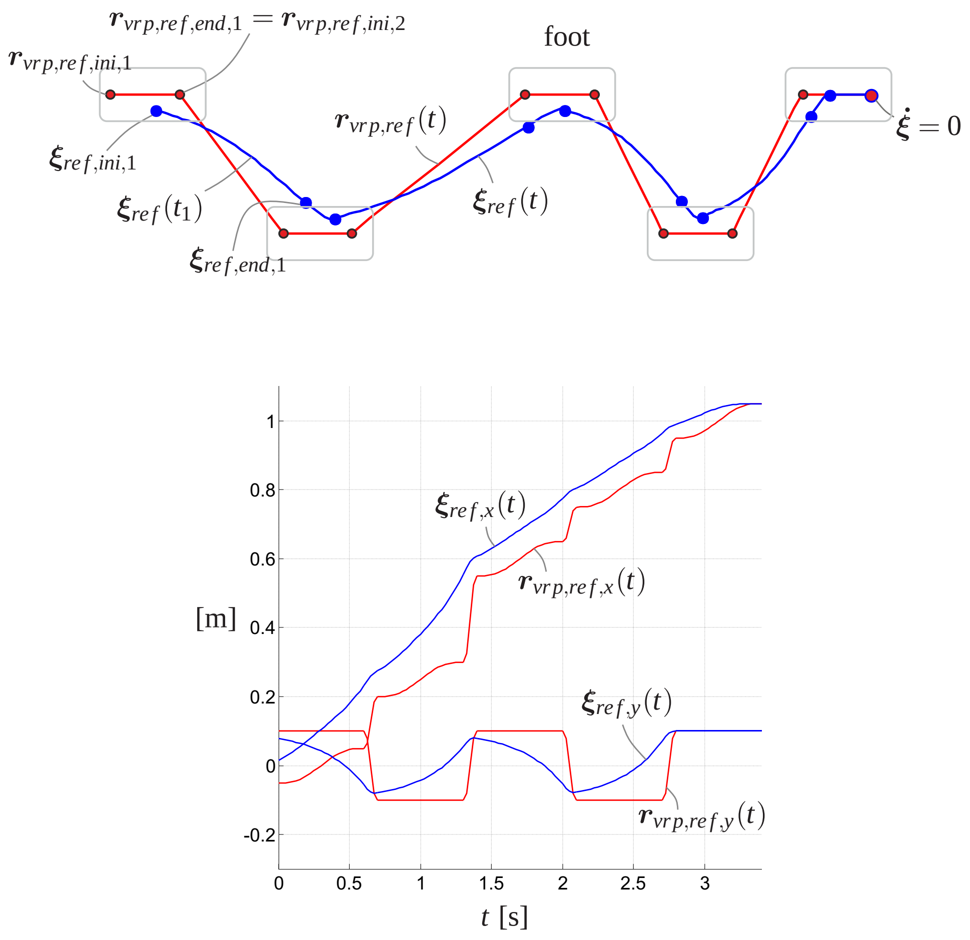

Figure 5.5: Generation of VRP/eCMP and DCM references; trajectories in space and time.

Virtual Repellent Point (VRP). These definitions had helped to raise the two-dimensional concepts of Capture Point [21, 24] and ZMP/CMP to the third dimension. The following sections will provide a detailed derivation and overview of the author's current state of research regarding DCM-based planning and control for three-dimensional walking.

5.3.1 Planning smooth DCM reference trajectories for multi-step previews

This section will provide a method for smooth reference trajectory generation. In the author's view, two key criteria for a sound walking reference generator are that it includes a preview of sufficiently many (for typical dynamic walking cases about three or more) future steps in order to avoid/ease discontinuities at the contact transitions and that it is well feasible with regard to the previewed contact sequences at hand. Feasibility here relates to the question if the forces that correlate to the reference trajectories can be generated by the available contacts, which is typically expressed as unilaterality constraints (no forces pulling towards ground allowed) or more generally as friction cone constraints. To reduce complexity, walking reference generators typically work with the assumption of zero nominal torque around the CoM. With that constraint and assuming sufficient friction, the feasibility problem in walking generators is reduced to the problem of finding eCMP trajectories (corresponding to ZMP trajectories in 2D) that lie well within the previewed base(s) of support. This work designs DCM reference trajectories that correspond to feasible eCMP trajectories while fulfilling a certain *terminal constraint*. To comply with minimal bases of support such as point or line contacts, the desired eCMP trajectories are implemented as smooth piecewise point-to-point interpolations. These are encoded via the following $(n_{prev}-1)$ fifth order polynomials between the n_{prev} previewed eCMP waypoints (see Fig. 5.5):

$$\boldsymbol{r}_{ecmp,ref,i}(t_i) = \boldsymbol{r}_{ecmp,ref,ini,i} + \left(\frac{t_i}{T_i}\right)^3 \left(10 - \frac{15\,t_i}{T_i} + 6\,\frac{t_i^2}{T_i^2}\right) (\boldsymbol{r}_{ecmp,ref,end,i} - \boldsymbol{r}_{ecmp,ref,ini,i}) \tag{5.34}$$

$$\dot{\boldsymbol{r}}_{ecmp,ref,i}(t_i) = \frac{30\,t_i^2}{T_i^3} \left(1 - \frac{t_i}{T_i}\right)^2 (\boldsymbol{r}_{ecmp,ref,end,i} - \boldsymbol{r}_{ecmp,ref,ini,i}) \tag{5.35}$$

$$\ddot{\boldsymbol{r}}_{ecmp,ref,i}(t_i) = \frac{60\,t_i}{T_i^3} \left(1 - \frac{3\,t_i}{T_i} + \frac{2\,t_i^2}{T_i^2}\right) (\boldsymbol{r}_{ecmp,ref,end,i} - \boldsymbol{r}_{ecmp,ref,ini,i}) \tag{5.36}$$

Here, $\boldsymbol{r}_{ecmp,ref,ini,i}$ denotes the starting eCMP waypoint (red points in Fig. 5.5) and $\boldsymbol{r}_{ecmp,ref,end,i}$ the end eCMP waypoint of the i-th transition phase[2]. Each time in transition is denoted by $t_i \in [0, T_i]$, where T_i is the total time for each transition phase. Equations (5.34)-(5.36) fulfill the following boundary conditions:

$$\begin{bmatrix} \boldsymbol{r}_{ecmp,ref,i}(0) \\ \dot{\boldsymbol{r}}_{ecmp,ref,i}(0) \\ \ddot{\boldsymbol{r}}_{ecmp,ref,i}(0) \\ \boldsymbol{r}_{ecmp,ref,i}(T_i) \\ \dot{\boldsymbol{r}}_{ecmp,ref,i}(T_i) \\ \ddot{\boldsymbol{r}}_{ecmp,ref,i}(T_i) \end{bmatrix} = \begin{bmatrix} \boldsymbol{r}_{ecmp,ref,ini,i} \\ 0 \\ 0 \\ \boldsymbol{r}_{ecmp,ref,end,i} \\ 0 \\ 0 \end{bmatrix} . \tag{5.37}$$

Both eCMP velocity and acceleration start from and end with zero, i.e., (5.35) smoothly interpolates between $\boldsymbol{r}_{ecmp,ref,ini,i}$ and $\boldsymbol{r}_{ecmp,ref,end,i}$ during the i-th transition phase. The term “transition phase” is used here instead of “step”, since a single step may consist of several transition phases (such as a heel-to-toe transition followed by another transition from toe to upcoming heel, or more complex). With (5.34) and (5.28), the corresponding VRP reference trajectories become

$$\boldsymbol{r}_{vrp,ref,i}(t_i) = \boldsymbol{r}_{ecmp,ref,i}(t_i) + [0\;\;0\;\;\Delta z_{vrp}]^T \;. \tag{5.38}$$

Now, to find the DCM reference trajectories that fit these smooth VRP trajectories, the following ordinary differential equation (ODE) (as in (5.30)) has to be solved:

$$\dot{\boldsymbol{\xi}}_{ref,i}(t_i) = \frac{1}{b}\,(\boldsymbol{\xi}_{ref,i}(t_i) - \boldsymbol{r}_{vrp,ref,i}(t_i)) \tag{5.39}$$

As boundary conditions, the DCM positions at the end of each transition phase $\boldsymbol{\xi}_{ref,i}(T_i) = \boldsymbol{\xi}_{ref,end,i}$ are used. The solution to ODE (5.39) is

$$\boldsymbol{\xi}_{ref,i}(t_i) = \boldsymbol{r}_{vrp,ref,ini,i} + \varepsilon_i(t_i)\,(\boldsymbol{\xi}_{ref,end,i} - \boldsymbol{r}_{vrp,ref,ini,i}) + s_i(t_i)\,(\boldsymbol{r}_{vrp,ref,end,i} - \boldsymbol{r}_{vrp,ref,ini,i}) \tag{5.40}$$

where $\varepsilon_i(t_i) = e^{\frac{t_i - T_i}{b}}$ and $s_i(t_i)$ is

$$\begin{aligned} s_i(t_i) \;&=\; \frac{1}{T_i^3}\Big(60\,b^2\,\gamma_i\,(t_i + b - \varepsilon_i\,(T_i + b)) + 30\,b\,\gamma_i\,(t_i^2 - \varepsilon_i\,T_i^2) + \\ &+\; 10\,\gamma_i\,(t_i^3 - \varepsilon_i\,T_i^3) + \Big(-\frac{15}{T_i} + \frac{30\,b}{T_i^2}\Big)(t_i^4 - \varepsilon_i\,T_i^4) + \frac{6}{T_i^2}\,(t_i^5 - \varepsilon_i\,T_i^5)\Big) \end{aligned} \tag{5.41}$$

[2]Note: in the more trivial case that the eCMP is supposed to *remain* at a certain eCMP waypoint during a “transition” phase, that eCMP waypoint is simply used as start and end point in equations (5.34)-(5.36)

with $\gamma_i = 1 - \frac{6\,b}{T_i} + \frac{12\,b^2}{T_i^2}$. To determine the actual solution for the DCM trajectories, the DCM trajectory are designed to nominally come to a stop at the n_{prev}-th previewed VRP waypoint, i.e., $\boldsymbol{\xi}_{ref,end,n_{prev}-1}(T_{n_{prev}-1}) = \boldsymbol{r}_{vrp,ref,end,n_p rev}$. This means that the robot "thinks" it will come to a stop after n_{prev} steps. Yet, due to the receding horizon, this terminal constraint is not forcingly required to ever be reached. Given the terminal constraint and using (5.40), the DCM trajectories are iterated backwards from the final (i.e. $(n_{prev}-1)$-th) to the first transition phase[3]. For all future steps other than the current one, it is sufficient to evaluate (5.40) for $t_i = 0$, i.e.,

$$\boldsymbol{\xi}_{ref,ini,i} = \boldsymbol{\xi}_{ref,i}(0) = \boldsymbol{r}_{vrp,ref,ini,i} + \varepsilon_{0,i}\,(\boldsymbol{\xi}_{ref,end,i} - \boldsymbol{r}_{vrp,ref,ini,i}) + s_{0,i}\,(\boldsymbol{r}_{vrp,ref,end,i} - \boldsymbol{r}_{vrp,ref,ini,i}) \tag{5.42}$$

where $\varepsilon_{0,i} = e^{\frac{-T_i}{b}}$ and $s_{0,i} = s_i(t_i = 0)$. In the presented backwards iteration, the final DCM reference position of the $(i-1)$-th transition phase is the i-th initial DCM reference position:

$$\boldsymbol{\xi}_{ref,end,i-1} = \boldsymbol{\xi}_{ref,ini,i} \quad . \tag{5.43}$$

Using (5.42) and (5.43) as backward iteration, all "end of step" reference DCMs $\boldsymbol{\xi}_{ref,end,i}$ are found until the first one. The first "end of step" reference DCM $\boldsymbol{\xi}_{ref,end,1}$ is used in (5.40) to compute the DCM reference position $\boldsymbol{\xi}_{ref,1}(t_1)$ for the current time in transition $t_1 \in [0, T_1]$:

$$\boldsymbol{\xi}_{ref,1}(t_1) = \boldsymbol{r}_{vrp,ref,ini,1} + \varepsilon_1(t_1)\,(\boldsymbol{\xi}_{ref,end,1} - \boldsymbol{r}_{vrp,ref,ini,1}) + s_1(t_1)\,(\boldsymbol{r}_{vrp,ref,end,1} - \boldsymbol{r}_{vrp,ref,ini,1}) \tag{5.44}$$

With $\boldsymbol{\xi}_{ref,1}(t_1)$, the current VRP reference $\boldsymbol{r}_{vrp,ref,1}(t_1)$ (computed from (5.34) and (5.38)). These DCM reference trajectories are drawn blue in figure 5.5. With (5.40), the current DCM reference velocity can be computed as

$$\dot{\boldsymbol{\xi}}_{ref,1}(t_1) = \frac{1}{b}\,(\boldsymbol{\xi}_{ref,1}(t_1) - \boldsymbol{r}_{vrp,ref,1}(t_1)) \quad . \tag{5.45}$$

The current DCM reference position $\boldsymbol{\xi}_{ref,1}(t_1)$ and velocity $\dot{\boldsymbol{\xi}}_{ref,1}(t_1)$ are used in the DCM tracking controller, which will be presented in the next section.

5.3.2 Three-dimensional DCM tracking control

By choosing the deviation $\Delta\boldsymbol{x} = \boldsymbol{x} - \boldsymbol{\xi}$ of the CoM from the DCM and the DCM error $\Delta\boldsymbol{\xi} = \boldsymbol{\xi} - \boldsymbol{\xi}_{ref,1}$ as new states, the open-loop system dynamics (5.31) can be rewritten as

$$\begin{bmatrix}\Delta\dot{\boldsymbol{x}}\\ \Delta\dot{\boldsymbol{\xi}}\end{bmatrix} = \begin{bmatrix}-\frac{1}{b} & -\frac{1}{b}\\ 0 & \frac{1}{b}\end{bmatrix}\begin{bmatrix}\Delta\boldsymbol{x}\\ \Delta\boldsymbol{\xi}\end{bmatrix} + \begin{bmatrix}\frac{\boldsymbol{r}_{vrp}-\boldsymbol{\xi}_{ref,1}}{b}\\ \frac{\boldsymbol{r}_{vrp,ref,1}-\boldsymbol{r}_{vrp}}{b}\end{bmatrix} \quad , \tag{5.46}$$

where $\boldsymbol{r}_{vrp}$ is the control input. Note that the matrix elements $\pm\frac{1}{b}$ could be replaced by matrices $\boldsymbol{B}^{-1}$, which might assign different dynamics to each spatial direction. In this work, all spatial directions are treated the same, such that the scalar notation is chosen. To stabilize the open-loop dynamics (5.46), the following feedback controller is applied:

$$\boldsymbol{r}_{vrp,des} = \boldsymbol{r}_{vrp,ref,1} + [r_1 \;\; r_2]\begin{bmatrix}\Delta\boldsymbol{x}\\ \Delta\boldsymbol{\xi}\end{bmatrix} \quad , \tag{5.47}$$

which with (5.45) leads to the following closed-loop dynamics:

$$\begin{bmatrix}\Delta\dot{\boldsymbol{x}}\\ \Delta\dot{\boldsymbol{\xi}}\end{bmatrix} = \underbrace{\begin{bmatrix}\frac{r_1-1}{b} & \frac{r_2-1}{b}\\ -\frac{r_1}{b} & -\frac{r_2-1}{b}\end{bmatrix}}_{\boldsymbol{A}_{x,\xi}}\begin{bmatrix}\Delta\boldsymbol{x}\\ \Delta\boldsymbol{\xi}\end{bmatrix} + \underbrace{\begin{bmatrix}-\dot{\boldsymbol{\xi}}_{ref,1}\\ 0\end{bmatrix}}_{\Delta_{feedforward}} \quad . \tag{5.48}$$

[3] As mentioned above: the i-th *transition phase* denotes the transition from the i-th to the $(i+1)$-th contact point.

The eigenvalues of the system matrix $\boldsymbol{A}_{x,\xi}$ are

$$\lambda_{1,2} = \frac{r_1 - r_2 \pm \sqrt{r_1^2 - 2\, r_1\, r_2 + (r_2-2)^2}}{2\, b} \quad . \tag{5.49}$$

Solving for the control gains r_1 and r_2 yields

$$r_1 = (1 + b\,\lambda_1)(1 + b\,\lambda_2) \tag{5.50}$$

$$r_2 = 1 + b^2\,\lambda_1\,\lambda_2 \;, \tag{5.51}$$

which can be seen as pole placement equations. For $\lambda_1 < 0$ and $\lambda_2 < 0$ the system is stable. This choice for the control gains leads to the following VRP feedback control law:

$$\boldsymbol{r}_{vrp,des} = \boldsymbol{r}_{vrp,ref,1} + (1 + b\,\lambda_1)(1 + b\,\lambda_2)\,(\boldsymbol{x} - \boldsymbol{\xi}) + (1 + b^2\,\lambda_1\,\lambda_2)\,(\boldsymbol{\xi} - \boldsymbol{\xi}_{ref,1}) \;, \tag{5.52}$$

which can be transformed into a desired CoM force via (5.32) as

$$\boldsymbol{F}_{com,des} = -m\,\lambda_1\,\lambda_2\,(\boldsymbol{x} - \boldsymbol{\xi}_{ref,1}) \;+\; m\,(\lambda_1 + \lambda_2)\,\dot{\boldsymbol{x}} \;+\; \frac{m}{b}\,\dot{\boldsymbol{\xi}}_{ref,1} \;. \tag{5.53}$$

Comparison to second order force-based tracking controllers

Equation (5.53) shows certain similarities to standard force-based tracking controllers. For the pure *regulation* case (i.e., $\boldsymbol{\xi}_{ref,1} = \boldsymbol{x}_{ref}$ and $\dot{\boldsymbol{\xi}}_{ref,1} = 0$), the desired force becomes

$$\boldsymbol{F}_{com,des} = -\underbrace{m\,\lambda_1\,\lambda_2}_{k_p}\,(\boldsymbol{x} - \boldsymbol{x}_{ref}) \;+\; \underbrace{m\,(\lambda_1 + \lambda_2)}_{-k_d}\,\dot{\boldsymbol{x}} \;, \tag{5.54}$$

i.e., the controller shows the exact same behavior as a standard force-based PD regulation controller. For that case, any desired PD-gains k_p and k_d can be achieved[4].

In some use cases, the DCM-based controller might be used to *track a general second order CoM reference* $[\boldsymbol{x}_{ref}, \dot{\boldsymbol{x}}_{ref}, \ddot{\boldsymbol{x}}_{ref}]$. To this end, the input reference is converted into corresponding DCM references $\boldsymbol{\xi}_{ref,1} = \boldsymbol{x}_{ref} + b\,\dot{\boldsymbol{x}}_{ref}$ and $\dot{\boldsymbol{\xi}}_{ref,1} = \dot{\boldsymbol{x}}_{ref} + b\,\ddot{\boldsymbol{x}}_{ref}$, which inserted into (5.53) yields

$$\boldsymbol{F}_{com,des} = -\underbrace{m\lambda_1\lambda_2}_{k_p}\,(\boldsymbol{x} - \boldsymbol{x}_{ref}) \;+\; \underbrace{m(\lambda_1 + \lambda_2)}_{k_d}\,(\dot{\boldsymbol{x}} - \dot{\boldsymbol{x}}_{ref}) \;+\; m\ddot{\boldsymbol{x}}_{ref} + \underbrace{\frac{m}{b}\,(1 + b\lambda_1)(1 + b\lambda_2)\,\dot{\boldsymbol{x}}_{ref}}_{\boldsymbol{F}_{\dot{\boldsymbol{x}}_{ref}}} \;. \tag{5.55}$$

It turns out that the corresponding controller differs from a standard force-based PD tracking controller by the term $\boldsymbol{F}_{\dot{\boldsymbol{x}}_{ref}}$. That extra term would corrupt the desired tracking behavior. In order for that term to vanish (i.e. $\boldsymbol{F}_{\dot{\boldsymbol{x}}_{ref}} = 0$), one of the two eigenvalues λ_1 and λ_2 has to be chosen as $-\frac{1}{b}$. Here, $\lambda_1 = -\frac{1}{b}$ and $\lambda_2 = -k_\xi$ is chosen, such that (5.55) turns into

$$\boldsymbol{F}_{com,des} = -\underbrace{\frac{m\,k_\xi}{b}}_{k_p}\,(\boldsymbol{x} - \boldsymbol{x}_{ref}) \;-\; \underbrace{m\left(\frac{1}{b} + k_\xi\right)}_{k_d}\,(\dot{\boldsymbol{x}} - \dot{\boldsymbol{x}}_{ref}) \;+\; m\,\ddot{\boldsymbol{x}}_{ref} \;, \tag{5.56}$$

[4]Note: PD gains that correspond to imaginary eigenvalues may be undesirable since they can lead to overswinging transient responses. This might make the desired eCMP leave the base of support and thus conflict with feasibility.

which achieves the desired tracking behavior for a second order CoM reference trajectory. It has to be noted though that for the particular choice of $\lambda_1 = -\frac{1}{b}$ the PD gains k_p and k_d are *coupled*:

$$k_d = b\, k_p + \frac{m}{b} \quad . \tag{5.57}$$

Note that (for $k_p > 0$) a lower bound for the damping gain $k_{d,min} = \frac{m}{b}$ exists. In practical applications this may cause problems since the sensitivity against noise in the CoM velocity estimate cannot be eased by choosing arbitrarily low damping gains k_d.

Specific choice of system eigenvalues

As shown in the previous section, while arbitrary eigenvalues for the system matrix $\boldsymbol{A}_{x,\xi}$ from (5.48) may be chosen theoretically, for proper tracking of a second order CoM reference one of the eigenvalues has to be chosen as $\lambda_1 = -\frac{1}{b}$. As shown later, the same choice is desirable for the DCM controller since other choices would lead to deviations from the desired DCM tracking dynamics. To this end, the left hand side of (5.48) is set to zero and - using the control gains r_1 and r_2 from (5.50) and (5.51) - solved for the steady state deviations

$$\begin{bmatrix} \Delta\boldsymbol{x} \\ \Delta\boldsymbol{\xi} \end{bmatrix}_{steadystate} = \underbrace{\begin{bmatrix} -b & -b \\ \frac{(1+b\lambda_1)(1+b\lambda_2)}{b\lambda_1\lambda_2} & \frac{\lambda_1+\lambda_2+b\lambda_1\lambda_2}{\lambda_1\lambda_2} \end{bmatrix}}_{\boldsymbol{A}_{x,\xi}^{-1}} \underbrace{\begin{bmatrix} -\dot{\boldsymbol{\xi}}_{ref,1} \\ 0 \end{bmatrix}}_{\Delta_{feedforward}} \quad . \tag{5.58}$$

It turns out that for the steady state DCM error $\Delta\boldsymbol{\xi}_{steadystate}$ to vanish, one of the two eigenvalues has to be chosen as $-\frac{1}{b}$. This is the exact same condition as the one from the previous section on second order CoM trajectory tracking. Again $\lambda_1 = -\frac{1}{b}$ and $\lambda_2 = -k_\xi$ are chosen. This leads to new control gains $r_1 = 0$ and $r_2 = 1 + b\, k_\xi$. The closed-loop system dynamics (5.48) becomes

$$\begin{bmatrix} \Delta\dot{\boldsymbol{x}} \\ \Delta\dot{\boldsymbol{\xi}} \end{bmatrix} = \underbrace{\begin{bmatrix} -\frac{1}{b} & k_\xi \\ 0 & -k_\xi \end{bmatrix}}_{\boldsymbol{A}_{x,\xi}} \begin{bmatrix} \Delta\boldsymbol{x} \\ \Delta\boldsymbol{\xi} \end{bmatrix} + \underbrace{\begin{bmatrix} -\dot{\boldsymbol{\xi}}_{ref,1} \\ 0 \end{bmatrix}}_{\Delta_{feedforward}} \quad . \tag{5.59}$$

The first row of (5.59) can also be expressed as $\dot{\boldsymbol{x}} = -\frac{1}{b}\,(\boldsymbol{x} - \boldsymbol{\xi})$. This is equivalent to the naturally stable CoM dynamics (5.21) that is not affected by the controller. In contrast, the instable DCM dynamics (5.30) is stabilized which results in the following stable first order dynamics:

$$\underbrace{\dot{\boldsymbol{\xi}} - \dot{\boldsymbol{\xi}}_{ref,1}}_{\Delta\dot{\boldsymbol{\xi}}} = -k_\xi\, (\underbrace{\boldsymbol{\xi} - \boldsymbol{\xi}_{ref,1}}_{\Delta\xi}), \tag{5.60}$$

It is obvious that for $k_\xi > 0$ the system is stable. It has to be noted, that this stability analysis neglects physical limitations such as limited base of support. The VRP control law (5.52) becomes

$$\boldsymbol{r}_{vrp,des} = \boldsymbol{r}_{vrp,ref,1} + (1 + b\, k_\xi)\, (\underbrace{\boldsymbol{\xi} - \boldsymbol{\xi}_{ref,1}}_{\Delta\xi}) \quad . \tag{5.61}$$

Just as in the corresponding Capture Point control law (5.19), the desired VRP is located at a certain deviation from the current VRP reference position $\boldsymbol{r}_{vrp,ref,1}$. Again, the DCM error $\Delta\xi$ is scaled by a factor $(1 + b\, k_\xi) > 1$ and added to the nominal VRP reference, which yields the desired DCM convergence. As the DCM error $\Delta\xi$ converges asymptotically, also the commanded VRP

$\boldsymbol{r}_{vrp,des}$ and its corresponding eCMP $\boldsymbol{r}_{ecmp,des}$ converge to their references $\boldsymbol{r}_{vrp,ref,1}$ and $\boldsymbol{r}_{ecmp,ref,1}$ after a perturbation. The force-based DCM control law (5.53) can finally be written as

$$\begin{aligned} \boldsymbol{F}_{com,des} &= -\frac{m\,k_\xi}{b}\,(\boldsymbol{x}-\boldsymbol{\xi}_{ref,1}) \;-\; m\left(\frac{1}{b}+k_\xi\right)\dot{\boldsymbol{x}} \;+\; \frac{m}{b}\,\dot{\boldsymbol{\xi}}_{ref,1} \\ &= \frac{m}{b^2}\left(\boldsymbol{x}-\boldsymbol{r}_{vrp,ref,1}-(1+b\,k_\xi)(\boldsymbol{\xi}-\boldsymbol{\xi}_{ref,1})\right) \;. \end{aligned} \tag{5.62}$$

Finally, the desired external (e.g. leg-) force $\boldsymbol{F}_{ext,des}$ becomes

$$\boldsymbol{F}_{ext,des} = \frac{m}{b^2}\left(\boldsymbol{x}-\boldsymbol{r}_{vrp,ref,1}-(1+b\,k_\xi)(\boldsymbol{\xi}-\boldsymbol{\xi}_{ref,1})\right) - \boldsymbol{F}_g \;. \tag{5.63}$$

Note that the only equations that are finally needed are (5.34), (5.42), (5.43) and (5.44) for three-dimensional DCM trajectory generation and (5.63) for force-based DCM tracking control. They can easily be computed in real-time on any computer.

If no actuation limits are violated and no unknown perturbations occur, the DCM tracking controller (5.63) is asymptotically stable. In [30], its robustness w.r.t. CoM error, unknown external perturbation forces, unknown force lag and errors in the total robot mass estimation was examined. It turns out that the nominal controller is powerful and well-behaved even under the mentioned imperfections. However, in case of severe perturbations, the physical constraints of the robot, such as its limited base of support, may corrupt the nominal performance of the controller and lead to divergence of the robot state. The next section introduces a method for footstep adjustment that considerably increases the robustness of the robot gait.

5.3.3 Increasing walking robustness via step adjustment

Depending on the perturbation at hand, the whole-body controller presented in chapter 4 translates the desired force $\boldsymbol{F}_{ext,des}$ from the DCM controller (5.63) into a leg force $\boldsymbol{F}_{leg}$ that either doesn't or does produce a torque τ_{com} around the CoM. These two modulation modes are referred to as "ankle strategy" and "hip strategy", although in both cases typically all robot joints are involved. However, in case of continuous or strong perturbations, the desired eCMPs (corresponding to the desired VRPs from (5.61)) are not necessarily feasible, i.e., within or close to the base of support. Thus, modification of eCMP within (or close to) the current and future preplanned bases of support may not suffice to keep the robot's CoM from diverging. Unlike the CoP, it is physically feasible for the eCMP to (temporarily) leave the base of support. Yet, the associated torque around the CoM leads to growing angular momentum and thus clashes with the body posture task (see chapter 4) in the long run. Therefore, the eCMP should usually be held inside the base of support, which in turn limits the possible controls actions.

An additional powerful stabilization mechanism is the adjustment of footsteps, i.e., the positions of one or more future footsteps - or more generally speaking: contact points - are adjusted such that the new contact situation is more suitable for stabilizing the given CoM state. Additionally, the step adjustment should lead to fast and smooth recovery to the original reference trajectories and preplanned contacts. To this end, in this section, a method for footstep adjustment (see Fig. 5.7) will be derived that leads to highly increased robustness against external perturbations.

The main idea here is to adjust the foot target locations in such a way that the new initial DCM reference location at the moment of step transition (or more precisely: at the switch to the first transition phase that is affected by the contact point adjustments) is equivalent to the previewed DCM at that moment. To this end, the effect an arbitrary adjustment of the adjustable subset of contact points on the corresponding initial DCM reference positions $\boldsymbol{\xi}_{ref,ini,i}$ is first computed and then the achieved relation is inverted. It would be possible to use multiple different contact

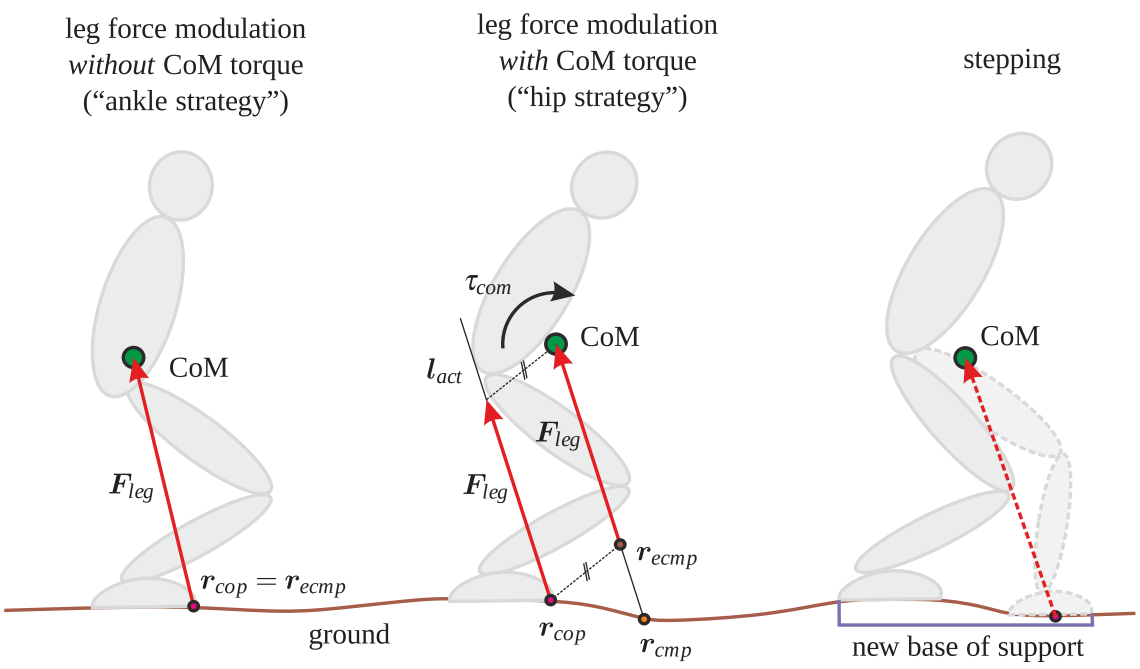

Figure 5.6: Stabilization strategies.

point adjustments, but for reasons of brevity and simplicity in the following derivation, this work assumes that all adjustable contact points are adjusted by the same offset $\Delta\boldsymbol{r}_{ecmp}$ (violett arrows in Fig. 5.7). To distinguish adjustable contact points from non-adjustable ones, an adjustment vector $\boldsymbol{v}_{adj} \in \mathbb{R}^{1\times n_c}$ is defined that indicates for all n_c contact points if they are adjustable ($\rightarrow v_{adj,i} = 1$) or non-adjustable ($\rightarrow v_{adj,i} = 0$). For *each adjustable* contact point, $\Delta\boldsymbol{r}_{ecmp}$ is added to the corresponding initial and final eCMP positions $\boldsymbol{r}_{ecmp,ref,ini,i}$ and $\boldsymbol{r}_{ecmp,ref,end,i}$ and recompute equation (5.42). That way, all adjusted initial DCM reference positions $\boldsymbol{\xi}_{ref,ini,adjusted,i}$ (i.e., the ones that incorporate the step adjustment $\Delta\boldsymbol{r}_{ecmp}$) can be computed via

$$\boldsymbol{\xi}_{ref,ini,adjusted,i} = \boldsymbol{\xi}_{ref,ini,i} + \alpha_i\, \Delta\boldsymbol{r}_{ecmp} \quad , \tag{5.64}$$

where the i-th adjustment gain α_i is denoted by

$$\alpha_i = v_{adj,i} + \varepsilon_{0,i}\left(v_{adj,i+1}\ \alpha_{i+1} - v_{adj,i}\right) + s_{0,i}\left(v_{adj,i+1} - v_{adj,i}\right) \ . \tag{5.65}$$

That way all n_c adjustment gains α_i are found by starting with $\alpha_{n_c} = 0$ and propagating backwards. As in (5.43), also the DCM reference is propagated backwards via $\boldsymbol{\xi}_{ref,end,i} = \boldsymbol{\xi}_{ref,ini,adjusted,i+1}$. Equation (5.64) would return the adjusted initial DCMs if the step adjustment was known. As mentioned above, here the goal is the opposite namely to find an appropriate step adjustment $\Delta\boldsymbol{r}_{ecmp}$ which corresponds to an adjustment of the original DCM trajectory. This adjustment should lead to perfect cancellation of the previewed initial DCM tracking error $\Delta\boldsymbol{\xi}_{ini,FA,previewed}$ at the beginning of the first transition phase that is affected by the step adjustment ("FA" stands for "first affected"). In other words: the starting point $\boldsymbol{\xi}_{ref,ini,FA}$ of the first affected DCM trajectory is adjusted to equal the previewed DCM position $\boldsymbol{\xi}_{ini,previewed}$ at the switching instant. That way any DCM tracking error that has accumulated until the end of the last non-modifiable transition phase will vanish (with respect to the adjusted DCM reference trajectory) after the step adjustment. Reordering (5.64) and

$$\Delta\boldsymbol{\xi}_{ref,ini,FA} = \boldsymbol{\xi}_{ref,ini,adjusted,FA} - \boldsymbol{\xi}_{ref,ini,FA} = \Delta\boldsymbol{\xi}_{ini,FA,previewed} \tag{5.66}$$

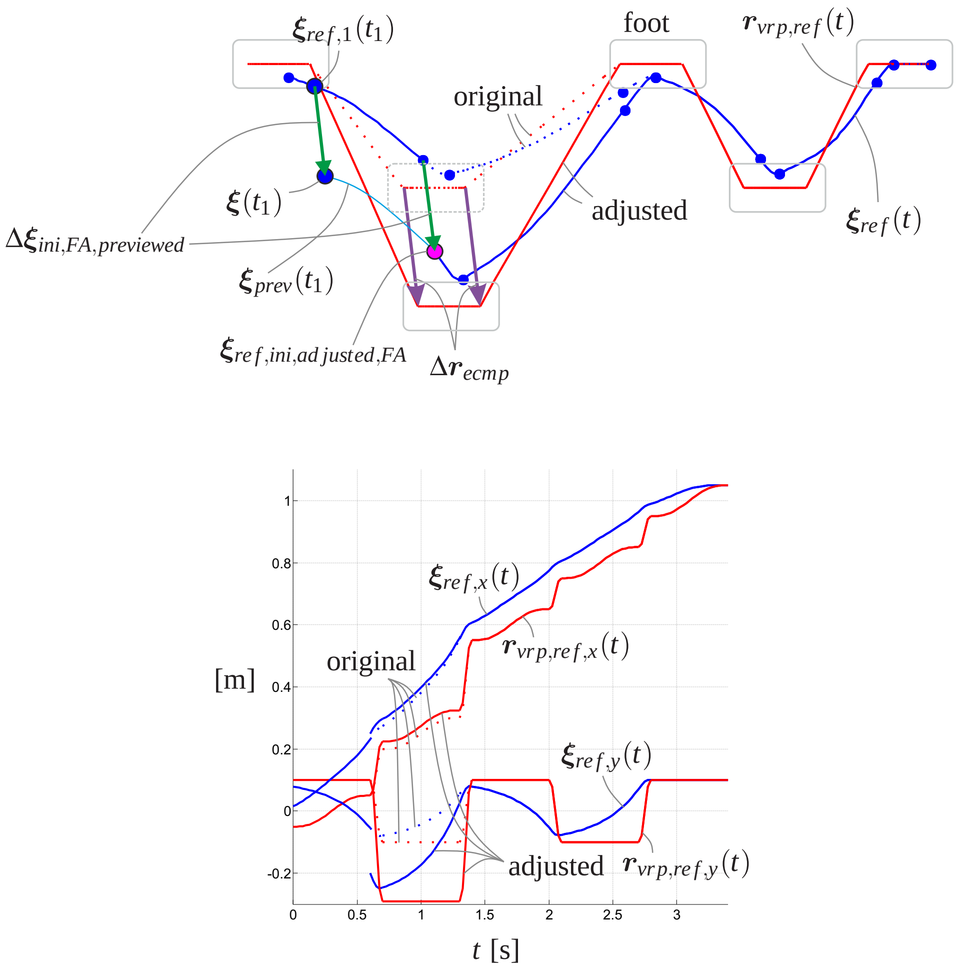

Figure 5.7: Step adjustment; trajectories in space and time.

yields the step adjustment (violett arrows in Fig. 5.7)

$$\Delta\boldsymbol{r}_{ecmp} = \frac{\Delta\boldsymbol{\xi}_{ini,FA,previewed}}{\alpha_{FA}} = \frac{\boldsymbol{\xi}(t_1) - \boldsymbol{\xi}_{ref,1}(t_1)}{\alpha_{FA}} \quad . \tag{5.67}$$

Here, the assumption that the initial DCM error $\Delta\boldsymbol{\xi}_{ini,FA,previewed}$ at the transition phase switch (green arrows in Fig. 5.7) will be equal to the *current* DCM error $\Delta\boldsymbol{\xi}(t_1) = \boldsymbol{\xi}(t_1) - \boldsymbol{\xi}_{ref,1}(t_1)$ is already inserted, i.e., the current DCM error is assumed to persist until the switch (see also $\boldsymbol{\xi}_{prev}(t_1)$ (bright blue curve) in Fig. 5.7). This assumption is a compromise between a more conservative approach that might preview further grows of the DCM error (e.g. due to under-actuation) and a more optimistic perspective that might preview DCM error convergence (e.g. following the nominal DCM tracking control dynamics). That way, the chosen approach is less sensitive and prone to noise as compared to the conservative approach while being less inert against actual perturbations as compared to the optimistic approach. In any case, at the instant of transition phase switching, the assumption $\Delta\boldsymbol{\xi}_{ini,FA,previewed} = \Delta\boldsymbol{\xi}(t_1)$ holds true, since for that instant the preview horizon has decreased to zero. Equation (5.67) is the main result of this section. From the current DCM tracking error $\Delta\boldsymbol{\xi}(t_1) = \boldsymbol{\xi}(t_1) - \boldsymbol{\xi}_{ref,1}(t_1)$ the appropriate step adjustment $\Delta\boldsymbol{r}_{ecmp}$ is computed that allows for state recovery even after strong perturbations.

Despite the discrete step adjustment the proposed method leads to smooth and continuous VRP/eCMP reference trajectories, because the step adjustment (and the corresponding adjustment of the swing foot trajectory) occurs *before* the first phase transition is initiated which is affected by the adjustment (see also Fig. 5.7). If no further perturbation occurs, DCM and VRP are smoothly routed back to comply with the originally planned footsteps (or contact points).

5.3.4 Increasing robustness against strong continuous perturbations

To increase robustness, a third stabilization method is applied. With the methods derived so far, a steady state deviation of both DCM and VRP (and eCMP accordingly) arises if the perturbation force persists. The desired eCMP from the DCM tracking law may even become unfeasible (or hard to produce for the QP) which may lead to a fall sooner or later. In this work, the solution is to estimate the external force via a disturbance observer and explicitly account for it in the DCM dynamics. The derivations for the DCM dynamics have to be adjusted accordingly to account for the external force. That way, the DCM reference trajectory is modified such that it “leans against the external force”, i.e., gravity compensates for the external force.

Momentum-based disturbance observer

In this section, a momentum-based disturbance observer will be derived which is based on the ideas of DeLuca [94]. Newton’s 2nd law gives a relation between the linear momentum $\boldsymbol{p} = m\,\dot{\boldsymbol{x}}$ and the total force $\boldsymbol{F}_{com}$ acting on the CoM as

$$\dot{\boldsymbol{p}} = \boldsymbol{F}_{com} = \boldsymbol{F}_{known} + \boldsymbol{F}_{perturb} \ , \tag{5.68}$$

i.e., the rate of change of the linear momentum $\dot{\boldsymbol{p}}$ equals the forces acting on the CoM. The CoM force $\boldsymbol{F}_{com}$ is composed of a known part $\boldsymbol{F}_{known}$ and an unknown perturbation force $\boldsymbol{F}_{perturb}$. If $\boldsymbol{F}_{perturb}$ would be known (e.g. measured) it could directly be included in the controller. This work assumes that it cannot be measured directly. Therefore, it will be estimated using a momentum-based observer. For bipedal walking the sum of forces acting on the CoM is typically assumed to be $\boldsymbol{F}_{known} = \boldsymbol{F}_{ext} + \boldsymbol{F}_{g}$ (i.e., the sum of all known external forces (e.g. leg forces) $\boldsymbol{F}_{ext}$ and the gravitational force $\boldsymbol{F}_{g}$). For brevity, the combined expression $\boldsymbol{F}_{known}$ will be used in this section. Equation (5.68) is reused for the observer:

$$\dot{\hat{\boldsymbol{p}}} = \boldsymbol{F}_{known} + \hat{\boldsymbol{F}}_{perturb} \ . \tag{5.69}$$

Here, $\dot{\hat{\boldsymbol{p}}}$ is the corresponding estimated rate of change of linear momentum and $\hat{\boldsymbol{F}}_{perturb}$ denotes the estimated perturbation force. Since no a priori knowledge about $\boldsymbol{F}_{perturb}$ exists, it is - for the observer design - assumed to be constant, i.e., its derivative to be zero. This motivates the formulation of the following desired observer dynamics:

$$\dot{\hat{\boldsymbol{F}}}_{perturb} = k_F\ (\boldsymbol{F}_{perturb} - \hat{\boldsymbol{F}}_{perturb}) \ , \tag{5.70}$$

which for $k_F > 0$ describes a stable PT1 behavior, i.e., the estimated perturbation force $\hat{\boldsymbol{F}}_{perturb}$ converges towards $\boldsymbol{F}_{perturb}$ if the latter is constant and follows it with a lag otherwise. An exact and lag-free estimation of non-constant perturbation forces would require knowledge on their derivatives and is thus not considered here. Integrating the desired behavior (5.70) yields

$$\hat{\boldsymbol{F}}_{perturb}(t) = k_F \left(\int_0^t \boldsymbol{F}_{perturb}(t)\, dt\ - \int_0^t \hat{\boldsymbol{F}}_{perturb}(t)\, dt \right) \ . \tag{5.71}$$

The actual and estimated momentum can be computed via integration of (5.68) and (5.69):

$$\boldsymbol{p}(t) = \boldsymbol{p}_0 + \int_0^t (\boldsymbol{F}_{known}(t) + \boldsymbol{F}_{perturb}(t))\, dt \tag{5.72}$$

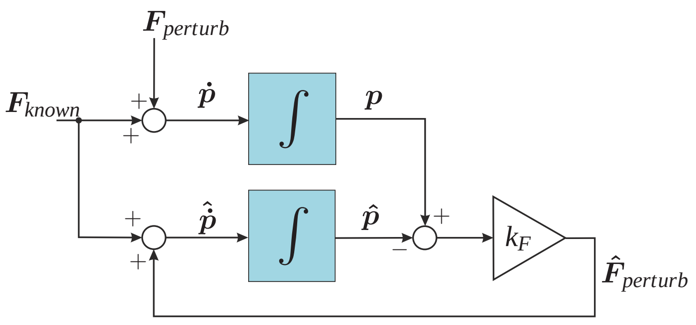

Figure 5.8: Overview of the momentum-based disturbance observer.

and

$$\hat{\boldsymbol{p}}(t) = \boldsymbol{p}_0 + \int_0^t (\boldsymbol{F}_{known}(t) + \hat{\boldsymbol{F}}_{perturb}(t))\, dt \quad . \tag{5.73}$$

Here, $\boldsymbol{p}_0 = m\,\dot{\boldsymbol{x}}(0)$ denotes the initial linear momentum. Combining (5.72) and (5.73) yields $\boldsymbol{p}(t) - \hat{\boldsymbol{p}}(t) = \int_0^t \boldsymbol{F}_{perturb}(t)\, dt - \int_0^t \hat{\boldsymbol{F}}_{perturb}(t)\, dt$, such that (5.71) can be written as

$$\hat{\boldsymbol{F}}_{perturb} = k_F\ (\boldsymbol{p} - \hat{\boldsymbol{p}}) = k_F\ (m\,\dot{\boldsymbol{x}} - \hat{\boldsymbol{p}}) \quad . \tag{5.74}$$

Here, the time dependancy "(t)" was omitted for brevity and $\boldsymbol{p} = m\,\dot{\boldsymbol{x}}$ was substituted. Equations (5.73) and (5.74) are the actual *observer equations*. The overall observer design is outlined in Fig. 5.8. In a discrete system, the observer equations have to be discretized, which yields

$$\hat{\boldsymbol{F}}_{perturb,k} = k_F\ (m\,\dot{\boldsymbol{x}} - \hat{\boldsymbol{p}}_k) \tag{5.75}$$

and

$$\hat{\boldsymbol{p}}_{k+1} = \hat{\boldsymbol{p}}_k + (\boldsymbol{F}_{known} + \hat{\boldsymbol{F}}_{perturb,k})\,\Delta t \quad . \tag{5.76}$$

Here, Δt denotes the sampling time. For $k = 0$, $\hat{\boldsymbol{p}}_k$ is initialized as $\hat{\boldsymbol{p}}_0 = m\,\dot{\boldsymbol{x}}(0)$.

In this section, a momentum-based disturbance observer was derived that allows to estimate external perturbations and thus to increase the robustness of the overall control framework. The two finally required observer equations (for discrete sampling times) are (5.75) and (5.76). Note that for practical purposes, it makes sense to limit the estimated force to a maximum to avoid over-compensation, e.g. in the case when the robot is perturbed by a rigid obstacle.

Leaning against perturbation forces

If the perturbing forces $\boldsymbol{F}_{perturb}$ are known (or estimated, $\hat{\boldsymbol{F}}_{perturb}$) they can be explicitly included in the definition of the Virtual Repellent Point (VRP). Equation (5.28) can be adjusted accordingly:

$$\boldsymbol{r}_{vrp} = \boldsymbol{r}_{ecmp} + [0\ 0\ \Delta z_{vrp}]^T - \frac{b^2}{m}\,\boldsymbol{F}_{perturb} \tag{5.77}$$

The force scaling term can also be written as $\frac{b^2}{m} = \frac{\Delta z_{vrp}}{m\,g} \geq 0$. The negative sign in front of this term implies that the VRP is shifted towards the perturbing force. Note: the nominal offset $[0\ 0\ \Delta z_{vrp}]^T$ accounting for gravity would also be achieved if the gravitational force was inserted as "perturbing force", i.e., $\boldsymbol{F}_{perturb} = \boldsymbol{F}_g = [0\ 0\ \ -m\,g]^T$. Likewise, if an estimate of the perturbing force is available, the original VRP reference trajectories $\boldsymbol{r}_{vrp,ref,i}$ from (5.34) are shifted by $-\frac{b^2}{m}\,\hat{\boldsymbol{F}}_{perturb}$

$$\boldsymbol{r}_{vrp,ref,i,shifted} = \boldsymbol{r}_{vrp,ref.i} - \frac{b^2}{m}\,\hat{\boldsymbol{F}}_{perturb} \quad , \tag{5.78}$$

i.e., they are shifted towards the perturbing force and the robot “leans into” the perturbation. Here, $\hat{\boldsymbol{F}}_{perturb}$ is used instead of $\boldsymbol{F}_{perturb}$ to indicate that in this section the estimated force is used instead of the actual unknown perturbing force. For brevity time dependancies are omitted in this section. While the original DCM dynamics (5.30) remains unaffected, due to the shift of the VRP references in (5.78), also the original DCM references $\boldsymbol{\xi}_{ref,i}$ from (5.40) are shifted by $-\frac{b^2}{m}\,\hat{\boldsymbol{F}}_{perturb}$:

$$\boldsymbol{\xi}_{ref,i,shifted} = \boldsymbol{\xi}_{ref,i} - \frac{b^2}{m}\,\hat{\boldsymbol{F}}_{perturb} \ . \tag{5.79}$$

For a given perturbing force estimate $\dot{\hat{\boldsymbol{F}}}_{perturb}$, (5.79) could be differentiated to yield

$$\dot{\boldsymbol{\xi}}_{ref,i,shifted} = \dot{\boldsymbol{\xi}}_{ref,i} - \frac{b^2}{m}\,\dot{\hat{\boldsymbol{F}}}_{perturb} \ . \tag{5.80}$$

Then, the following DCM tracking dynamics (similar to (5.60)) could be asked for:

$$\dot{\boldsymbol{\xi}} - \dot{\boldsymbol{\xi}}_{ref,1,shifted} = -k_\xi\,\left(\boldsymbol{\xi} - \boldsymbol{\xi}_{ref,1,shifted}\right) \ , \tag{5.81}$$

which would track the shifted DCM trajectories asymptotically for $k_\xi > 0$. The following ideal DCM velocity would correspond to the ideal DCM tracking behavior (5.81):

$$\dot{\boldsymbol{\xi}}_{ideal} = \dot{\boldsymbol{\xi}}_{ref,1} - \frac{b^2}{m}\,\dot{\hat{\boldsymbol{F}}}_{perturb} - k_\xi\,\left(\boldsymbol{\xi} - \boldsymbol{\xi}_{ref,1} + \frac{b^2}{m}\,\hat{\boldsymbol{F}}_{perturb}\right) \ . \tag{5.82}$$

Since no estimate of the perturbation force derivative $\dot{\hat{\boldsymbol{F}}}_{perturb}$ is available in the presented setup, it is assumed to be zero, which yields the new desired DCM velocity

$$\dot{\boldsymbol{\xi}}_{des} = \dot{\boldsymbol{\xi}}_{ref,1} \;-\; k_\xi\,\left(\boldsymbol{\xi} \;\underbrace{-\,\boldsymbol{\xi}_{ref,1} + \frac{b^2}{m}\,\hat{\boldsymbol{F}}_{perturb}}_{-\boldsymbol{\xi}_{ref,1,shifted}}\right) \ . \tag{5.83}$$

Using this desired DCM rate and with (5.70), the following closed-loop dynamics is found:

$$\begin{bmatrix} \dot{\hat{\boldsymbol{F}}}_{perturb} \\ \Delta\dot{\boldsymbol{\xi}} \end{bmatrix} = \begin{bmatrix} -k_F & 0 \\ -\frac{k_F\,b^2}{m} & -k_\xi \end{bmatrix} \begin{bmatrix} \hat{\boldsymbol{F}}_{perturb} \\ \Delta\boldsymbol{\xi} \end{bmatrix} + \begin{bmatrix} k_F \\ \frac{k_F\,b^2}{m} \end{bmatrix} \boldsymbol{F}_{perturb} \ . \tag{5.84}$$

The DCM error $\Delta\boldsymbol{\xi} = \boldsymbol{\xi} - \boldsymbol{\xi}_{ref,1,shifted}$ is here defined as the difference between the actual DCM and the shifted DCM reference from (5.79) and $\Delta\dot{\boldsymbol{\xi}}$ denotes its derivative. The eigenvalues of the system matrix are stable for $k_F > 0$ and $k_\xi > 0$ such that for a bounded $\boldsymbol{F}_{perturb}$ the preconditions for bounded input bounded output (BIBO) stability are fulfilled. With $\boldsymbol{r}_{vrp,des} = \boldsymbol{\xi} - b\,\dot{\boldsymbol{\xi}}_{des}$ from (5.30), the desired VRP, that corresponds to the desired DCM rate from (5.83), is found as

$$\boldsymbol{r}_{vrp,des} = (1 + k_\xi\,b)\,\boldsymbol{\xi} \;-\; k_\xi\,b\,\boldsymbol{\xi}_{ref,1} \;-\; b\,\dot{\boldsymbol{\xi}}_{ref,1} \;+\; \frac{k_\xi\,b^3}{m}\,\hat{\boldsymbol{F}}_{perturb} \ . \tag{5.85}$$

Substituting (5.85) in (5.33) and with (5.45), the desired external force, that includes the compensation of the estimated external force, can be written in the following condensed form:

$$\boldsymbol{F}_{ext,des} = \frac{m}{b^2}\left(\boldsymbol{x} \;-\; \boldsymbol{r}_{vrp,ref,1} \;-\; (1 + b\,k_\xi)(\boldsymbol{\xi} - \boldsymbol{\xi}_{ref,1})\right) \;-\; \boldsymbol{F}_g \;-\; (1 + b\,k_\xi)\,\hat{\boldsymbol{F}}_{perturb} \ . \tag{5.86}$$

Note that the original non-shifted DCM reference $\boldsymbol{\xi}_{ref,1}$ from (5.40) is used here. When comparing (5.86) to (5.63), the effect of the perturbation force estimate on the force control law is apparent.

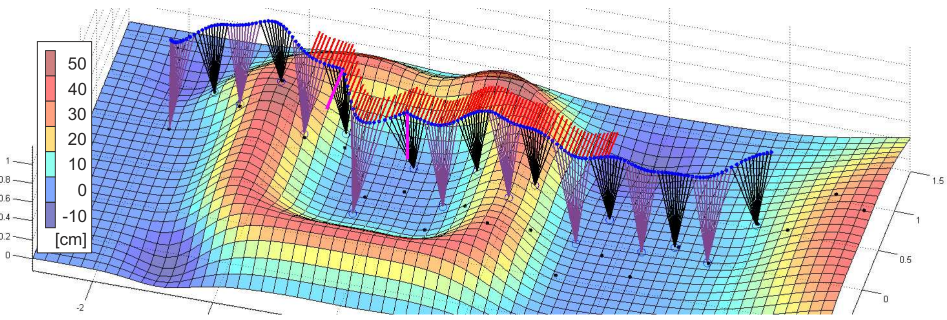

Figure 5.9: Simulation of a Prismatic Inverted Pendulum [29]. Walking direction: left to right. The biped is modeled as a point mass with two point feet. The red lines denote a lateral disturbance corresponding to 10 % of the robot's weight. The pink lines denote a horizontal and a vertical push, each corresponding to a sudden velocity change of 1 m/s. The black points on the ground denote the preplanned footstep locations.

Comparison to integral controllers

Standard integral controllers (combined with PD components in a PID controller) typically increase the robustness against external perturbations. For constant or slowly changing perturbations, they typically lead to good tracking of the desired reference trajectories with no (or low) steady state deviations. In case of a constant external force for example, the integral part of a corresponding force PID controller gradually builds up a counterforce which compensates for the perturbation force. Yet, such an integral compensation comes with a drawback when applied to walking robots with limited support. The compensation of strong perturbations may correspond to eCMPs far outside the base of support, which would require high and continuous rates of change of angular momentum or otherwise be unfeasible, i.e., the desired force could not be produced.

In contrast, the method presented in this chapter shifts the nominal VRP and DCM trajectories such that the robot "leans against" the external force. That way gravity helps compensating for the external perturbation, while the commanded eCMP converges back to the nominal one, which increases the system's robustness w.r.t. new perturbations.

5.4 Evaluation of DCM-based walking

5.4.1 Prismatic Inverted Pendulum simulations (point mass)

To evaluate the performance of the DCM controller (5.63), which the walking in this work is based on, simulations based on the Prismatic Inverted Pendulum model [29] (the robot is modeled as *point mass* with two massless *point feet*) were carried out. For DCM planning and feedback control, the methods from Sec. 5.3 are used. Yet, for simplicity, instantaneous transitions of the eCMPs from one foot to the next are simulated (no double support). Also, these transitions are assumed to be impact-free. Figure 5.9 shows a screenshot of a simulation, in which the robot traverses a crater. The stepping time is set to 0.5 seconds. The modeled mass is 60kg. The approximate "average height" of the CoM over the ground is set to $\Delta z_{vrp} = 0.8m$. The surface height varies between plus 50cm and minus 10cm. An unknown lateral disturbance of 58.86 N (10 % of robot's weight) - active from $t = 2.25s$ till $t = 6.75s$ - was perturbing the robot. Additionally, at $t = 2.75s$ and $t = 3.75s$ the robot suffered lateral and vertical pushes, each resulting in a sudden velocity change of 1 m/s.

The desired footprints are preplanned on the known three-dimensional ground surface (black

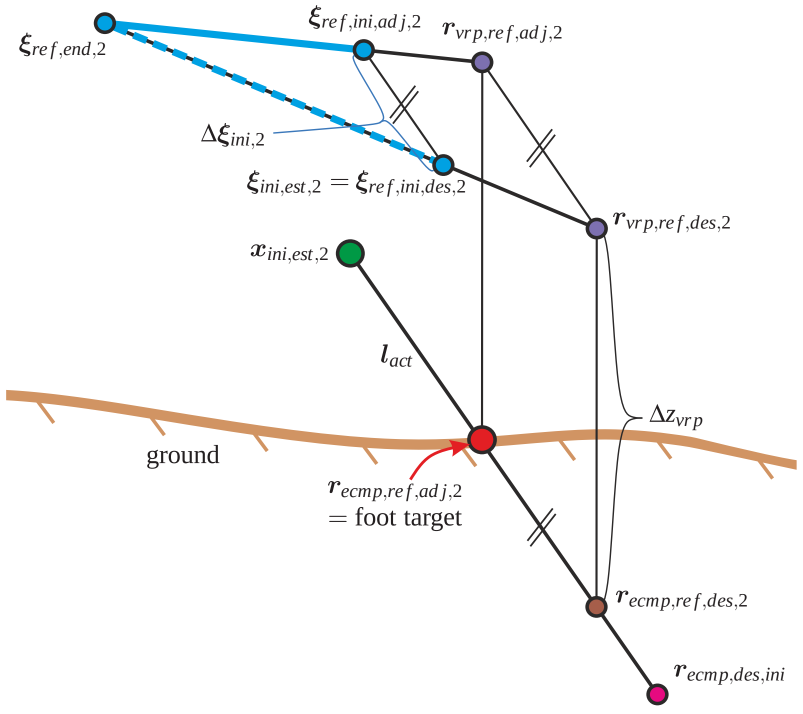
$\boldsymbol{\xi}_{ref,end,2}$
$\boldsymbol{\xi}_{ref,ini,adj,2}$
$\boldsymbol{r}_{vrp,ref,adj,2}$
$\Delta\boldsymbol{\xi}_{ini,2}$
$\boldsymbol{\xi}_{ini,est,2} = \boldsymbol{\xi}_{ref,ini,des,2}$
$\boldsymbol{r}_{vrp,ref,des,2}$
$\boldsymbol{x}_{ini,est,2}$
l_{act}
Δz_{vrp}
ground
$\boldsymbol{r}_{ecmp,ref,adj,2}$
= foot target
$\boldsymbol{r}_{ecmp,ref,des,2}$
$\boldsymbol{r}_{ecmp,des,ini}$

Figure 5.10: Appropriate step adjustment for point-foot robots on 3D ground surface.

points in the figure)[5]. Due to the model assumptions (point mass and point foot), only the direction CoM-to-foot is feasible for the external forces, while the other two spatial directions are unactuated (robot follows its unstable natural dynamics). Therefore, the desired eCMP $\boldsymbol{r}_{ecmp,des}$ has to be projected via

$$\boldsymbol{r}_{ecmp,feas} = \boldsymbol{x} + \boldsymbol{u}_{x,f}\boldsymbol{u}_{x,f}^T(\boldsymbol{r}_{ecmp,des} - \boldsymbol{x}) \tag{5.87}$$

where $\boldsymbol{r}_{ecmp,feas}$ is the best feasible eCMP and $\boldsymbol{u}_{x,f}$ is the unit vector pointing from the CoM to the point foot. To comply with the constraint that the feet can only push the CoM off, desired eCMPs “above” the CoM are projected onto the CoM itself, resulting in zero external force (i.e. free-falling robot). To compensate for the lack of controllability, the robot adjusts its target location for the subsequent step (see Fig. 5.10) using the method presented in Sec. 5.3.3[6]. Throughout each stance phase, the current state is numerically integrated to provide estimates for the initial DCM $\boldsymbol{\xi}_{ini,est,2}$ and CoM $\boldsymbol{x}_{ini,est,2}$ at the step transition. Reordering (5.42), the ideal VRP $\boldsymbol{r}_{vrp,ref,des,2}$ (and the corresponding eCMP $\boldsymbol{r}_{ecmp,ref,des,2}$) is computed, which would shift the DCM from the estimated initial DCM $\boldsymbol{\xi}_{ini,est,2}$ to the final desired DCM $\boldsymbol{\xi}_{ref,end,2}$ of that step within the next stepping time (dashed blue line in Fig. 5.10). The intersection $\boldsymbol{r}_{ecmp,ref,adj,2}$ of the line $\boldsymbol{x}_{ini,est,2}$-to-$\boldsymbol{r}_{ecmp,ref,des,2}$ with the ground is chosen as *target location* for the adjusted step. The adjusted nominal VRP $\boldsymbol{r}_{vrp,ref,adj,2}$ is found Δz_{vrp} further above. With $\boldsymbol{r}_{vrp,ref,adj,2}$ and $\boldsymbol{\xi}_{ref,end,2}$, the desired initial DCM $\boldsymbol{\xi}_{ref,ini,adj,2}$ is computed via (5.42). Now, the adjusted DCM reference (solid blue line in Fig. 5.10) is computed via (5.40) and (5.45). This step adjustment has two beneficial properties: i) Since the initial commanded eCMP $\boldsymbol{r}_{ecmp,des,ini}$, computed from the initial DCM error $\Delta\boldsymbol{\xi}_{ini,2}$ via (5.61) and (5.28), is perfectly in the actuated direction (line CoM-to-foot, note parallelisms), good convergence in the beginning of the subsequent step

[5]Note that online planned footsteps could be handled by the control framework as well.

[6]As mentioned above, constant eCMP reference positions and instantaneous transitions between them were chosen for the point-mass simulations.

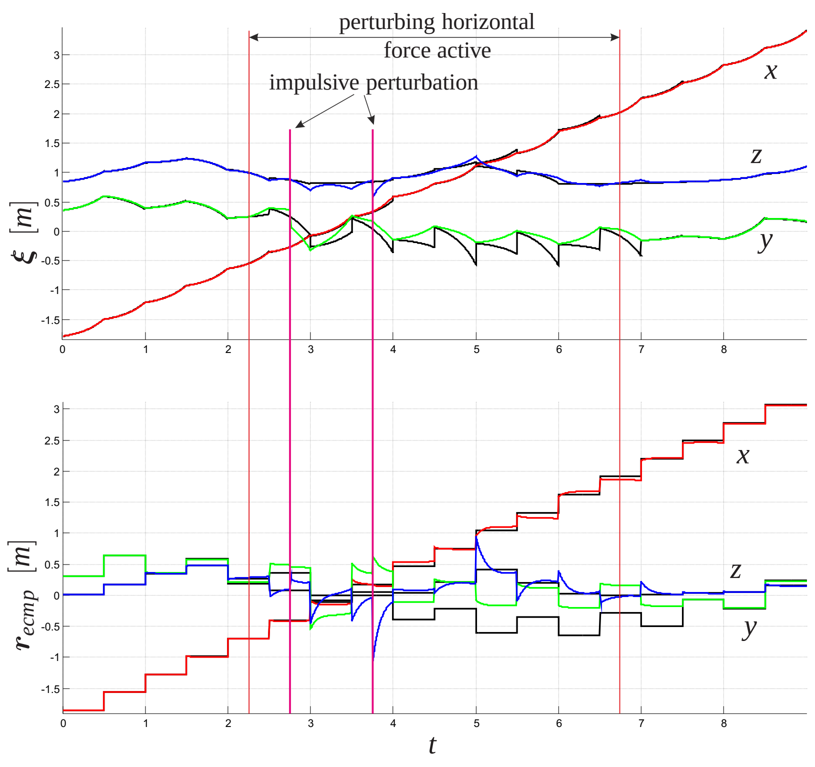

Figure 5.11: Prismatic Inverted Pendulum simulation. Top: desired (black) and real DCMs, Bottom: desired (point foot position, black) and real eCMPs

is assured. ii) If the DCM error converges to zero, the commanded eCMP becomes perfectly consistent with the point-foot constraint.

Figure 5.11 presents the achieved DCM and eCMP tracking. The robot was able to compensate for the perturbations. After the perturbations, the desired foot locations were tracked very well after a couple of steps. Note that in this simulation neither a finite-sized foot nor torques around the CoM were available, showing the robustness of the proposed underlying control laws and the proposed step adjustment method.

Other simulation setups including (unknown) constant and impulsive perturbations in different directions as well as mass estimation errors also showed a very robust performance of the simulated robot. The robustness increases for shorter times per step t_{step} and bigger DCM gains k_ξ.

5.4.2 Whole-body simulations and experiments

The proposed DCM trajectory generator and tracking controller from sections 5.3.1 and 5.3.2 were thoroughly tested in numerous simulations and experiments with DLR's humanoid robot TORO ([1,95], see Fig. 5.14, left) and IHMC's Atlas robot [96]. The following sections describe several different whole-body simulations and experiments.

Whole-body simulations

The walking algorithms, described in this thesis, were initially based on Capture Point (i.e. the two-dimensional equivalent of the DCM, see Sec. 5.1) control that was embedded into an admittance-based whole-body controller. Toro was operated using position control, while an admittance controller (see [25] for details) was used to achieve the required ZMP control. After

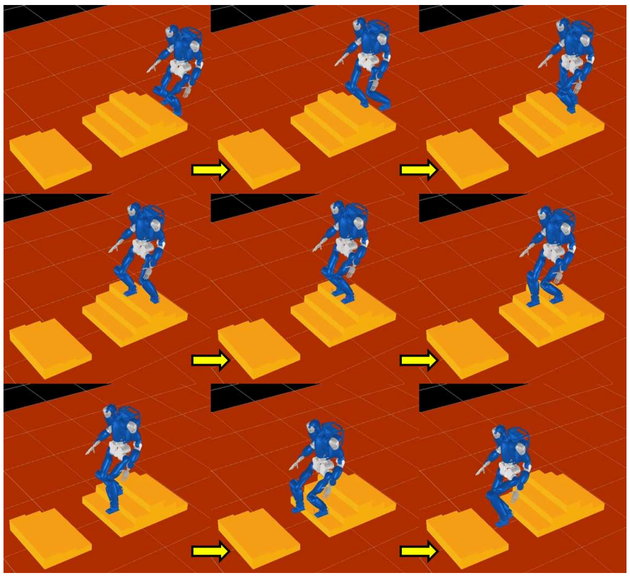
Figure 5.12: OpenHRP3 simulation [2] of DLR's humanoid robot TORO. Walking over a set of stairs of variable height. (step height differences: $[+12,+12,+12,-12,-12,-12,+10,+5,+3,-18]$ cm)

the extension of the concept of Capture Point to the three-dimensional DCM, the corresponding controllers were adjusted. Figure 5.12 shows Toro walking over a set of stairs in OpenHRP3 [2] while using 3D DCM and admittance control. Figure 5.13 presents the corresponding DCM and eCMP trajectories, which are tracked precisely. Note the vertical components (blue) that smoothly follow the stair surface.

During the preparations for DARPA's Virtual Robotics Challenge (VRC) [30, 81], two DCM-based trajectory generators were developed [30, 81] (the DCM reference trajectory generator presented in Sec. 5.3.1 is an advancement of these original generators). Based on these walking pattern generators, the simulated Atlas robot achieved a maximum step length of up to 70 cm (see Fig. 5.15 (left)) in both IHMC's simulation environment [97] and the official VRC simulator Gazebo [98].

Figure 5.14, right, show a simulation, in which Toro is subject to an unknown sidewards pointing perturbation force. As shown in the lower plot in Fig. 5.16, this force continuously grows from 0 to 150 N within ten seconds. The disturbance observer presented in section 5.3.4 is used to estimate the perturbation force. Toro is well able to counteract this perturbation by leaning towards it. The upper plot in Fig. 5.16 shows the three-dimensional DCM tracking and the corresponding CoM motion. In the y-direction, the leaning of the DCM towards the perturbation becomes apparent.

Walking experiments

The Capture Point (CP) based walking controller, as presented in the author's early publications [24, 25] and recapitulated in Sec. 5.1, was extensively tested in hardware experiments, including online obstacle avoidance [99] and foot trajectory optimizations [100], and applied in different demonstration scenarios. Figure 5.17 shows a plot of corresponding ZMP, CP and CoM trajectories from experiments in which Toro was controlled via a joystick input. The CP references are tracked well, while the CoM smoothly follows the actual DCM. The desired ZMPs are tracked well by the underlying ZMP controller.

The same algorithms as for the Virtual Robotics Challenge (VRC) were also used by team

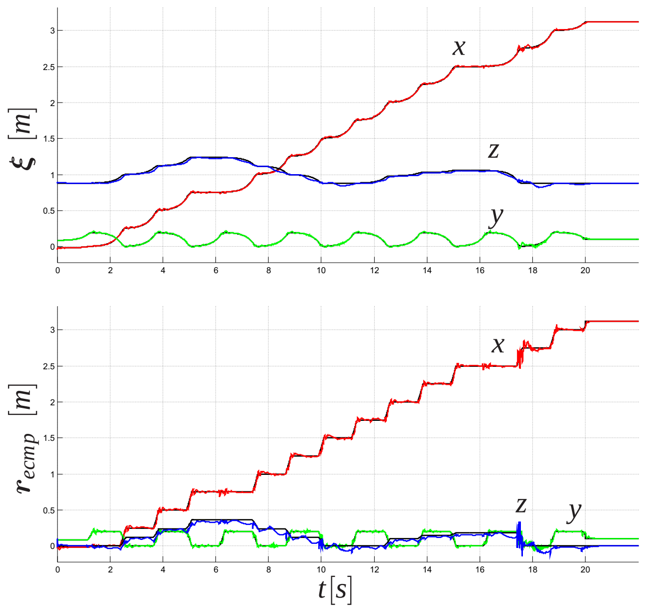

Figure 5.13: OpenHRP3 simulation of humanoid robot TORO. Top: desired (black) and real DCMs, Bottom: desired (black) and real eCMPs

IHMC for the DARPA Robotics Challenge Trials [3] and Finals [4]. Figure 5.18 shows an experimental result of Atlas walking on flat ground while using the Continuous Double Support (CDS) trajectory generator presented in [30]. The plot shows a series of seven steps. It displays estimated and desired eCMP and DCM trajectories in x (forward) and y (sideward) direction. The estimated DCM and eCMP are attained via a Kalman filter [81] and by mapping joint torques to ground reaction forces. Apart from deviations due to impacts at the foot touch-downs, both eCMP and DCM trajectories are tracked well. The eCMP transition during double support is continuous, resulting in continuous leg forces and joint torques. The whole-body controller used by IHMC for these experiments is similar to the one presented in chapter 4, the most prominent difference being that IHMC's problem formulation was fully determined, in contrast to the over-determined formulation in this work.

Experiments that evaluate the DCM controller, embedded into the WBC control framework presented in chapter 4, are subject to the author's current research. Figure 5.19 shows a first result of Toro taking a single step using WBC-embedded DCM control. Due to problems with the state estimation, the actual DCM trajectories are shaky. Apart from this effect, the desired DCM trajectories (black) are tracked very well, which is a promising intermediary result. Stable torque-based walking is expected to be achieved in the near future. While the Capture Point and admittance-based walking algorithm is executed on the real robot with a 1ms rate, the torque-based WBC framework is running at three different rates: the dynamics computations (based on [70]) run in a 3ms rate, the QP-based WBC runs at a 2ms rate and the high level torque controller uses a 1ms rate.

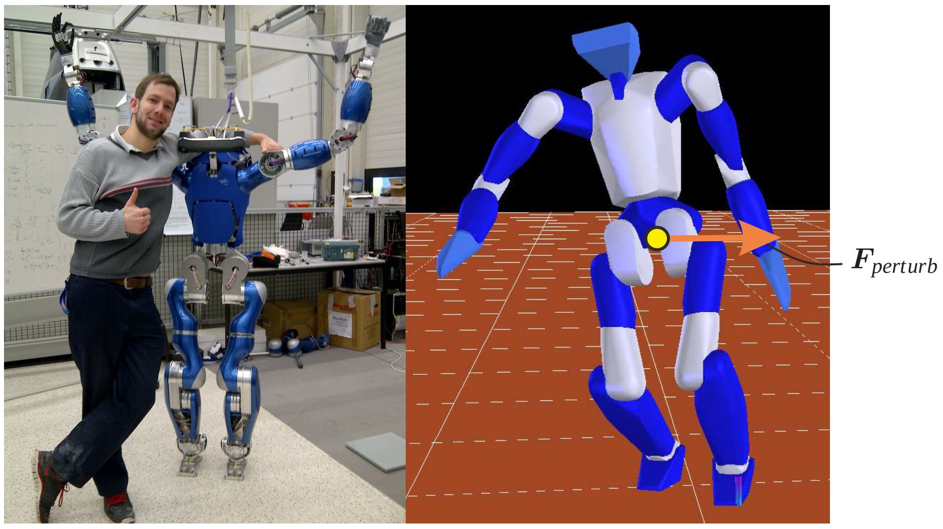

Figure 5.14: left: humanoid robot Toro and author; right: Toro fighting against perturbation force in OpenHRP simulation.

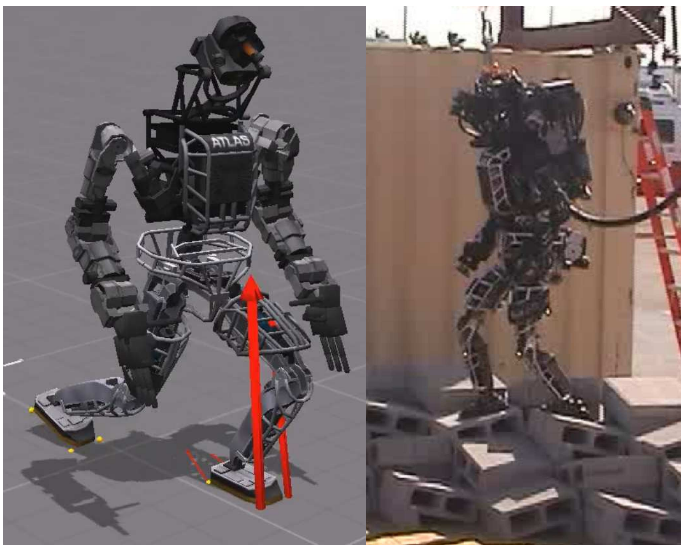

Figure 5.15: left: Atlas robot [96] doing 70 cm steps in IHMC's simulation environment (SimulationConstructionSet [97]) using toe-off motion.
right: Physical Atlas robot walking over cinder blocks during DRC Trials.

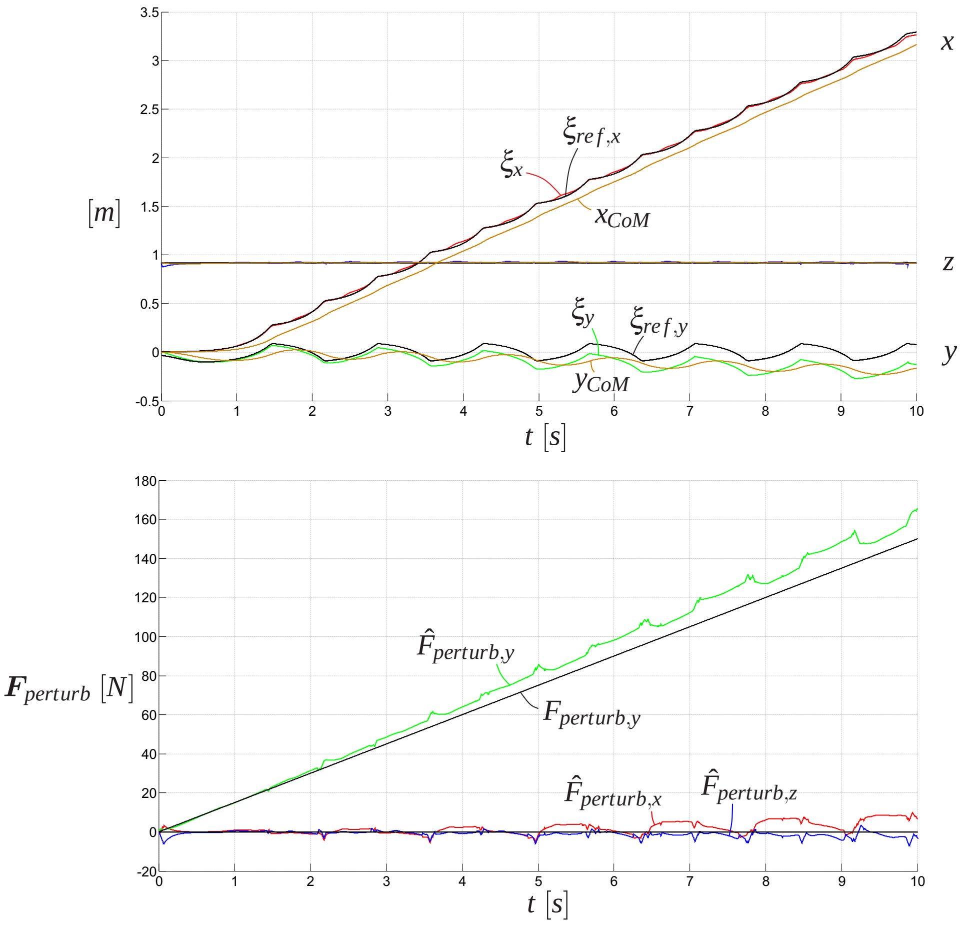

Figure 5.16: OpenHRP simulation of Toro leaning against unknown external perturbation force. Top: desired and achieved DCM trajectories and corresponding CoM motion. Bottom: actual and estimated perturbation force.

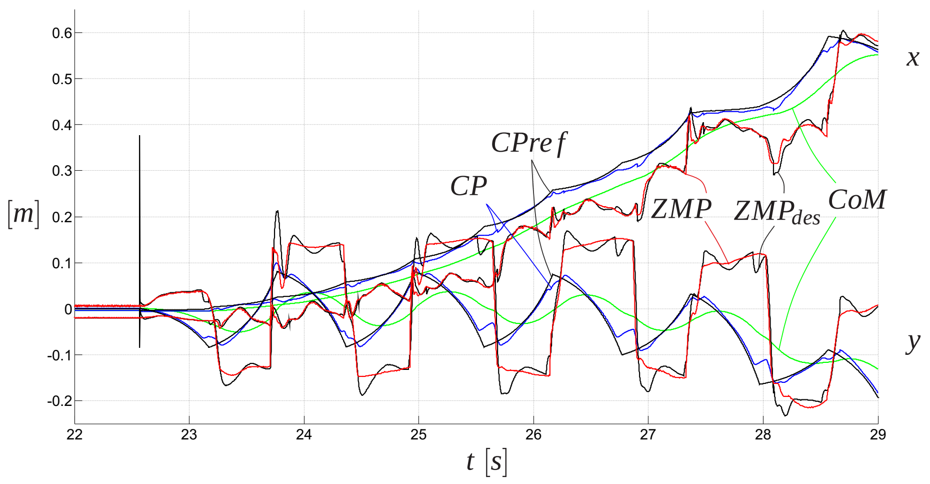

Figure 5.17: Real experiment: Toro walking based on CP control (position controlled).

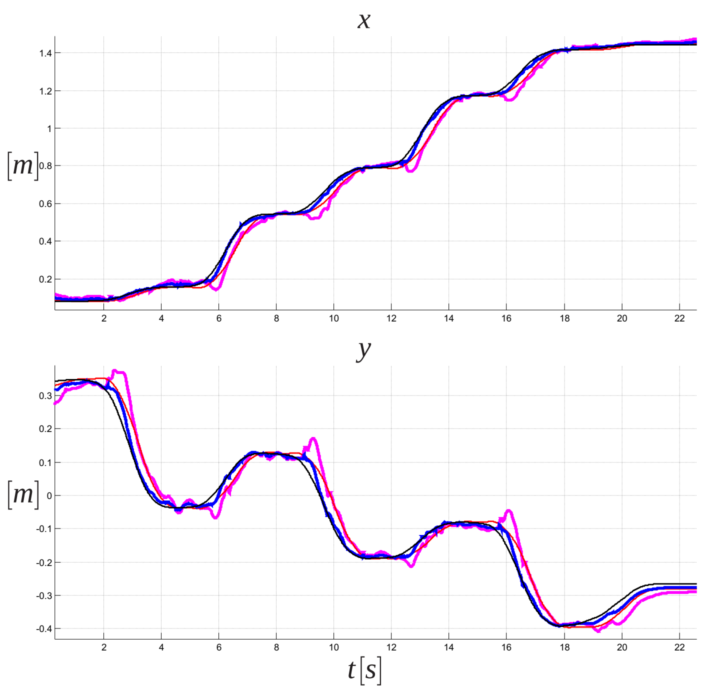

Figure 5.18: Real experiment: IHMC's Atlas robot walking during DRC Trials. Pink/red: estimated/desired eCMP, blue/black: estimated/desired DCM

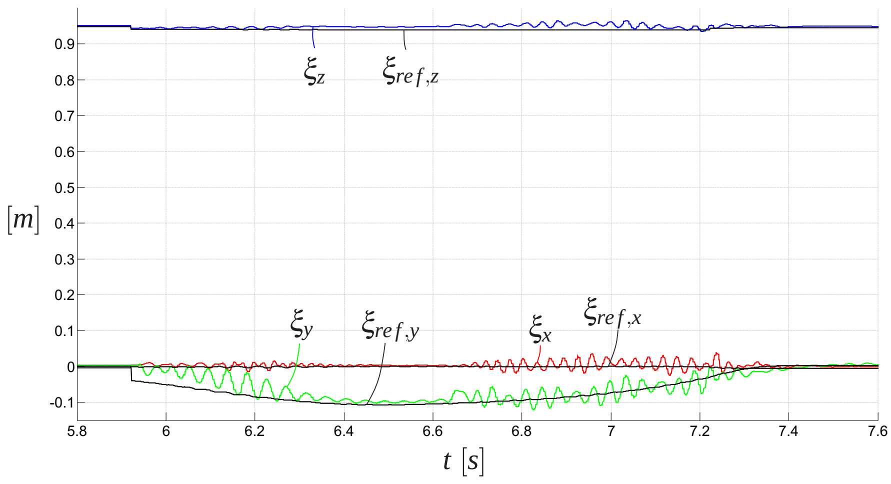

Figure 5.19: Real experiment: Toro takes single step using WBC-embedded DCM-control.

CHAPTER 6

Bipedal running based on Biologically Inspired Dead-beat control

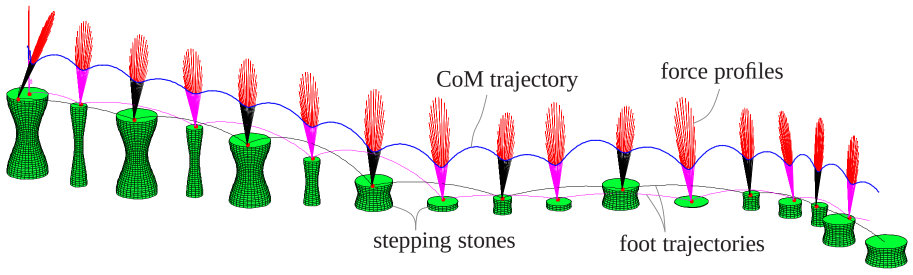

Figure 6.1: Bipedal point-mass model running on 3D stepping stones based on Biologically Inspired Deadbeat (BID) control.

In terms of locomotion speed, bipedal running typically exceeds the locomotion form of walking, as presented in the previous section, considerably, i.e. for certain applications bipedal humanoid running may be of interest. Humans and certain running birds have developed astonishing mechanisms and control approaches that allow for fast and agile running gaits. In the research presented in this thesis, inspired by human running experiments in a gait laboratory, a new concept for bipedal running, the so called Biologically Inspired Deadbeat (BID) control, was developed, which is a major contribution with regard to the state of the art. BID control encodes the CoM motion of a robot and the corresponding leg forces during stance via polynomials and uses intuitive boundary conditions as design parameters. All required equations can be solved analytically. This results in significant advantages over the use of the spring-loaded inverted pendulum (SLIP, [6]) for running, due to the non-linearity of the latter. This chapter will present the BID method in detail.

6.1 Human running experiments as motivation

The main idea used in this chapter is to *design desired CoM trajectories* that produce *approximately natural GRF profiles* while fulfilling several *boundary conditions*. It is well known that some physical template models, such as the SLIP, generate ground reaction forces (GRF) similar to the ones observed in human running. Back in 1985, Raibert stated in his book “Closed form expressions relating forward foot placement to net forward acceleration for the one-legged machine are not known” [40]. The lack of closed form solutions e.g. for the SLIP motivates the search for

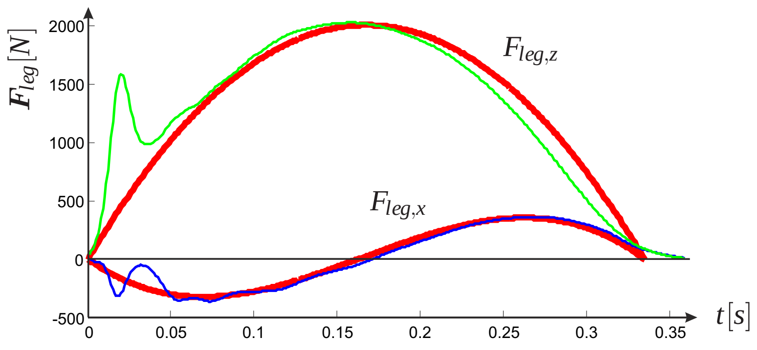

Figure 6.2: Comparison of experimentally measured human leg forces (blue/green) and polynomial approximations (red).

alternative ways of encoding the leg force ($\boldsymbol{F}_{leg}$, equivalent to GRF). Figure 6.2 shows a typical GRF profile recorded during a human running experiment via a force plate. Except for the impact phenomenon at the beginning and the lower slope in the end of stance, the human GRF profiles can be approximated quite well by polynomials of order 2 in the vertical direction and of order 3 in the $x-$direction. Therefore, the original idea in [67] was to approximate the leg force profile during stance via polynomials. The total force $\boldsymbol{F}_{CoM}$ acting on the CoM can be computed from the leg force $\boldsymbol{F}_{leg}$ and the gravitational force $\boldsymbol{F}_g$ via

$$\boldsymbol{F}_{CoM} = \boldsymbol{F}_{leg} + \boldsymbol{F}_g = \boldsymbol{F}_{leg} + m\,\boldsymbol{g} \quad . \tag{6.1}$$

Here, m is the robot's total mass and $\boldsymbol{g} = [0\;0\;-g]^T$ denotes the gravitational acceleration vector. The constant offset between $\boldsymbol{F}_{CoM}$ and $\boldsymbol{F}_{leg}$ in (6.1) and Newton's 2nd law (CoM acceleration $\ddot{\boldsymbol{x}} = \frac{\boldsymbol{F}_{CoM}}{m}$) motivates - during stance - the use of a 4th order polynomial to encode the vertical CoM position z and 5th order polynomials to encode the horizontal CoM positions x and y, as this correlates to 2nd and 3rd order polynomials for the CoM accelerations $\ddot{x}$, $\ddot{y}$, $\ddot{z}$ and thus leg forces. This polynomial encoding can be written as:

$$\begin{bmatrix}\sigma(t)\\ \dot{\sigma}(t)\\ \ddot{\sigma}(t)\end{bmatrix} = \underbrace{\begin{bmatrix}1 & t & t^2 & t^3 & t^4 & t^5\\ 0 & 1 & 2t & 3t^2 & 4t^3 & 5t^4\\ 0 & 0 & 2 & 6t & 12t^2 & 20t^3\end{bmatrix}}_{\begin{bmatrix}\boldsymbol{t}_{\sigma}^T(t)\\ \boldsymbol{t}_{\dot{\sigma}}^T(t)\\ \boldsymbol{t}_{\ddot{\sigma}}^T(t)\end{bmatrix}} \boldsymbol{p}_{\sigma}, \quad \sigma \in \{x,y,z\} \tag{6.2}$$

Here, $\boldsymbol{t}_{\sigma}^T(t)$, $\boldsymbol{t}_{\dot{\sigma}}^T(t)$ and $\boldsymbol{t}_{\ddot{\sigma}}^T(t)$ denote the time-mapping row vectors that - for a given time t - map the polynomial parameter vectors $\boldsymbol{p}_{\sigma}$ to CoM positions $\sigma(t)$, velocities $\dot{\sigma}(t)$ and accelerations $\ddot{\sigma}(t)$. The last elements of the vectors are greyed out to indicate that they are only used for the horizontal directions, but not for the vertical one.

6.2 Outline of BID control method

This work uses a preview of at least two upcoming stance and flight phases, as shown in Fig. 6.3. The desired relative apex and touch-down heights $\Delta z_{apex,des}$ and $\Delta z_{TD,des}$ are used as design parameters. They indicate how high over the floor the apex of each flight curve (i.e. $\dot{z} = 0$) should be and at what CoM height the touch-down (TD) is supposed to happen. $z_{floor,i}$ denotes the height level of the i-th step. Another design parameter, used in this work, is the total stance time T_s (it can vary from step to step), whereas the total flight time T_f results from the boundary conditions chosen in section 6.4.1. To keep track of the current running state, a state machine is used. It switches from flight to stance, if the CoM is below $z_{TD} = z_{floor,i} + \Delta z_{TD}$ and the vertical velocity is negative, and from stance to flight when the total stance time is over. A timer provides the time in stance $t_s \in [0, T_s]$ and the time in flight $t_f \in [0, T_f]$. They are reset at state transitions.

6.3 CoM dynamics during flight

Running is a locomotion pattern, which employs alternate flight and (single leg supporting) stance phases. During flight, the CoM cannot be controlled, i.e. it follows its natural dynamics (parabolic path through space). For a given time t, the CoM position $\boldsymbol{x}(t) = [x(t), y(t), z(t)]^T$ and velocity $\dot{\boldsymbol{x}}(t) = [\dot{x}(t), \dot{y}(t), \dot{z}(t)]^T$ can be computed as

$$\boldsymbol{x}(t) = \boldsymbol{x}_0 + \dot{\boldsymbol{x}}_0\, t + \boldsymbol{g}\, \frac{t^2}{2} \quad , \tag{6.3}$$

$$\dot{\boldsymbol{x}}(t) = \dot{\boldsymbol{x}}_0 + \boldsymbol{g}\, t \, , \tag{6.4}$$

where $\boldsymbol{x}_0$ and $\dot{\boldsymbol{x}}_0$ are the initial CoM position and velocity. One typical task in running control is to achieve a certain apex height. The apex is the highest point in the ballistic flight curve, i.e. vertical CoM velocity $\dot{z}_{apex} = 0$. Using this condition and the current vertical CoM velocity $\dot{z}$ instead $\dot{z}_0$ in the third row of (6.4), the current time to apex Δt_{apex} is found as

$$\Delta t_{apex} = \frac{\dot{z}}{g} \quad . \tag{6.5}$$

The remaining time until touch-down (TD) is computed as

$$\Delta t_{TD} = \Delta t_{apex} + \sqrt{\Delta t_{apex}^2 + \frac{2}{g}\,(z - z_{TD})} \quad . \tag{6.6}$$

Here, $z_{TD} = z_{floor} + \Delta z_{TD}$ is the CoM height at which the touch-down (flight to stance transition) is previewed to happen. With (6.3), (6.4) and (6.6), the previewed touch-down state can be precomputed for any CoM state $[\boldsymbol{x}, \dot{\boldsymbol{x}}]$ as

$$\begin{bmatrix} \boldsymbol{x}_{TD} \\ \dot{\boldsymbol{x}}_{TD} \end{bmatrix} = \begin{bmatrix} \boldsymbol{x} + \Delta t_{TD}\, \dot{\boldsymbol{x}} + \frac{\Delta t_{TD}^2}{2}\, \boldsymbol{g} \\ \dot{\boldsymbol{x}} + \Delta t_{TD}\ \boldsymbol{g} \end{bmatrix} \quad . \tag{6.7}$$

In this work, the relative touch-down height is computed as

$$\Delta z_{TD} = min(\Delta z_{TD,des},\ z - z_{floor} + \frac{\dot{z}^2}{2g} - \Delta_{apex,TD,min}) \quad , \tag{6.8}$$

i.e. nominally the desired relative touch-down height $\Delta z_{TD,des}$ is achieved, while in case of perturbations a minimum height difference between apex and touch-down $\Delta_{apex,TD,min}$ is guaranteed and the solution of (6.6) is assured to be real.

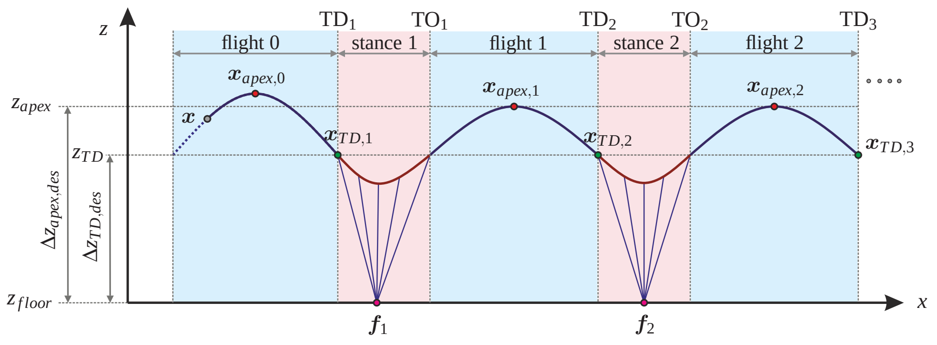

Figure 6.3: Preview of upcoming flight and stance phases (planar sketch) - used for design of boundary conditions. For readability, a constant floor height z_{floor} is shown here.

6.4 Deadbeat control via boundary condition satisfaction

6.4.1 Vertical planning and boundary conditions

As mentioned above, the vertical CoM trajectory during stance is encoded via a 4th order polynomial, i.e. it has 5 polynomial parameters. These can be derived using 5 boundary conditions. Fig. 6.3 graphically displays the used preview of upcoming flight and stance sequences and the corresponding boundary conditions. In this work, - for each previewed contact phase - four linear vertical boundary conditions are used that can be combined as

$$\underbrace{\begin{bmatrix} z_{TD,i} \\ \dot{z}_{TD,i} \\ -g \\ -g \end{bmatrix}}_{\boldsymbol{b}_{z,i}} = \underbrace{\begin{bmatrix} \boldsymbol{t}_z^T(0) \\ \boldsymbol{t}_{\dot{z}}^T(0) \\ \boldsymbol{t}_{\ddot{z}}^T(0) \\ \boldsymbol{t}_{\ddot{z}}^T(T_{s,i}) \end{bmatrix}}_{\boldsymbol{B}_{z,i}} \boldsymbol{p}_{z,i} \ . \tag{6.9}$$

Here, i denotes the index of the considered step and $\boldsymbol{b}_{z,i}$, $\boldsymbol{B}_{z,i}$ and $\boldsymbol{p}_{z,i}$ denote the corresponding boundary condition vector, boundary condition mapping matrix and vertical polynomial parameter vector, respectively. The first two elements in $\boldsymbol{b}_{z,i}$ imply that CoM position and velocity at the beginning of stance equal the CoM touch-down state. The other two elements say that the CoM acceleration at beginning and end of stance equals minus gravity, i.e. the vertical leg force is zero.

The general solution of the linear system $\boldsymbol{B}_{z,i}\,\boldsymbol{p}_{z,i} = \boldsymbol{b}_{z,i}$ is

$$\boldsymbol{p}_{z,i} = \boldsymbol{B}_{z,i}^T(\boldsymbol{B}_{z,i}\,\boldsymbol{B}_{z,i}^T)^{-1}\,\boldsymbol{b}_{z,i} + \boldsymbol{r}_{z,i}\,\tilde{p}_{z,i} \ . \tag{6.10}$$

The (reduced) nullspace base vector[1] $\boldsymbol{r}_{z,i}$ ensures that $\boldsymbol{B}_{z,i}\,\boldsymbol{r}_{z,i} = 0$. The whole (one-dimensional) nullspace of $\boldsymbol{B}_{z,i}$ is represented by the scalar variable $\tilde{p}_{z,i}$. The vector $\boldsymbol{r}_{z,i}$ is computed as

$$\boldsymbol{r}_{z,i} = \begin{bmatrix} -\boldsymbol{B}_{z,i,square}^{-1}\,\boldsymbol{B}_{z,i,final} \\ 1 \end{bmatrix} \ , \tag{6.11}$$

where $\boldsymbol{B}_{z,i,final}$ is the last column in $\boldsymbol{B}_{z,i}$, while $\boldsymbol{B}_{z,i,square}$ consists of all other columns. Equation (6.9) encodes the four *linear* previously described vertical boundary conditions. The fifth boundary condition used in this work is the apex height $z_{apex,i}$ of the CoM during the i-th upcoming flight

[1]see Sec. 3.5.4 on reduced nullspace matrices.

phase (see Fig. 6.3). The vertical CoM state of the i-th take-off (at end of i-th stance time $T_{s,i}$) can be computed via (6.2) as

$$z_{TO,i} = \boldsymbol{t}_z^T(T_{s,i})\ \boldsymbol{p}_{z,i} \tag{6.12}$$

$$\dot{z}_{TO,i} = \boldsymbol{t}_{\dot{z}}^T(T_{s,i})\ \boldsymbol{p}_{z,i} \quad . \tag{6.13}$$

With (6.3) and (6.5), the i-th apex height can be computed as

$$z_{apex,i} = z_{TO,i} + \frac{\dot{z}_{TO,i}^2}{2g} \ . \tag{6.14}$$

The sought-after parameter vector $\boldsymbol{p}_{z,i}$ has to correspond to the desired apex height $z_{apex,i,des}$, which can be computed as

$$z_{apex,i,des} = z_{floor,i+1} + \Delta z_{apex,des} \ . \tag{6.15}$$

Note that here the height $z_{floor,i+1}$ of the upcoming step is used. Inserting (6.12) and (6.13) into (6.14) leads to a quadratic equation in the unknown scalar variable $\tilde{p}_z$

$$\begin{aligned} 0 \quad &= \quad \frac{\boldsymbol{r}_{z,i}^T\, t_{\dot{z}}\, t_{\dot{z}}^T\, \boldsymbol{r}_{z,i}}{2g}\, \tilde{p}_{z,i}^2 + (t_z^T\, \boldsymbol{r}_{z,i} + \frac{t_{\dot{z}}^T\, p_{z,i,0}\ \ t_{\dot{z}}^T\, \boldsymbol{r}_{z,i}}{g})\ \tilde{p}_{z,i} + \\ &+ \quad \frac{(t_{\dot{z}}^T\, p_{z,i,0})^2}{2g} - z_{apex,i,des} \ + t_z^T\ p_{z,i,0} \end{aligned} \tag{6.16}$$

The only valid solution to (6.16) (yielding positive vertical take-off velocities) is

$$\begin{aligned} \tilde{p}_{z,i} \quad &= \quad \frac{2\,\dot{z}_{TD,i} - gT_{s,i} - \Gamma}{4\,T_{s,i}^3} \ , \\ \Gamma \quad &= \quad \sqrt{g(gT_{s,i}^2 - 4\,\dot{z}_{TD,i}\ T_{s,i} + 8\,(z_{apex,i,des} - z_{TD,i}))} \end{aligned} \tag{6.17}$$

Note: finally only (6.11) and (6.17) are necessary as inputs for (6.10) to compute polynomial parameters $\boldsymbol{p}_{z,i}$ for each previewed step that fulfill all desired vertical boundary conditions.

6.4.2 Horizontal planning and boundary conditions

In this work, the derivation for the x- and y-component is equivalent. The letter χ is used to indicate horizontal quantities, i.e. $\chi \in \{x, y\}$. For each previewed contact phase, the following five linear horizontal boundary conditions are used:

$$\underbrace{\begin{bmatrix} \chi_{TD,i} \\ \dot{\chi}_{TD,i} \\ 0 \\ 0 \\ \chi_{TD,i+1,des} \end{bmatrix}}_{\boldsymbol{b}_{\chi,i}} = \underbrace{\begin{bmatrix} \boldsymbol{t}_{\chi}^T(0) \\ \boldsymbol{t}_{\dot{\chi}}^T(0) \\ \boldsymbol{t}_{\ddot{\chi}}^T(0) \\ \boldsymbol{t}_{\ddot{\chi}}^T(T_{s,i}) \\ \boldsymbol{t}_{\chi}^T(T_{s,i}) + T_{f,i}\ \boldsymbol{t}_{\dot{\chi}}^T(T_{s,i}) \end{bmatrix}}_{\boldsymbol{B}_{\chi,i}} \boldsymbol{p}_{\chi,i} \quad . \tag{6.18}$$

Here, $\boldsymbol{b}_{\chi,i}$, $\boldsymbol{B}_{\chi,i}$ and $\boldsymbol{p}_{\chi,i}$ denote the horizontal boundary condition vector, boundary condition mapping matrix and polynomial parameter vector, respectively. As in Sec. 6.4.1, the first two elements of $\boldsymbol{b}_{\chi,i}$ imply that the initial CoM state is equal to the CoM touch-down state. The next two elements assure that initial and final CoM acceleration are zero, i.e. horizontal leg forces are zero. The fifth element - as intermediate control target - specifies the horizontal CoM touch-down position $\chi_{TD,i+1,des}$ of the upcoming step. Since - in case of no perturbations - the horizontal

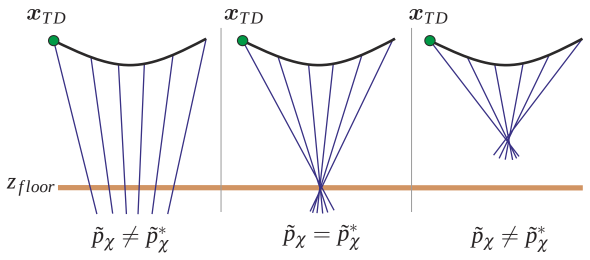
x_{TD}
x_{TD}
x_{TD}
z_{floor}
$\tilde{p}_\chi \neq \tilde{p}^*_\chi$
$\tilde{p}_\chi = \tilde{p}^*_\chi$
$\tilde{p}_\chi \neq \tilde{p}^*_\chi$

Figure 6.4: Effect of $\tilde{p}_\chi$ on force ray focusing (lines of action).

velocity during flight is constant, the take-off state can be propagated to each upcoming touch-down position via

$$\chi_{TD,i+1,des} = \chi_{TO,i} + T_{f,i}\,\dot{\chi}_{TO,i} = (\boldsymbol{t}^T_{\chi,i}(T_{s,i}) + T_{f,i}\,\boldsymbol{t}^T_{\dot{\chi}}(T_{s,i}))\,\boldsymbol{p}_{\chi,i} \quad , \tag{6.19}$$

The i-th time of flight $T_{f,i}$ is computed via (6.5) and (6.6). Note: $z_{TO,i}$ and $\dot{z}_{TO,i}$ (used as z and $\dot{z}$ in (6.5) and (6.6)) depend on the vertical polynomial parameter vector $\boldsymbol{p}_{z,i}$. Thus, the vertical boundary conditions are solved before the horizontal ones. The general solution of (6.18) is

$$\boldsymbol{p}_{\chi,i} = \underbrace{\boldsymbol{B}^T_{\chi,i}(\boldsymbol{B}_{\chi,i}\,\boldsymbol{B}^T_{\chi,i})^{-1}\,\boldsymbol{b}_{\chi,i}}_{\boldsymbol{p}_{\chi,i,0}} + \boldsymbol{r}_{\chi,i}\,\tilde{p}_{\chi,i} \quad . \tag{6.20}$$

The nullspace base vector $\boldsymbol{r}_{\chi,i}$ is computed via the equivalent of (6.11). The horizontal directions have one more polynomial parameter than the vertical one, i.e. one more degree of freedom (DOF). This DOF, represented by the scalar $\tilde{p}_{\chi,i}$ in (6.20), has an effect on the geometry of the leg force rays in space (see Fig. 6.4). Now, the goal is to find the value for $\tilde{p}_{\chi,i}$, which produces the best possible focusing of leg forces, such that these are best feasible for finite-sized (or even point-) feet. To this end, for each previewed step, the time-dependent intersection point $\boldsymbol{x}_{int,i} = [x_{int,i}, y_{int,i}, z_{floor,i}]$ of the leg force with the floor is computed and the integral of the mean square deviation from its mean value $\bar{x}_{int,i}$ is minimized. For a given time in the i-th stance $t_s \in [0, T_{s,i}]$, its horizontal components are

$$\begin{aligned} \chi_{int,i}(t_s) &= \chi(t_s) - \frac{f_{leg,\chi,i}(t_s)}{f_{leg,z,i}(t_s)}\,(z(t_s) - z_{floor,i}) \\ &= \underbrace{(\boldsymbol{t}^T_\chi(t_s) - \frac{(\boldsymbol{t}^T_z(t_s)\boldsymbol{p}_{z,i} - z_{floor,i})\;\boldsymbol{t}^T_{\ddot{\chi}}(t_s)}{\boldsymbol{t}^T_{\ddot{z}}(t_s)\boldsymbol{p}_{z,i} + g}\;)}_{\boldsymbol{d}^T_{\chi,i}(t_s)}\;\boldsymbol{p}_{\chi,i} \quad . \end{aligned} \tag{6.21}$$

Here, $f_{leg,\chi,i}(t_s)$ and $f_{leg,z,i}(t_s)$ are the horizontal and vertical components of the leg force $\boldsymbol{F}_{leg,i}$ and $z(t_s)$ is the height of the CoM. The explicit solution to (6.10) is given by

$$\boldsymbol{p}_{z,i} = [z_{TD,i}\;,\;\dot{z}_{TD,i}\;,\;-\frac{g}{2}\;,\;-2T_{s,i}\,\tilde{p}_{z,i}\;,\;\tilde{p}_{z,i}]^T \quad . \tag{6.22}$$

Remember: the scalar nullspace variables $\tilde{p}_{z,i}$ are given in (6.17). Although the horizontal polynomial parameter vectors $\boldsymbol{p}_{\chi,i}$ are not determined yet, - using the third and fourth line of (6.18) as constraints - they can be expressed as

$$\boldsymbol{p}_{\chi,i} = \begin{bmatrix} p_{\chi,i,1} \\ p_{\chi,i,2} \\ 0 \\ -2T_{s,i}\; p_{\chi,i,5} \;-\; \frac{10T_{s,i}^2}{3}\; p_{\chi,i,6} \\ p_{\chi,i,5} \\ p_{\chi,i,6} \end{bmatrix} . \tag{6.23}$$

Using (6.2) and (6.22), the term $\boldsymbol{d}_{\chi,i}^T(t_s)$ in (6.21) is expanded to

$$\boldsymbol{d}_{\chi,i}^T(t_s) = [1, t_s, \square, t_s^3, t_s^4, t_s^5] + \beta(t_s)\; \underbrace{\frac{[0, 0, \square, 6t_s, 12t_s^2, 20t_s^3]}{12\; \tilde{p}_{z,i}\; t_s\; (T_{s,i} - t_s)}}_{\boldsymbol{f}_{\chi,i}^T(t_s)} \;, \tag{6.24}$$

where $\beta(t_s) = z_{TD,i} - z_{floor,i} + \dot{z}_{TD,i}\, t_s - \frac{gt_s^2}{2} - 2\tilde{p}_{z,i}T_{s,i}\, t_s^3 + \tilde{p}_{z,i}\, t_s^4$. The $\square$ symbol indicates that the corresponding term has no effect on the product $\boldsymbol{d}_{\chi,i}^T(t_s)\, \boldsymbol{p}_{\chi,i}$ due to the 0 in the third element of $\boldsymbol{p}_{\chi,i}$ and is thus not considered. Now, cancelling $2t_s$ yields $\boldsymbol{f}_{\chi,i}^T(t_s) = \frac{1}{6\tilde{p}_{z,i}(T_{s,i}-t_s)}[0, 0, \square, 3, 6t_s, 10t_s^2]$. When the product $\boldsymbol{f}_{\chi,i}^T(t_s)\, \boldsymbol{p}_{\chi,i}$ is evaluated, the term $(T_{s,i} - t_s)$ can be cancelled as well and the product is equivalently written as

$$\boldsymbol{f}_{\chi,i}^T(t_s)\, \boldsymbol{p}_{\chi,i} = \underbrace{[0, 0, 0, 0, \frac{-1}{\tilde{p}_{z,i}}, \frac{-5}{3\tilde{p}_{z,i}}(t_s + T_{s,i})]}_{\check{\boldsymbol{f}}_{\chi,i}^T(t_s)}\; \boldsymbol{p}_{\chi,i} \;. \tag{6.25}$$

Setting $\boldsymbol{f}_{\chi,i}^T(t_s) = \check{\boldsymbol{f}}_{\chi,i}^T(t_s)$ and $\square = 0$ in (6.24), one finds

$$\boldsymbol{d}_{\chi,i} = [1\ ,\ t_s\ ,\ 0\ ,\ t_s^3\ ,\ t_s^4 + \frac{-\beta(t_s)}{\tilde{p}_{z,i}}\ ,\ t_s^5 + \frac{-5\beta(t_s)}{3\tilde{p}_{z,i}}(t_s + T_{s,i})]^T \;. \tag{6.26}$$

Now, the horizontal components of the mean intersection point $\overline{\boldsymbol{x}}_{int,i} = [\overline{x}_{int,i}, \overline{y}_{int,i}, z_{floor,i}]$ can be computed via

$$\overline{\chi}_{int,i} = \frac{1}{T_{s,i}} \int_{t_s=0}^{T_{s,i}} \chi_{int,i}(t_s)\; dt_s = \underbrace{\frac{1}{T_{s,i}} \int_{t_s=0}^{T_{s,i}} \boldsymbol{d}_{\chi,i}^T(t_s)\; dt_s}_{\boldsymbol{e}_{\chi,i}^T}\; \boldsymbol{p}_{\chi,i} \;. \tag{6.27}$$

Here, $\boldsymbol{e}_{\chi,i}$ is a constant vector. The simple form of (6.26) as compared to (6.24) facilitates the integration which yields the following analytical solution for $\boldsymbol{e}_{\chi,i}$:

$$\boldsymbol{e}_{\chi,i} = \begin{bmatrix} 1 \\ \frac{T_{s,i}}{2} \\ 0 \\ \frac{T_{s,i}^3}{4} \\ \frac{z_{floor,i} - z_{TD,i} - \frac{\dot{z}_{TD,i}T_{s,i}}{2} + \frac{gT_{s,i}^2}{6}}{\tilde{p}_{z,i}} + \frac{T_{s,i}^4}{2} \\ \frac{5T_{s,i}}{2\tilde{p}_{z,i}}(z_{floor,i} - z_{TD,i} - \frac{5\dot{z}_{TD,i}T_{s,i}}{9} + \frac{7gT_{s,i}^2}{36}) + \frac{19T_{s,i}^5}{18} \end{bmatrix} \tag{6.28}$$

where $\tilde{p}_{z,i}$ is the scalar variable from (6.17). The deviation of the i-th time-varying intersection point from its mean value is

$$\Delta\chi_{int,i}(t_s) = \chi_{int,i}(t_s) - \overline{\chi}_{int,i} = \underbrace{(\boldsymbol{d}^T_{\chi,i}(t_s) - \boldsymbol{e}^T_{\chi,i})}_{\boldsymbol{k}^T_{\chi,i}(t_s)}\ \boldsymbol{p}_{\chi,i} \quad . \tag{6.29}$$

The square of the deviation at a given time t_s is

$$\Delta\chi^2_{int,i}(t_s) = \boldsymbol{p}^T_{\chi,i}\ \underbrace{\boldsymbol{k}_{\chi,i}(t_s)\ \boldsymbol{k}^T_{\chi,i}(t_s)}_{\boldsymbol{L}_{\chi,i}(t_s)}\ \boldsymbol{p}_{\chi,i} \quad . \tag{6.30}$$

In order to obtain the mean square of the deviation $\chi_{int,i,ms}$, 6.30 is integrated once more and (6.20) is inserted to achieve

$$\begin{aligned} \chi_{int,i,ms} &= \boldsymbol{p}^T_{\chi,i}\ \underbrace{\frac{1}{T_{s,i}} \int_{t_s=0}^{T_{s,i}} \boldsymbol{L}_{\chi,i}(t_s)\ dt_s}_{\boldsymbol{M}_{\chi,i}}\ \boldsymbol{p}_{\chi,i} \qquad (6.31)\\ &= \boldsymbol{r}^T_{\chi,i}\boldsymbol{M}_{\chi,i}\ \boldsymbol{r}_{\chi,i}\ \tilde{p}^2_{\chi,i} + 2\ \boldsymbol{r}^T_{\chi,i}\boldsymbol{M}_{\chi,i}\ \boldsymbol{p}_{\chi,i,0}\ \tilde{p}_{\chi,i} + \boldsymbol{p}^T_{\chi,i,0}\ \boldsymbol{M}_{\chi,i}\ \boldsymbol{p}_{\chi,i,0} \quad . \end{aligned}$$

The analytical solution (6.28) facilitates the (also analytical) computation of matrix $\boldsymbol{M}_{\chi,i}$ which is finally found as

$$M_{\chi,i} = \boldsymbol{\Phi}_{focus,i}\ \boldsymbol{T}_{focus,i}\ \boldsymbol{\Phi}^T_{focus,i} \quad . \tag{6.32}$$

Here, the matrix $\boldsymbol{\Phi}_{focus,i}$ is

$$\boldsymbol{\Phi}_{focus,i} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ -\frac{\dot{z}_{TD,i}}{\tilde{p}_{z,i}} & \frac{g}{2\tilde{p}_{z,i}} & 2T_{s,i} & 0 & 0 \\ \frac{5}{3\tilde{p}_{z,i}}\Big[v_1 & v_2 & v_3\Big] & \frac{5T_{s,i}}{3} & -\frac{2}{3} \end{bmatrix} \quad , \tag{6.33}$$

With $v_1 = (z_{floor,i} - z_{TD,i} - \dot{z}_{TD,i}T_{s,i})$, $v_2 = \frac{gT_{s,i}}{2} - \dot{z}_{TD,i}$ and $v_3 = 2\tilde{p}_{z,i}T^2_{s,i} + \frac{g}{2}$. All time integrals are combined in

$$\boldsymbol{T}_{focus,i} = \begin{bmatrix} \frac{T^2_{s,i}}{12} & \frac{T^3_{s,i}}{12} & \frac{3T^4_{s,i}}{40} & \frac{T^5_{s,i}}{15} & \frac{5T^6_{s,i}}{84} \\ \frac{T^3_{s,i}}{12} & \frac{4T^4_{s,i}}{45} & \frac{T^5_{s,i}}{12} & \frac{8T^6_{s,i}}{105} & \frac{5T^7_{s,i}}{72} \\ \frac{3T^4_{s,i}}{40} & \frac{T^5_{s,i}}{12} & \frac{9T^6_{s,i}}{112} & \frac{3T^7_{s,i}}{40} & \frac{5T^8_{s,i}}{72} \\ \frac{T^5_{s,i}}{15} & \frac{8T^6_{s,i}}{105} & \frac{3T^7_{s,i}}{40} & \frac{16T^8_{s,i}}{225} & \frac{T^9_{s,i}}{15} \\ \frac{5T^6_{s,i}}{84} & \frac{5T^7_{s,i}}{72} & \frac{5T^8_{s,i}}{72} & \frac{T^9_{s,i}}{15} & \frac{25T^{10}_{s,i}}{396} \end{bmatrix} \tag{6.34}$$

With the matrix $\boldsymbol{M}_{\chi,i}$ and differentiating (6.31) with respect to $\tilde{p}_{\chi,i}$, the optimal parameter is found

$$\tilde{p}^*_{\chi,i} = -\frac{\boldsymbol{r}^T_{\chi,i}\ \boldsymbol{M}_{\chi,i}\ \boldsymbol{p}_{\chi,i,0}}{\boldsymbol{r}^T_{\chi,i}\ \boldsymbol{M}_{\chi,i}\ \boldsymbol{r}_{\chi,i}} \quad , \tag{6.35}$$

which minimizes the mean square deviation as defined above. With (6.35), (6.20) turns into

$$\boldsymbol{p}_{\chi,i} = \underbrace{(\boldsymbol{I} - \frac{\boldsymbol{r}_{\chi,i}\ \boldsymbol{r}^T_{\chi,i}\ \boldsymbol{M}_{\chi,i}}{\boldsymbol{r}^T_{\chi,i}\ \boldsymbol{M}_{\chi,i}\ \boldsymbol{r}_{\chi,i}})}_{\boldsymbol{\Omega}_{\chi,i}}\ \underbrace{\boldsymbol{B}^T_{\chi,i}(\boldsymbol{B}_{\chi,i}\ \boldsymbol{B}^T_{\chi,i})^{-1}}_{\boldsymbol{B}^+_{\chi,i}}\ \boldsymbol{b}_{\chi,i} \quad , \tag{6.36}$$

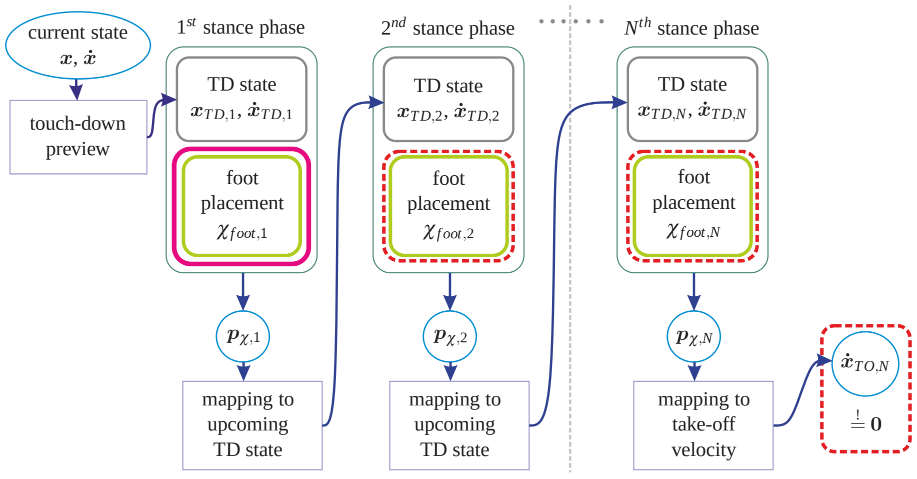
current state
$\boldsymbol{x}, \dot{\boldsymbol{x}}$
touch-down preview
1st stance phase
2nd stance phase
Nth stance phase
TD state
$\boldsymbol{x}_{TD,1}, \dot{\boldsymbol{x}}_{TD,1}$
$\boldsymbol{x}_{TD,2}, \dot{\boldsymbol{x}}_{TD,2}$
$\boldsymbol{x}_{TD,N}, \dot{\boldsymbol{x}}_{TD,N}$
foot placement
$\chi_{foot,1}$
$\chi_{foot,2}$
$\chi_{foot,N}$
$\boldsymbol{p}_{\chi,1}$
$\boldsymbol{p}_{\chi,2}$
$\boldsymbol{p}_{\chi,N}$
mapping to upcoming TD state
mapping to take-off velocity
$\dot{\boldsymbol{x}}_{TO,N}$
$\overset{!}{=} \boldsymbol{0}$

Figure 6.5: CoM touch-down adjustment for foot targeting.

which directly maps the horizontal boundary conditions $\boldsymbol{b}_{\chi,i}$ to appropriate polynomial parameter vectors $\boldsymbol{p}_{\chi,i}$ (including best force focus). If - as in [67] - horizontal CoM touch-down target positions (or similarly: take-off velocities) are used as boundary conditions, (6.36) provides the solution to the problem.

6.4.3 Foot step targeting and leg cross-over avoidance

In [67], foot positions could not be controlled directly, which caused problems with leg cross-over (see Fig. 6.7). Also when precise foot placement is required - e.g. when running over stepping stones as in Fig. 6.1 - the method failed to provide any guarantee of safe stepping. To address these drawbacks, this work aims at an explicit solution for foot-step targeting. Setting $\overline{\chi}_{int,i} = \chi_{foot,i}$ in (6.27), and inserting (6.36), one can solve for the desired upcoming CoM touch-down position[2] $\chi_{TD,i+1,des}$, which corresponds to the desired foot location $\chi_{foot,i}$. Re-substituting this particular $\chi_{TD,i+1,des}$ in (6.36) yields

$$\boldsymbol{p}_{\chi,i} = [\underbrace{(\boldsymbol{I} - \boldsymbol{e}^{\oplus}_{\chi,i}\, \boldsymbol{e}^{T}_{\chi,i})\ \boldsymbol{\Omega}_{\chi,i}\, \boldsymbol{\Pi}_{\chi,i},}_{\boldsymbol{A}_{TD,\chi,i}} \quad \boldsymbol{e}^{\oplus}_{\chi,i}] \begin{bmatrix} \chi_{TD,i} \\ \dot{\chi}_{TD,i} \\ \chi_{foot,i} \end{bmatrix} \quad . \tag{6.37}$$

Here, $\boldsymbol{A}_{TD,\chi,i}$ maps the i-th touch-down state to $\boldsymbol{p}_{\chi,i}$ and the specific pseudo-inverse $\boldsymbol{e}^{\oplus}_{\chi,i} = \frac{\boldsymbol{\Omega}_{\chi,i}\, \boldsymbol{\pi}_{\chi,i}}{\boldsymbol{e}^{T}_{\chi,i}\, \boldsymbol{\Omega}_{\chi,i}\, \boldsymbol{\pi}_{\chi,i}}$ of $\boldsymbol{e}_{\chi,i}$ maps the i-th foot position. The matrix $\boldsymbol{\Pi}_{\chi,i}$ combines the first two column vectors of $\boldsymbol{B}^{+}_{\chi,i}$, while $\boldsymbol{\pi}_{\chi,i}$ is its final column. Note: the third and fourth boundary conditions in (6.18) are implicitly accounted for. Now, all previewed *desired footholds* $\chi_{foot,i} = \chi_{foot,des,i}$ (excluding the first one) and the final take-off velocity $\dot{\chi}_{TO,N} = \boldsymbol{0}$ will be used as constraints and the first foothold $\chi_{foot,1}$ (*control variable*) and all future horizontal polynomial parameter vectors $\boldsymbol{p}_{\chi,i}$ will be solved for, which yield perfect tracking of the future desired footholds. By combining the touch-down state into $\boldsymbol{\tau}_{\chi,i} = [\chi_{TD,i}, \dot{\chi}_{TD,i}]^{T}$, (6.37) becomes

$$\boldsymbol{p}_{\chi,i} = \boldsymbol{A}_{TD,\chi,i}\ \boldsymbol{\tau}_{\chi,i} + \ \boldsymbol{e}^{\oplus}_{i}\ \ \chi_{foot,i} \ . \tag{6.38}$$

[2]This is why $\chi_{TD,i+1,des}$ was called an “intermediate control target” earlier.

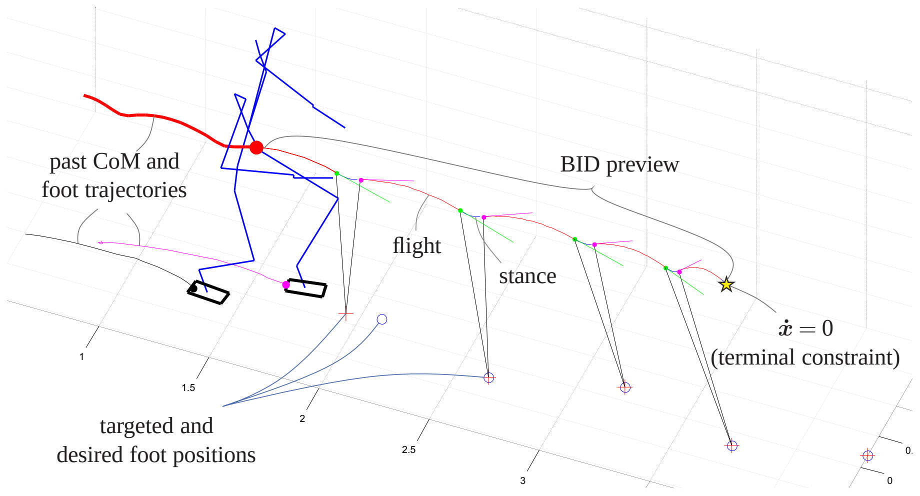

Figure 6.6: BID preview of Toro (displayed as stickman) running in OpenHRP. All desired foot positions (except the first one) are previewed to be perfectly tracked.

Each upcoming touch-down state can be previewed as

$$\boldsymbol{\tau}_{\chi,i+1} = \begin{bmatrix} \chi_{TD,i+1} \\ \dot{\chi}_{TD,i+1} \end{bmatrix} = \underbrace{\begin{bmatrix} \boldsymbol{t}_{\chi,i}^T(T_{s,i}) + T_{f,i}\ \boldsymbol{t}_{\dot{\chi}}^T(T_{s,i}) \\ \boldsymbol{t}_{\dot{\chi}}^T(T_{s,i}) \end{bmatrix}}_{\boldsymbol{S}_i} \boldsymbol{p}_{\chi,i} \ . \tag{6.39}$$

Starting from $i = 1$ and propagating (6.38) and (6.39) forward, the following expression for the polynomial parameter vector of the N-th (i.e. final) previewed stance phase is found:

$$\boldsymbol{p}_{\chi,N} = \boldsymbol{G}_{\chi,1}\ \boldsymbol{A}_{TD,\chi,1}\ \boldsymbol{\tau}_{\chi,1} + \sum_{i=1..N} (\boldsymbol{G}_{\chi,i}\ \boldsymbol{e}_{\chi,i}^{\oplus}\ \chi_{foot,i}) \ . \tag{6.40}$$

The matrices $\boldsymbol{G}_{\chi,i} = \boldsymbol{G}_{\chi,i+1}\ \boldsymbol{A}_{TD,\chi,i+1}\ \boldsymbol{S}_i$ are evaluated by starting with $\boldsymbol{G}_{\chi,N} = \boldsymbol{I}_{6x6}$ and iterating backwards until $i = 1$. Now, with (6.2) and (6.40), the horizontal take-off velocity after the final previewed stance phase is found as

$$\dot{\chi}_{TO,N} = \boldsymbol{t}_{\dot{\chi}}^T(T_{s,N})\ \boldsymbol{p}_{\chi,N} \ . \tag{6.41}$$

To guarantee stability, $\dot{\chi}_{TO,N} = 0$ is chosen as terminal constraint, i.e. the controller presumes the CoM to come to a full stop after the final previewed contact phase.
Note that in the same way for the computation of the final vertical polynomial parameter vector, the terminal constraint $\dot{z}_{TO,N} = 0$ is used instead of a desired apex height.
As mentioned above and shown in Fig. 6.5, the terminal constraint and all desired foot targets $\chi_{foot,des,i}$ are used as target quantities, other than the first one (which is sacrificed in order to serve as a control variable). Therefore, (6.40)-(6.41) are solved for $\chi_{foot,1}$ which finally yields the sought-after first foot placement

$$\chi_{foot,1} = \frac{-\boldsymbol{t}_{\dot{\chi}}^T(T_{s,N})\ (\boldsymbol{G}_{\chi,1}\ \boldsymbol{A}_{TD,\chi,1}\ \boldsymbol{\tau}_{\chi,1} + \boldsymbol{\eta})}{\boldsymbol{t}_{\dot{\chi}}^T(T_{s,N})\ \boldsymbol{G}_{\chi,1}\ \boldsymbol{e}_{\chi,1}^{\oplus}} \ , \tag{6.42}$$

$$\boldsymbol{\eta} = \sum_{i=2..N} (\boldsymbol{G}_{\chi,i}\ \boldsymbol{e}_{\chi,i}^{\oplus}\ \chi_{foot,des,i}) \ .$$

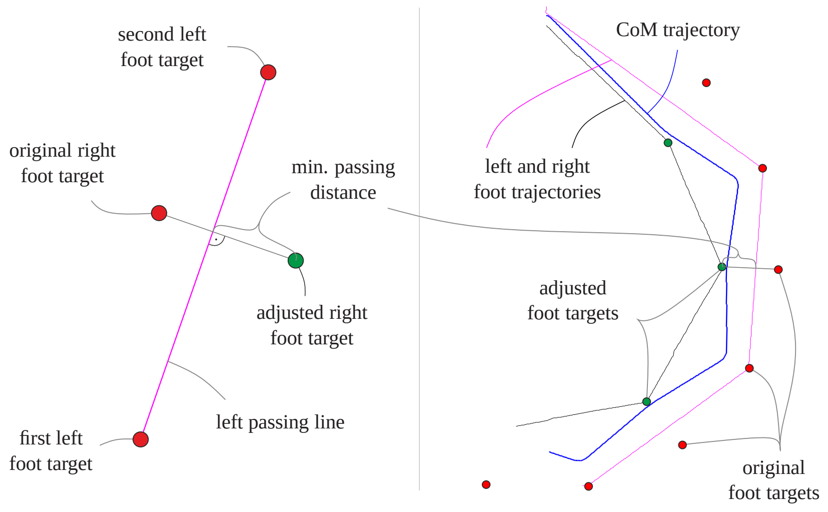

Figure 6.7: Leg cross-over avoidance,
left: scheme (depicted for left pass), right: simulation output

Now all horizontal polynomial parameter vectors $\boldsymbol{p}_{\chi,i}$ are solved for by alternately evaluating (6.38) and (6.39). As foot positions in (6.38), $\chi_{foot,i} \in \{\chi_{foot,1}, \chi_{foot,des,2}, \cdots \chi_{foot,des,N}\}$ are used. During stance the first foot position $\chi_{foot,1}$ and the polynomial parameter vector $\boldsymbol{p}_{\chi,1}$ (feed-forward) are frozen and the second foothold $\chi_{foot,2}$ is used as control variable instead. That way - even in face of unknown perturbations - the foot targets are continuously adjusted. Equation (6.42) is adopted accordingly.

One feature of the presented framework is that due to the multi-step preplanning, both future foot aim points $\chi_{foot,1}$ and $\chi_{foot,2}$ (i.e. the aim points of the left and right foot) are known at all times, which facilitates foot trajectory generation. In this work, the foot trajectories are implemented as polynomials. The achieved precise foot targeting is particularly interesting for running over 3D stepping stones or other restricted surfaces.

An additional feature of precise foothold targeting is that leg cross-over can be explicitly avoided. This is especially helpful for running in sharp turns (see Fig. 6.7). Therefore, the originally preplanned footholds can be adjusted such that the left foot always passes by the right foot on the left, and vice versa. At the same time, the Euclidean distance of the adjusted footholds from the originally planned ones should be minimal. This way, the legs can be prevented from twisting around each other. To achieve this goal, an adjustment heuristic as shown in Fig. 6.7 is used. In the shown example, four foot positions are previewed, i.e. two for each foot. The method adjusts the second/third desired footstep (i.e. the projection shown in Fig. 6.7, left, is applied twice), such that the swing feet can safely swing from the first/second foothold to the third/fourth one. The fourth foothold remains unchanged to achieve good long term tracking of the original desired foot locations.

6.5 State feedback control

In the nominal case (no perturbations), the force profiles and foot aim points as derived in the previous sections assure that - for any initial conditions -, after the first stance phase, all desired boundary conditions from sections 6.4.1 and 6.4.2 are fulfilled (deadbeat control). Therefore, planning once per step or even pre-planning a whole sequence of upcoming steps would be sufficient. Yet, to cope with perturbations, this work proposes a state feedback control method, which is based on continuous re-planning of the desired contact forces throughout both flight and stance phases. During flight the previewed CoM touch-down state is updated (see Fig. 6.8) via (6.7). In contrast to [67] (no feedback during stance), during stance, the first take-off state is predicted via

$$\begin{bmatrix}\sigma_{TO,1}\\ \dot{\sigma}_{TO,1}\end{bmatrix} = \underbrace{\begin{bmatrix}\sigma\\ \dot{\sigma}\end{bmatrix}}_{feedback} + \underbrace{\begin{bmatrix}\boldsymbol{t}_{\sigma}^{T}(T_{s,1}) - \boldsymbol{t}_{\sigma}^{T}(t_s)\\ \boldsymbol{t}_{\dot{\sigma}}^{T}(T_{s,1}) - \boldsymbol{t}_{\dot{\sigma}}^{T}(t_s)\end{bmatrix}\boldsymbol{p}_{\sigma,1}}_{preview} \ , \quad \sigma \in \{x,y,z\} \tag{6.43}$$

Here, $\boldsymbol{t}_{\sigma}^{T}(t)$ and $\boldsymbol{t}_{\dot{\sigma}}^{T}(t)$ are the time-mapping row vectors from (6.2). They are evaluated for the first total stance time $T_{s,1}$ and the current time in stance $t_s \in [0, T_{s,1}]$ to predict how much of an offset is expected if for the remaining time in step the current force profile (encoded by $\boldsymbol{p}_{\sigma,1}$) is applied. This offset is added to the current measured state to predict the take-off state, which in turn is used to compute the upcoming CoM touch-down state. Note: after touch-down, the force profile of the current stance phase is frozen and commanded to the robot as feed-forward. The main advantage of the presented state feedback during stance is that the foot aim points are continuously updated to avoid discontinuities in the foot reference trajectories.

During flight, the first upcoming foot position is one of the *control inputs* (see (6.42)). Whilst all other future footsteps are previewed to coincide with the desired foot target locations (see Fig. 6.6), the nominal position of the first foot is an output of the controller. Depending on the limitations at hand (e.g. limited allowable supporting area) - this nominal foot aim point may have to be projected to a feasible one, resulting in deviations from the nominal deadbeat behavior.

6.6 Guaranteeing leg force feasibility

The desired 3D force acting on the CoM can be computed for a given time in stance t_s as

$$\boldsymbol{F}_{CoM,des}(t_s) = m \begin{bmatrix}\boldsymbol{t}_{\ddot{x}}^{T}(t_s)\ \boldsymbol{p}_{x,1}\\ \boldsymbol{t}_{\ddot{y}}^{T}(t_s)\ \boldsymbol{p}_{y,1}\\ \boldsymbol{t}_{\ddot{z}}^{T}(t_s)\ \boldsymbol{p}_{z,1}\end{bmatrix} \ , \tag{6.44}$$

i.e. the polynomial of the first force profile is evaluated. The corresponding desired external (e.g. leg-) force $\boldsymbol{F}_{ext,des}$ is found by reordering (6.1):

$$\boldsymbol{F}_{ext,des} = \boldsymbol{F}_{CoM,des} - \boldsymbol{F}_{g} \quad . \tag{6.45}$$

The polynomial parameters were chosen in order to achieve the best focus of the leg forces with the ground. Yet, for physical robots feasibility is not guaranteed.

Point-mass point-feet model

One obvious example is when the robot is modeled as point-mass with point feet. In that case, the leg force is constrained to point along the unit vector $\boldsymbol{u}_{x,f}$ from CoM to point foot. As the other two spatial directions are unactuated, the desired external (i.e. leg-) force $\boldsymbol{F}_{ext,des}$ has to be projected to the feasible direction[3]:

$$\boldsymbol{F}_{ext,f} = \boldsymbol{u}_{x,f}\boldsymbol{u}_{x,f}^{T}\boldsymbol{F}_{ext,des} \quad . \tag{6.46}$$

Assuming sufficient ground friction, $\boldsymbol{F}_{leg,f}$ can be safely commanded to the point-mass/foot model.

[3]Note: for more complex robots this projection may not be necessary.

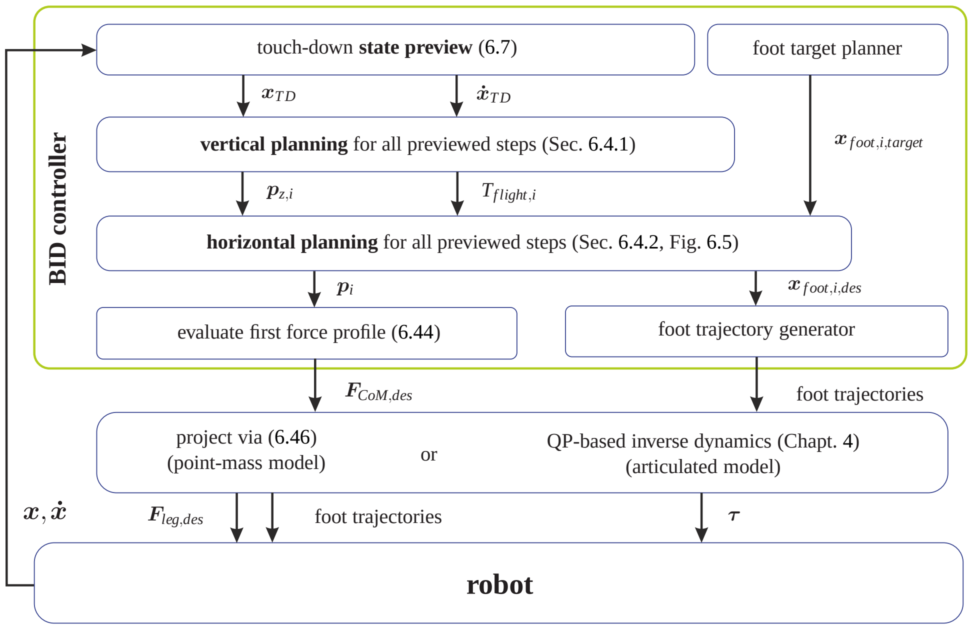
BID controller
touch-down **state preview** (6.7)
foot target planner
x_{TD}
$\dot{x}_{TD}$
vertical planning for all previewed steps (Sec. 6.4.1)
$x_{foot,i,target}$
$p_{z,i}$
$T_{flight,i}$
horizontal planning for all previewed steps (Sec. 6.4.2, Fig. 6.5)
p_i
$x_{foot,i,des}$
evaluate first force profile (6.44)
foot trajectory generator
$F_{CoM,des}$
foot trajectories
project via (6.46) (point-mass model)
or
QP-based inverse dynamics (Chapt. 4) (articulated model)
$x, \dot{x}$
$F_{leg,des}$
foot trajectories
τ
robot

Figure 6.8: Computation flow of BID controller (outline).

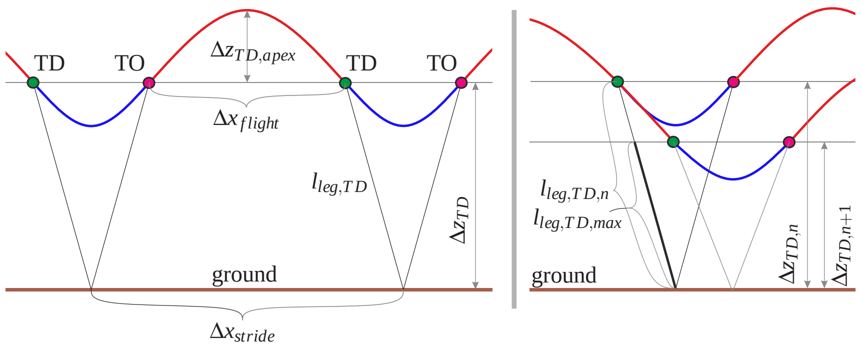

Figure 6.9: Correlations for stationary running.

Articulated multi-body model

As in chapter 5 on walking, the main idea of the BID control concept is to first focus on the robot's CoM dynamics and the problem of foot placement, which in the author's view are the key challenges of locomotion. Once CoM dynamics and foot placement are solved, they need to be embedded into a more general control framework, such as the QP-based whole-body control (WBC) framework presented in chapter 4, to make them available for complex robots, such as humanoids. The mentioned WBC framework tries to reconcile the specified tasks as best as possible while guaranteeing feasibility. Note that feasibility here only relates to ground reaction wrenches and joint torques, whereas stability or balance (depending on the physical limitations of the given robot) cannot be guaranteed.

6.7 Enhancing kinematic feasibility

A major issue concerning the porting of BID control to kinematically restricted robot models (such as humanoid robots) is that the BID controller does not naturally consider any kinematic limitations. In case of high desired velocities and accelerations or strong perturbations, BID control may result in unrealistic high required leg lengths. In the next two subsections, two methods to ease this problem (while not guaranteeing feasibility) will be presented: one for finding nominally feasible gait parameters and one for online touch-down leg length adjustment. Both methods assume that the distance from CoM to foot in the BID preview correlates with the leg length in a multi-body model. This is an approximation of course.

6.7.1 Nominally feasible gait design

For a periodic running gait (assumed here), the CoM height at touch-down z_{TD} equals the one at take-off z_{TO} (see Fig. 6.9). With energy conservation ($\dot{z}_{TO}^2 = 2\,g\,\Delta z_{TD,apex}$) and with (6.5), the time of flight T_f (i.e. from TO to TD) can be derived as

$$T_f = \sqrt{\frac{8\,\Delta z_{TD,apex}}{g}} \quad . \tag{6.47}$$

Here, $\Delta z_{TD,apex}$ denotes the height difference between apex and touch-down. For a desired flight percentage $f_{flight} = \frac{T_f}{T_s+T_f}$ and with the mean horizontal speed (e.g. derived from a joystick input) $v_{mean} = \frac{\Delta x_{stride}}{T_s+T_f}$ one gets

$$T_f = f_{flight}\;\underbrace{(T_s + T_f)}_{\frac{\Delta x_{stride}}{v_{mean}}} \;\leq\; f_{flight}\,\frac{\Delta x_{stride,max}}{v_{mean}} \quad . \tag{6.48}$$

Here, the inequality indicates, that the time of flight should be small enough, such that a maximum desirable stride length $\Delta x_{stride,max}$ is not exceeded. By combining (6.47) and (6.48), a condition for the maximum allowable height difference between apex and touch-down is found:

$$\Delta z_{TD,apex} \;\leq\; \underbrace{\frac{g}{8}\,(f_{flight}\,\frac{\Delta x_{stride,max}}{v_{mean}})^2}_{\Delta z_{TD,apex,max}} \quad . \tag{6.49}$$

A second condition for nominal kinematic feasibility is that a maximum allowable touch-down leg length $l_{leg,TD,max}$ is not exceeded. By inspection of Fig. 6.9 (left), the following condition for the CoM touch-down height is determined to be

$$\Delta z_{TD} \leq \underbrace{\sqrt{l_{leg,TD,max}^2 - (\frac{\Delta\hat{x}_{stride} - \Delta\hat{x}_{flight}}{2})^2}}_{\Delta z_{TD,max}} \quad . \tag{6.50}$$

Here, $\Delta\hat{x}_{stride} = v_{mean}\,(T_s + T_f)$ and $\Delta\hat{x}_{flight} = v_{mean}\,T_f$ denote the approximated (assuming constant horizontal velocity) distances traveled during a whole stride and during a single flight phase, respectively. With the described adjustments of apex (6.49) and touch-down height difference (6.50), the nominal desired touch-down and apex height difference become

$$\Delta z_{TD,des} = min(\Delta z_{TD,nominal}\,,\;\; \Delta z_{TD,max}) \qquad \textit{and} \tag{6.51}$$

$$\Delta z_{apex,des} = \Delta z_{TD,des} + min(\Delta z_{TD,apex,nominal}\,,\;\; \Delta z_{TD,apex,max}) \quad ,$$

where $\Delta z_{TD,nominal}$ and $\Delta z_{TD,apex,nominal}$ act as upper limits. Also, the nominal stance time can be computed with (6.47) as

$$T_s = \frac{1 - f_{flight}}{f_{flight}} \sqrt{\frac{8\ \Delta z_{TD,apex}}{g}} \quad . \tag{6.52}$$

That way, the design parameters introduced in Sec. 6.2 (i.e. $\Delta z_{TD,des}$, $\Delta z_{apex,des}$ and T_s) are deduced from desired flight percentage f_{flight}, maximum desired stride length $\Delta x_{stride,max}$, maximum desired touch-down leg length $l_{leg,TD,max}$ and mean horizontal speed v_{mean} to maximize the kinematic feasibility.

6.7.2 Active leg length control

In case of strong perturbations, the leg lengths resulting from the BID controller may not comply with kinematic limitations of multi-body robots. To ease this problem, the original BID plan is adjusted via the following iteration scheme (see Fig. 6.9 (right))

$$\Delta z_{TD,i,n+1} = min(\Delta z_{TD,des}\ ,\ \frac{l_{leg,TD,max}}{l_{leg,TD,i,n}}\ \Delta z_{TD,i,n})) \quad . \tag{6.53}$$

The touch-down height difference is iteratively adjusted (if it doesn't exceed the nominal touch-down height $\Delta z_{TD,des}$) such that for each stance phase i the resulting touch-down leg length $l_{leg,TD,i}$ does not exceed the maximum desired touch-down leg length $l_{leg,TD,max}$ (similar the rest length of SLIP models). Here, n denotes the iteration count. For each iteration the complete BID preview has to be re-evaluated.

6.8 Simulations and evaluation of BID control

6.8.1 BID-based point-mass simulations

To test the performance and robustness of the proposed control framework, numerous simulations were performed. For the first set of simulations, a point-mass with two massless point-feet was considered. Figure 6.10 shows the results of a robustness examination for three different constant external forces. From top to bottom, the figure shows phase plots for three simulations. Each simulation was setup in the following way: no perturbation during the first 4 seconds, then 4 seconds of constant force acting (magnitude: -50N (corresponding to $\approx$ 10% of the robot's mass (here 50kg)), force direction: purely x, y and z, respectively), followed by 4 seconds without perturbation. Here, $\Delta x = x - x_{joystick}$ and $\Delta y = y - y_{joystick}$ denote the errors w.r.t. the nominal horizontal CoM position $\boldsymbol{x}_{joystick} = [x_{joystick}, y_{joystick}]^T$, which was computed from a virtual joystick input. The stars denote the initial states. The phase plots show that for perturbed and unperturbed phases, the system quickly converges to corresponding limit cycles. Note: the perturbation forces in the shown simulations were kept comparably low to increase readability of the plots.

Many further BID-based simulations with a bipedal point-mass robot were performed, which showed a very high robustness of the basic BID controller. It has to be mentioned, that for extreme perturbations, the leg length could grow to unrealistic levels (due to the constant touch-down height). To assure leg length feasibility, the method from Sec. 6.7.2 can be applied. The controller is most sensitive against strong unknown perturbations that point towards the ground. In the simulation, the maximum permanent force the controller could withstand was $-750N$, i.e. 1.5 times the robots weight. For higher forces, the robot's CoM would hit the ground.

Figure 6.11 shows the result of a simulation in which the point-mass robot was running over three-dimensional stepping stones (see also Fig. 6.1). The left subplot shows the robot's foot

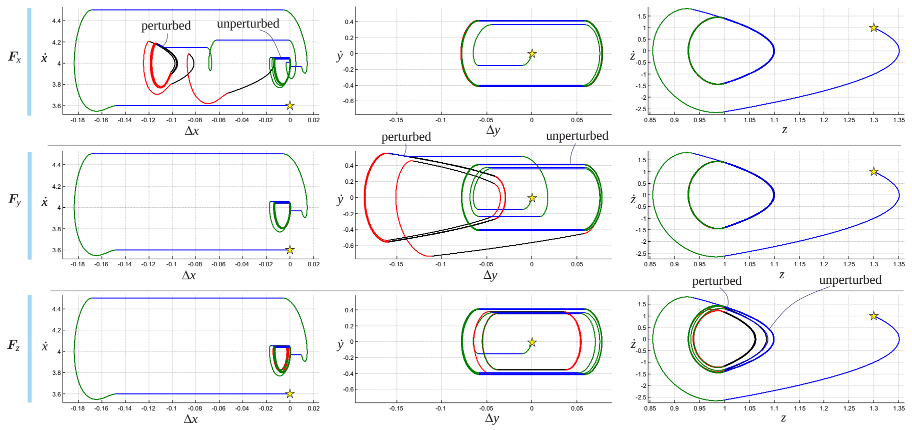

Figure 6.10: Robustness examination (point-mass) for different constant external forces. Perturb. inactive: stance green, flight blue. Perturb. active: stance red, flight blue.

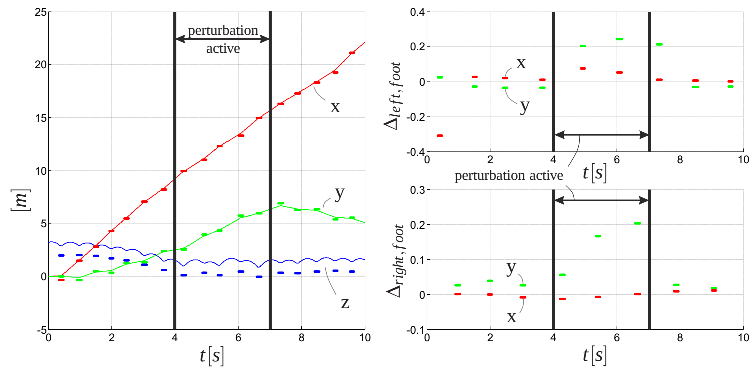

Figure 6.11: CoM trajectories and foot target tracking performance in nominal and perturbed case (point mass simulation over stepping stones). Perturbing force: 30N in y-direction.

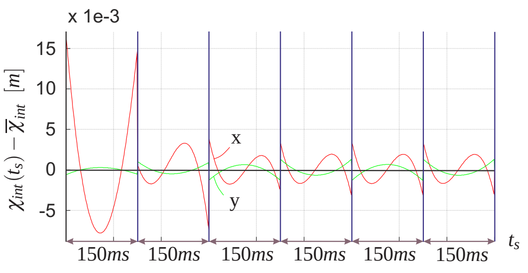

Figure 6.12: Deviation of force intersection point $\chi_{int}(t_s)$ from mean intersection point $\overline{\chi}_{int}$ in case that the *forces are not projected (in contrast to other point-mass simulations)*.

positions (bars, only active during stance) and CoM positions (continuous curves). The right subplots show the difference between desired and achieved foot positions. Nominally, the foot target positions are tracked well, whereas in case of perturbations they deviate. This is necessary to stabilize the CoM motion against the perturbation. After the perturbation is removed, good tracking is regained after a single step.

Figure 6.12 shows how far the force intersection point $\chi_{int}(t_s)$ deviates from the mean intersection point $\overline{\chi}_{int}$ (i.e. the stance foot position) for the case that the *desired force profiles* are *not projected*. In the shown simulation, the robot starts at zero speed and then runs at $2\frac{m}{s}$. The stance time is set to $150ms$[4]. The initial range of deviation is about $22mm$, while for stationary running it is about $6mm$. This shows that the original (non-projecting) method is well applicable for small-footed robots and that equation (6.46) typically has minor influence.

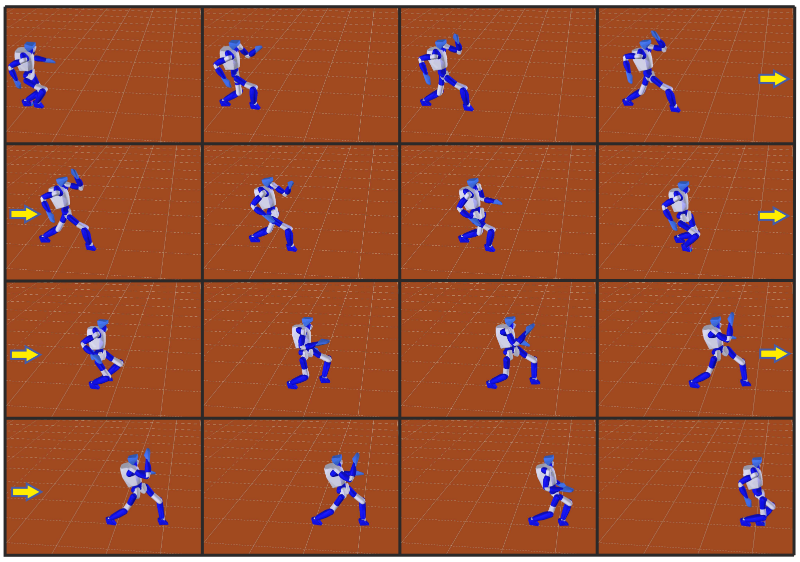

Figure 6.13: Toro [1] running in OpenHRP [2] at 5 m/s.

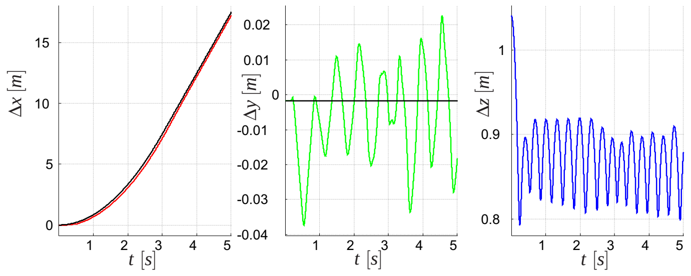

Figure 6.14: Toro's CoM while running in OpenHRP at $0-5\ \frac{m}{s}$.

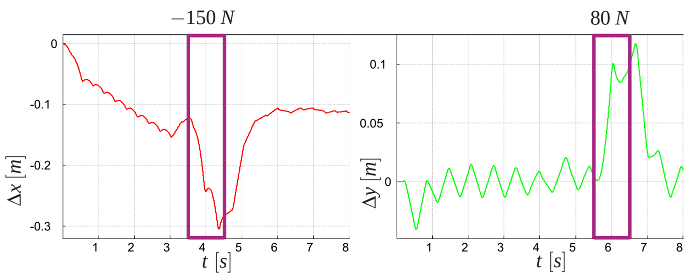

Figure 6.15: CoM error during push-recovery simulation.

[4]Along the time-axis, the stance phases are pieced together.

6.8.2 QP-based multi-body simulations

To proof the applicability of the biologically inspired deadbeat (BID) control framework, it was embedded into the QP-based whole-body controller from Sec. 4 and full-body simulations of the humanoid robot Toro [1] in OpenHRP [2] were performed. It has to be noted, that Toro's joint torque and velocity limits were omitted in the simulations. Figure 6.13 shows Toro running at $5m/s$. The gait parameters $\Delta z_{TD,des}$, $\Delta z_{apex,des}$ and T_s were computed via the method from Sec. 6.7.1 to make such high running speed kinematically feasible for Toro. Following intuitive design parameters (as described in Sec. 6.7.1) were chosen: desired flight percentage $f_{flight} = 0.7$, maximum desired stride length $\Delta x_{stride,max} = 1.4\,m$, maximum desired touch-down leg length $l_{leg,TD,max} = 0.86\,m$, nominal touch-down height $\Delta z_{TD,nominal} = 0.86\,m$ and nominal height difference between touch-down and apex $\Delta z_{TD,apex,nominal} = 0.06\,m$. The target velocity (derived from a virtual joystick input and used as v_{mean}) ramped up from $0\ m/s$ to $5\ m/s$ until second 3 and then stayed constant. Two important human-like features *evolved*: first, natural arm swing motions (see also multimedia attachment) that facilitate the angular momentum regulation and contribute to the CoM manipulation and second, stretched hind legs at the end of stance. This shows that the combination of BID and whole-body control can automatically create human-like motions, such that the effect of the various cost functions and their weights can be examined.

The foot targets were derived from a joystick input. The CoM motion (see Fig. 6.14, colored) follows the desired joystick reference (black) nicely. Knowing the vertical previewed dynamics and thus the times to each upcoming touch-down, the foot targets were placed at lateral offsets from the nominal sway-free and continuous joystick reference.

A very important quality of a running controller is its reactiveness and robustness. Without that quality, the OpenHRP running simulations would fail due to the overdeterminedness of tasks (such as CoM force and angular momentum control, posture control etc.), tracking errors and energy losses at impact. To investigate this quality of the presented combined BID and whole-body control framework, multiple simulations were performed where the robot was subject to external perturbations. One of these simulations is shown in Fig. 6.15. It displays the errors in horizontal CoM position with regard to the joystick reference. Toro runs at $3\ \frac{m}{s}$ (after ramping up from $0\ \frac{m}{s}$ until second 3). From second $3.5-4.5$ it is subject to a backwards pointing external force of $-150\ N$ and between second 5.5 and 6.5 to a lateral force of $80\ N$ (both constant and unknown). The controller is well able to compensate for these perturbations and recovers after just a few steps. The steady state error of about $0.1\ m$ in x-direction can be explained by the fact that the *foot step* (not the CoM) is planned to coincide with the joystick reference (aside from a sideward offset) at the *instant of touch-down*, while the continuous joystick reference keeps moving throughout stance. The kinematic feasibility of the running gait under these strong perturbations was facilitated by the methods from Sec. 6.7.
The OpenHRP simulations of Toro running show the control framework's robustness and reliability. It is thus a promising concept for future more detailed comparison between human and humanoid running and prediction of human behavior.

6.9 Comparison to human experiment

The BID controller had been inspired by observations from human running experiments. In the previous section, its high robustness was shown, which substantiates its applicability for humanoid running control. Now the question arises, how well the BID control outputs fit the ones observed in human running experiments. Thus, the loop is closed by comparing the corresponding forces and CoM trajectories. Figure 6.16 (left) shows a human subject running on a force plate treadmill,

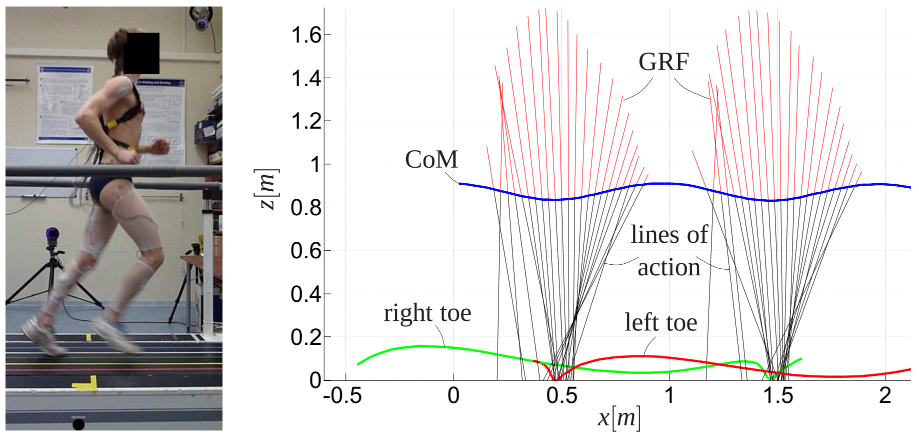

Figure 6.16: Human running experiment. Left: subject running on force plate treadmill, right: trajectories and GRF.

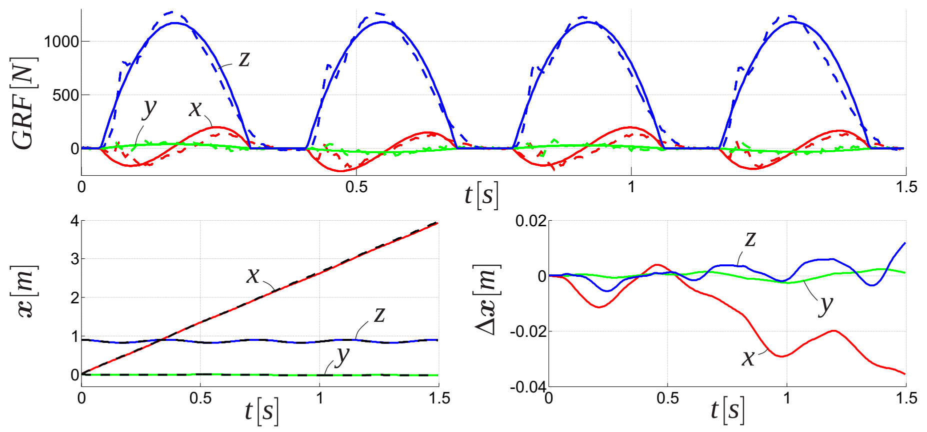

Figure 6.17: Comparison of human experiment data [101, 102] (dashed) and output of matched BID simulation. top: ground reaction forces (GRF), left: CoM position, right: CoM error.

its posture being tracked via markers. On the right side of the figure, the corresponding CoM and toe trajectories are shown. It becomes apparent that the lines of action of the ground reaction forces (GRF) in humans are not as strictly focused as the ones designed in the BID controller (compare to figures 6.3 and 6.4). This shows that humans make use of angular momentum during running, while the CoP remains in the ball of the foot (compare toe trajectories). Figure 6.17 shows the corresponding force profiles and CoM trajectories and overlays them to a "matched"[5] BID simulation. The force profiles match quite well. The main differences are the initial impacts, slightly higher vertical force maximum and lower final force slope in humans as compared to the BID simulation. The CoM positions are very consistent. The errors are in the range of several millimeters. The x direction is drifting due to slight timing mismatch. From these observations it is inferred that BID control sufficiently approximates the GRF in human running to allow for decent insights into human running control. Yet, the observed differences motivate further examination of human running control.

[5] "matching" means that basic gait parameters such as stance time, apex and touch-down height and also the human subject's and the model's mass have to be aligned. Otherwise a comparison - especially in time domain - would be impossible/useless.

6.10 BID control: discussion and outlook

6.10.1 Strengths and limitations of current control framework

In this chapter a closed-form solution to 3D running was provided. The control framework yields leg force profiles that are independent of the specific hardware design of a particular robot, i.e. it is generic. Also, it might be used to identify required actuator characteristics for new robots.

For the running simulations, a standard PC (3.3 GHz, quad-core, Win7 64bit) was used. In the Matlab/Simulink simulation setup, 1 *ms* was used as sampling time. All BID control related computations were executed in real-time.

The force profiles as derived in sections 6.4.1 and 6.4.2 nominally lead to perfect tracking after just one stance phase (deadbeat control), i.e. the controller is perfectly stable. In case of actuation limits, the control commands may have to be adjusted (e.g. via (6.46) for point-mass point-feet robots), so stability cannot be guaranteed. Yet, the simulations show the high robustness of the controller even in case of constraints.

In the presented control framework, impact-free state transitions are assumed (compare Fig. 6.2). The impact losses in real systems will cause perturbations. Notwithstanding, due to its high robustness in simulations, good performance of the controller is expected.

A drawback of the current control setup, when compared to human running, is the missing toe-off motion. In the current setup the feet are aligned with the ground during contact. Toe-off motion (especially during single support) is usually classified as a challenging task. It has to be tackled in future research to enhance the capabilities of humanoid running and make it more comparable to its natural counterpart.

6.10.2 Comparison to other works

When compared to SLIP control, the main feature of the presented BID controller is its analyticity, which allows for explicit solutions for 3D CoM trajectories and foot-step placement during running. Some features of the work of Raibert [40] such as apex height control and forward speed control via foot placement show major similarities to this work. Yet, BID control provides analytic solutions for planning and control as compared to Raibert's three-parted and rather heuristic running controller.

6.10.3 Potential usage, extensions and future work

One interesting aspect in human running is the center of pressure's (CoP) motion from heel to toe (as observed in jogging). This effect can be observed in Fig. 6.16 (intersection of the black force lines with the ground). This means that, while in humanoid locomotion the nominal CoP is kept as close to the foot center as possible (as in Fig. 6.4) to increase the likelihood of feasible desired leg forces, it can be advantageous to move the CoP from heel to toe during stance. A simple trick to produce such nominal CoP motion using the proposed BID control framework is to set the virtual foot positions below the actual ground. That way the intersection points of the force lines with the actual ground (corresponding to the CoP) shows a heel-to-toe motion.

In this work, locally flat stepping stones (see figure 6.1) are used. However, the controller is expected to handle more complex ground surfaces. Naturally robust foot trajectories for blind running are another interesting research topic. The BID algorithm may also be applied to problems such as hopping and jumping. Also quadrupedal gaits such as bounding/galloping and trotting are expected to be achievable.

6.10.4 Running related acknowledgements

We want to cordially thank Christian Rode, Roy Müller and Moritz Maus from the Locomotion Laboratory (Technical University of Darmstadt) and Imperial College London for kindly providing the human running data and the photo used in this work. We also thank Debora Clever from the Optimization in Robotics & Biomechanics group (University of Heidelberg) for providing human experiment data and for the fruitful discussions. Special thanks to our student Alexandra Mercader who significantly contributed to find the purely analytical solution.

CHAPTER 7

Discussion and conclusion

7.1 Discussion

The main goal of this thesis is to provide control methods that focus on the most critical component of a robot's dynamics, namely its CoM dynamics, and to then embed these reduced dynamics controllers into a whole-body control framework. Although the methods presented for DCM-based walking in chapter 5 and BID-based running in chapter 6 cover different forms of locomotion, on closer inspection these control methods show certain similarities. The first analogy is the preview of several (typically three or more) future footsteps and the derivation of feasible force profiles that nominally track them. Note that feasibility here is purely related to the question, if the line of action of the leg forces passes through the base of support or not, while frictional constraints and actuator limits are not considered at this stage, but only later in the quadratic programming (QP)-based whole-body controller from chapter 4. Both the walking and the running algorithm are purely analytical, such that the nominal trajectories, DCM trajectories for walking and CoM trajectories for running, are explicitly known over the whole multi-step horizon. Both methods provide a high degree of robustness and allow for interesting insights into the dynamics of the two locomotion forms. With regard to BID control, the feature of explicit footstep targeting, to the best of the author's knowledge, is a unique feature among any other existing (online) running controllers. The second analogy is related to the modulation and potential projection of the desired forces, such that they comply with the contact constraints. In case of DCM control, this modulation/projection can consist of leg force modulation and projection of a desired center of pressure (CoP) to the feasible foot supporting area, respectively. The proposed BID controller, in comparison, modulates the first upcoming stance foot position and all previewed leg force profiles, while projecting the foot position to a feasible one in case of a limited allowable contact area (see Fig. 7.1).

The proposed locomotion controllers are embedded into the quadratic programming (QP) based whole-body control (WBC) framework presented in chapter 4. This framework shows major similarities to other inverse-dynamics based works such as [56,58,60]. In contrast to most other works, where joint accelerations are chosen as control variables and torques are treated as outputs, in this work the joint torques are chosen as *control inputs*. Both formulations are basically equivalent, yet, the use of joint torques as control variables may be advantageous in certain cases, since they are the actual physical control quantity. For that choice of control input, joint torque constraints

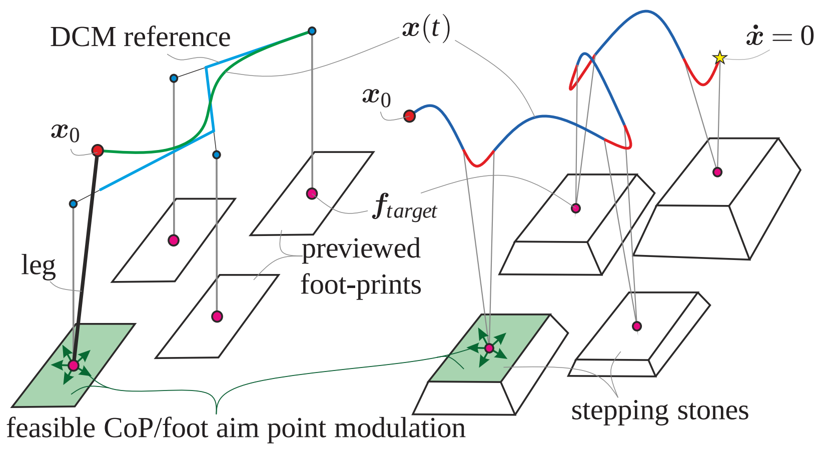

Figure 7.1: Analogy of DCM (walking) and BID (running); stepping stones vs. finite-sized feet.

of the robot can be treated as box constraints, while they have to be handled as linear constraints otherwise, which can be computationally more expensive. On the other side, the inversion of the mass matrix in (4.7) can be avoided by the joint acceleration-based formulation.

The combination of the presented walking and running controllers into the WBC framework yields smooth and natural looking behaviors of the robot. Especially the self-emergent (i.e. not manually programmed) arm motions resemble natural locomotion as observed in humans. One can conclude that the implemented motions and dynamics are close to their natural counterparts, such that insights into the control of the latter can be deduced. With regard to naturalness of achieved motions, it has to be noted that the presented work has room for improvement in the field of foot and leg motions. These are based on polynomial splines that are tracked quite accurately while the natural counterparts, as observed in humans and animals, show major dissimilarities. Also natural features such as toe-off motions and push-off are missed out in this work. These are expected to have a very positive effect on the centroidal momentum balance of the robot and should thus be investigated in future research.

The different active whole-body tasks are traded-off in the QP-based controller and are locally optimized, i.e. without any preview of the future of the whole-body motion. It has to be noted though, that the term “optimal” only means that the specified cost function is minimized. Yet, the task weights and also of the desired task dynamics (e.g. chosen eigenvalues for foot trajectory tracking) are typically chosen arbitrarily and may thus require extensive manual gain tuning. Offline optimization of these control gains, which may include an evaluation of the overall long-term behavior of the robot (such as the occurrence of singularities or falls during locomotion), might yield highly improved controller performance.

One of the major issues in whole-body control, especially for highly dynamic locomotion such as running or galloping, is the interplay of angular momentum and posture control. Pure posture control without considering angular momentum leads to a build-up of the latter, which quickly leads to constraint violations and a fall of the robot. Strict regulation of the angular momentum to zero also quickly leads to failure, since the leg swing motions either require torques around the center of mass or, if weighted strong enough, interfere with the posture control. Online generation of consistent angular momentum and postural trajectories, to the author’s knowledge, does not yet exist and would be an exciting field for future research efforts.

Another interesting research question is, if humanoid robots should be position or torque controlled. Recently ([17, 38, 103]), several position-based controllers have been shown to produce reliable locomotion performance. These controllers typically rely on high quality contact force sensing and control (via force-torque sensors in the feet and active force control), accurate state estimation and strong and fast actuation. On the other hand, torque control promises several assets as well, including the natural and compliant interaction with unknown or unforeseen elements of the environment, especially when such unforeseen contacts are made with robot links that have no sensing equipment such as torque sensors. In that case, position controlled robots behave rigidly and thus pose a threat to humans and other element of the environment. Torque controlled robots, in contrast, typically show a compliant behavior, which is most appropriate for direct human-robot interaction. It has to be noted though, that torque controllers have a higher potential of diverging in case of model errors or miss-estimated contact assumptions, as compared to position controlled approaches. These use explicitly designed motions that - apart from the free-floating base - can be controlled to not diverge. In the author's view, the most appropriate approach for humanoid control might be impedance control, which finds a good balance between desired positions and torques. With regard to safety, passivity-based or energy-bounded control approached may achieve desirable behaviors.

7.2 Conclusion

In the first chapters of this thesis, mathematical basics of robotics are recapitulated and useful tools for robotics and motion design, which the later higher-level controllers are based on, are presented.

The main contributions of this thesis are two control approaches, one for humanoid walking and one for running, that are based on the reduced dynamics model of CoM motion and are embedded into a quadratic programming (QP) based whole-body control framework (see chapter 4). The general idea pursued in both the presented walking and running methods is the following: Instead of explicitly previewing future constraint violations, constraint-compatible reference trajectories for multi-step previews are designed and tracked via feedback control. Both the presented walking and running control frameworks design reference trajectories that maximize the distance to the constraints and thus the margin of stability. That way in many cases the tracking controllers ask for control actions that do not violate the constraints, such that the nominal, stable controller behavior is actually achieved. Omitting the consideration of constraints in the preview allows for analytical and thus inexpensive computations. The embedding of the CoM-based controllers into the presented QP-based whole-body control framework then guarantees feasibility even in case of strong perturbations, while stability cannot be guaranteed.

The walking control framework is based on the concept of Divergent Component of Motion (DCM), which is extended to 3D in this work. Corresponding DCM trajectory generation and feedback control methods are presented (see Chap. 5), which facilitate the process of bipedal walking generation. Even walking trajectories over unstructured three-dimensional terrain can be generated and tracked in a clean and consistent way. The main idea, that DCM control is based on, is to split the center of mass dynamics into a stable and an unstable part (the DCM) and to control only that unstable part, while the naturally stable component remains untouched. That way, the second order control problem is reduced to a first order one, i.e. complexity is reduced while comprehensibility is increased. Due to the simplicity and analyticity of DCM control, it can be applied to plan and track three-dimensional walking trajectories in real-time. This work also introduced two new points, the enhanced Centroidal Moment Pivot point (eCMP) and the Virtual Repellent Point (VRP), respectively, with allow for the encoding of external (e.g. leg-) forces and total force acting on the robot's CoM. These points can be seen as generalizations of the Zero Moment Point (ZMP) and the Centroidal Moment Pivot point to 3D. A DCM trajectory generator is proposed (see chapter 5.3.1), that uses predefined smooth eCMP trajectories and a DCM terminal constraint as input and computes a corresponding consistent DCM reference trajectory, which is tracked by a nominally asymptotically stable DCM tracking controller (see chapter 5.3.2). Additionally, to increase the robustness of the walking gait against unknown external perturbations, methods for step adjustment and perturbation force estimation and counteraction are introduced.

In addition to the DCM-based walking controller, the second main contribution of this work is the introduction of Biologically Inspired Deadbeat (BID) control (see chapter 6). BID control uses polynomial splines to encode CoM motion and leg forces during stance, which was originally inspired by observations from human running experiments. Intuitive boundary conditions, chosen by the control designer, are explicitly solved for, which yields deadbeat behavior of the controller. In contrast to the spring-loaded inverted pendulum model (SLIP, [6]), the use of polynomials in BID control allows for purely analytical solutions for an arbitrary number of future stance and flight phases. Explicit footstep targeting in 3D can be achieved, which - to the author's knowledge - is a unique feature as compared to any other online running controller. BID control is real-time capable, facilitates the design of versatile running motions and is very robust due to the deadbeat behavior. The explicit knowledge about upcoming required foot locations facilitates the design of corresponding foot trajectories. Comparison to human running gaits shows promising similarities.

The embedding of the DCM-based walking controller and BID-based running controller into the whole-body control (WBC) framework presented in chapter 4 is another contribution of this work. The implemented WBC framework uses a single weighted quadratic program (QP) to solve an inverse dynamics problem, which contains the walking or running task besides other tasks, such as foot tracking, torso orientation control and overall body posture control. The combination of these tasks yields a robust and flexible framework that is able to reproduce several different agile locomotion modes such as walking and running. All control components can be computed in real-time. The whole-body controller assures feasibility of the finally commanded control outputs. The proposed walking and running controllers and the whole-body control framework are tested in numerous simulations and partially also in experiments with real robots.

Bibliography

[1] J. Englsberger, A. Werner, C. Ott, B. Henze, M. A. Roa, G. Garofalo, R. Burger, A. Beyer, O. Eiberger, K. Schmid, and A. Albu-Schäffer, "Overview of the torque-controlled humanoid robot toro," in *IEEE-RAS Int. Conf. on Humanoid Robots*, 2014, pp. 916–923. (Cited on pages 12, 17, 79, 101, and 102)

[2] F. Kanehiro, K. Fujiwara, S. Kajita, K. Yokoi, K. Kaneko, H. Hirukawa, Y. Nakamura, and K. Yamane, "Open architecture humanoid robotics platform," in *IEEE Int. Conf. on Robotics and Automation*, 2002, pp. 24–30 vol.1. (Cited on pages 12, 80, 101, and 102)

[3] "Darpa robotics challenge." [Online]. Available: http://www.theroboticschallenge.org/ (Cited on pages 15, 18, and 81)

[4] M. Johnson, B. Shrewsbury, S. Bertrand, T. Wu, D. Duran, M. Floyd, P. Abeles, D. Stephen, N. Mertins, A. Lesman, J. Carff, W. Rifenburgh, P. Kaveti, W. Straatman, J. Smith, M. Griffioen, B. Layton, T. de Boer, T. Koolen, P. Neuhaus, and J. Pratt, "Team ihmc's lessons learned from the darpa robotics challenge trials," *Journal of Field Robotics*, vol. 32, no. 2, pp. 192–208, 2015. (Cited on pages 15 and 81)

[5] S. Collins, A. Ruina, R. Tedrake, and M. Wisse, "Efficient bipedal robots based on passive dynamic walkers," *Science*, vol. 307, no. 5712, pp. 1082–1085, 2005. (Cited on page 16)

[6] H. Geyer, A. Seyfarth, and R. Blickhan, "Compliant leg behaviour explains basic dynamics of walking and running," *Proceedings of the Royal Society B: Biological Sciences*, pp. 2861–2867, 2006. (Cited on pages 16, 17, 85, and 110)

[7] G. Garofalo, C. Ott, and A. Albu-Schäffer, "Walking control of fully actuated robots based on the bipedal slip model," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, 2012, pp. 1456–1463. (Cited on page 16)

[8] Y. Liu, P. M. Wensing, J. P. Schmiedeler, and D. E. Orin, "Terrain-blind humanoid walking based on a 3-d actuated dual-slip model," *IEEE Robotics and Automation Letters*, vol. 1, no. 2, pp. 1073–1080, 2016. (Cited on page 16)

[9] M. Vukobratovic and Y. Stepanenko, "On the stability of anthropomorphic systems," *Mathematical Biosciences*, vol. 15, pp. 1–37, 1972. (Cited on pages 16, 58, and 59)

[10] 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 *IEEE Int. Conf. on Robotics and Automation*, 2003, pp. 1620–1626. (Cited on page 16)

[11] B. Stephens and C. Atkeson, “Push recovery by stepping for humanoid robots with force controlled joints,” in *IEEE-RAS Int. Conf. on Humanoid Robots*, 2010, pp. 52–59. (Cited on pages 16 and 59)

[12] P.-B. Wieber, “Trajectory free linear model predictive control for stable walking in the presence of strong perturbations,” in *IEEE-RAS Int. Conf. on Humanoid Robots*, 2006, pp. 137–142. (Cited on pages 16 and 59)

[13] A. Herdt, N. Perrin, and P.-B. Wieber, “Walking without thinking about it,” in *Int. Conf. on Intell. Robots and Systems*, 2010, pp. 190–195. (Cited on page 16)

[14] K. Nishiwaki and S. Kagami, “Online walking control system for humanoids with short cycle pattern generation,” *Int. J. of Robotics Research*, vol. 28, no. 6, pp. 729–742, 2009. (Cited on pages 16 and 59)

[15] T. Sugihara, “Standing stabilizability and stepping maneuver in planar bipedalism based on the best com-zmp regulator,” in *IEEE Int. Conf. on Robotics and Automation*, 2009, pp. 1966–1971. (Cited on pages 16 and 59)

[16] S. Kajita, M. Morisawa, K. Miura, S. Nakaoka, K. Harada, K. Kaneko, F. Kanehiro, and K. Yokoi, “Biped walking stabilization based on linear inverted pendulum tracking,” in *Int. Conf. on Intell. Robots and Systems*, 2010, pp. 4489–4496. (Cited on pages 16 and 59)

[17] J. Urata, K. Nshiwaki, Y. Nakanishi, K. Okada, S. Kagami, and M. Inaba, “Online walking pattern generation for push recovery and minimum delay to commanded change of direction and speed,” in *Int. Conf. on Intell. Robots and Systems*, 2012. (Cited on pages 16 and 108)

[18] R. Tedrake, S. Kuindersma, R. Deits, and K. Miura, “A closed-form solution for real-time zmp gait generation and feedback stabilization,” in *Humanoid Robots (Humanoids), 2015 IEEE-RAS 15th International Conference on*, 2015, pp. 936–940. (Cited on page 16)

[19] A. L. Hof, “The ’extrapolated center of mass’ concept suggests a simple control of balance in walking,” *Human Movement Science*, vol. 27, pp. 112–125, 2008. (Cited on pages 16, 18, 60, and 63)

[20] J. Pratt, J. Carff, S. Drakunov, and A. Goswami, “Capture point: A step toward humanoid push recovery,” in *IEEE-RAS Int. Conf. on Humanoid Robots*, 2006, pp. 200–207. (Cited on pages 16, 18, 59, 60, and 63)

[21] T. Koolen, T. D. Boer, J. Rebula, A. Goswami, and J. E. Pratt, “Capturability-based analysis and control of legged locomotion. part 1: Theory and application to three simple gait models,” *Int. J. of Robotics Research*, vol. 31, no. 9, pp. 1094–1113, 2012. (Cited on pages 16, 59, 60, 63, and 66)

[22] J. E. Pratt, T. Koolen, T. D. Boer, J. Rebula, S. Cotton, J. Carff, M. Johnson, and P. Neuhaus, “Capturability-based analysis and control of legged locomotion, part 2: Application to m2v2, a lower-body humanoid,” *IJRR*, vol. 31, no. 10, pp. 1117–1133, 2012. (Cited on pages 16, 59, 60, and 63)

[23] T. Takenaka, T. Matsumoto, and T. Yoshiike, “Real time motion generation and control for biped robot, 1st report: Walking gait pattern generation,” in *Int. Conf. on Intell. Robots and Systems*, 2009. (Cited on pages 16, 18, 60, and 63)

[24] J. Englsberger, C. Ott, M. A. Roa, A. Albu-Schäffer, and G. Hirzinger, “Bipedal walking control based on capture point dynamics,” in *Int. Conf. on Intell. Robots and Systems*, 2011, pp. 4420–4427. (Cited on pages 16, 60, 63, 66, and 80)

[25] J. Englsberger and C. Ott, “Integration of vertical COM motion and angular momentum in an extended capture point tracking controller for bipedal walking,” in *IEEE-RAS Int. Conf. on Humanoid Robots*, 2012. (Cited on pages 16, 60, 62, 63, 79, and 80)

[26] M. Morisawa, S. Kajita, F. Kanehiro, K. Kaneko, K. Miura, and K. Yokoi, “Balance control based on capture point error compensation for biped walking on uneven terrain,” in *IEEE-RAS Int. Conf. on Humanoid Robots*, 2012, pp. 734–740. (Cited on pages 16 and 60)

[27] K. Seo, J. Kim, and K. Roh, “Towards natural bipedal walking: Virtual gravity compensation and capture point control,” in *Int. Conf. on Intell. Robots and Systems*, 2012, pp. 4019–4026. (Cited on pages 16 and 60)

[28] J. Englsberger, C. Ott, and A. Albu-Schäffer, “Three-dimensional bipedal walking control using divergent component of motion,” in *Int. Conf. on Intell. Robots and Systems*, 2013, pp. 2600–2607. (Cited on page 16)

[29] Y. Zhao and L. Sentis, “A three dimensional foot placement planner for locomotion in very rough terrains,” in *IEEE-RAS Int. Conf. on Humanoid Robots*, 2012, pp. 726–733. (Cited on pages 16, 17, 62, and 77)

[30] J. Englsberger, C. Ott, and A. Albu-Schäffer, “Three-dimensional bipedal walking control based on divergent component of motion,” *Robotics, IEEE Transactions on*, vol. 31, no. 2, pp. 355–368, 2015. (Cited on pages 16, 17, 62, 63, 71, 80, and 81)

[31] A. D. Kuo, “Energetics of actively powered locomotion using the simplest walking model,” *Journal of Biomechanical Engineering*, vol. 124, pp. 113–120, 2002. (Cited on page 17)

[32] Y. Ogura, K. Shimomura, H. Kondo, A. Morishima, T. Okubo, S. Momoki, H. ok Lim, and A. Takanishi, “Human-like walking with knee stretched, heel-contact and toe-off motion by a humanoid robot,” in *Int. Conf. on Intell. Robots and Systems*, Oct 2006. (Cited on page 17)

[33] M. A. Hopkins, D. W. Hong, and A. Leonessa, “Humanoid locomotion on uneven terrain using the time-varying divergent component of motion,” in *2014 IEEE-RAS International Conference on Humanoid Robots*, 2014, pp. 266–272. (Cited on page 17)

[34] A. Werner, R. Lampariello, and C. Ott, “Optimization-based generation and experimental validation of optimal walking trajectories for biped robots,” in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 4373–4379. (Cited on page 17)

[35] ——, “Trajectory optimization for walking robots with series elastic actuators,” in *53rd IEEE Conference on Decision and Control*, 2014, pp. 2964–2970. (Cited on page 17)

[36] D. Clever and K. Mombaur, “A new template model for optimization studies of human walking on different terrains,” in *2014 IEEE-RAS International Conference on Humanoid Robots*, 2014, pp. 500–505. (Cited on page 17)

[37] K. H. Koch, D. Clever, K. Mombaur, and D. Endres, “Learning movement primitives from optimal and dynamically feasible trajectories for humanoid walking,” in *Humanoid Robots (Humanoids), 2015 IEEE-RAS 15th International Conference on*, 2015, pp. 866–873. (Cited on page 17)

[38] R. Wittmann, A. C. Hildebrandt, D. Wahrmann, D. Rixen, and T. Buschmann, “Real-time nonlinear model predictive footstep optimization for biped robots,” in *Humanoid Robots (Humanoids), 2015 IEEE-RAS 15th International Conference on*, 2015, pp. 711–717. (Cited on pages 17 and 108)

[39] S. Lohmeier, T. Buschmann, and H. Ulbrich, “Humanoid robot lola,” in *Robotics and Automation, 2009. ICRA ’09. IEEE International Conference on*, 2009, pp. 775–780. (Cited on page 17)

[40] M. Raibert, *Legged robots that balance*. The MIT Press,Cambridge, MA, Jan 1985. (Cited on pages 17, 85, and 104)

[41] K. Nagasaka, Y. Kuroki, S. Suzuki, Y. Itoh, and J. Yamaguchi, “Integrated motion control for walking, jumping and running on a small bipedal entertainment robot,” in *IEEE Int. Conf. on Robotics and Automation*, 2004, pp. 3189–3194. (Cited on page 17)

[42] T. Takenaka, T. Matsumoto, T. Yoshiike, T. Hasegawa, S. Shirokura, H. Kaneko, and A. Orita, “Real time motion generation and control for biped robot, 4th report: Integrated balance control,” in *Int. Conf. on Intell. Robots and Systems*, 2009, pp. 1601–1608. (Cited on page 17)

[43] S. Cotton, I. M. C. Olaru, M. Bellman, T. van der Ven, J. Godowski, and J. Pratt, “Fastrunner: A fast, efficient and robust bipedal robot. concept and planar simulation,” in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE, 2012, pp. 2358–2364. (Cited on page 17)

[44] D. Lakatos, C. Rode, A. Seyfarth, and A. Albu-Schäffer, “Design and control of compliantly actuated bipedal running robots: Concepts to exploit natural system dynamics,” in *IEEE-RAS Int. Conf. on Humanoid Robots*, 2014, pp. 930–937. (Cited on page 17)

[45] G. Schultz and K. Mombaur, “Modeling and optimal control of human-like running,” *IEEE/ASME Transactions on Mechatronics*, vol. 15, no. 5, pp. 783–792, 2010. (Cited on pages 17 and 18)

[46] B. Dadashzadeh, H. Vejdani, and J. Hurst, “From template to anchor: A novel control strategy for spring-mass running of bipedal robots,” in *Int. Conf. on Intell. Robots and Systems*, 2014, pp. 2566–2571. (Cited on page 17)

[47] S. G. Carver, N. J. Cowan, and J. M. Guckenheimer, “Lateral stability of the spring-mass hopper suggests a two-step control strategy for running,” *Chaos*, vol. 19, no. 2, 2009. (Cited on page 17)

[48] H. R. Vejdani, Y. Blum, M. A. Daley, and J. W. Hurst, “Bio-inspired swing leg control for spring-mass robots running on ground with unexpected height disturbance,” *Bioinspiration and Biomimetics*, vol. 8, no. 4, 2013. (Cited on page 17)

[49] A. Wu and H. Geyer, “The 3-d spring-mass model reveals a time-based deadbeat control for highly robust running and steering in uncertain environments,” *Robotics, IEEE Transactions on*, vol. 29, no. 5, pp. 1114–1124, Oct 2013. (Cited on page 17)

[50] D. Koepl and J. Hurst, “Impulse control for planar spring-mass running,” *Journal of Intelligent & Robotic Systems*, vol. 74, no. 3-4, pp. 589–603, 2014. (Cited on page 17)

[51] P. M. Wensing and D. E. Orin, “High-speed humanoid running through control with a 3d-slip model,” in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*. IEEE, 2013, pp. 5134–5140. (Cited on pages 17 and 54)

[52] P. Wensing and D. Orin, “3d-slip steering for high-speed humanoid turns,” in *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, 2014, pp. 4008–4013. (Cited on pages 17 and 54)

[53] H.-W. Park, S. Park, and S. Kim, “Variable-speed quadrupedal bounding using impulse planning: Untethered high-speed 3d running of mit cheetah 2,” in *IEEE Int. Conf. on Robotics and Automation*, 2015, pp. 5163–5170. (Cited on page 18)

[54] H. Dai, A. Valenzuela, and R. Tedrake, “Whole-body motion planning with centroidal dynamics and full kinematics,” in *2014 IEEE-RAS International Conference on Humanoid Robots*, 2014, pp. 295–302. (Cited on page 18)

[55] L. Sentis and O. Khatib, “A whole-body control framework for humanoids operating in human environments,” in *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, 2006, pp. 2641–2648. (Cited on page 18)

[56] ——, “Synthesis of whole-body behaviors through hierarchical control of behavioral primitives,” *International Journal of Humanoid Robotics*, vol. 2, no. 4, pp. 505–518, 2005. (Cited on pages 18, 45, and 107)

[57] O. E. Ramos, N. Mansard, and P. Souères, “Whole-body motion integrating the capture point in the operational space inverse dynamics control,” in *2014 IEEE-RAS International Conference on Humanoid Robots*, 2014, pp. 707–712. (Cited on page 18)

[58] M. A. Hopkins, V. L. Orekhov, C. S. Knabe, B. Y. Lattimer, and D. W. Hong, “Humanoid balancing on unstable terrain using whole-body momentum control and series elastic actuation,” in *2014 IEEE-RAS International Conference on Humanoid Robots*, 2014, pp. 1097–1097. (Cited on pages 18 and 107)

[59] S. Feng, X. Xinjilefu, C. G. Atkeson, and J. Kim, “Optimization based controller design and implementation for the atlas robot in the darpa robotics challenge finals,” in *Humanoid Robots (Humanoids), 2015 IEEE-RAS 15th International Conference on*, 2015, pp. 1028–1035. (Cited on page 18)

[60] P. Wensing and D. Orin, “Generation of dynamic humanoid behaviors through task-space control with conic optimization,” in *IEEE Int. Conf. on Robotics and Automation*, 2013, pp. 3103–3109. (Cited on pages 18, 46, and 107)

[61] S. h. Hyon and G. Cheng, “Passivity-based full-body force control for humanoids and application to dynamic balancing and locomotion,” in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct 2006, pp. 4915–4922. (Cited on pages 18 and 45)

[62] C. Ott, M. A. Roa, and G. Hirzinger, “Posture and balance control for biped robots based on contact force optimization,” in *IEEE-RAS Int. Conf. on Humanoid Robots*, 2011, pp. 26–33. (Cited on pages 18 and 45)

[63] A. Dietrich, T. Wimböck, A. Albu-Schäffer, and G. Hirzinger, “Reactive whole-body control: Dynamic mobile manipulation using a large number of actuated degrees of freedom,” *IEEE Robotics & Automation Magazine*, vol. 19, no. 2, pp. 20–33, 2012. (Cited on pages 18 and 45)

[64] A. Dietrich, T. Wimbock, A. Albu-Schaffer, and G. Hirzinger, “Integration of reactive, torque-based self-collision avoidance into a task hierarchy,” *IEEE Transactions on Robotics*, vol. 28, no. 6, pp. 1278–1293, 2012. (Cited on pages 18 and 45)

[65] B. Henze, C. Ott, and M. A. Roa, “Posture and balance control for humanoid robots in multi-contact scenarios based on model predictive control,” in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sept 2014, pp. 3253–3258. (Cited on pages 18 and 45)

[66] M. B. Popovic, A. Goswami, and H. Herr, “Ground reference points in legged locomotion: Definitions, biological trajectories and control implications,” *Int. J. of Robotics Research*, vol. 24, no. 12, 2005. (Cited on pages 19 and 63)

[67] J. Englsberger, P. Kozlowski, and C. Ott, “Biologically inspired dead-beat controller for bipedal running in 3d,” in *IEEE/RSJ Int. Conf. on Intell. Robots and Systems*, 2015, pp. 989–996. (Cited on pages 19, 86, 93, and 96)

[68] ——, “Biologically inspired deadbeat control for running on 3d stepping stones,” in *IEEE-RAS Int. Conf. on Humanoid Robots*, 2015, pp. 1067–1074. (Cited on page 19)

[69] J. Englsberger, P. Kozlowski, C. Ott, and A. Albu-Schäffer, “Biologically inspired deadbeat control for running: from human analysis to humanoid control and back,” *Robotics, IEEE Transactions on (accepted)*, 2016. (Cited on page 19)

[70] G. Garofalo, C. Ott, and A. Albu-Schäffer, “On the closed form computation of the dynamic matrices and their differentiations,” in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013, pp. 2364–2359. (Cited on pages 25, 31, and 81)

[71] R. Murray, Z. Li, and S. Sastry, *A Mathematical Introduction to Robotic Manipulation*. CRC Press, 1994. (Cited on pages 25, 26, and 51)

[72] O. Rodrigues, “Des lois géometriques qui regissent les déplacements d’ un systéme solide dans l’ espace, et de la variation des coordonnées provenant de ces déplacement considérées indépendent des causes qui peuvent les produire,” *journal de mathématiques pures et appliquées*, vol. 1, no. 5, pp. 380–440, 1840. (Cited on page 29)

[73] S. R. Ploen, “Geometric algorithms for the dynamics and control of multibody systems,” Ph.D. dissertation, University of California Irvine, 1997. (Cited on page 30)

[74] J. Ackermann, *Robust Control: The Parameter Space Approach*, 2nd ed. Springer-Verlag, 2002. (Cited on page 34)

[75] M. D. Shuster, “A survey of attitude representations,” *The Journal of the Astronautical Science*, vol. 41, no. 4, pp. 439 – 517, 1993. (Cited on page 36)

[76] J. Park, “Analysis and control of kinematically redundant manipulators: An approach based on kinematically decoupled joint space decomposition,” Ph.D. dissertation, Pohang University of Science and Technology (POSTECH), 1999. (Cited on page 43)

[77] Ch. Ott, “Cartesian impedance control of flexible joint manipulators,” Ph.D. dissertation, Saarland University, November 2005. (Cited on page 43)

[78] J. Pratt, C.-M. Chew, A. Torres, P. Dilworth, and G. Pratt, “Virtual model control: An intuitive approach for bipedal locomotion,” *The International Journal of Robotics Research*, vol. 20, no. 2, pp. 129–143, 2001. (Cited on page 45)

[79] M. Hutter, C. D. Remy, M. A. Hoepflinger, and R. Siegwart, “Scarleth: Design and control of a planar running robot,” in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2011, pp. 562–567. (Cited on page 45)

[80] L. Righetti and S. Schaal, “Quadratic programming for inverse dynamics with optimal distribution of contact forces,” in *2012 12th IEEE-RAS International Conference on Humanoid Robots (Humanoids 2012)*, Nov 2012, pp. 538–543. (Cited on pages 45, 46, and 54)

[81] T. Koolen, J. Smith, G. Thomas, S. Bertrand, J. Carff, N. Mertins, D. Stephen, P. Abeles, J. Englsberger, S. McCrory *et al.*, “Summary of team ihmc’s virtual robotics challenge entry,” in *Int. Conf. on Humanoid Robots*, 2013. (Cited on pages 45, 46, 54, 80, and 81)

[82] L. Righetti, J. Buchli, M. Mistry, M. Kalakrishnan, and S. Stefan, “Optimal distribution of contact forces with inverse dynamics control,” *Int. J. of Robotics Research*, vol. 32, no. 3, pp. 280–298, 2013. (Cited on pages 45, 46, and 54)

[83] S. Kuindersma, F. Permenter, and R. Tedrake, “An efficiently solvable quadratic program for stabilizing dynamic locomotion,” in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, May 2014, pp. 2589–2594. (Cited on pages 45 and 46)

[84] 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 *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, 2015, pp. 3346–3351. (Cited on page 45)

[85] D. E. Orin, A. Goswami, and S.-H. Lee, “Centroidal dynamics of a humanoid robot,” *Auton. Robots*, vol. 35, no. 2-3, pp. 161–176, 2013. (Cited on pages 48 and 49)

[86] A. Herzog, N. Rotella, S. Schaal, and L. Righetti, “Trajectory generation for multi-contact momentum control,” in *Humanoid Robots (Humanoids), 2015 IEEE-RAS 15th International Conference on*, 2015, pp. 874–880. (Cited on page 49)

[87] B. Henze, A. Dietrich, and C. Ott, “An approach to combine balancing with hierarchical whole-body control for legged humanoid robots,” *IEEE Robotics and Automation Letters*, vol. 1, no. 2, pp. 700–707, 2016. (Cited on page 54)

[88] A. Escande, N. Mansard, and P.-B. Wieber, “Hierarchical quadratic programming: Fast online humanoid-robot motion generation,” *The International Journal of Robotics Research*, 2014. (Cited on page 54)

[89] H. Ferreau, H. Bock, and M. Diehl, “An online active set strategy to overcome the limitations of explicit mpc,” *International Journal of Robust and Nonlinear Control*, vol. 18, no. 8, pp. 816–830, 2008. (Cited on page 55)

[90] H. Ferreau, C. Kirches, A. Potschka, H. Bock, and M. Diehl, “qpOASES: A parametric active-set algorithm for quadratic programming,” *Mathematical Programming Computation*, vol. 6, no. 4, pp. 327–363, 2014. (Cited on page 55)

[91] P.-B. Wieber, *Fast Motions in Biomechanics and Robotics*, ser. Lecture Notes in Control and Information Sciences. Springer Berlin / Heidelberg, 2006, vol. 340, ch. Holonomy and Nonholonomy in the Dynamics of Articulated Motion, pp. 411–425. (Cited on page 58)

[92] T. Sugihara, Y. Nakamura, and H. Inoue, “Realtime humanoid motion generation through zmp manipulation based on inverted pendulum control,” in *IEEE Int. Conf. on Robotics and Automation*, 2002. (Cited on page 58)

[93] J. Slotine and W. Li, “On the adaptive control of robot manipulators,” *Int. J. of Robotics Research*, vol. 6, no. 3, pp. 49–59, 1987. (Cited on page 63)

[94] A. De Luca, A. Albu-Schäffer, S. Haddadin, and G. Hirzinger, “Collision detection and safe reaction with the dlr-iii lightweight manipulator arm,” in *Int. Conf. on Intell. Robots and Systems*, 2006, pp. 1623–1630. (Cited on page 74)

[95] B. Henze, A. Werner, M. A. Roa, G. Garofalo, J. Englsberger, and C. Ott, “Control applications of TORO - a torque controlled humanoid robot (video),” in *IEEE-RAS Int. Conf. on Humanoid Robots*, 2014. (Cited on page 79)

[96] “Atlas - the agile anthropomorphic robot.” [Online]. Available: http://www.bostondynamics.com/robot_Atlas.html (Cited on pages 79 and 82)

[97] “Simulation construction set - ihmc’s simulation environment for humanoid robots.” [Online]. Available: http://ihmc.us/groups/scs/ (Cited on pages 80 and 82)

[98] “Gazebo - a multi-robot simulator for outdoor environments.” [Online]. Available: http://gazebosim.org/about.html (Cited on page 80)

[99] N. Perrin, C. Ott, J. Englsberger, O. Stasse, F. Lamiraux, and D. G. Caldwell, “A continuous approach to legged locomotion planning.” [Online]. Available: http://hal.inria.fr/docs/00/80/59/81/PDF/\continuous_locomotion_perrin.pdf (Cited on page 80)

[100] H. Kaminaga, J. Englsberger, and C. Ott, “Kinematic optimization and online adaptation of swing foot trajectory for biped locomotion,” in *IEEE-RAS Int. Conf. on Humanoid Robots*, 2012, pp. 593–599. (Cited on page 80)

[101] H.-M. Maus, S. Revzen, J. Guckenheimer, C. Ludwig, J. Reger, and A. Seyfarth, “Constructing predictive models of human running,” *Journal of The Royal Society Interface*, vol. 12, no. 103, 2014. (Cited on page 103)

[102] ——, “Data from: Constructing predictive models of human running. dryad digital repository.” 2014. [Online]. Available: http://dx.doi.org/10.5061/dryad.r9v30 (Cited on page 103)

[103] “Boston dynamics incorporated.” [Online]. Available: http://www.bostondynamics.com (Cited on page 108)