Fakultät für Maschinenwesen
Lehrstuhl für Angewandte Mechanik

Robust Walking Robots in Unknown Environments

– Dynamic Models, State Estimation and Real-Time Trajectory Optimization

Robert Wittmann

Vollständiger Abdruck der von der Fakultät für Maschinenwesen der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktor-Ingenieurs (Dr.-Ing.)

genehmigten Dissertation.

Vorsitzender: Prof. Dr.-Ing. Florian Holzapfel

Prüfer der Dissertation:

1. Prof. Dr.-Ing. Dr.-Ing. habil. Heinz Ulbrich, i.R.
2. Prof. Dr.-Ing. habil. Boris Lohmann

Die Dissertation wurde am 05.04.2017 bei der Technischen Universität München eingereicht und durch die Fakultät für Maschinenwesen am 11.08.2017 angenommen.

Abstract

This thesis presents new methods that aim to increase the robustness of bipedal walking robots in unknown environments. The main part is a strategy to adapt future motion according to the current state of the robot. A model for humanoid robots is proposed that allows an accurate and fast prediction. The state obtained from a state observer and the prediction model are used to calculate a reaction. All methods are applied to the real robot LOLA and are evaluated in experiments.

KEYWORDS: Bipedal Robots, modeling, state estimation, trajectory optimization, model predictive control.

Zusammenfassung

Diese Arbeit stellt neue Methoden zur Erhöhung der Robustheit von zweibeinigen Laufrobotern in unbekanntem Gelände vor. Kern der Arbeit ist dabei eine Strategie, welche auf Basis des aktuellen Roboterzustands die zukünftige Bewegung anpasst. Ein Prädiktionsmodell für humanoide Roboter wird eingeführt, welches eine genaue und schnelle Vorhersage ermöglicht. Der über einen Beobachter gewonnene Zustand des Roboters und das Prädiktionsmodell werden anschließend verwendet, um in Echtzeit eine Reaktion zu berechnen. Alle Methoden wurden in Experimenten am Laufroboter LOLA getestet und evaluiert.

STICHWORTE: Zweibeinige Roboter, Modellierung, Zustandsschätzung, Trajektorienoptimierung, Modellprädiktive Regelung.

Acknowledgements

This thesis is the result of my research activities at the Chair of Applied Mechanics, Technical University Munich. It received important results from the four-year research project "Flexibles und Robustes Gehen in unbekannten Umgebungen. During the project I got supported by many people, without whom this work would not have been possible.

First of all I want to thank my supervisor professor Ulbrich for his guidance and interest in my reserach – even after your retirement. You gave me the opportunity to work in a very fruitful environment on an interesting project. Thus, I had the chance to develop my own ideas. I am also very grateful to his successor professor Rixen. Not only did you give me the freedom to proceed with my research directly at the Chair but also you were actively interested in the progress and the work of the robotics group. Thank you for your valuable advice and support! I would also like to acknowledge Professor Boris Lohmann for serving on my thesis defense committee.

I am deeply grateful for having had the chance to work with a number of very talented and highly motivated people. I am especially thankful to the research group working on the robot Lola. A person without whom my work would not have been possible is Thomas Buschmann. He drew my interest and fascination for humanoid robots during my Diploma thesis under his guidance. I learned so much about motion planning and control of bipedal robots from his long experience with bipedal robots. Thank you for the inspiring discussions and your advice. I warmly thank Arne-Christoph Hildebrandt who was working on footstep planning and obstacle avoidance throughout the project and Daniel Wahrmann who developed the vision system for LOLA. You both were a very valuable and motivating project partners. Thank you for all the discussions, your motivated work and all the time we spent in the lab. I am also very grateful to Felix Sygulla for helping with the development of the new ETHERCAT based communication system. All final experiments would not have been possible without this upgrade. I would like to thank the other robotics team members Felix Ellensohn, Philipp Seiwald and Christoph Schütz for all the inspiring discussions on robotics research topics.

I also would like to express my gratitude to Sebastian Lohmeier, Thomas Buschmann, Markus Schwienbacher and Valerio Favot for providing such a great robot to do my research. Experimental robotics research is impossible without decent hardware and I am especially grateful for Simon Gerer, Georg König and Georg Mayr's work in repairing and manufacturing LOLA. I owe special thanks to Georg Mayr. His long experience with legged robots, his help in maintaining and helping with the ETHERCAT-upgrade for the robot LOLA were invaluable.

I would also like to thank my (ex-) colleagues Thomas Buschmann, Arne-Christoph Hildebrandt, Felix Sygulla and Philipp Seiwald for proofreading this thesis and giving helpful comments.

Table of Contents

List of Abbreviations

BVP Boundary Value Problem

CAN Controller Area Network
CoG Center of Gravity
CoP Center of Pressure

DDP Differential Dynamic Programming
DoF Degree of Freedom
DRC Darpa Robotics Challenge
DS Double Support

EoM Equation of Motion

FIFO First-In First-Out
FoR Frame of Reference
FRI Foot Rotation Indicator
FTS Force-Torque Sensors

IMU Inertial Measurement Unit

LIPM Linear Inverted Pendulum Model

MBS Multibody System
MPC Model Predictive Control

ODE Ordinary Differential Equation

RMSE Root Mean Squared Error

SLIPM Spring Loaded Inverted Pendulum Model
SS Single Support
SSV Swept-Sphere-Volume
SVD Singular Value Decomposition

wrt. with respect to

ZMP Zero Moment Point

Chapter 1

Introduction

In the past years robotic systems advanced in terms of their autonomy and versatility. Mobile robots acting in unknown environments are a good example for the progress. One important property of these robots is that they have the ability to move their base which is on the one hand a huge benefit and increases the range of applications. On the other hand several safety issues arise since the robots have to interact and adapt to changing environmental conditions. Several perception and motion planning problems have to be solved to enable a safe operation. The robot has to gather information of its environment in addition to its own state. Depending on this information it has to decide whether it has to adapt or stop its planned behavior. Additionally, the robot has to detect unforeseen errors, decide what to do and, if necessary, plan a reaction. The overall time for detection, decision and reaction is a crucial point since the robot has to react instantaneously.

Bipedal robots belong to the class of mobile robots. They have human-like capabilities and can perform in environments designed for humans. In environments with stairs, doors or unmovable obstacles they are potentially superior to wheeled robotic systems since legged systems require only discrete foothold positions. Nevertheless bipedal robots introduce additional challenges to the ones of mobile robots. Recent research in the field of humanoid robots includes perception, motion planning, feedback control or manipulation.

Application fields of humanoid robots are e.g. prosthesis development, service robots and disaster operation. The latter was initiated by the Fukushima Accident in 2011. There was a lack of available tele-operated robots that can perform repair work inside the contaminated nuclear power plant. This inspired the Darpa Robotics Challenge (DRC) from 2012 to 2015[1]. During the challenge several teams from all over the world had to solve tasks with humanoid robots only by using tele-operation. One main drawback of the impressive solutions was their lack of autonomy. Almost all decisions were controlled by the human operators.

While the DRC focuses on solving a wide range of different tasks (open valve, use tools, open doors, climb stairs), the work presented in this thesis aims to increase the autonomy of bipedal walking. It was conducted during the project "Flexible and robust biped walking in uneven terrain" funded by the DFG (Deutsche Forschungsgemeinschaft).

1.1 Problem Statement

One important requirement to bring bipedal robots to real world applications is a reliable hardware and software system that solves its locomotion tasks autonomously. This thesis covers software methods to increase the robustness of bipedal robots that act in unknown

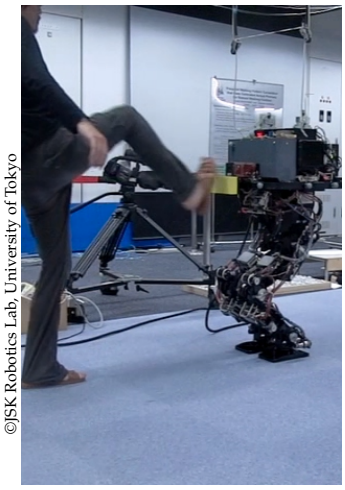
[1] http://www.darpa.mil/program/darpa-robotics-challenge

Figure 1.1: Bipedal robot LOLA of the chair of applied mechanics.

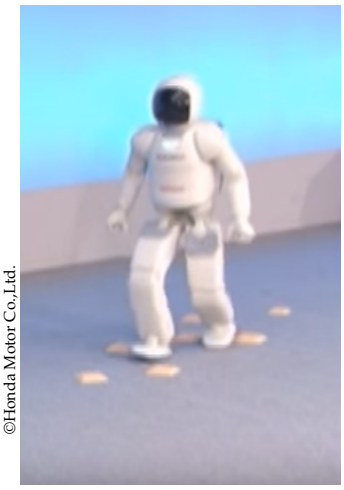
environments. Robustness means the robot's ability to recover from severe disturbances resulting from external forces or errors and uncertainties in the environment model. This can be assigned to the research field of bipedal walking stabilization. The robot's mechanical properties have to be accounted for developing new stabilization methods. A bipedal robot has many joints compared to conventional industrial robots. To generate a stabilizing motion all joints have to be coordinated in the right way and the robot's nonlinear kinematics and dynamics have to be considered. Furthermore the biped is not fixed to the environment. This is a necessary requirement since bipedalism consists of continuously closing and breaking contacts between the feet and the ground in order to move the overall system. However this introduces limitations on the reaction forces the robot can apply to the environment. In particular, the robot can not pull the ground. When increasing the robustness of bipedal robots several general questions arise:

- What is the robot's current state?
  Since the robot is not fixed to the environment the absolute position and orientation with respect to (wrt.) the world is mainly of interest.
- Will the robot fall?
  Given the current state and the desired motion the future behavior of the robot has to be predicted with an appropriate model.
- How to adapt future motion?
  When the robot is predicted to fall the final question is how to adapt its overall motion in order to pretend this.

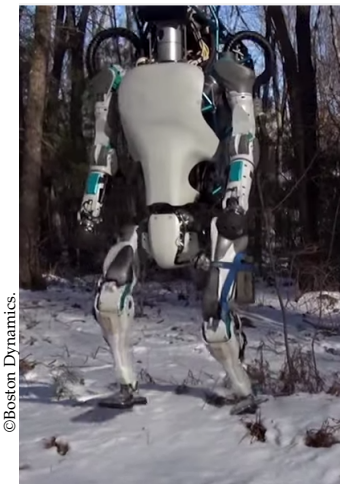
Another crucial point is real-time capability of the overall control system. This is especially important for the prediction model and the adaptation of future motion. The main research platform for this work is the robot LOLA (Figure 1.1), developed at the Chair of Applied Mechanics, Technical University of Munich. The developed model-based methods are applied and tested with this robot but can be used for other fully actuated humanoids.



a) HRP3L-JSK



b) Asimo



c) Atlas next Generation

d) HRP2

e) Pet-Proto

Figure 1.2: Humanoid robots in uneven terrain or with unknown disturbances.

1.2 Related Work

The following part is intended to give a brief overview of research groups whose work is considered to be most relevant for this thesis. In particular only work with full sized humanoid robots and methods that are applied to real hardware are considered in this summary. Special effort is put towards the robot's ability to overcome an unknown disturbance, with focus on the robustness of the method. Some examples are shown in Figure 1.2.

Tokyo University

The JSK-laboratory of the Tokyo University has a long tradition in developing humanoid robots and walking control methods. In 2011 Urata presented a powerful trajectory optimization to react to huge external pushes. Their robot is a modified version of the HRP3L robot with high power leg joints. They can achieve very fast and accurate leg motions which increased the overall system performance (Urata et al. 2010). During the DRC a revised robot called Schaft was developed. It performed best during the trials.

The JSK-laboratory entered the DRC-finals with a different team and a modified version of the HRP2-robot (Kaneko et al. 2004).

Honda

Honda realized many electrically powered robots since the 1990s that show very impressive results. Beside manipulation, recognition and high level control they introduced fundamental control concepts for walking control and stabilization. Those methods inspired much other research work. With their current robot Asimo they achieved high walking and running speeds as well as robust walking. Their basic model-based approach uses Inertial Measurement Unit (IMU) and Force-Torque Sensors (FTS) data to stabilize the overall robot (Hirose and Ogawa 2007). The basic ideas are published in few papers and patent applications.

Boston Dynamics

With impressive videos Boston Dynamics gained a lot of attention. Using the hydraulically actuated bipedal robots Petman and Pet-Proto, they showed the bipeds recovering from severe pushes, performing walking with additional support of the arms in very difficult situations. Details to the hardware and control design are except one vague publication not available (Nelson et al. 2012). Recently, they provided the Atlas robot which served for several teams of the DRC as hardware platform for their algorithms. There are videos of a revised version of the robot (Atlas next Generation) walking fully autonomously through snow and forest. This can be seen as one of the most impressive walking performances over the world. Unfortunately there are no publications concerning the walking control.

IHMC and CMU

Florida Institute of Human & Machine Cognition (IHMC) started with robots using serial elastic actuators. Carnegie Mellon University (CMU) built with Raibert several hopping robots during the 90s (Raibert 1986). As each of them received an Atlas robot for the DRC those laboratories made a large progress in the field of fully actuated human sized humanoids. Both groups show promising results with the robot walking over unmodelled uneven terrain, receiving external pushes and stabilizing using the arms (Feng et al. 2014; Kuindersma et al. 2015). IHMC also showed static walking experiments where the robot has to walk over partial footholds such as line contacts.

KAIST

The Humanoid Robot Research Center (HUBO Labs) was founded in 1985 at the Korean Institute of Technology (KAIST). They developed several generations of the HUBO robot (Park et al. 2005). Most recently a modified version to participate in the DRC was shown. The team from KAIST showed convincing results during the finals and won the challenge. One reason is the robust well tested control framework which results from many years of experience.

1.3 Contributions of this Thesis

The main objective of this thesis is the development of methods to increase the robustness of bipedal robots. These methods stabilize the robot (prevent falling) when unforeseen events like external disturbances occur. They use measurement data to calculate a reaction that adapts the robot's current and future motion. All algorithms are developed with special effort to their applicability on the real robot. This requires that they can be performed in real-time[2].

A secondary objective is a robust communication system for the overall mechatronics and low level control. This is especially related to synchronizing several processes, detecting errors of the overall control system and exchanging data with the sensor-actor network of the robot. The low level control for the joints is decentralized and has a huge influence to the system's overall walking performance. Especially for large disturbances there are fast reaction motions that have to be realized by the joints. The main contributions of this thesis are:

- The development of a sensor-based trajectory adaptation framework with non-standard approach that can be integrated into the overall walking control system of LOLA. Interaction with collision avoidance methods is also considered.
- Development of a new class of fast and accurate dynamic prediction models that include fundamental properties of the biped. Those models allow a online trajectory planning for humanoids which does not use the Zero Moment Point (ZMP) (cf. Subsection 2.2.1).
- State estimation for bipeds using nonlinear models. This is mainly used to filter the IMU data.
- Real-time trajectory optimization methods, including:
  - Parameter optimization by a direct shooting method
  - Trajectory optimization by a conjugate gradient method
  - Combination of both methods by an indirect formulation
  - Consideration of inequality constraints that result from kinematic limits and online detected obstacles
- A robust ETHERCAT-based real-time system and low level joint control. Both increase the robustness and reliability of the overall mechatronic system.
- Experimental verification of the stabilization methods on the bipedal robot LOLA.

The thesis' structure is as follows: Chapter 2 describes the background on the dynamics, feasibility and stability of bipedal robots. An overview of the experimental platform LOLA and of the control system extension is part of Chapter 3. It also presents an improved joint control. A new class of dynamic prediction models is presented in Chapter 4. The models can be applied for state estimation (Chapter 5) and trajectory adaptation (Chapter 6). Different methods to add sensor feedback to the robot's trajectory plannning are presented. Chapter 7 shows experimental results for the overall walking stabilization in different experiments. Finally, Chapter 8 concludes the thesis with a summary and a discussion of the results.

[2] Real-time means that there is a hard response deadline for the algorithms which is in this case within few milliseconds.

Chapter 2

Feasibility and Stability of Bipedal Robots

Dealing with stability of biped robots requires to understand the underlying dynamics and properties of such systems. This chapter gives an overview of the dynamics of humanoids and discusses resulting challenges. One is the limitation of possible interaction forces with the environment, which arises from the fact that the robot is not rigidly fixed to the environment. It introduces limits for feasible motions of the robot. A brief overview is given, how it can be dealt with in motion planning. The second challenge is the underactuation of bipedal robots being mainly a concern of stability. Existing stability criteria and stabilization approaches are discussed in the later part.

2.1 Dynamics of Bipedal Locomotion

Bipedal robots present a class of mechanical systems with different challenging properties for planning and control. Beside their nonlinear multibody dynamics with many Degrees of Freedom (DoFs) these robots are in contrast to industrial manipulators not fixed to the environment. This is necessary for locomotion as it requires to "regularly breaking and recovering contacts in order to obtain a displacement of the whole system" (Wieber 2002). This structure-varying property means that the robot has to be able to open and close its contacts. In mechanics those contacts are classified as unilateral, which introduces inequality constraints for possible contact forces. If the contact state changes unintentionally e.g. when a strong disturbance occurs, or the contacts are considered to be compliant the system is also underactuated as it has less actuators than DoFs. It can be described mathematically by stating the overall Equation of Motion (EoM) of the robot as presented in Fujimoto et al. (1998). The overall DoFs $\boldsymbol{q} \in \mathbb{R}^n$ are split into the free floating base $\boldsymbol{q}_T \in \mathbb{R}^6$ part and the joints $\boldsymbol{q}_J \in \mathbb{R}^{n-6}$ of the robot

$$\begin{bmatrix} \boldsymbol{M}_{TT} & \boldsymbol{M}_{TJ} \\ \boldsymbol{M}_{JT} & \boldsymbol{M}_{JJ} \end{bmatrix} \begin{bmatrix} \ddot{\boldsymbol{q}}_T \\ \ddot{\boldsymbol{q}}_J \end{bmatrix} + \begin{bmatrix} \boldsymbol{h}_T \\ \boldsymbol{h}_J \end{bmatrix} = \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{\tau} \end{bmatrix} + \begin{bmatrix} \boldsymbol{J}_{\lambda,T}^T \\ \boldsymbol{J}_{\lambda,J}^T \end{bmatrix} \boldsymbol{\Lambda}. \tag{2.1}$$

The mass matrix entries are denoted by $\boldsymbol{M}_{ij}$, $\boldsymbol{h}_i$ describe the nonlinear vectors for Coriolis, centrifugal and gravitational forces and $\boldsymbol{\tau} \in \mathbb{R}^{n-6}$ the joint torques from the actuators. The contact forces $\boldsymbol{\Lambda} \in \mathbb{R}^{12}$ are projected with the matrices $\boldsymbol{J}_{\lambda,T}$, $\boldsymbol{J}_{\lambda,J}$ to the directions of $\boldsymbol{q}_T$ and $\boldsymbol{q}_J$ respectively. There are several works dealing with control of such underactuated mechanical systems (Huang et al. 2015; Romano et al. 2014) but without the switching of the contact states.

The way how to deal with the contact forces can be basically split into two different categories namely assuming them as rigid or compliant. Rigid contacts are often used for *whole-body control* approaches (Kuindersma et al. 2015; Ott et al. 2011; Ramos et al. 2014, e.g.) but also for analytic approaches (Chevallereau et al. 2008; Grizzle et al. 2003;

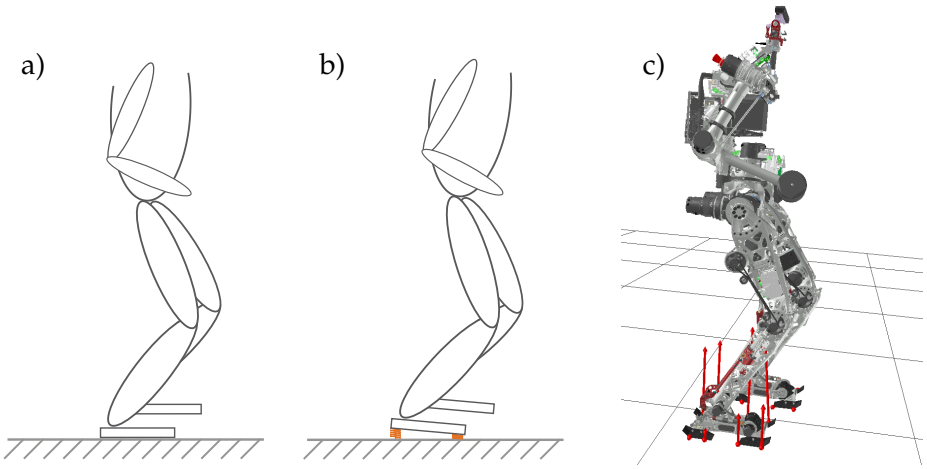


Figure 2.1: Contact models for bipeds. a) rigid body contacts b) compliant contacts and c) compliant contact model of LOLA (red arrows visualize forces).

Huang et al. 1999). These works mainly try to avoid tipping over one edge by constraining the contact forces and moments, which can be formulated with inequality constraints or non-smooth equality constraints (Buschmann 2010; Leine and Nijmeijer 2004). As long as the contact forces and moments are inside the set of feasible values, the foot remains flat on the ground and the system can be treated as fully actuated. Nevertheless there is always a finite compliance between foot and ground and for some robots compliant material is intentionally used at the sole Lohmeier (2009) and Yamaguchi and Takanishi (1996). One postulated advantage is to obtain a shock absorbing property which is crucial for situations when the swing-leg touches the ground too early due to disturbances such as external pushes, rough terrain or simply modeling errors. This is at the expense of obtaining a system being additionally *all the time* underactuated which has to be considered especially in walking stabilization. However compliant contacts allow to apply robust admittance control for the reaction forces (Buschmann et al. 2009; Hashimoto et al. 2012) at moderate update rates. The two different contact types and the compliant contact model of the bipedal robot LOLA are shown in Figure 2.1. Assuming a discretization of each foot with n_c contact elements that produce a force $\boldsymbol{f}_j$ with a displacement $\triangle \boldsymbol{r}_{f_j}$ to the reference point of foot i, the contact wrench can be summed up with

$$\boldsymbol{\lambda}_i = \begin{bmatrix} \boldsymbol{F}_i \\ \boldsymbol{T}_i \end{bmatrix} \qquad \text{where } \boldsymbol{F}_i = \sum_{j \in C_i} \boldsymbol{f}_j(\boldsymbol{q}, \dot{\boldsymbol{q}}), \quad \boldsymbol{T}_i = \sum_{j \in C_i} \triangle \boldsymbol{r}_{f_j} \times \boldsymbol{f}_j(\boldsymbol{q}, \dot{\boldsymbol{q}}). \tag{2.2}$$

The set of all active contacts $C_i \leq n_c$ has to be determined at each time instant by verifying the sign of the contact forces and the gap function. Note that the element forces $\boldsymbol{f}_j$ depend on the robot's state $\boldsymbol{q}, \dot{\boldsymbol{q}}$ if a compliant contact model is used, e.g. a linear spring–damper model. The overall contact force vector $\boldsymbol{\Lambda}$ in (2.1) is finally constructed with the entries of both feet

$$\boldsymbol{\Lambda} = [\boldsymbol{\lambda}_1^T, \boldsymbol{\lambda}_2^T]^T. \tag{2.3}$$

The above shown dynamics description is visualized in Figure 2.2. Details will be explained in the following parts. It can be summarized by the following two statements:

a) unilateral contacts and the resulting constraints for possible motions are a concern of feasibility in bipedal locomotion

b) underactuation and the resulting inability to directly control the full state are a concern of stability in bipedal locomotion.

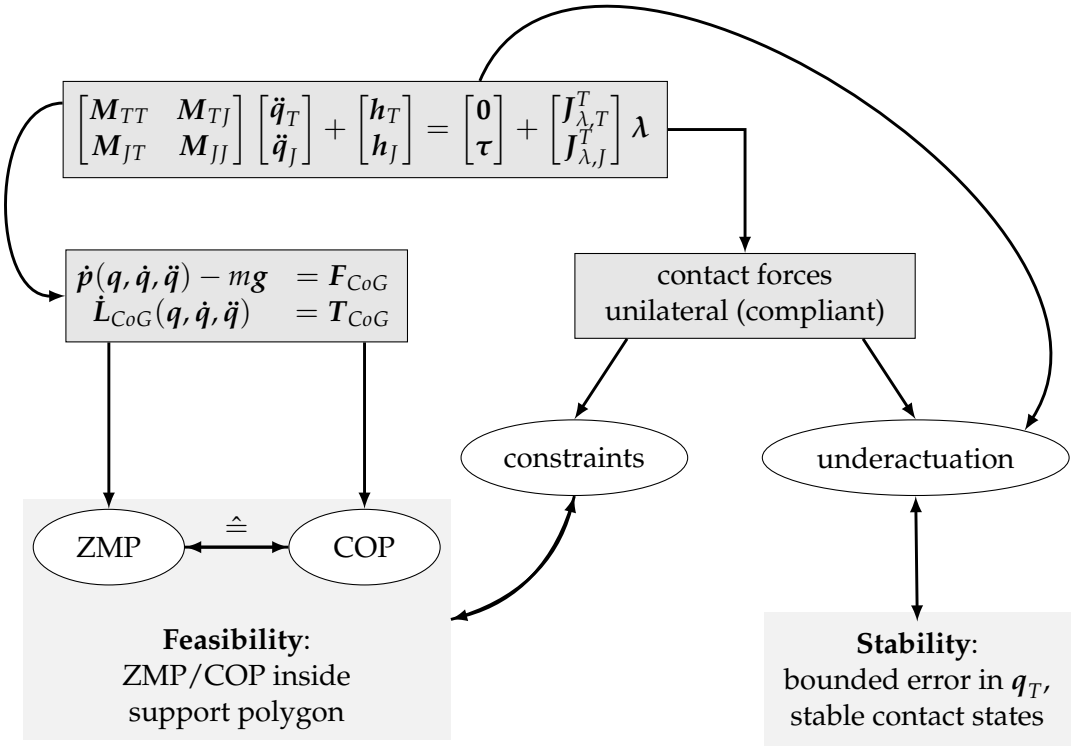


Figure 2.2: Overview – different concepts for feasibility and stability. Relations to the overall multibody dynamics are shown as well.

2.2 Feasibility in Bipedal Locomotion

The first line in (2.1) is considered for the discussion of feasibility. It gathers the Newton and Euler equations of the whole robot (Wieber 2008) which can be split into the change of linear ($\dot{p}$) and angular momentum ($\dot{L}_{CoG}$) about the system's overall Center of Gravity (CoG)

$$\dot{p}(q,\dot{q},\ddot{q}) - mg = F_{CoG}, \tag{2.4}$$
$$\dot{L}_{CoG}(q,\dot{q},\ddot{q}) = T_{CoG}. \tag{2.5}$$

The forces F_{CoG} and torques T_{CoG} correspond to the right hand side of the overall EoM. They are limited due to unilateral contacts, i.e. they have to be inside a valid subset of the contact wrench. It was shown in Buschmann (2010) and Kajita et al. (2014) that this corresponds to the widely used ZMP concept (Vukobratovic and Borovac 2004).

2.2.1 Constraints – The Zero Moment Point

The concept of the ZMP was introduced more than 40 years ago by Vukobratovic and Stepanenko (1972) and is to this day a widely used tool for trajectory generation and control in humanoid robotics. *"The ZMP is defined as that point on the ground at which the net moment of the inertial forces and the gravity forces has no component along the horizontal axes"*(Vukobratovic and Borovac 2004). The assumptions are that there are only coplanar contacts (normally related to contact surface between foot and ground) and there is no

slippage. Assuming that the total acting force $\boldsymbol{F}_0$ and torque $\boldsymbol{T}_0$ are known for a fixed reference point 0 i.e. using the result from (2.4) and (2.5)

$$\boldsymbol{T}_0 = \boldsymbol{T}_{CoG} - \boldsymbol{r}_{0,CoG} \times \boldsymbol{F}_{CoG} \quad \text{and } \boldsymbol{F}_0 = \boldsymbol{F}_{CoG} \tag{2.6}$$

the ZMP position $\boldsymbol{r}_{0,zmp} = [x_{zmp}, y_{zmp}, 0]$ can be computed with the relation

$$\boldsymbol{T}_{zmp} = \boldsymbol{T}_0 - \boldsymbol{r}_{0,zmp} \times \boldsymbol{F}_0 = \begin{bmatrix} 0 & 0 & T_{zmp,z} \end{bmatrix}^T. \tag{2.7}$$

Solving (2.7) for the horizontal position delivers finally the ZMP equations

$$x_{zmp} = -\frac{T_{0,y}}{F_z}, \qquad y_{zmp} = \frac{T_{0,x}}{F_z}. \tag{2.8}$$

Note that the ZMP is assumed to lie on a horizontal plane which intersects with the point 0 such that $z_{zmp} = 0$. This can be realized by simply shifting the appropriate Frame of Reference (FoR). Defining the support polygon as the minimal convex hull of all active contact points (see Figure 2.3 for one leg or two legs on the ground) the ZMP criterion states that the robot is *dynamically balanced* or its motion is *physically feasible* if the ZMP lies strictly inside it (Vukobratovic and Stepanenko 1972; Wieber 2002). For a given motion $(\boldsymbol{q}(t), \dot{\boldsymbol{q}}(t), \ddot{\boldsymbol{q}}(t))$ the total change of linear and angular momentum can be computed with (2.4) and (2.5) and the necessary reaction forces and torques can be afterwards evaluated in terms of feasibility using the ZMP relations of (2.8) with the current support polygon. That means that those trajectories can be tracked by a control algorithm as long as this criterion is fulfilled. Note that this criterion is not a necessary condition for stable walking, e.g. see Pratt (2001). Huang et al. (1999) used this approach to generate a walking pattern for a 7-link planar model of a biped. They parametrize the hip trajectory via cubic splines and modify initial and final values of the x position until the ZMP condition is met. The ZMP concept can be extended to rough terrain situation (Sardain and Bessonnet 2004) by a more general formulation where the ZMP can exist on any surface (not only horizontal) and by introducing a virtual surface. This is a weighted mean surface of all surfaces that are currently in contact with the feet. Nevertheless the conditions of sufficient friction and coplanar contact of each foot remain. The authors also showed that the Center of Pressure (CoP) is identical to the ZMP. The only difference is their computation, since the CoP is determined from the reaction forces and the ZMP from the motion, i.e. right and left hand side of (2.4) and (2.5).

The authors of (Goswami 1999) introduced the Foot Rotation Indicator (FRI) point which is a point on the foot/ground contact surface where the net ground reaction force would have to act to keep the foot stationary. This point coincides with the ZMP/CoP as long as the foot contact is stationary. If the foot rotates over an edge the FRI moves outside the support polygon. It can be therefore seen as a generalization of the ZMP criterion.

The ZMP criterion can be used in an inverse manner by first determining a feasible ZMP trajectory for a given sequence of steps and the corresponding support polygons in order to calculate afterwards a corresponding CoG trajectory. This inverse problem is commonly solved using a simplified model and is subject of the next section.

2.2.2 Center of Gravity Trajectory Planning Concepts

Optimization based methods which solve the overall EoM of the robot showed to take too much computational time to be applied in real-time, e.g. Bessonnet (2004) and Buschmann et al. (2005). There are several groups that work on the computational efficiency to reduce the time to obtain a solution (Kuindersma et al. 2015; Tassa et al. 2012) or use machine

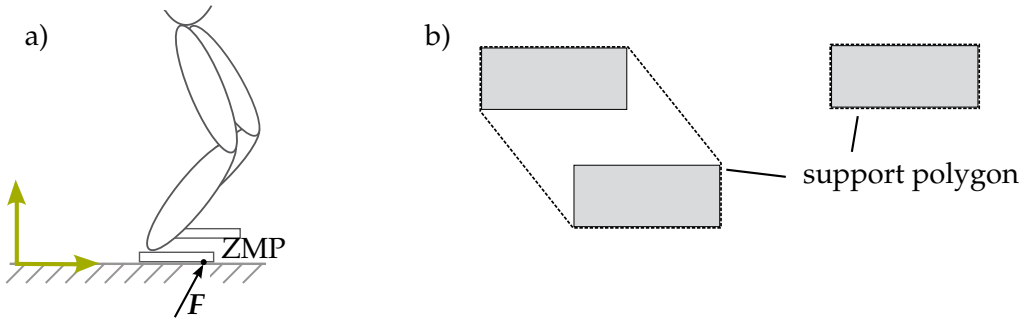


Figure 2.3: Physical feasible ZMP position (a) and support polygon definition for one leg and two legs on the ground (b).

learning techniques to generate motions from a database (Koch et al. 2015). While (2.4) and (2.5) describe the dynamics of a biped exactly, in real-time motion generation it is often assumed that $|\dot{\boldsymbol{L}}_{CoG}| \ll 1$. The simplification can be inserted in (2.6). This delivers the Linear Inverted Pendulum Model (LIPM) which is introduced in Kajita and Tani (1995) and can be used to describe the horizontal motion of the CoG. Writing the CoG vector $\boldsymbol{r}_{0,CoG} = [x_c, y_c, z_c]^T$, the x and y components are denoted by

$$T_{0,x} = 0 + my_c(\ddot{z}_c + g) - mz_c\ddot{y}_c, \tag{2.9}$$
$$T_{0,y} = 0 + mz_c\ddot{x}_c - mx_c(\ddot{z}_c + g). \tag{2.10}$$

The simplification that the CoG height is constant $z(t) = h = \text{const.}$ and inserting the relations (2.8) finally delivers the LIPM dynamics

$$\ddot{x}_c = \frac{g}{h}(x_c - x_{zmp}) \quad \text{and} \quad \ddot{y}_c = \frac{g}{h}(y_c - y_{zmp}). \tag{2.11}$$

Since the simplified equations for the x and y direction are identical, the following considerations only treat the x direction. The same results for motions in y direction can be obtained be simply replacing the corresponding variables. The analytic solution for x_c from (2.11) can be stated if the ZMP trajectory is assumed to be constant or linear by superposition of the homogeneous and particular solution

$$x_c(t) = c_1 e^{\omega t} + c_2 e^{-\omega t} + x_{zmp}(t). \tag{2.12}$$

The constant $\omega = \sqrt{g/h}$ describes the natural eigenfrequency of the LIPM. Equation (2.12) suggests to consider the ZMP as input in order to generate a CoG trajectory. Following this idea and assuring that the ZMP trajectory is always inside the support polygon results in a feasible $x_c(t)$ by solving (2.12) for an initial state $x_c(0), \dot{x}_c(0)$. One problem in this straight forward procedure is the unstable eigenvalue resp. divergent solution which results in an unbounded behavior of the CoG motion. This can be avoided be constraining the final position to some desired value $x_c(t_e)$. If the generated trajectory should ensure C^1-continuity this results in an over-constraint Boundary Value Problem (BVP). One possibility to solve this problem is to introduce a modification of the ZMP trajectory with an additional free parameter γ (Buschmann 2010)

$$x_{zmp}(t) = x_{zmp,id}(t) + \Delta x_{zmp}(\gamma, t). \tag{2.13}$$

This results finally in a linear equation system that can be solved very efficiently (cf. Figure 2.4).

The work of Harada et al. (2004) extended this idea to arbitrary Spline-trajectories for the ZMP. The work in Takenaka et al. (2009a) extended it by a point mass located at each

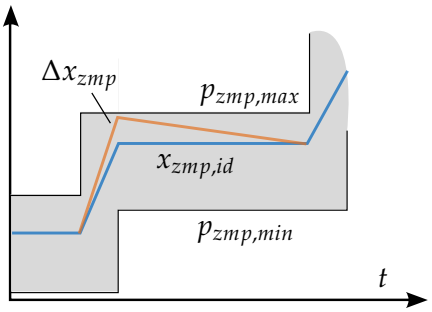


Figure 2.4: Example ZMP trajectory within allowable region with an additional modification.

foot resulting in a three-mass model. The inertia of the feet is considered as a disturbance term in the ZMP trajectory. The authors solve their problem with an arbitrary but known right hand side by a shooting method. Another interesting point is that they do not choose to set final values for position and/or velocity of the CoG but set a final value for the divergent component of motion. This avoids too restrictive constraints for the resulting BVP. The formulation based on motion decomposition is also used for CoG trajectory planning in Englsberger et al. (2011) and Morisawa et al. (2012).

In contrast to above stated methods, input and output can be shifted for a different problem formulation. This was first introduced in Kajita et al. (2003) and many other works followed this idea (Dimitrov et al. 2008; Lanari and Hutchinson 2015; Sugihara and Nakamura 2005; Tajima et al. 2009; Wieber 2006). The problem is formulated by introducing an input $u_x = \dddot{x}_c$ and applying a preview control for the resulting ZMP trajectory which is considered to be the output (also known as *cart-table model*)

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix} x_c \\ \dot{x}_c \\ \ddot{x}_c \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}\begin{bmatrix} x_c \\ \dot{x}_c \\ \ddot{x}_c \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ u_x \end{bmatrix}, \qquad x_{zmp} = \begin{bmatrix} 1 & 0 & -\frac{h}{g} \end{bmatrix}\begin{bmatrix} x_c \\ \dot{x}_c \\ \ddot{x}_c \end{bmatrix}. \tag{2.14}$$

Note that the inversion of the unstable system dynamics (2.11) results in a system with a positive zero at ω (Lanari and Hutchinson 2015), which creates a system that is non-minimum phase. The preview controller for the ZMP position takes future reference values into account. The work of Dimitrov et al. (2008) extends the formulation to a direct collocation method with additional inequality constraints for allowable the ZMP positions. They apply an efficient active set solver for the resulting problem which performs in real-time rates. In Nishiwaki and Kagami (2006) and Tajima et al. (2009) the current sensed state of the robot is used as initial value and the optimization is solved at a high frequency in real-time. The former work uses a database for the Riccati-solutions to reduce computational time. There is also the possibility to extend the formulation by additional variables for a step length modification (Stephens and Atkeson 2010; Urata et al. 2011; Wieber 2006) which can be used for large disturbance rejection.

Those works already include the idea to adapt the overall set of trajectories (walking pattern) according to the current state of the robot. Tracking a feasible set of trajectories is not sufficient for stable walking. This will be subject of the next section.

2.3 Stability in Bipedal Locomotion

The starting point of the following stability considerations in bipedal locomotion are the overall EoM (2.1) partitioned into actuated and unactuated DoFs. As stated before a bipedal robot is underactuated since there are less input variables than DoFs of the robot. Assuming that the $\boldsymbol{q}_J$ can be controlled via the motor torques $\boldsymbol{\tau}$, the basic control flow is as follows: The system input $\boldsymbol{\tau}$ controls the joints' DoFs $\boldsymbol{q}_J$ and is used to modify contact

forces $\boldsymbol{\lambda}_i$. This has to be done in such a way that the unactuated DoFs $\boldsymbol{q}_T$ follow the desired values. Note that this implies that the generated motion $\boldsymbol{q}$ accounts for limitations of $\boldsymbol{\lambda}_i$. In other words the motion has to be feasible. In addition the inertia $\boldsymbol{M}_{TJ}$ of the links can be used to produce a modified reaction force and moment which ends in *wind-milling* strategies (Goswami and Kallem 2004) or future contact force limits can be modified by shifting next footstep positions.

In the following a short overview of stability criteria is given and feedback stabilization concepts for bipedal robots are summarized. Those concepts include local feedback controllers as well as predictive modification strategies.

2.3.1 Stability Criteria

Viability Kernel

In order to transcript the avoid-falling-objective into a mathematical description Wieber (2008) proposes the use of viability theory (Aubin 1991). The author introduces the set $\mathcal{F}(t)$ of all configurations $\boldsymbol{q}$ where the robot has fallen or tips over, in other words configurations that have to be avoided and that are non-viable

$$\forall t, \quad \boldsymbol{q}(t) \notin \mathcal{F}(t). \tag{2.15}$$

Additionally all configurations that lead to a fall, whatever the robot is going to do, are also labeled as non-viable. Thus the set of all viable configurations $\mathcal{V}(t)$ includes all $\boldsymbol{q}(t)$ that are neither a fallen state nor lead to a fall. The distance between a given configuration of the system to the closest non-viable configuration is the viability margin and can be used as stability margin of the system. Unfortunately for such complex systems as humanoid robots the set $\mathcal{V}(t)$ can not be determined numerically. Nevertheless this introduces the idea to use Model Predictive Control (MPC) with a simplified model. Non-viable configurations are represented by appropriate cost functions and inequality constraints. WIEBER proposes to use the LIPM together with a minimization of the CoG velocity and the constraints that the ZMP has to remain inside the support polygon $\mathcal{Z}(t)$

$$\begin{aligned} \min_{x_c(t)} \int_{t_0}^{\infty} |\dot{x}_c(t)|^2 \mathrm{d}t \qquad \text{s.t.} \quad & \ddot{x}_c = \omega^2 (x_c - x_{zmp}), \\ & x_{zmp}(t) \in \mathcal{Z}(t) \quad \forall t \geq t_0. \end{aligned} \tag{2.16}$$

This means that future motion will not diverge and will remain feasible as long as the biped can be represented by a pendulum. However the question how to describe a viable configuration remains which is the same concern as in standard stability theory. Describing a stable state for a humanoid robot in a mathematical way is still an open question in current research. It is also related to the used model. In this thesis a model with passive DoFs is used that allows to represent the underactuated behavior of the robot more precisely. One of those is the absolute inclination of the model wrt. the world which is then used in a MPC control scheme to avoid non-viable states by minimizing the inclination over a certain time horizon.

Periodic Motions

Bipedal locomotion with constant velocity can be seen as a cyclic motion which includes two steps (also called one stride). Cyclic trajectories starting at the state $\boldsymbol{x}(t)$ return after a time period T to the initial state $\boldsymbol{x}$

$$\boldsymbol{x}(t + kT) = \boldsymbol{x}(t) \quad \forall k \in \mathbb{N}. \tag{2.17}$$

In order to obtain such a *limit cycle* the system has to behave in such a way that deviations from the periodic limit cycle due to disturbances is going to return to it. This can be analyzed using Poincaré Maps (Khalil and Grizzle 2002) where a linear relation can be stated for small deviations $\Delta \boldsymbol{x}$ of the state

$$\Delta \boldsymbol{x}(t+T) = \boldsymbol{K} \Delta \boldsymbol{x}(t). \tag{2.18}$$

The matrix $\boldsymbol{K}$ describes a linear return matrix with one eigenvalue at 1.0 and the remaining ones have to be less than one in order to receive a stable limit cycle. This is often used for limit cycle walking robots (Grizzle et al. 2003; M'Closkey and Burdick 1993; Westervelt et al. 2003) in order to study their stability or design feedback laws. Hobbelen and Wisse (2009) successfully applied the cyclic stability criterion to their two-legged robot FLAME. They also combined it with an active lateral foot placement strategy. Another successful application can be found in Renjewski et al. (2015) for the robot ATRIAS. They extended the strategy of limit cycles with additional control laws and enabled the robot to walk on the grass applying a spring-mass model. However the eigenvalue analysis of the Poincaré return maps assumes periodicity and is only valid for small deviations. Walking arbitrary in unknown environments where also large disturbances occur this seems to be not the right criterion to analyze the stability of a bipedal system.

Capturability

The capture point is first introduced in Pratt et al. (2006) and Pratt and Tedrake (2006) and describes the ability of a robot to place its foot in such a way that the CoG will come to rest over it. Additionally they state that the trajectory leading to this *captured state* has to be feasible and the point has to be reachable. In the context of their capturability analysis the authors also introduce the capture region which is the set of all possible capture points. For larger disturbances they introduce the N-step capture point. The later means a capture point that can be reached within N steps.
They derive the capture point for a biped modeled as LIPM from the conservation of the orbital energy (Kajita et al. 1990). For a given state of the CoG $(x_c, \dot{x}_c)$ it was shown in Englsberger et al. (2011) that the capture point p_x can be computed with

$$p_x = x_c + \frac{\dot{x}_c}{\omega}. \tag{2.19}$$

It should be noted that this is the divergent component of motion previously introduced and used in Matsumoto et al. (2004). Assuming a constant ZMP position x_{zmp} in (2.12), the solution for a given initial state $(x_c(0), \dot{x}_c(0))$ is

$$x_c(t) = \frac{1}{2}\left(x_c(0) - x_{zmp} - \frac{\dot{x}_c(0)}{\omega}\right) e^{-\omega t} + \frac{1}{2}\left(x_c(0) - x_{zmp} + \frac{\dot{x}_c(0)}{\omega}\right) e^{\omega t} \tag{2.20}$$

which will diverge if the factor of $e^{\omega t}$ is not equal to zero. This leads finally to the same condition as stated in (2.19) (Buschmann 2010, p.52) and is also related to the usage of the divergent component of motion for trajectory planning. The divergent component of motion $q(t)$ is the unstable solution of the LIPM dynamics which can be obtained by a decomposition into a convergent and a divergent part $p(t)$ and $q(t)$ (Takenaka et al. 2009a). The two parts are defined as follows

$$\begin{bmatrix} p \\ q \end{bmatrix} = \begin{bmatrix} 1 & -\frac{1}{\omega} \\ 1 & \frac{1}{\omega} \end{bmatrix} \begin{bmatrix} x_c \\ \dot{x}_c \end{bmatrix}. \tag{2.21}$$

With (2.21) the LIPM-dynamics (2.11) can be transformed to

$$\begin{bmatrix} \dot{p} \\ \dot{q} \end{bmatrix} = \begin{bmatrix} -\omega & 0 \\ 0 & \omega \end{bmatrix} \begin{bmatrix} p \\ q \end{bmatrix} + \begin{bmatrix} \omega \\ -\omega \end{bmatrix} x_{zmp}. \tag{2.22}$$

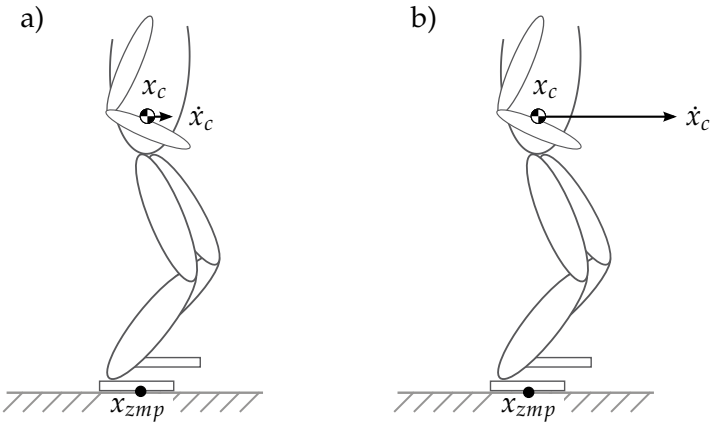


Figure 2.5: Stable (a) and unstable (b) state of the biped with the same CoG and ZMP position.

An application of the capturability-theory to the robot M2V2 is shown in Koolen et al. (2012). The conclusion drawn from (2.20) shows that the ZMP criterion (cf. Subsection 2.2.1) can not be used as stability criterion as it does not consider the CoG velocity of the robot. This is illustrated in Figure 2.5 where the robot is depicted with an identical CoG and ZMP position but in the first case with $\dot{x}_c \approx 0$ and in the second case with $\dot{x}_c \gg 0$. From (2.19) it can be seen that the necessary p_x^* easily exceeds possible values inside the support polygon (or a kinematic possible future support polygon) if $\dot{x}_c$ increases, while the ZMP criterion is fulfilled for both situations. This is the reason why the authors of (Pratt and Tedrake 2006) name the capture point a velocity based stability criterion.

2.3.2 Feedback Control in Bipedal Walking

The above introduced stability considerations are necessary when the bipedal robot does not exactly follow the ideal planned motion. Reasons are modeling errors and other external disturbances such as unknown rough terrain or pushes. Therefore the ideal motion plan has to be adapted according to the current state of the robot in order to avoid instability and a possible fall. There exist many feedback control methods in robotics (Siciliano et al. 2009) that are not directly applicable due to underactuation and the hybrid nature of bipedal walking. Strategies from literature that have been applied to bipedal locomotion can be divided in local modifications and (global) model predictive modifications for a given time horizon. In addition one can distinguish between formulations in workspace (Siciliano et al. 2009, p.84) and in configuration space of the robot which can also appear in mixed formulations.

Hirai et al. (1998) present a stabilizer of the humanoid robot ASIMO. The main feedback variable is the absolute inclination of the upper body which is treated as horizontal displacement error of the CoG and is used to calculate a reaction moment. This reaction moment aims to restore an upright posture. The regulation of the moment is then distributed on several control strategies which are a direct *ground reaction force control*, *model ZMP control* and *foot landing position control*. They are activated one after each other if the admissible limit of the former is reached. The *ground reaction force control* modifies the foot rotation depending on the tracking error of the reaction moment until the physical limit due to the foot geometry is reached. The remaining stabilizing moment that can not be regulated directly is then fed into the *model ZMP control* which accelerates the upper body's horizontal position in order to compensate for the moment. If the ZMP that results from this modified CoG trajectory exceeds the physical limits (reaches the edge of the support polygon), the *foot landing position control* adjusts the next stance position in order to make the CoG plan feasible. There is an extension to running motions with details given in Takenaka et al. (2009c). A short overview of the feedback control is

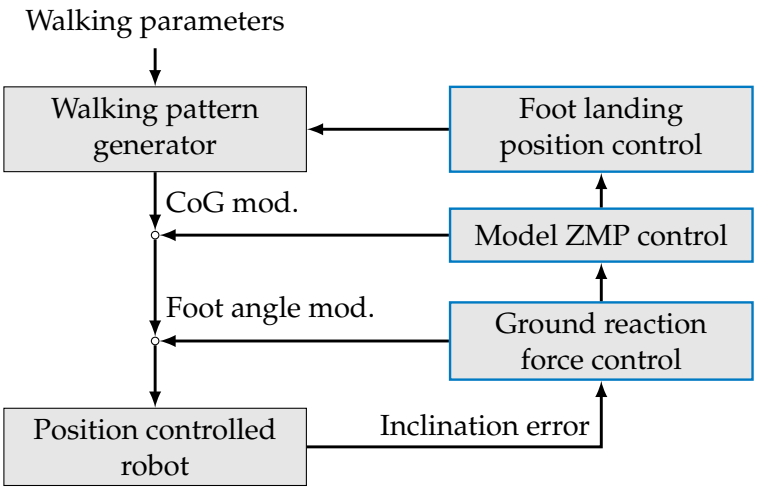


Figure 2.6: Feedback control of humanoid robot Asimo (adapted from Takenaka et al. (2009c)).

depicted in Figure 2.6. In this diagram the block position controlled robot includes the inverse kinematics solution.

A feedback control framework for the biped HRP2 is presented in Nishiwaki and Kagami (2009a) and Nishiwaki and Kagami (2009b). It is based on a continuous recalculation of the walking pattern with a moderate frequency and local adaption of the trajectories with a high frequency. The authors use an IMU in order to estimate current position, velocity and acceleration of the CoG. These values are used as initial values for the trajectory planning problem which is formulated as a preview control of the LIPM and runs with a cycle time of 10 ms. The local feedback runs at a frequency of 1 kHz and consists of several control strategies. The authors use an I-controller in order to follow the desired ZMP trajectory which is compared to the current acting ZMP calculated with contact force measurements. The output is a displacement of the horizontal torso position. In Nishiwaki and Kagami (2007a) a local modification for the swing foot trajectory depending on the inclination error is presented. It basically shifts the swing foot's height and orientation in order to ensure a touch down at the planned time and with the desired orientation. For large disturbances there exists an additional adaptation of future step parameters as the step time and next footstep position. An overview of the described control scheme is shown in Figure 2.7.

Another interesting approach is described in Tajima et al. (2009) which shows convincing results for a running humanoid robot. The authors also use a continuous recalculation of the trajectories with the estimated current state of the robot as initial value. In addition they reduce toe, ankle and knee joint stiffness just before expected landing time in order to absorb shocks at touchdown. This is realized by lowering the position control gains for the mentioned joints. A notable fact is that they do not use any information of the current acting contact wrench for their stabilization. In Tajima and Suga (2006) they roughly describe a method to adjust next foot landing position depending on the robot's current state which is based on the LIPM.

Compared to the acceleration of the CoG to produce a certain reaction moment modification a different approach is to directly control interaction forces and torques by modifying the trajectories of the feet that are in contact with the ground. The authors of Fujimoto et al. (1998) present a hybrid position/force control by using a tracking control in taskspace for the overall dynamics of the biped (2.4) and (2.5). Kajita et al. (2010) uses a damping control for the horizontal contact moments and vertical contact force by modify-

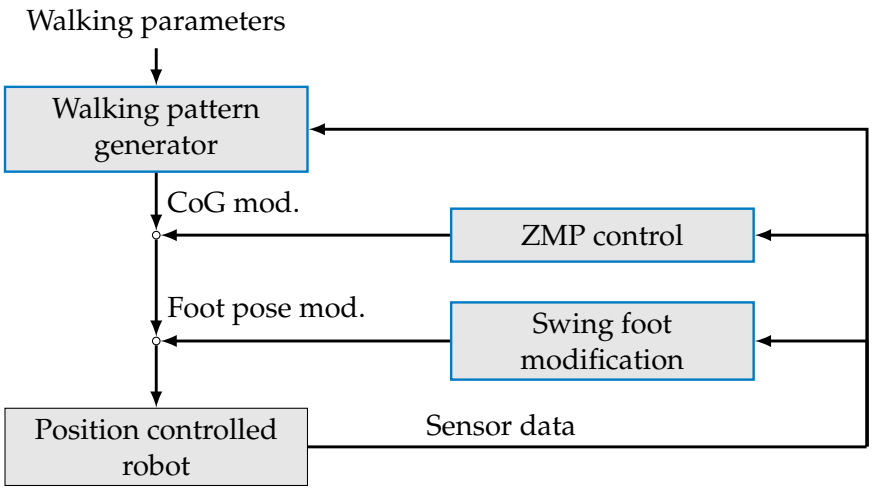


Figure 2.7: Feedback control of humanoid robot HRP2 (adapted from Nishiwaki and Kagami (2009b)).

ing the according angles and vertical displacement of the feet. There is a similar approach which considers a simple model for the contact dynamics in order to design an interaction force controller (Hashimoto et al. 2012). The problem of how to distribute a desired contact force or moment over both feet is commonly solved by using a preplanned force-distribution depending on the relative time of the walking cycle (Buschmann et al. 2009; Kajita et al. 2010; Nishiwaki and Kagami 2009b) but there is also a work that tries to solve it by an optimization formulation (Ott et al. 2011).

Position Control versus Torque Control

The basic difference between stiff position control and the torque control concept is visualized in Figure 2.8. For the former the inner cascade is the position loop which can be formulated for the joint angles $\boldsymbol{\Theta}$. The outer loop is consequently the force feedback that produces a modification $\Delta \boldsymbol{w}$ for the ideal taskspace trajectories $\boldsymbol{w}_{id}$. The desired joint angles $\boldsymbol{\Theta}_d$ can be computed for example with an inverse kinematics formulation on velocity level. This is shifted for the torque control where the position feedback is the outer cascade and the inner loop tries to track a desired force $\boldsymbol{\tau}_d$. Those are computed by an inverse dynamics model by feedback linearization. Note that in both concepts the quantities $\boldsymbol{\Theta}_d$ and $\boldsymbol{\tau}_d$ have to be computed central and can be sent then to local joint controllers. As a consequence the update rate for both quantities is of the same order and one has to decide what is the main objective in order to choose the most suitable framework.

A walking controller that is based on the full robot model was introduced in Löffler et al. (2002) for the biped robot JOHNNIE. The authors use a feedback-linearization technique to impose a linear behavior for the tracking errors. In Löffler (2006) it was reported that due to limited bandwidth of the inner force control loop a different stabilization that is based on an impedance control (Ott et al. 2010) with an inner position loop shows better results for fast walking tasks. It correlates with the principal design for most of the successful and powerful walking controllers for bipedal walking robots. This is an important result which shows that when using cascaded feedback loops the bandwidth of the inner loop limits the possible bandwidth of the outer ones and should be considered in the overall controller design. This is especially the case for whole-body controller that use a feedback on position and velocity level (Kuindersma et al. 2014; Sherikov et al. 2014) and compute with an inverse dynamics based method desired torques that are then regulated for each joint. This rises the general question what is the main objective for the whole system. If the robot should perform a certain task (walk in a commanded direc-

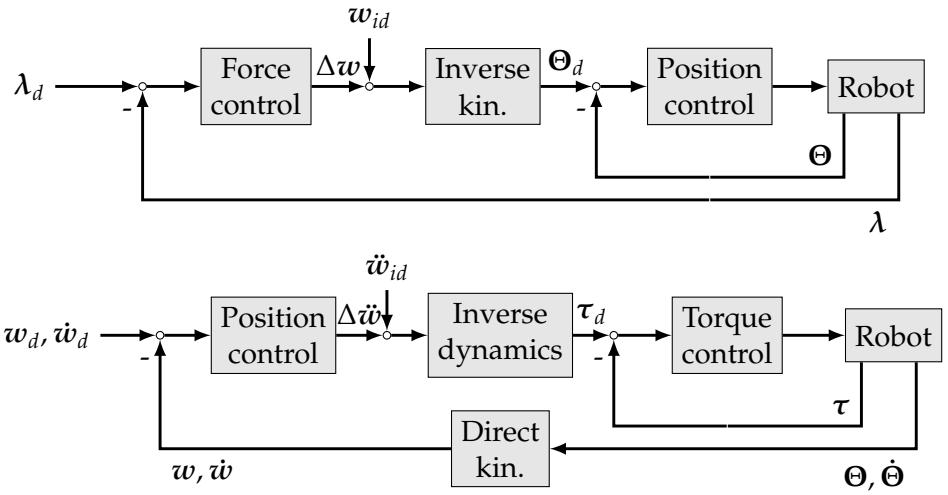


Figure 2.8: Concept comparison: *Stiff position control* (top) versus *torque control* (bottom).

tion or walk fast) a good tracking performance for the position level is preferable. If the terrain gets very unknown and rough and the robot should be compliant in its behavior a certain tracking for commanded torques are the main objective. But this requires models that are more complex for the control and the sensors have to fulfill certain bandwidth characteristics. This may be especially crucial for 6-axis force torque sensors and the IMU as well as for communication issues. Another question that has to be answered is the validity of the LIPM as planning tool because it assumes a stiff position controlled robot which was shown in Furusho and Masubuchi (1987) and will be a concern in Chapter 4.

While the focus of this part was mainly to describe overall sensor feedback frameworks for bipedal walking the next section treats methods that drastically change the future motion by modifying next foot landing positions. This aims mainly to increase the stability for large disturbances that can not be stabilized with local feedback laws and can be classified as a global stabilization strategy.

2.3.3 State Dependent Foot Placement

A modification of next footstep positions for stabilizing the robot is a challenging task which is in the author's opinion not yet completely solved. Given a current disturbed state of the overall robot the adaptation for the following footstep positions has to be computed in real-time. The solution of this problem has to consider the following constraints and properties

a) foot positions can not be changed immediately due to velocity limitations in the joints.

b) modified foot positions have to be reachable (inside the kinematic limits)

c) the effect of the next footstep position on the overall state evolution of the robot is in general nonlinear

d) modified footstep affects the feasibility of already planned CoG trajectories.

A stabilizing solution for a disturbed state that would lead to a divergence of the robot is schematically shown in Figure 2.9. The consequences of the above stated issues a) – d) can be seen in this diagram: the robot's absolute inclination at time t_1 is the same for the ideal and stabilizing trajectories (if the feet's inertia is neglected). From the end of

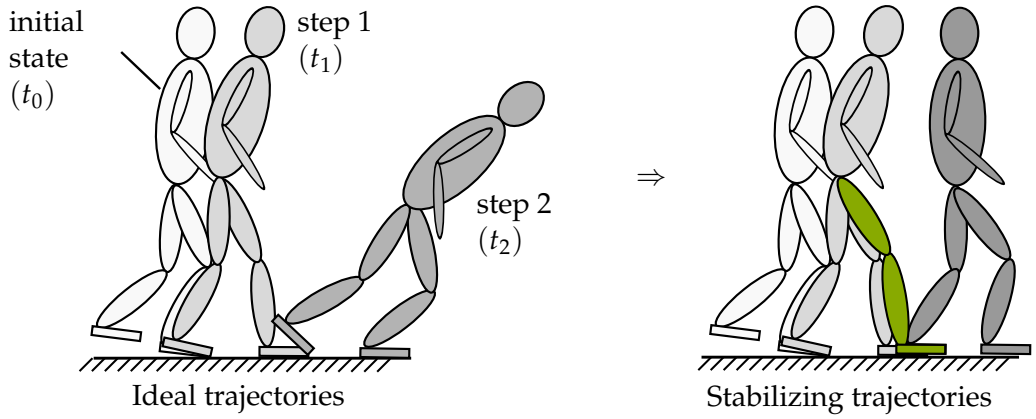


Figure 2.9: State evolution of a biped with disturbed initial state (current state) with snapshots taken at $t = t_0, t_1, t_2$. A stabilizing solution is shown on the right hand side.

the first step (t_1) until the second step (t_2) the modified step length has an influence to the robot's behavior and can stabilize it. A consequence is that footstep modifications are normally determined in a model predictive way. Due to the fact that the swing foot requires a certain amount of time to move to its final position the necessary prediction horizon is long compared to other real-time model predictive methods.

One often applied concept to determine footstep modifications is the capture point introduced in Pratt et al. (2012) and Pratt et al. (2006). Following the definition of the capture point and using the LIPM it can be computed with (2.19). In a walking controller it can be used to calculate a footstep modification depending on the measured CoG position and velocity error which is then used to generate a feasible new CoG trajectory. In Hodgins and Raibert (1991) and Raibert (1986) some very efficient heuristics for hopping robots are presented. They also introduce the concept of the neutral point which is a point the robot has to step in order to maintain its velocity. Based on the model of a two-link walker with point feet Wight et al. (2008) derived their foot placement estimator which is based on an energy conservation approach. Another way to generalize the efficiency of foot placement strategies is the usage of learning methods. Rebula et al. (2007) extended the capture point framework by an additionally learned offset which showed for learned situations a better performance. There exist also work for online learning techniques (Missura and Behnke 2015) that showed for planar considerations impressive results. Nevertheless there remains the question, how to extend those methods to general walking situations. In Aftab et al. (2012) the authors compared the results of a LIPM based stepping criterion with human balance measurements for similar perturbations.

The problem for a state dependent foot placement can be solved together with the problem to determine a feasible CoG trajectory. This can be done by formulating an overall optimization problem for both unknown quantities in a MPC formulation. There are several works solving the problem by approximating the robot with the LIPM and using a direct optimization method (Diedam et al. 2008; Stephens and Atkeson 2010; Urata et al. 2011). Sherikov et al. (2014) extended this approach by an inverse dynamics stabilization methods which is integrated in the stated optimization problem. The work shown in Urata et al. (2011) shows very impressive results which is also affected by a sophisticated hardware design using high power joint actuators (Urata et al. 2010) and very efficient calculation techniques. The authors state the optimal control problem for the pendulum in a different way by choosing the time derivative of the CoP as input and setting the weight of the input to zero. This allows to give an explicit solution of the problem and to compute more than hundred iterations of the optimization in each control cycle. To the author's knowledge this is the only work that includes an additional

step time optimization.

The methods presented in this work are related to the above mentioned strategies. Based on a new proposed nonlinear and more accurate prediction model an optimization based method to determine stabilizing foot placements will be presented. This is also extended to additionally optimize the CoG trajectory. In the author's opinion one main drawback in existing approaches is the inaccuracy of the LIPM and its restriction to flat ground/flat foot walking. These assumptions are removed with the newly introduced prediction model in Chapter 4.

Divergent component of motion with prediction

Beside using the divergent component of motion $q(t)$ computed from the current state it is also possible to predict its evolution into the future and use the predicted value in order to calculate a footstep modification similar to (2.7). The solution of $q(t)$ can be written as

$$q(t) = q(0)e^{\omega t} - \omega \int_0^t e^{\omega(t-\tau)} x_{zmp} \, \mathrm{d}\tau = q(0)e^{\omega t} + x_{zmp}(1 - e^{\omega t}). \tag{2.23}$$

Starting with the divergent component of motion and the end of the step $q(t_1)$, one can solve (2.23) for x_{zmp} for a given end value $q(t_1 + T) = q_d(t_1 + T)$ after a time period T

$$x_{zmp} = \frac{q_d(t_1 + T)}{1 - e^{\omega T} - q(t_1)e^{\omega T}} \tag{2.24}$$

The stepping criterion reduces for $T \to \infty$ to $x_{zmp} = q(t_1)$ which corresponds to using the predicted divergent component at t_1 (for $q_d = 0$) as desired footstep modification. In Takenaka et al. (2009c) a similar criterion for a footstep modification is stated.

2.4 Chapter Summary

This chapter gave an overview of the underlying dynamics of bipedal locomotion. Feasibility and stability criteria are introduced and some well known concepts for motion planning and stabilization of bipedal robots are reviewed. The relations are visualized in Figure 2.2. The ZMP/CoP can be computed from the Newton and Euler equations of the whole system and verified whether they lie inside the support polygon or not. This is a popular tool to validate feasibility of the planned motion as long as one restricts to flat footed gaits with no slippage. Otherwise one has to consider the force constraints introduced by unilateral contacts directly on the Multibody System (MBS). As the biped has less inputs than DoFs it is underactuated. This property results also from the unilateral (compliant) contacts and is mainly a concern of stability. Considerations of cascaded feedback loops that stabilize bipedal locomotion and how to design a framework that fits best to an existing hardware were shown. They will be used in the following chapter to extend the control system of LOLA to increase its robustness.

Chapter 3

Control Framework for Robust Walking

3.1 Introduction

This chapter starts with a brief overview of the mechanics, electronics and communication system of the bipedal robot LOLA. The basic structure of the planning and control system is presented. The description includes the general procedure to generate a walking pattern, existing feedback loops and models used in specific methods. This overview is the basis to introduce the control framework extensions for unknown environments in the second part of this chapter.

Unknown environments raise multiple issues for robotic systems: (1) Motion generation problems in complex environments have to be solved autonomously. (2) Robots have to be able to react quickly to dynamically changing environments or user input. (3) Robots have to be reliable and robust - even when subjected to unknown disturbances. The issues are solved by adapting the robot's motion according to sensor data. The sensor data can be classified into two categories: First category is data from a vision system that can be used to adapt the robot's motion to avoid collisions with the environment. The second category includes sensor data that is related to the state of the robot and its contact forces. Depending on this state the walking control system has to adapt the robot's overall motion. This is mainly a concern of the robot's ability to recover from severe disturbances resulting from external forces or model errors.

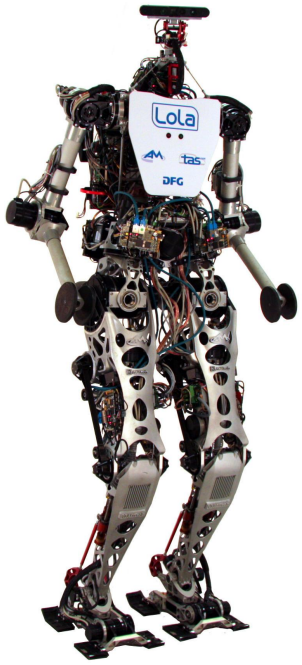
The second part of this chapter summarizes a hierarchical approach for using sensor data in LOLA's walking control system. First, methods for disturbance rejection are introduced. Afterwards, an integration and combination with collision avoidance methods is discussed. Details on the real-time system implementation are given since most algorithms are computationally expensive. Additionally a method to improve the overall positional tracking of the joint control is presented.

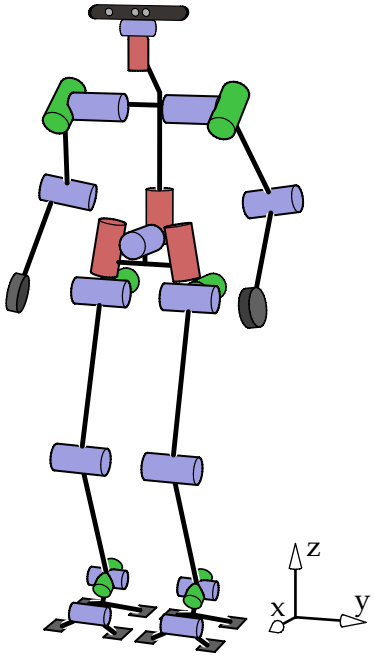
3.2 The Bipedal Robot Lola – System Overview

This section gives a summary of important properties and design concepts of the bipedal robot LOLA. The focus is on the overall mechatronic system including mechanics, electronics and control.

3.2.1 Mechanical Design

The aim of the LOLA-project is to develop a robot with human-like capabilities with a focus on bipedal locomotion. It is a platform to develop new methods for fast, flexible and robust walking. The robot weights approximately 60 kg and is 1.8 m tall. The mechanical structure of the robot is designed to be lightweight but stiff. The legs are designed to





Joint	DoF
Head	2
Shoulder	2
Elbow	1
Pelvis	2
Hip	3
Knee	1
Ankle	2
Toe	1
Total	**24**

Figure 3.1: Photograph and kinematic structure of LOLA. The table on the right hand side gives an overview of the robot's joints.

have low inertia. This increases the robot's capabilities to perform fast walking motions and moves the effective CoG height upwards. The mechanical design and its motivation can be summarized with the statement given in Lohmeier (2009):

> "Both static and dynamic behaviors of the mechanical system have a strong influence on the walking capabilities. In particular, structural weak points of the locomotor system can degrade walking performance by causing structural vibrations and deviations from the reference trajectories, which can destabilize the robot."

The kinematic structure with 24 joints is shown in Figure 3.1. Each leg consists of seven joints including an active toe. To increase the robot's redundancy there are two pelvis joints connecting the legs with the upper body. The arms are mainly used to compensate for the leg's motion and to minimize the angular momentum. They have three DoFs. All joints are electrically actuated using high power brushless DC motors. Most of the robot's joints are equipped with stiff high ratio Harmonic Drive transmission gears except for the ankle and knee. The ankle joints are designed by a parallel kinematics using two spatial slider crank mechanisms. Each is actuated with a motor mounted to the thigh. This introduces three main advantages, (1) the motors are shifted upwards which decreases the overall inertia of the leg. (2) by using parallel kinematics the peak torques of the motors can be reduced by approximately one third and (3) an increased workspace of the joints is realized. The knee joints are based on a roller-screw based linear drive with a four-bar linkage mechanism. Both ankle and knee kinematics have a nonlinear kinematics and therefore a nonlinear torque-speed characteristics. Mechanical data of all different joint types are listed in Table 3.1. Load ratings and top speed refer to joint side measurement units. The joints' ratio is chosen low compared to other robotic systems. This fact and the high power and low weight joint design especially for ankle, knee and hip fits perfect for large and fast movements during walking tasks of the robot.

Table 3.1: Mechanical joint actuator data. A: hip flexion, B: hip adduction, C: hip rotation, D: pelvis adduction/rotation, E: shoulder flexion, F: shoulder adduction/elbow flexion, G: toe flexion (data taken from Lohmeier (2010, p.179)). For ankle and knee joints only maximal values are given of the nonlinear kinematics.

Drive	Mass [kg]	Top speed [rad/s]	Continous load [Nm]	Peak load [Nm]	Gear ratio [-]
A	3.010	12.06	122	243	50
B	1.438	8.16	108	284	100
C	1.522	8.16	108	284	100
D	0.933	8.16	78	147	100
E	1.082	8.16	78	147	100
F	0.973	9.17	54	110	100
G	0.531	10.68	24	48	100
Knee	1.931	10.68	≤ 189	≤ 378	≤ 72
Ankle	0.516	10.68	≤ 115	≤ 230	≤ 70
Pan	0.237	8.90	0.90	7.50	100
Tilt	0.066	10.47	0.44	1.40	100

3.2.2 Sensor and Communication System

The biped LOLA is equipped with different sensors used to control its motion and recognize disturbances from the environment. Every sensor is chosen to be lightweight, precise and to have a high update rate. The last property is also related to the communication system and includes the time to transmit the sensor data to a feedback controller. This directly influences the bandwidth of the closed loop system. Considering for example the IMU whose data is used in the central controller the overall transmission time consists of the digitization of the measurements, sensor data fusion and sending the data with the communication system to the central computer. In the following a summary of available sensors (cf. Figure 3.2) and the utilized communication systems is given.

Inertial Measurement Unit

The IMU used on the robot is the commercial high precision system iVRU-FC-C167 from iMAR Navigation. The sensor is rigidly fixed to the upper body of the robot and consists of three fiber-optic gyroscopes and three MEMS accelerometers. The system runs with a frequency of 200 Hz including internal sensor fusion algorithms and error compensation models (e.g. velocity, acceleration based switching between different compensation models/methods) which provide accurate and drift-free measurements for the absolute inclination and inclination rate. The generated data can be accessed via Controller Area Network (CAN).

Force/Torque Sensors

LOLA's feet are equipped with 6 axis FTS developed with an optimized shear-beam geometry and strain gauges to measure deformations. They are mounted between ankle joint and foot in order to measure the reaction forces acting on the robot (considering the feet massless). The FTS data is read with the frequency of the high level control and has an accuracy of approx. 0.6 %. The discrete contact state is resolved with contact switches

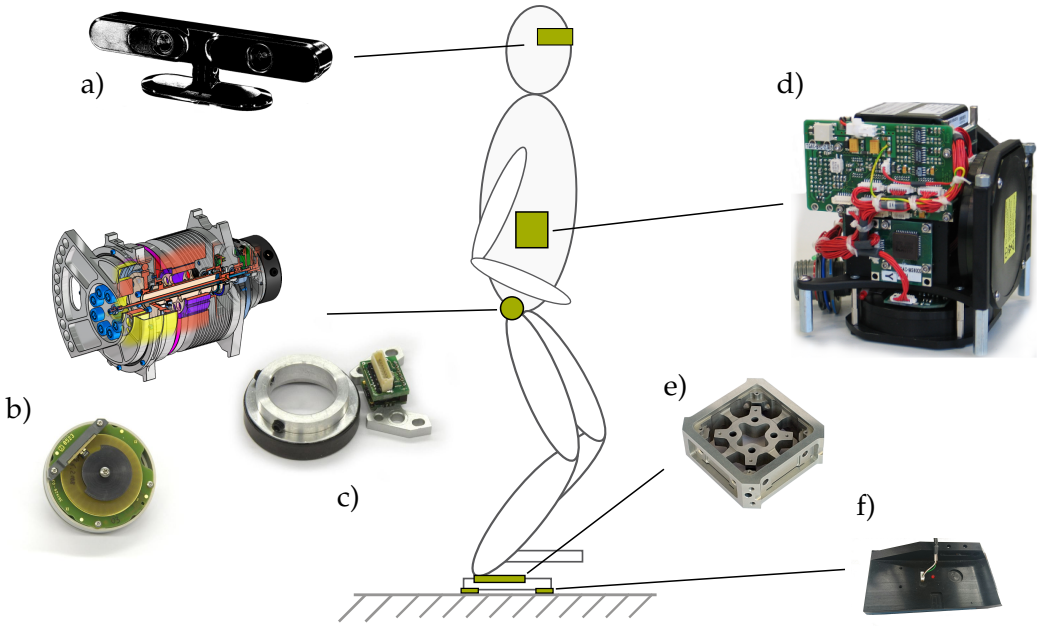


Figure 3.2: Mounted sensors and their locations: RGB-D sensor (a), absolute encoder with integrated limit switch (b), incremental encoder (c), IMU (d), FTS (e) and contact sensors (f).

that are directly located at the contact elements of each foot. The FTS data and the contact state is read with an in-house developed microcontroller-board (Cortex-M4) and is sent via CAN messages to the network.

Joint Sensors

Each joint is equipped with an absolute encoder on the link side and an incremental encoder on the motor side. The encoder evaluation is performed on local servo controller (servo controller from ELMO Motion Control are used) and is directly used for independent joint control. For safety reasons a limit switch is mounted on joint side enabling hardware collision checking. The resolution for incremental and absolute encoder for all joints is resp. 11520 and 131072 counts/rev..

Communication System and On-Board computer

The desired joint motion (target data) of the biped is planned on a central control unit and is sent to local servo controllers. The overall sensor data has to be transmitted to the central control unit in order to observe the overall state of the robot and adapt its motion according to its environment. Updated values for sensor and target data have to be available with a frequency of 1 kHz. A high performance communication system is required to transmit all mentioned data within the specified cycle time. Therefore the central control unit is connected with the servo controllers (ELMO Gold[1]) by an ETHERCAT-bus[2]. The IMU and FTS are accessed with a CAN-gateway running with 1000 kBds. An additional GPIO slave with A/D inputs is integrated which allows to include additional measurements or triggers of external sensors and devices.

The vision system is mounted together with the central control unit on the back of the robot. It is connected with the RGB-D sensor (Asus Xtion PRO LIVE [3]) located at the head. Both on-board computers consist of a mini-ITX embedded board with Intel Core

[1] www.elmomc.com
[2] www.beckhoff.de
[3] https://www.asus.com/de/Multimedia/Xtion_PRO_LIVE

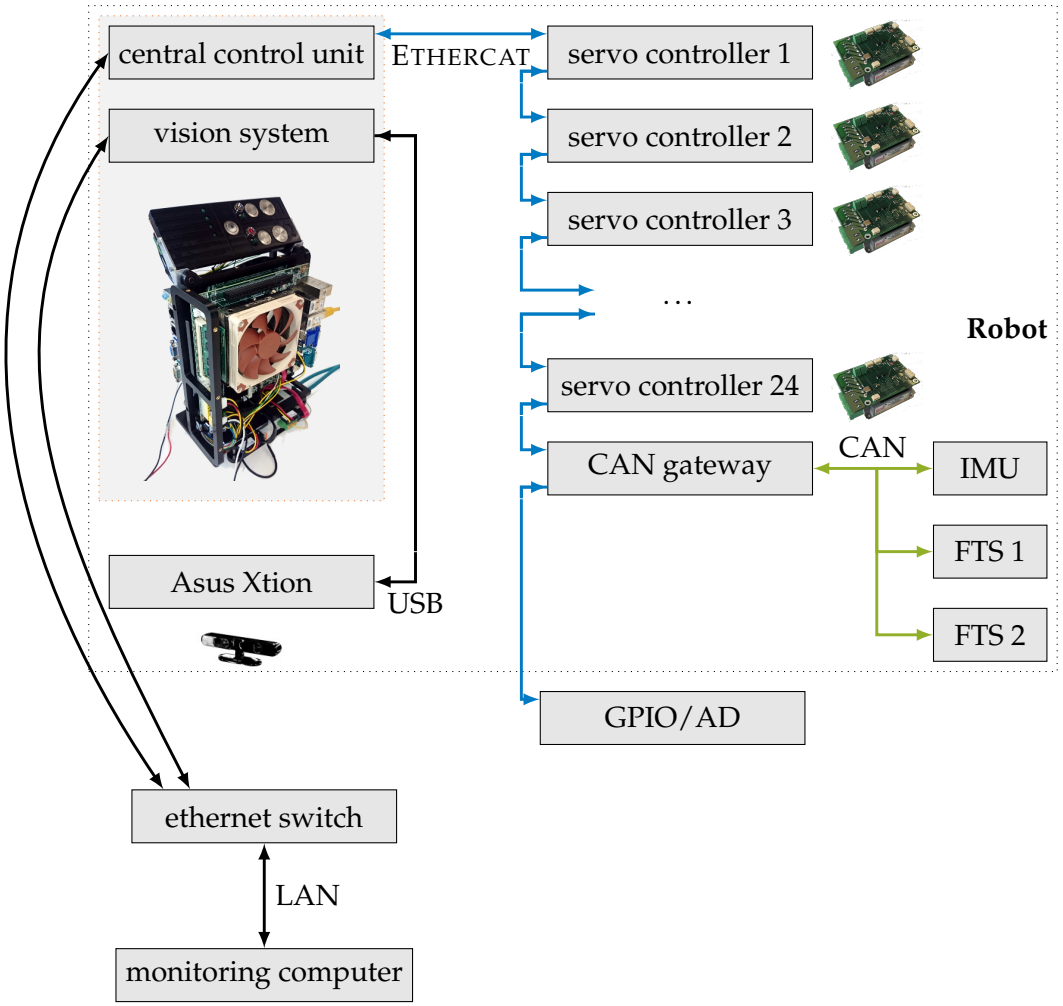


Figure 3.3: Overview of the communication system.

i7-4770S@3.1GHz (4x) processor and 8GB RAM. The computer with the vision processing software runs under a Linux OS and the other, on which the walking control is executed, runs under QNX NEUTRINO 6.6. Both computers use TCP to communicate via Ethernet. The overall architecture is summarized in Figure 3.3. The procedure of generating new desired data and using sensor data will be part of the following section.

3.2.3 Planning and Control System

The planning and control system receives high level walking commands from the monitoring computer and generates target data for the robot's servo controllers. Motion commands can be a 2D-velocity vector with a desired curvature or desired step lengths and step time. The generated motion has to assure that the robot will follow the given commands and will remain stable. It has to be computed in real-time since the commands are sent during execution and the walking control system has to consider sensor data. Remembering the basic challenges of the dynamics of bipedal locomotion this is a non trivial task. In order to fulfill the real-time requirement the state-of-the-art solution is to split the problem into sub problems that are solved hierarchically. The following description summarizes the work presented in Buschmann (2010) and Buschmann et al. (2012, 2007, 2009). The main parts of the planning and control system are shown in Figure 3.4.

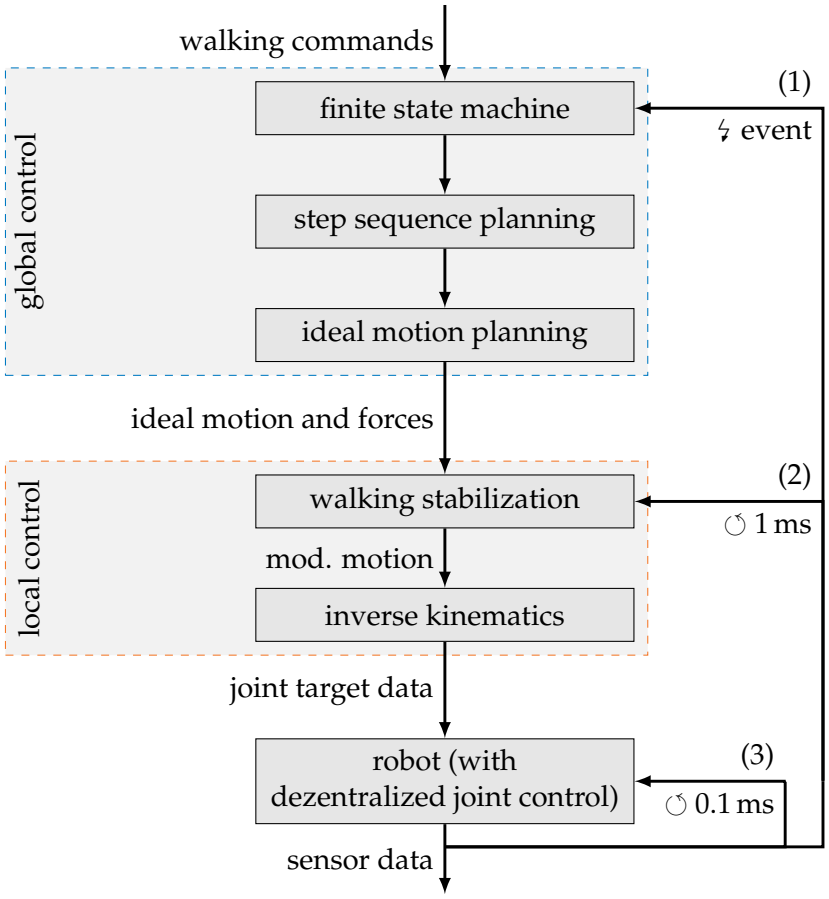


Figure 3.4: Overview of the existing planning and control system of LOLA (adapted from Buschmann (2010)).

Global Control

Coordination of the global control is done with a finite state machine that can change its state between *standing*, *start walking*, *walking* and *stop walking*. The *walking*–state is further partitioned in a Single Support (SS) phase, a Double Support (DS) phase and a contact event when the current swing leg touches the ground. An event based control is used (Buschmann et al. 2012) in order to increase the system's robustness. Sensor data namely the contact force measurements (or the contact switch information) are observed and can cause a transition from SS to DS before the ideal planned transition would occur. This is called an early contact event and increases the robustness of the overall system especially in unknown environments.

The step sequence planning uses the determined state together with the walking command to calculate a step sequence. Each step is described with a set of parameters including step lengths in x and y direction, maximal swing foot height, step timing for SS and DS, reference CoG height and others. The sequence consists of four to ten future steps and is determined once after every completed step. This parameter set is used to plan the ideal taskspace trajectories $\boldsymbol{w}_d(t)$. They consist of

- the foot positions wrt. the robot's CoG ($_T\boldsymbol{r}_{CoG,FR}$, $_T\boldsymbol{r}_{CoG,FL}$),
- the foot orientations wrt. the upper body ($_T\boldsymbol{s}_{TFR}$, $_T\boldsymbol{s}_{TFL}$),
- the absolute CoG position ($_T\boldsymbol{r}_{CoG}$),
- the absolute upper body orientation ($_T\boldsymbol{s}_T$),

- toe angles (q_{TR}, q_{TL}) and
- pan and tilt angles (q_{pan}, q_{tilt}) for the camera head.

Note that the first four parts are written in a reference system "T" which is an ideal planning FoR explained in Subsection 3.2.4. The foot trajectories as well as the vertical CoG trajectory are planned using fifth order polynomials that connect several target points specified through the step parameters. Using fifth order polynomials satisfies $\mathcal{C}^2$–continuity, which is mandatory for planning the horizontal CoG trajectories. Those trajectories have to be feasible in terms of considering the constraints caused by unilateral contacts (see Chapter 2). For LOLA this is solved by determining ideal feasible force or ZMP trajectories in advance and specifying initial and final values for the horizontal CoG position and velocity. This is done for a time horizon of two steps. The robot is approximated with a three mass model in order to state the BVP. The CoG trajectories are represented by a set of exponential splines which can be used to transform the problem into a linear equation system (Buschmann et al. 2007). This can be solved efficiently in real-time. The solution includes a time-varying CoG height and considers inertia effects of the foot motion which is important for fast walking. The global control provides finally ideal motion and forces to the local control.

Local Control

The local control runs with a cycle time of $t_c = 1\,\text{ms}$ and uses the planned motion ($\boldsymbol{w}_d$, $\dot{\boldsymbol{w}}_d$) and total forces and moments ($\boldsymbol{\lambda}_d = [\boldsymbol{F}_0, \boldsymbol{T}_0]^T$) for the current time instant. It modifies them depending on sensor data and finally generates joint target data which is sent to the local joint controllers. The walking stabilization (Buschmann et al. 2009) is done in two steps. The global aim is to keep the upper body in an upright position. Therefore the measured absolute inclinations (φ_x, φ_y) and inclination rates ($\dot{\varphi}_x$, $\dot{\varphi}_y$) in x and y direction from the IMU are used to generate stabilizing forces and moments

$$\begin{aligned} T_{y,stab} &= K_{p,y}(\varphi_x - \varphi_{x,d}) + K_{d,y}(\dot{\varphi}_x - \dot{\varphi}_{x,d}) \\ T_{x,stab} &= K_{p,x}(\varphi_{y,d} - \varphi_y) + K_{d,x}(\dot{\varphi}_{y,d} - \dot{\varphi}_y) \\ F_{z,stab} &= K_{p,z}\triangle z_{CoG} + K_{d,z}\triangle \dot{z}_{CoG}. \end{aligned} \tag{3.1}$$

Note that the index for the moments indicates a rotation about the corresponding axis. The additional modification of the desired vertical force is based on the error $\triangle z_{CoG}$ between the planned and the resulting CoG height which is computed using the output of the inverse kinematics. This prevents drift of the CoG height which is caused by model inaccuracies of the robot's total mass and foot motion caused by vertical force control on uneven terrain or when the robot is tilting. The drift would be produced by the subsequent force control. The adapted forces and moments $\boldsymbol{\lambda}_{mod} = \boldsymbol{\lambda}_d + \boldsymbol{\lambda}_{stab}$ are distributed to both feet using the desired force distribution ($\boldsymbol{\Lambda}_{mod} = [\rho_{f1}\boldsymbol{\lambda}_{mod}, \rho_{f2}\boldsymbol{\lambda}_{mod}]^T$) and are forwarded to a hybrid position/force control. The control uses an explicit contact model of the feet that consists of decoupled point contacts with a linear stiffness (2.2). By means of this model the controller modifies the planned taskspace target data according to the tracking error between target $\boldsymbol{\Lambda}_{mod}$ and measured forces $\boldsymbol{\Lambda}$. The controller is designed to impose a linear error dynamics

$$\boldsymbol{S}_\lambda \left(\triangle\dot{\boldsymbol{\Lambda}} + \boldsymbol{K}_\lambda \triangle\boldsymbol{\Lambda}\right) = \boldsymbol{O} \qquad \text{where } \triangle\boldsymbol{\Lambda} = \boldsymbol{\Lambda}_{mod} - \boldsymbol{\Lambda} \tag{3.2}$$

with the time constant $\boldsymbol{K}_\lambda$. $\boldsymbol{S}_\lambda$ is a selection matrix that specifies which components are used for the control. For LOLA the horizontal contact moments and the vertical force is chosen for each foot. The relation (3.2) and the contact model can be used to derive a

control law for the taskspace modification ($\boldsymbol{w}_{stab}, \dot{\boldsymbol{w}}_{stab}$). It is restricted to the vertical foot position and horizontal foot orientations with a selection matrix $\boldsymbol{S}_x$:

$$\dot{\boldsymbol{w}}_{mod} = \dot{\boldsymbol{w}}_d + \boldsymbol{S}_x \dot{\boldsymbol{w}}_{stab}, \tag{3.3}$$

$$\boldsymbol{w}_{mod} = \boldsymbol{w}_d + \boldsymbol{S}_x \boldsymbol{w}_{stab} \qquad \text{where } \boldsymbol{w}_{stab} = \int \dot{\boldsymbol{w}}_{stab} \mathrm{d}t. \tag{3.4}$$

When the swing foot leaves the ground the control for its components blends from force to position control. This is done by using a hybrid position/force control. Next the modified taskspace trajectories have to be transformed into the robot's joint space. The robot is eight times redundant ($\boldsymbol{w} \in \mathbb{R}^{22}, \boldsymbol{q} \in \mathbb{R}^{30}$). The method used for LOLA is based on *automatic supervisory control* originally proposed by (Liégeois 1977). It is a velocity level inverse kinematics algorithm developed for redundant manipulators. The problem is formulated as optimization where joint velocities and an arbitrary chosen cost function $H(\boldsymbol{q})$ are minimized subject to inverse kinematics as equality constraints. The cost function H is used to avoid joint limits and impose a comfort pose. Schwienbacher et al. (2011) extended the method by self-collision avoidance and vertical angular momentum minimization. The closed form solution is

$$\dot{\boldsymbol{q}}_d = \boldsymbol{J}^{\#}_{w,W} \dot{\boldsymbol{w}}_{mod} + \left(\boldsymbol{E} - \boldsymbol{J}^{\#}_{w,W} \boldsymbol{J}_w\right) \boldsymbol{W}^{-1} \frac{\partial H}{\partial \boldsymbol{q}}^T \tag{3.5}$$

with the Jacobian of the total system $\boldsymbol{J}_w = \frac{\partial \dot{\boldsymbol{w}}}{\partial \dot{\boldsymbol{q}}}$. The matrix $\boldsymbol{W}$ is used as weighting matrix for the joint velocities $\dot{\boldsymbol{q}}$ in the cost function and $\boldsymbol{J}^{\#}_{w,W}$ describes the $\boldsymbol{W}$-weighted pseudo-inverse of $\boldsymbol{J}_w$. The resulting joint target data ($\boldsymbol{q}_d, \dot{\boldsymbol{q}}_d$) is transformed to motor angles/velocities ($\boldsymbol{\theta}_d, \dot{\boldsymbol{\theta}}_d$) and sent via the ETHERCAT-bus to the local joint controllers.

Decentralized Joint Control

The control of each joint is running as cascaded position, velocity and current feedback control on the ELMO servo controllers (Siciliano et al. 2009, pp. 319). An overview is shown in Figure 3.5. It consists of a P-control for the motor position, a PI-control for the velocity and PI-control for current. Sampling times are 0.1 ms for position and velocity and 50 µs for current feedback. Feedback of the motor motion ($\theta, \dot{\theta}$) is obtained from the incremental encoders and current (I) is measured with an integrated current sensor. Additional feedforward values for velocity ($\dot{\theta}_d$) and current (I_d) can be commanded to the controllers. At the time of writing (November 2016) only velocity feedforward is used from the inverse kinematics output. Using the current feedforward part to include computed torque feedforward (inverse dynamics from planned motion) did not improve the used joint control system in terms of position tracking errors.

Summary of Feedback loops

The control system of LOLA can be summarized with the following three main feedback loops

(1) event based feedback from the contact switches to the finite state machine,

(2) continuous central feedback to current motion (in taskspace) depending on IMU and FTS data,

(3) and continuous decentralized joint control for the motor position.

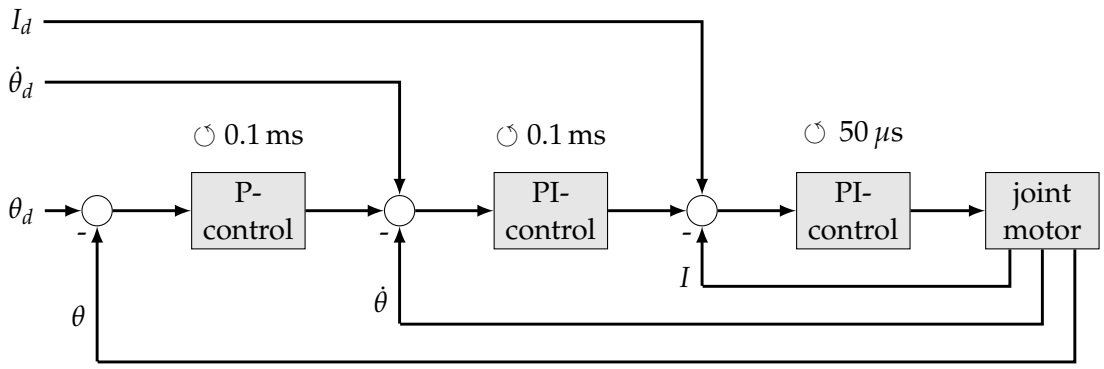


Figure 3.5: Decentralized cascaded joint control of LOLA.

Note that the inner cascades show a higher bandwidth than the outer ones. It follows the concept of stiff position control for humanoids shown in Figure 2.8. One main advantage is that the planned motion is adapted according to the reaction forces with a moderate bandwidth (it is mainly limited due to compliance in the contacts and the used I-controller for the force tracking) in an outer cascade while the commanded position is controlled with a high bandwidth. Following this control system concept, an additional continuous feedback to the global motion planning is added. The update frequency is chosen to be more than one magnitude lower than the local control.

3.2.4 Coordinate Systems and Orientation Errors

This section reviews details on the main coordinate systems and how orientation errors of the robot are represented. For the taskspace definition a planning frame of reference (denoted by index T) is used. This planning frame has its origin at the toe of the robot's stance foot and for an upright posture the x axis points horizontally forward and the z axis vertically upward. During a step it does not move in x direction. The planning frame is used to calculate direct and inverse kinematics of the robot, calculate the inverse dynamics and represent planned trajectories in taskspace. If the robot has a non-zero posture error, this FoR does not coincide with a world fixed frame (in the following subscript I assigns quantities that are represented in this world FoR). This is schematically shown in Figure 3.6. A special representation of the upper body's orientation is used for the walking control of LOLA originally proposed by Löffler (2006). This allows for treating the inclination in the x and y direction independently. The kinematics are explained in Buschmann (2010). Defining the axis of the world frame with ${}_I\boldsymbol{e}_x$, ${}_I\boldsymbol{e}_y$, ${}_I\boldsymbol{e}_z$ and of the planning frame with ${}_T\boldsymbol{e}_x$, ${}_T\boldsymbol{e}_y$, ${}_T\boldsymbol{e}_z$ the inclination angles are defined as follows

$$\phi_x = \angle({}_I\boldsymbol{e}_z, {}_T\boldsymbol{e}_x), \tag{3.6}$$

$$\phi_y = \angle({}_I\boldsymbol{e}_z, {}_T\boldsymbol{e}_y), \tag{3.7}$$

$$\alpha = \angle({}_I\boldsymbol{e}_x, {}_T\boldsymbol{e}'_x). \tag{3.8}$$

Orientation measurements from the IMU (represented as roll-pitch-yaw) are converted to the above stated parametrization. The final data of the absolute inclinations (ϕ_x, ϕ_y) and inclination rates ($\dot{\phi}_x, \dot{\phi}_y$) is available to be used in the walking control of the robot. For the prediction models introduced within this thesis slightly different variables are used to represent orientation errors. The inclination angles (3.6) and (3.7) are shifted by the value $\pi/2$ and are defined in the opposite direction:

$$\varphi_x = \frac{\pi}{2} - \phi_x, \tag{3.9}$$

$$\varphi_y = \frac{\pi}{2} - \phi_y. \tag{3.10}$$

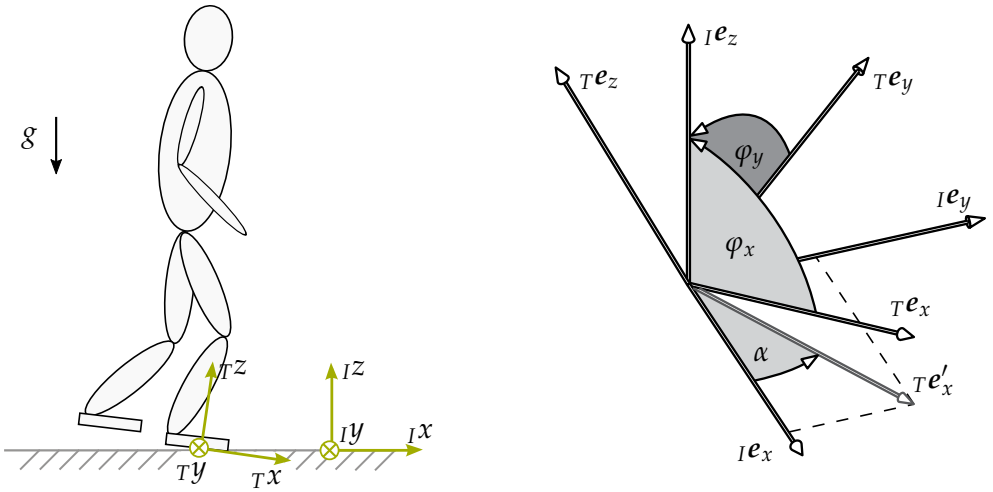


Figure 3.6: Left: inertial (index I) and planning (index T) frame of reference for an arbitrary robot posture. Right: representation of the upper body orientation (Buschmann 2010, p.69).

3.3 Control System Extensions for Robust Walking

The presented control system is extended in this work by several methods to improve the robot's walking capabilities in unknown environments. The main part is a model predictive trajectory adaptation. Integration details with collision avoidance methods are presented as well. Finally the real-time system will be explained and an improved joint feedforward control will be presented.

3.3.1 Model Predictive Trajectory Adaptation

This section describes how the existing control system is extended to increase the robustness of the biped in rough unknown environments. The main contribution is the introduction of a **continuous sensor based trajectory adaptation** (Nishiwaki and Kagami 2009a) which is done in a model predictive way. The method is summarized in Figure 3.7. The ideal planned motion and forces are adapted in the global control module according to current sensor data. The leg trajectories and the horizontal CoG trajectories are modified with an update frequency of 50 Hz. The main input variable is the measured error of the upper body inclination and inclination rate. With the assumption that the posture error of the upper body corresponds to the transformation between the planning and inertial FoR, the inclination errors are part of the robot's unactuated DoF q_T. These are one of the main state errors that have to be considered and minimized if large disturbances occur during walking in an unknown environment. Note that this directly extends the idea of the existing walking stabilization by an additional predictive approach since the same feedback variables are used. Further, stability issues of the system due to underactuation discussed in Chapter 2 are considered. It also matches the approach to describe the planned motion in an ideal FoR and adapt it according to the orientation error wrt. the world (cf. Subsection 3.2.4). With the sensor feedback being not a state error that is included in the taskspace w, computation times for the modification larger than one control cycle do not introduce discontinuities in the commanded trajectories.

A **prediction model** is used that includes the inclination error between robot and the environment as unactuated DoF. Additionally the model has to include the planned foot and CoG trajectories and should include the effect of the walking stabilization. The last point is important because otherwise results obtained by the model would be always too pessimistic. Speaking from a control theory point of view, the controller of the outer cascade has to use the plant with the effect of the inner feedback control loops. These

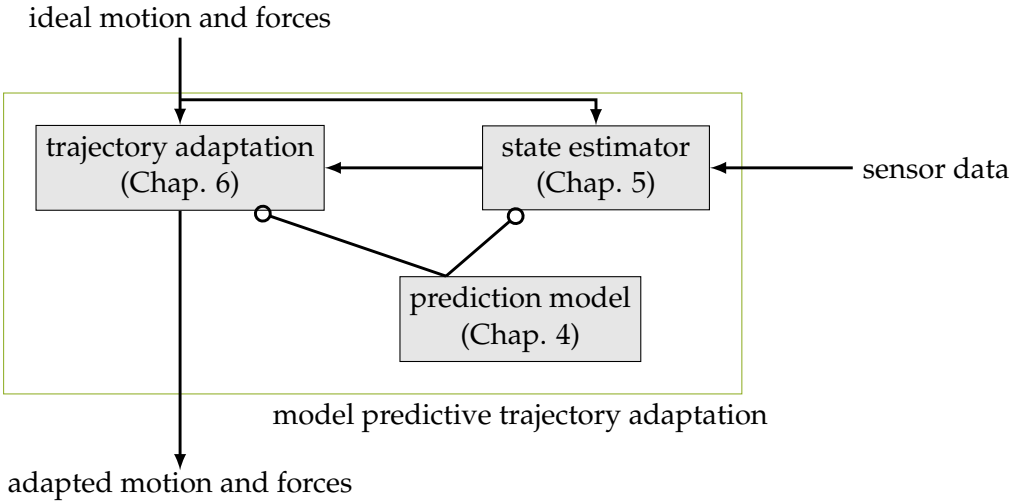


Figure 3.7: Main parts of the model predictive trajectory adaptation. Relations to the following chapters are given.

considerations are used in Chapter 4 to introduce a nonlinear prediction model. The derivation of the model emphasizes a real-time application.

The IMU data is not directly used as input for the trajectory modification. Chapter 5 describes a **state estimator** that extracts the global trend of the motion. It increases the robustness of the overall method. The estimator reduces high frequecy oscillations caused by structural vibrations and the local joint control. It finally acts like a low-pass filter but with less time delay.

The prediction model is used in the **trajectory adaptation** presented in Chapter 6. Leg trajectories are adapted by optimizing final values for the x, y and z position as well as the final horizontal orientation. Further the horizontal CoG trajectories are optimized. The optimization can be classified as online nonlinear model predictive control (Diehl et al. 2009). Figure 3.8 shows the integration of the method into the overall control system of LOLA. The depicted control scheme also includes methods for collision avoidance which will be described in the next section.

3.3.2 Integration with Collision Avoidance Methods

This part shows how a combination of the model predictive trajectory adaptation with collision avoidance methods can be realized within a hierarchical control system architecture. The realization on the real-time system is subject of the subsequent section. Splitting the problem into smaller pieces that are solved one after each other allows for generating a solution in real-time. Before the method of the combination is presented, an overview of LOLA's obstacle avoidance is given. One central point is that all geometries (robot as well as obstacles) are approximated by Swept-Sphere-Volume (SSV) objects (Hildebrandt et al. 2014; Schwienbacher 2014). There are three shape primitives (point, line and triangle SSV) used to build the collision models. One main advantage is that those allow for very efficient distance calculations and can be used for real-time collision avoidance. The collision model of the robot is fixed concerning its structure and only updated with the robot's configuration while the obstacles are generated, updated and removed online during walking by the vision system (Wahrmann et al. 2016). The environment information is obtained from the RGB-D sensor located at the head and is represented with a 3D point cloud. After pre-processing, the point cloud is segmented in surfaces and obstacles. Each object is represented with several SSVs depending on its distance to the robot. This way no prior knowledge of the environment in terms of color or models is required.

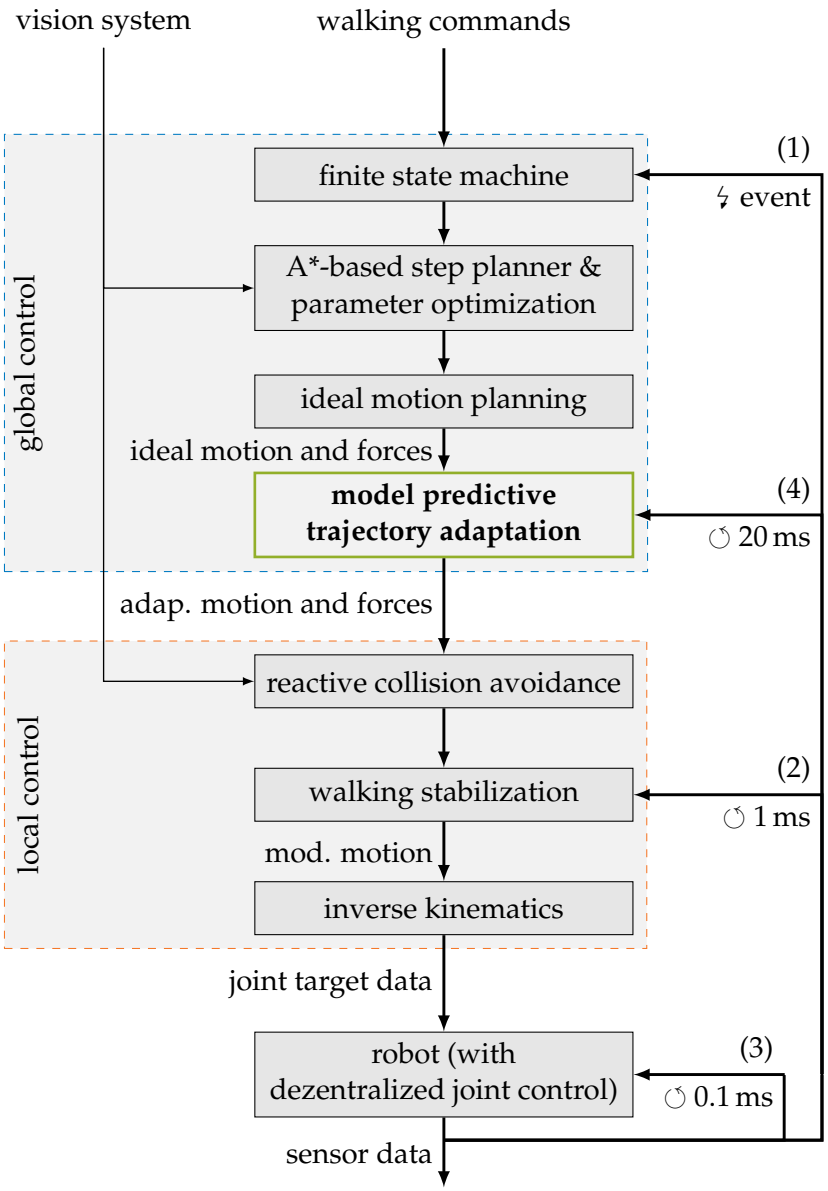


Figure 3.8: Extended walking control system of LOLA. Additional methods for flexible walking (A*-based step planner & parameter optimization, reactive collision avoidance) and robust walking (model predictive trajectory adaptation) are included.

Figure 3.9 shows a real setup and how the environment representation of the walking control looks like.

The obstacles are continuously sent to the real-time walking control system. First an A*-based step planner uses this information together with walking commands to determine a collision free step sequence for the next steps. Additional parameters for the step sequence as maximal swing leg height, CoG height, step time, etc. are also determined. This is done once before each step of the robot. In order to limit computational time, a simplified collision model of the robot is used (only the lower limb of the swing leg). This allows to perform a search for the next maximal 7 steps within 400 ms (number of steps and computing time strongly depend on the number of obstacles). The validity of the kinematic movement and possible collisions need to be checked additionally. The parameter optimization integrates the movement of the robot's next step using the whole kinematic model. It takes into account the local methods for collision avoidance and, consequently, the approximation of the robot's whole structure for collision checking.

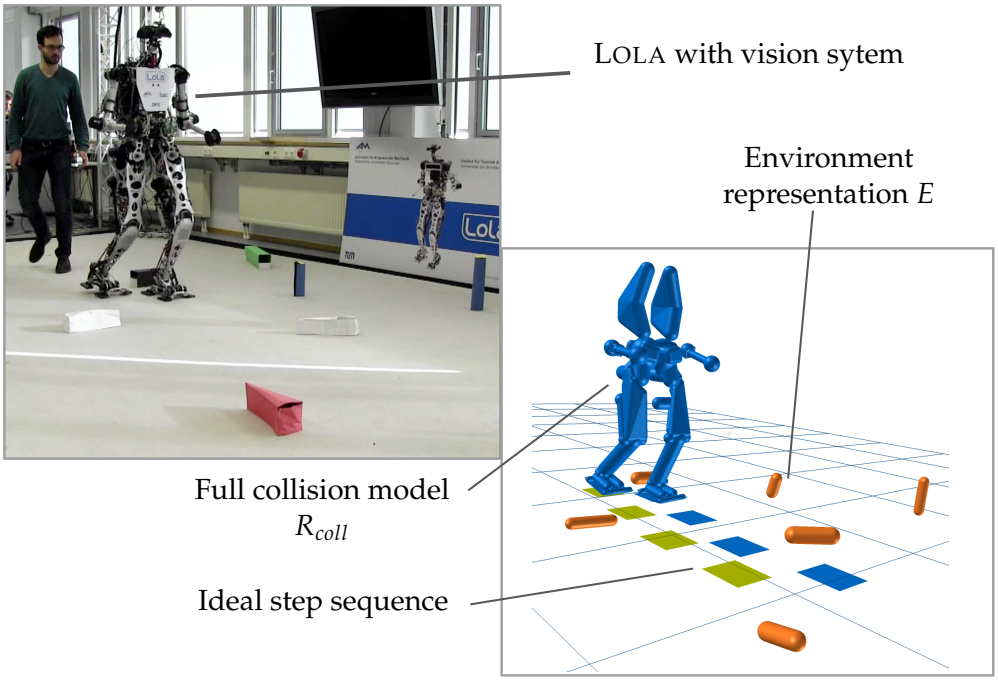


Figure 3.9: Left: real setup with LOLA and obstacles. Right: collision model of the biped (blue), obstacles in the field of view (orange) and current collision free step sequence (Hildebrandt et al. 2016).

Second, a reactive collision avoidance uses the environment information to locally modify swing foot trajectories during execution. This is mainly because the initial parameter set and the resulting trajectoroies are not necessarily collision free since simplified models are used (see Hildebrandt et al. (2015) for further explanation). The method is based on a projection of the cost function of the inverse kinematics (3.5) and extended by the obstacle avoidance into taskspace. Note that this method does not modify the final swing foot position (Hildebrandt et al. 2014).

A combination of collision avoidance methods with stabilization methods that also modify foot step positions requires a prioritization of one goal. At the end the algorithm has to decide whether it avoids collisions or prevents falling if an obstacle intersects with the optimal step length from the trajectory adaptation. The proposed method chooses collisions as main goal and places the footstep position as close as possible to the optimal one. Such a saturation of the stabilization acts like another disturbance that has to be compensated for in the following steps. Two general strategies will be presented in in Chapter 6:

- Generate a collision free ideal plan, adapt the footsteps according to sensor data and finally check versus collisions and kinematic limits. This corresponds to the method described in Chestnutt and Takaoka (2010).
- Generate a collision free ideal plan, describe the restricted areas of the environment by inequality constraints. Those are then directly considered in the sensor based footstep optimization.

3.3.3 Improved Joint Feedforward Control

For LOLA's control system there is one basic approach concerning the joint state. The feedback to global and local control uses only data from IMU and FTS. Actual values for all joint angles and velocities are only considered in the local joint control. Consequently,

the models and control strategies in the central control assume perfect joint tracking. In other words the joints are expected to follow planned motion ideally. One advantage of this approach is that global stability issues of the biped are treated decoupled from the feedback control of the joints. The following paragraph presents an improved joint control scheme that drives the robot's real behavior closer to above given assumption.

Figure 3.10 shows a measurement of the tracking error and the according desired joint velocity for the right hip flexion for two physical steps. One step where the right leg is the swing leg and a second step where it is the stance leg. The data is computed with the incremental encoder on motor side and converted via the gear ratio (cf. Table 3.1) to joint side quantities. From this figure it can be seen that the tracking error is directly related to the joint velocity and mainly occurs during swing phase. This is the main motivation for the method presented in the following to improve the joint's positional tracking performance.

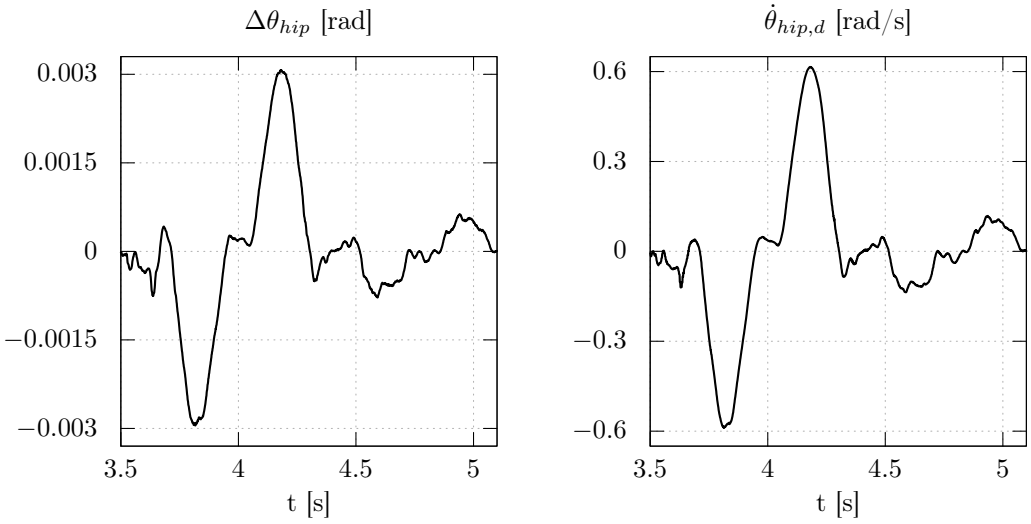


Figure 3.10: Position tracking error and joint velocity for the right hip flexion joint.

One way to use this knowledge is to identify an additional feedforward filter. A transfer function $G_e(s)$ from the desired joint velocity $\dot{\theta}_{hip,d}$ to the error $\Delta\theta_{hip} = \theta_{hip,d} - \theta_{hip}$ is used

$$\Delta\theta_{hip}(s) = G_e(s)\dot{\theta}_{hip,d}(s). \tag{3.11}$$

Several pole/zero configurations were tested to determine filter coefficients. All identified transfer functions show a fit of approximately 95-96 %, a constant gain $k_e = 0.005$ and poles p_i that are at the very left complex half plane ($\mathrm{Re}(p_i) \ll -100$). Consequently a constant gain k_e is used for the transfer function

$$G_e(s) = k_e. \tag{3.12}$$

With (3.12) it is possible to compute the tracking error of the controlled joint in advance. It can be used to improve the joint's tracking performance by adding the filter output to the target joint angle. In a discrete time setting this means that the tracking error of the next time step is already known and can be used as input to the position control. Figure 3.11 shows on the left side the corresponding control scheme. The system can be transformed by using a filter $\hat{G}_e(s)$ for the overall desired velocity that is computed with

$$\hat{G}_e(s) = 1 + G_e(s)G_p(s). \tag{3.13}$$

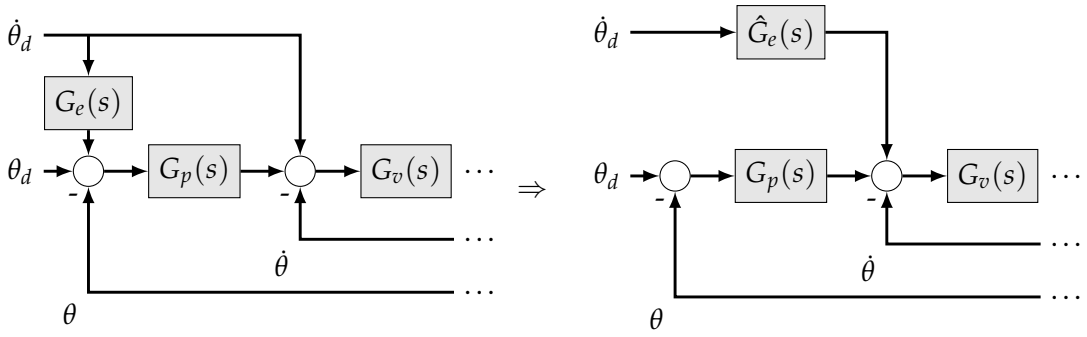


Figure 3.11: Improved feedforward control: on the left side with the identified filter $G_e(s)$ and on the right side the transformed control block diagram with a modified feedforward velocity.

Same procedure can be applied for the other joints which gives identical results only with different constant gain values k_e.

To enable an automatic computation of the feedforward gains a reinforcement learning (Sutton and Barto 1998) based strategy is used. This has the advantage that the optimal gains can be identified online when the joint control system or motor are changed. The velocity feedforward filter $\hat{G}_e(s) = 1 + k_{rl}$ is added to the walking control system with an inital value $k_{rl} = 0$. The robot is stepping in place and its joint tracking errors are recorded for the learning process. A cost function is defined with the Root Mean Squared Error (RMSE) over a time period of two steps for joint i

$$J_{\theta,i} = \sqrt{\frac{1}{N} \sum_{k=0}^{k=N-1} \Delta\theta_i^2} \tag{3.14}$$

The simple policy is used to increase k_{rl} by 0.1 increments as long as the cost function (3.14) decreases. This is performed for all joints simultaneously. The learning progress for the hip joint is depicted in Figure 3.12. The algorithm finds the best gain $k_{rl} = 1.3$ and can reduce the tracking error by approximately 90 %. Inserting the position controller $G_p(s) = 250$ into (3.13) results for the identified $\hat{G}_e(s) = 2.25$. Finally, the tracking performance is evaluated in a fast walking experiment. The robot is commanded to walk 5 m with a maximum step length of 0.6 m and a step time of 0.8 s. Figure 3.13 shows the tracking error for the hip joint without (ref.) and with the velocity feedforward gain adaptation (opt.). It can be seen that the filter produces much better results for the experiment. Results for other joints are provided in Appendix A.

3.3.4 Real–Time System

The control system is running on the real-time operating system QNX NEUTRINO 6.6[4] which is based on a microkernel design and has full multi-core support. The walking control is split in three main processes running in parallel (see Figure 3.14). Process 1 includes the global control part, process 2 the local control and process 3 is responsible for the ETHERCAT communication with sensor and target data exchange. To enable the combination with collision avoidance methods, process 1 includes a separate thread that calls the A*-based step planner. It computes the following n_{step} steps during the execution of step k. The ideal motion planning is called before a new step starts (vertical lines in Figure 3.14) to ensure that new target data for the ideal motion and forces is available. The model predictive trajectory adaptation (Adapt) is performed sequentially in the main

[4] http://www.qnx.com/

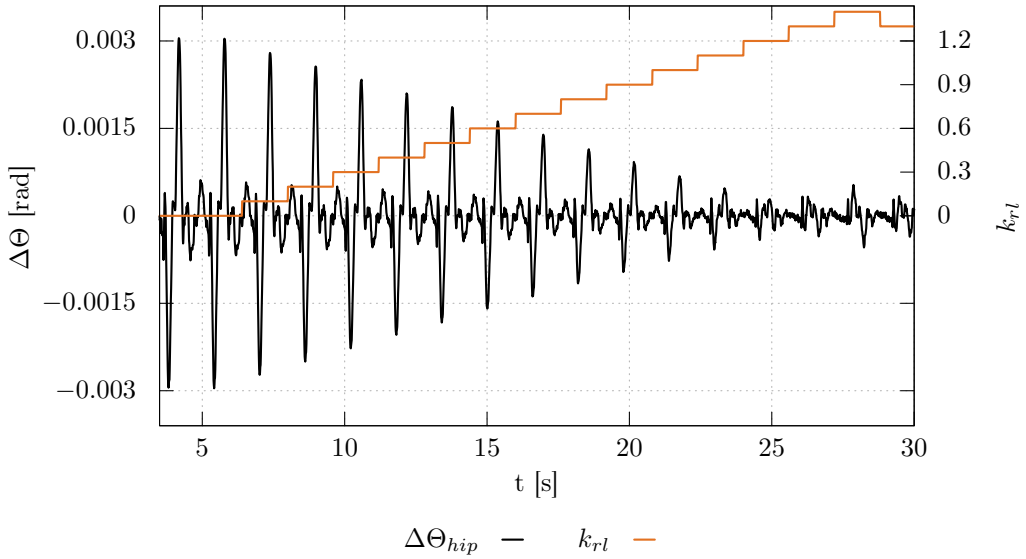


Figure 3.12: Tracking error and gain k_{rl} during the learning process of the hip joint.

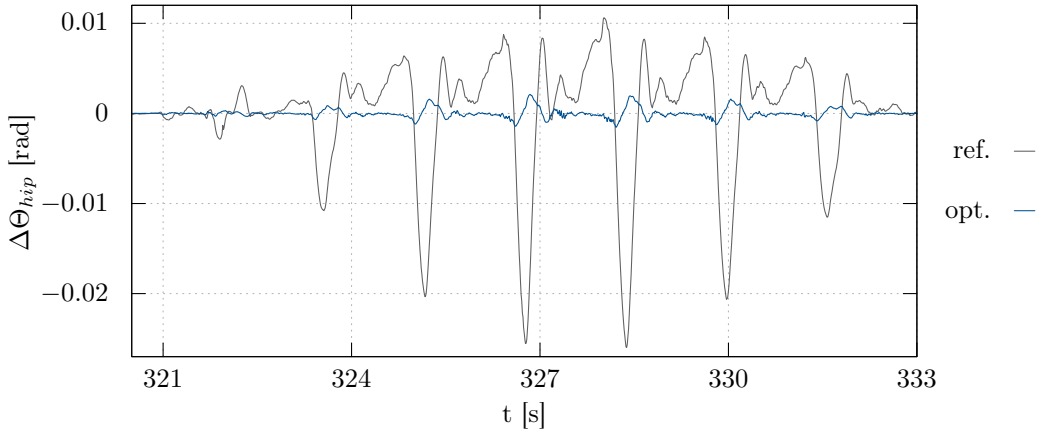


Figure 3.13: Hip flexion joint tracking performance evaluation for a fast walking experiment. With $k_{rl} = 0$ (ref.) and $k_{rl} = 1.3$ (opt.).

process. The SSV-map of Process 1 receives updates from the vision computer (Upd.SSV) with approx. 20 Hz. Inter-process communication is realized via shared memory. Process 1 writes new target values for the walking pattern of the next 10 time stamps into a First-In First-Out (FIFO) buffer based shared memory object which is read from process 2. Note that the time horizon of the buffer is half of the maximal admissible value of 20 that comes from the update rate of the trajectory adaptation. The buffer size is chosen such that it fits to the worst case computational time. A reduction of the size has the advantage that an updated trajectory comes earlier to the local control and consequently to the robot. The generated target data for current time stamp is written into the shared memory from the local control and it is read from the ETHERCAT process (ec-driver). The ETHERCAT bus runs with 4 kHz which is four times faster than the local control generates new target data. For timesteps where no new target data is available it interpolates the target joint positions and keeps the target velocities constant. The target data is finally sent to the joint controllers. The ec-driver writes updated sensor data to the shared memory which can be accessed from process 1 and 2.

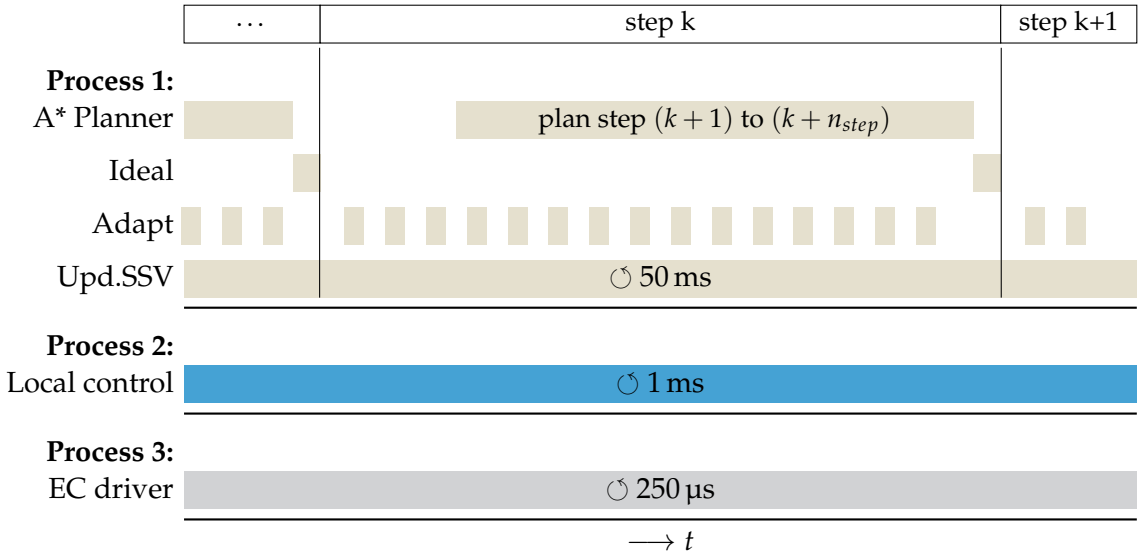


Figure 3.14: Overview of the three main processes of the walking control. The A*-based step planner runs parallel to the ideal trajectory planning and the trajectory adaptation in Process 1.

3.4 Chapter Summary

This chapter gave an overview of the mechanics, electronics and communication system of the bipedal robot LOLA. Further, the planning and control system is described and the integration of methods presented in this thesis are shown. This includes a short overview of the main parts of a model predictive trajectory adaptation: the prediction model, a state estimator and an algorithm to adapt trajectories (Figure 3.7). Finally details for the integration with collision avoidance methods, the real-time implementation and an improved joint feedforward control are given.

Chapter 4

Models for Real-Time Control

4.1 Introduction

Models are a mathematical description of a real system and are important to calculate the system's behavior. This chapter describes models that can be used in real-time control of bipedal walking. The main objectives for the models are (1) produce an accurate prediction of the robot's state evolution for a certain time horizon. Therefore underactuation and the ideal planned motion has to be taken into account. (2) Solve this long-term prediction in real-time. (3) Extendable by free variables to use them for trajectory optimization. Due to their properties it will be shown that presented models are also suitable for state estimation of the unactuated DoF of the robot. The classification into the model predictive trajectory adaptation is shown in Figure 4.1.

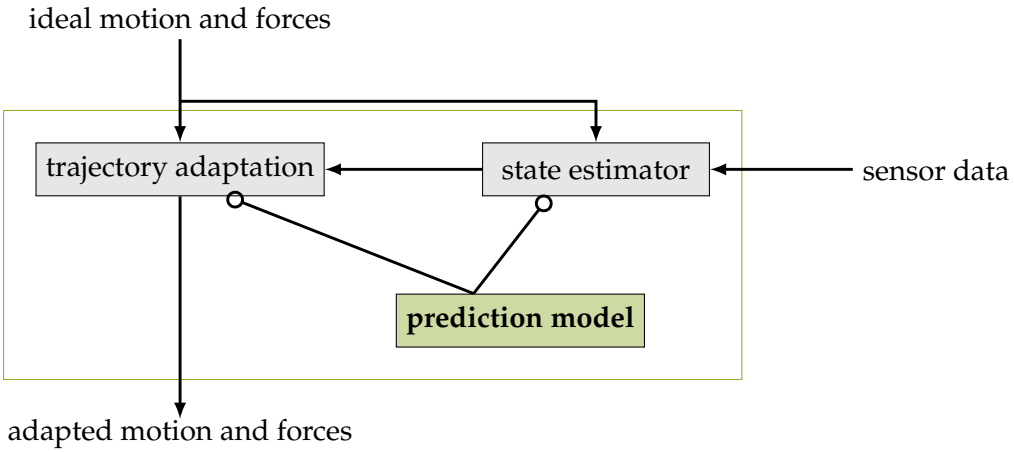


Figure 4.1: Chapter classification into the model predictive trajectory adaptation.

At the beginning a review of models for bipedal robots is given. They are discussed concerning their assumptions and accuracy. In the following sections the prediction model developed and used in this work is presented. Next to the basic concept there are two main variants are described that differ by the foot model. The models are verified using a model order reduction technique showing that the dominating error dynamics of a full rigid multibody model of the biped is included in the model. In addition prediction results are compared to simulation and measurement data in order to verify the stated model. Two model extensions are presented which introduce free inputs that can be modified with optimization methods, namely a modification of the swing foot trajectory and a modification of the CoG trajectory. The presented models are used to formulate a state estimator in Chapter 5 and real-time model predictive control in Chapter 6.

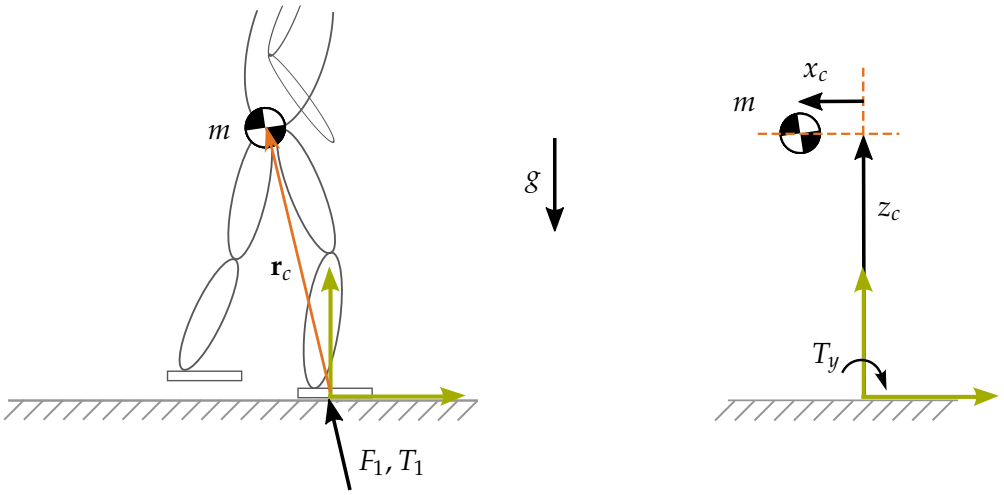


Figure 4.2: Left: global forces acting on a robot. Right: the LIPM.

4.2 Related Work

A model that is often used in real-time planning and control of long-term motion for humanoid robots with stiff position control is the LIPM. The dynamics in sagittal and lateral direction is considered to be decoupled and the biped is modeled with a single point mass located at the CoG. Additionally, multibody effects are neglected (Englsberger and Ott 2012; Mayr et al. 2012) which will be explained in the following. Stating the overall dynamics of a biped depicted in Figure 4.2, the total change of linear ($\dot{\boldsymbol{p}}$) and angular momentum ($\dot{\boldsymbol{L}}_{CoG}$) about the CoG is

$$\dot{\boldsymbol{p}} = m\boldsymbol{a}_C = m\boldsymbol{g} + \sum_{i=1}^{2} \boldsymbol{F}_i \tag{4.1}$$

$$\dot{\boldsymbol{L}}_{CoG} = \sum_{i=1}^{2} (\boldsymbol{T}_i - \boldsymbol{r}_{F_i,CoG} \times \boldsymbol{F}_i) = \sum_{i=1}^{n_{bodies}} m_i \boldsymbol{r}_{i,S_i} \times \boldsymbol{a}_i + \boldsymbol{I}_i^{0_i} \dot{\boldsymbol{\omega}}_i + \boldsymbol{\omega}_i \times \boldsymbol{I}_i^{0_i} \boldsymbol{\omega}_i \tag{4.2}$$

with the overall reaction force (F_i) and torque (T_i) acting on foot i. The vector from the i-th external force to the CoG is denoted by $\boldsymbol{r}_{F_i,Cog}$. The right hand side of (4.2) describes the angular momentum change due to multibody dynamics. The considered system consists of n_{bodies} bodies with the angular velocity $\boldsymbol{\omega}_i$, rotational inertia $\boldsymbol{I}_i^{0_i}$ about the body origin and $\boldsymbol{a}_i$ the acceleration of the respective body origin (Ulbrich 1996, p.59). Assuming that only foot 1 is in contact with the ground and inserting (4.1) into (4.2) yields for the first two rows

$$\dot{L}_{CoG,x} = T_x - m y_c(\ddot{z}_c + g) + m z_c \ddot{y}_c, \tag{4.3}$$

$$\dot{L}_{CoG,y} = T_y - m z_c \ddot{x}_c + m x_c(\ddot{z}_c + g). \tag{4.4}$$

The FoR can be set arbitrarily and is chosen to coincide with the force acting point which results in $\boldsymbol{r}_{F_1,CoG} = \boldsymbol{r}_c = [x_c, y_c, z_c]^T$. The assumption that leads to the LIPM (4.5) is to neglect change of angular momentum ($\dot{\boldsymbol{L}}_{CoG} = \boldsymbol{0}$). This correlates with omitting multibody effects (cf. (4.2)). The work presented in Kuindersma et al. (2015) includes the described multibody dynamics in their motion planning at the expense of having no real-time solution for the resulting problem.

Furusho and Masubuchi (1987) analyze in their work a five link bipedal planar model with point feet and one passive DoF. Assuming a stiff position control they show that the two slowest eigenvalues of the closed loop system correspond to the eigenvalues of

the center of mass dynamics. These eigenvalues correspond to the dynamics of the LIPM which represents consequently the dominant subsystem of a (stiff position controlled) bipedal robot. Those two eigenmodes are also described in the work of Takenaka et al. (2009a) where the pendulum dynamics is split into a convergent and a divergent part (cf. Subsection 2.3.1). In their work, Kajita et al. (1990) apply the pendulum model to produce potential energy conserving orbits which are used as CoG trajectories for a bipedal robot. Kajita et al. (2001) present a combination of the LIPM as minimal model of a biped with the existing ZMP concept. They use the ZMP as input for the pendulum and add a model predictive control to follow a predefined ZMP reference trajectory. This is used to plan CoG trajectories over a certain time horizon (several steps of the robot). There are numerous works that follow this idea and use the same linear model, e.g. Englsberger et al. (2011), Löffler et al. (2002), Morisawa et al. (2007), Stephens and Atkeson (2010), and Wieber (2006).

There exist several publications that extend the LIPM to increase the accuracy. Kajita et al. (2010) present a first order delay approximation for the controlled ZMP. This is used to design a state feedback control law for the humanoid HRP-4C. Pratt et al. (2006) added a flywheel with an additional rotational DoF to the LIPM including also constraints for the rotation. This idea is also used in Takenaka et al. (2009c) to build a two-staged feedback control of the robot. An extension that aims to increase the accuracy of the LIPM especially for fast walking is to use additional point masses located at the feet (Buschmann et al. 2007; Takenaka et al. 2009a). Buschmann et al. also removes the fixed CoG height from the trajectory planning problem. Under the premise that the foot and CoG height trajectories are known this results in a linear but time-varying differential equation. There exist planning methods that use the pendulum model with a varying CoG height in their planning problem (Tajima and Suga 2006). In Kajita et al. (2014, pp.152-153) the authors add an unactuated rotation angle to the linear pendulum model resulting in a nonlinear model. They use the obtained model to verify the ZMP controller concept by showing that the rotation angle can be controlled via the ZMP. This means for the real robot that an undesired inclination error which is modelled by the unactuated rotation angle is controllable.

The authors of Westervelt et al. (2003) and Westervelt et al. (2007) introduce planar two to five link models with rigid bodies. The contact between the robot and the ground is modelled with a point contact and an unactuated DoF. Impacts are assumed to be perfectly inelastic, which results in an instantaneous double support phase. Additionally they neglect slippage and rebound on impact. These optimistic assumptions simplify a stability analysis of control methods. They name the resulting model with impulse effects *Hybrid Zero Dynamics*. Another model that is mainly used for running or hopping robots is the Spring Loaded Inverted Pendulum Model (SLIPM). This model has two compliant legs and the angle of the swing leg is considered as control input. Raibert (1986) used the model to verify their effective control heuristics on several hopping robots.

Discussion of Model Accuracy

A conclusion that can be drawn is that for online motion planning of fully actuated bipedal robots and long term stabilization mainly the LIPM is used. In some cases it is used with extensions. In contrast to its popularity the prediction accuracy can be quite poor as can be seen in Figure 4.3. At time t_0 the current disturbed state of the robot is used to perform a prediction of the full MBS and the LIPM for three different disturbance cases (state disturbances result from external pushes with a peak force of 130-160 N). The MBS is the full multibody model of the robot with unilateral contacts and the feedback control described in Subsection 3.2.3. The LIPM (mass m, CoG position x_c and height z_c)

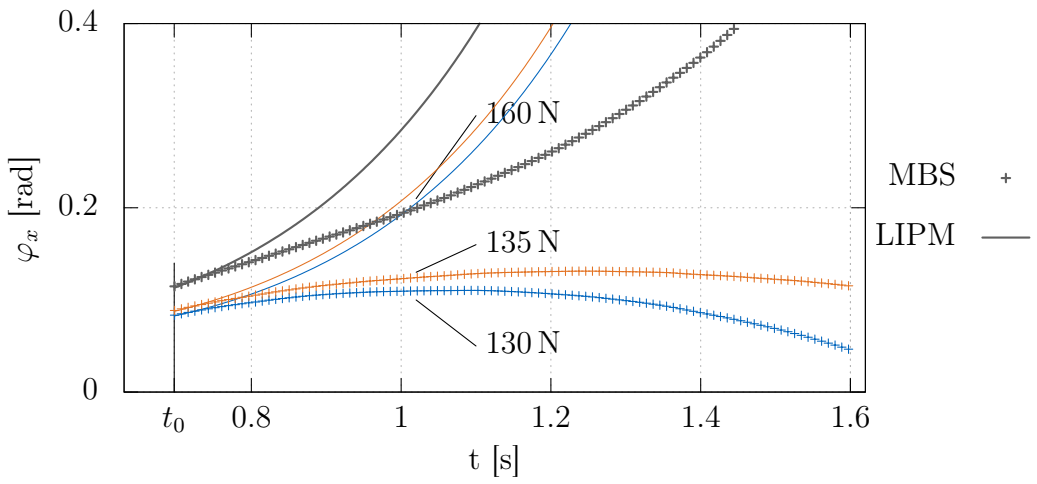


Figure 4.3: Upper body inclination in saggital plane for three different disturbance cases. Comparison of the state evolution for the full MBS and the LIPM. At $t = t_0$ the pendulum state is initialized with the measurements of $\varphi_x(t_0)$, $\dot{\varphi}_x(t_0)$.

is of the form

$$\ddot{x}_c = \frac{g}{z_c} x_c + \frac{1}{m z_c} T_{id}. \tag{4.5}$$

The ideal desired contact moment T_{id} is set by trajectory generation using e.g. the LIPM or – on LOLA – a three-mass model. It can be seen that the pendulum has a completely different state evolution compared to the full MBS of the robot. It predicts divergence for cases where the robot remains stable and overestimates divergence when the robot becomes unstable. Even an additional saturated PD-type feedback control (which is normally not included in the model) increases the prediction accuracy only lightly which will be shown in the later part of this chapter. This is the main motivation for the new prediction model presented in this work.

4.3 Proposed Model

4.3.1 Two Degrees of Freedom Prediction Model

In the following a description of the prediction model is given based on the work originally published in Wittmann et al. (2014). The main motivation for the proposed model is based on the observation that the LIPM does not reliably predict the robot's behavior under large disturbances (also with model extensions like flywheel, torque controller, etc.). It is observed that the model has to include the unactuated DoFs between robot and ground and that the unilateral and compliant contacts need to be properly accounted for. The planned CoG and foot trajectories are assumed to be perfectly tracked in the robot's planning FoR which rotates with the robot but does not translate in the horizontal plane (refer to Subsection 3.2.4 for the definition of coordinate systems). This is motivated by the fact that stiff position controlled bipedal robots are considered in this work. It is an adaptation of the idea in Furusho and Masubuchi (1987) to include only the slowest unknown parts of the MBS dynamics in the prediction. In this context unknown means that the real behavior deviates from the ideal planned motion. A model with these properties allows to predict the slow dynamics into the future, considering the ideal trajectories. In other words one can evaluate the system's behavior for the current sensed state and the planned motion and whether the state will diverge or not. Unknown disturbances that

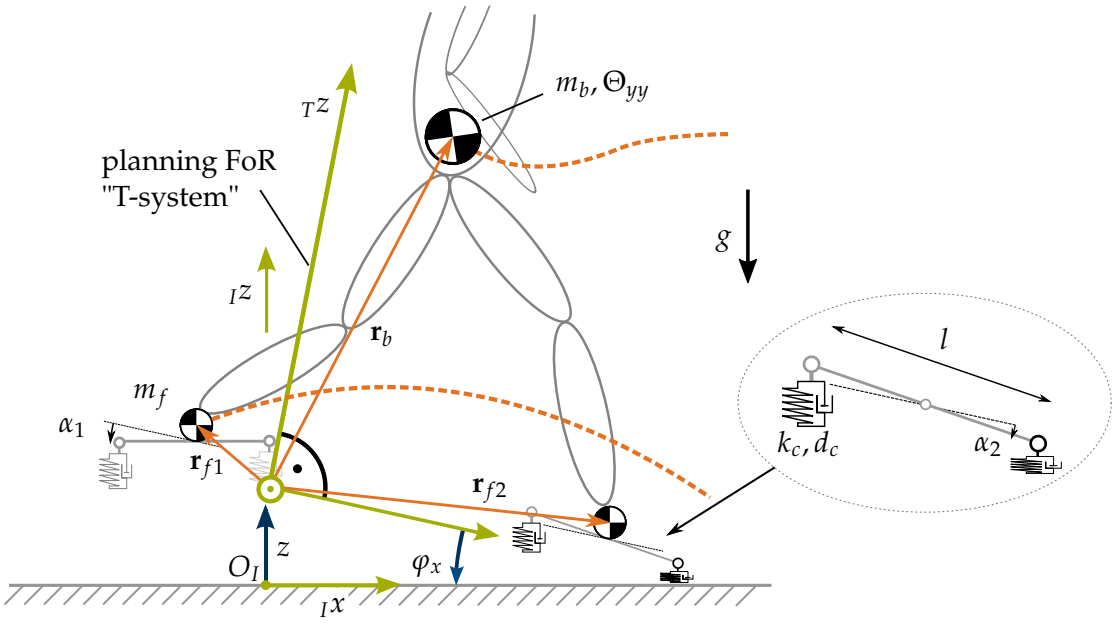


Figure 4.4: Prediction model for the x-z plane.

cause an inclination error of the robot can be mapped to the model as initial values. Using a two DoF model instead of the full multibody model ensures that this can be done in real-time for a sufficient long time horizon (approx. 1-2 s).

In the following the model description is related to the x-z plane shown in Figure 4.4 and can be formulated similarly for the y-z plane. The unactuated DoFs are the inclination φ_x about the y-axis and the vertical translation z. They describe the transformation from an inertial (index I) to a FoR rotating with the robot (index T). The translation in x direction is neglected to reduce computational time. A vector $\boldsymbol{r}_P$ written in the T-system coordinates can then be transformed into the I-system with

$$ {}_I\boldsymbol{r}_P = \begin{bmatrix} 0 & 0 & z \end{bmatrix}^T + \boldsymbol{A}_y(-\varphi_x)\, {}_T\boldsymbol{r}_P \,. \tag{4.6} $$

The matrix $\boldsymbol{A}_y(a)$ describes a rotation about the y-axis with the angle a. Furthermore, the trajectories for the robot's body mass $\boldsymbol{r}_b$[1] and the feet $\boldsymbol{r}_{f1}$, $\boldsymbol{r}_{f2}$ correspond to the ideal planned trajectories from the walking pattern generation (Figure 3.4). Based on the idea that all joints perfectly track the ideal motion, those time-dependent vectors are given in the robot FoR $({}_T\boldsymbol{r}_b\,(t),\, {}_T\,\boldsymbol{r}_{f1}(t),\, {}_T\boldsymbol{r}_{f2}\,(t))$. The mass m_b is extended by a constant mean inertia of the multibody system Θ_{yy}. Each foot has a point mass m_f at its center, the foot length l and two point contacts located at the toe and the heel. The contacts are linear spring/damper pairs with stiffness k_c and damping d_c with the values identified from the real robot's rubber sole. They act always only in ${}_I z$ direction and have to be considered unilateral due to the properties of walking robots. Additionally this enables the model to predict a falling behavior. In order to take the effect of the hybrid position/force control into account, two actuated DoFs α_1, α_2 are added. They describe a rotation of the feet wrt. the T-system and serve as input set by a feedback control. Consequently the prediction also includes the stabilizing effect and is supposed to be closer to the real behavior of the robot.

The EoM for the above described system are derived using the Lagrange's equation of the second kind

$$ \left[\frac{\mathrm{d}}{\mathrm{d}t} \left(\frac{\partial\, E_{kin}}{\partial\, \dot{\boldsymbol{q}}} \right) - \frac{\partial\, E_{kin}}{\partial\, \boldsymbol{q}} + \frac{\partial\, E_{pot}}{\partial\, \boldsymbol{q}} \right]^T = \boldsymbol{Q}_{NC} \tag{4.7} $$

[1] The body mass position is calculated such that it is equal to the robot's CoG position excluding the foot masses.

introducing the generalized coordinates $\boldsymbol{q} = [z,\, \varphi_x]^T$, the system's kinetic and potential energies E_{kin}, E_{pot} and the non-conservative forces denoted by $\boldsymbol{Q}_{NC}$. The two angles $\boldsymbol{\alpha} = [\alpha_1,\, \alpha_2]$ are treated in the following as time-varying but known quantities. The control law and its integration into the mechanical system are described in Subsection 4.3.2. The kinetic energy is composed of the sum of the proportion of all point masses and of the rotational inertia. Defining the set of all bodies $N_B = \{b, f1, f2\}$, it yields

$$E_{kin} = \frac{1}{2}\left(\sum_{j \in N_B} m_j \dot{\boldsymbol{r}}_j^2 + \Theta_{yy}\dot{\varphi}_x^2\right). \tag{4.8}$$

The computation of the velocity for a point mass j in the planar case uses the given time varying vector ${}_T\boldsymbol{r}_j\,(t) = [x_j(t),\, 0,\, z_j(t)]^T$ and the relationship (4.6) to calculate the absolute position

$${}_I\boldsymbol{r}_j\,(\boldsymbol{q}, t) = \begin{bmatrix} x_j(t)\cos\varphi_x + z_j(t)\sin\varphi_x \\ 0 \\ z - x_j(t)\sin\varphi_x + z_j(t)\cos\varphi_x \end{bmatrix}, \qquad \forall j \in N_B \tag{4.9}$$

that can be derived wrt. time. The potential energy is composed of two parts one due to gravity and another part due to contact deformation. Note that each contact state can be active or inactive depending on the system's state and the contact states change over time. The vertical deformation and relative velocity for one contact i of foot k is denoted by

$$\triangle z_i(\boldsymbol{q}, t) = z - x_{fk}(t)\sin\varphi_x + z_{fk}(t)\cos\varphi_x + (\text{-}1)^i\frac{l}{2}\sin(\alpha_k + \varphi_x) - z_0, \tag{4.10}$$

$$\begin{aligned}\triangle \dot{z}_i(\boldsymbol{q}, \dot{\boldsymbol{q}}, t) =& \dot{z} - \dot{x}_{fk}(t)\sin\varphi_x - x_{fk}(t)\dot{\varphi}_x\cos\varphi_x + \dot{z}_{fk}(t)\cos\varphi_x \\ & - z_{fk}(t)\dot{\varphi}_x\sin\varphi_x + (\text{-}1)^i\frac{l}{2}(\dot{\alpha}_k + \dot{\varphi}_x)\cos(\alpha_k + \varphi_x)\end{aligned} \tag{4.11}$$

with the contacts assigned according to Table 4.1 and the unstressed spring length z_0.

Table 4.1: Contact assignment.

i	foot variables	name
1/2	x_{f1}, α_1	right toe/right heel
3/4	x_{f2}, α_2	left toe/left heel

The time-dependent quantities $x_{f1/2}, z_{f1/2}$ are the planned foot trajectories evaluated at a certain time t. This is important as this introduces an up and down movement of the feet during walking. This enables the model to follow state transitions that are similar to the ones of the real robot. Additionally this introduces the possibility to evaluate a modification of the swing foot trajectories $\boldsymbol{r}_{fs}$ in the presence of disturbances. Naming the set of all active contacts N_C, the total potential energy of the system can be written as

$$E_{pot} = \frac{1}{2}\sum_{i \in N_C} k_{ci}(\triangle z_i)^2 + \sum_{j \in N_B} m_j\, {}_I\boldsymbol{r}_j^T\, \boldsymbol{g}. \tag{4.12}$$

The non-conservative forces for all contacts are

$$\boldsymbol{Q}_{NC} = \sum_{i \in N_C}\left(\frac{\partial \triangle z_i}{\partial \boldsymbol{q}}\right)^T(-d_{ci}\triangle \dot{z}_i). \tag{4.13}$$

Finally (4.7) can be evaluated with (4.8) - (4.13). Simplifying and rearranging results in the overall EoM of the prediction model

$$\begin{bmatrix} m & -\overline{mx}c_\varphi - \overline{mz}s_\varphi \\ -\overline{mx}c_\varphi - \overline{mz}s_\varphi & \overline{m(x^2+z^2)} + \Theta_{yy} \end{bmatrix} \begin{bmatrix} \ddot{z} \\ \ddot{\varphi}_x \end{bmatrix} + \begin{bmatrix} mg + \overline{m\ddot{z}}c_\varphi - \overline{m\ddot{x}}s_\varphi - \overline{mz}\dot{\varphi}_x^2 c_\varphi \\ \overline{mx}\ddot{z} + \overline{mz}\ddot{x} + 2\overline{mx}\dot{x}\dot{\varphi}_x \\ +\overline{mx}\dot{\varphi}_x^2 s_\varphi - 2\overline{m\dot{x}}\dot{\varphi}_x c_\varphi - 2\overline{m\dot{z}}\dot{\varphi}_x s_\varphi \\ +2\overline{mz}\dot{z}\dot{\varphi}_x - \overline{mx}gc_\varphi - \overline{mz}gs_\varphi \end{bmatrix} = \sum_{i \in N_C} \left(\frac{\partial \triangle z_i}{\partial \boldsymbol{q}} \right)^T F_{zi} \tag{4.14}$$

with the abbreviations

$$\overline{ma} = \sum_{j \in N_B} m_j a_j, \quad F_{zi} = -d_{ci}\triangle \dot{z}_i - k_{ci}\triangle z_i, \quad s_\varphi = \sin\varphi_x, \quad c_\varphi = \cos\varphi_x.$$

They are written in short form

$$\boldsymbol{M}(\boldsymbol{q},t)\ddot{\boldsymbol{q}} + \boldsymbol{h}(\boldsymbol{q},\dot{\boldsymbol{q}},t) = \boldsymbol{\lambda}(\boldsymbol{q},\dot{\boldsymbol{q}},t) \tag{4.15}$$

where $\boldsymbol{\alpha}(t)$ is treated as time varying parameter. The state of each unilateral contact (open or closed) has to be resolved continuously as it depends on the system state and the time-dependent foot trajectories. A positive deformation (wrt. the unstressed spring length z_0) or negative force result in an open contact state and otherwise in a closed state. This check has to be done in every timestep when the model is solved numerically. This and the integration of a force control is part of the next sections.

4.3.2 Controlled Model

To increase the prediction accuracy, the effect of the inertial stabilization and force control has to be included in the model in an appropriate way. With the additional foot angle $\boldsymbol{\alpha}$ it is possible to build a similar controller as the one used for the real robot. The control input is the inclination error $\Delta\varphi_x = \varphi_{x,d} - \varphi_x$ ($\varphi_{x,d}$ is the desired upper body inclination) of the upper body which is strongly related to the inclination error of the overall robot wrt. the world. The control output is a modification of the foot orientation relative to the robot. The control law is divided into two parts to realize that. First a modification of the ideal contact moment T_{id} is calculated using a PD-type control law

$$T_d = T_{id} + K_p\Delta\varphi_x + K_d\Delta\dot{\varphi}_x. \tag{4.16}$$

This contact moment is projected to a feasible region corresponding to the location and length of the feet, that is $T_d \in [T_{min}\, T_{max}]$. In a second step the force controller modifies the foot angles in order to control the contact torque by using a model for the interaction between foot and ground. According to the planned load factors ρ_{f1} and ρ_{f2} the desired torque for each leg is calculated. With the control gain K_F, the desired reaction torque $T_{d,i} = \rho_{fi}T_d$ and the currently acting torque $T_i(\boldsymbol{q},\dot{\boldsymbol{q}})$ for foot i, linear dynamics for the tracking error $\triangle T_i = T_{d,i} - T_i$ is chosen

$$\triangle\dot{T}_i + K_F\triangle T_i = 0 \tag{4.17}$$

that fades asymptotically to zero (for $K_F > 0$). Inserting $\dot{T}_i = \frac{\partial T_i}{\partial \alpha_i}\dot{\alpha}_i$ and assuming $\dot{T}_{d,i} = 0$ yields the control law $\boldsymbol{u}_c$ for the foot rotations

$$\dot{\boldsymbol{\alpha}} = \alpha_\lambda \left(\frac{\partial T}{\partial \alpha} \right)^{-1} K_\lambda \begin{bmatrix} \rho_{f1}T_d - T_1 \\ \rho_{f2}T_d - T_2 \end{bmatrix} + \alpha_x K_x \left[\boldsymbol{\alpha}_d - \boldsymbol{\alpha}\right] = \boldsymbol{u}_c. \tag{4.18}$$

(4.18) is extended by a second part that presents an additional position control. This

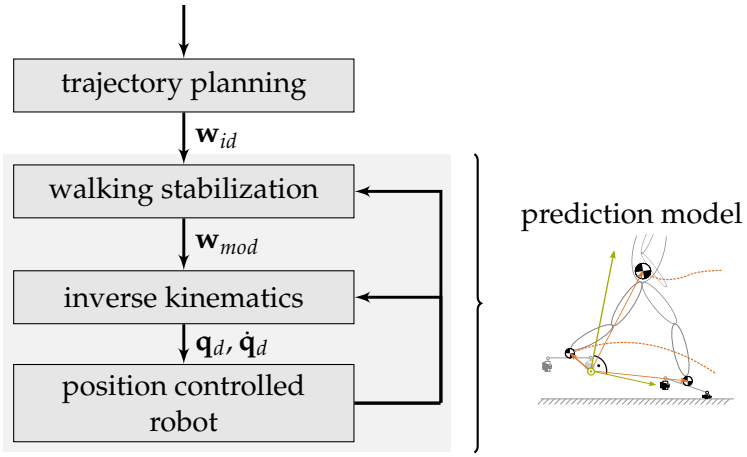


Figure 4.5: Walking control system and the corresponding part that is included in the prediction model.

is necessary since the swing foot is not in contact with its environment and it has to be ensured that the angle α_i of the swing leg returns to its desired value. The factors α_λ and α_x activate the force and position control for the stance and swing leg according to their current state. This implementation is a simplified but similar implementation of Buschmann et al. (2009). Therefore the control gains K_p, K_d, K_λ and K_x are equal to the ones used for the real robot. Introducing the extended state vector $\boldsymbol{z} = [\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{\alpha}]^T$ the equations (4.15), (4.18) can be written as a set of differential equations of order one

$$\dot{\boldsymbol{z}} = \begin{bmatrix} \dot{\boldsymbol{q}} \\ \boldsymbol{M}^{-1}\left[\boldsymbol{\lambda} - \boldsymbol{h}\right] \\ \boldsymbol{u}_c \end{bmatrix} := \boldsymbol{f}_{full}(\boldsymbol{z}, t). \tag{4.19}$$

This system approximates the overall controlled robot with its inertial stabilization, inverse kinematics and the joint control including also the interaction with the environment (cf. Figure 4.5). Additionally it includes parts of the planned trajectories and enables to evaluate the controlled robot's behavior for a given initial state and desired trajectories this way. Using finite sized feet in the model extends the possibilities to use it for several applications as well as to include terrain information in the prediction. Especially for long-term prediction a model simplification that reduces computational time by one third is proposed in the next section.

4.3.3 Reduced Controlled Model

The model introduced in Subsection 4.3.1 is simplified in order to increase the computation speed which is especially for an real-time MPC application necessary. The feet with two point contacts and foot length l are replaced by point feet (cf. Figure 4.6). This reduces the number of contacts to two and removes the additional variables $\boldsymbol{\alpha}$. The force control (4.18) is removed and the desired stabilization torque of (4.16) is applied directly in a saturated form to the robot

$$T_{stab} = \begin{cases} T_{min} & \text{for } T_{stab} < T_{min} \\ T_{id} + K_p \triangle\varphi_x + K_d \triangle\dot{\varphi}_x & \text{for } T_{min} \leq T_{stab} \leq T_{max} \\ T_{max} & \text{for } T_{max} < T_{stab} \end{cases}. \tag{4.20}$$

This is equal to the assumption to perfectly track the desired torques. The torque limits T_{min}, T_{max} result from the geometry and position of the feet. The damping and stiffness

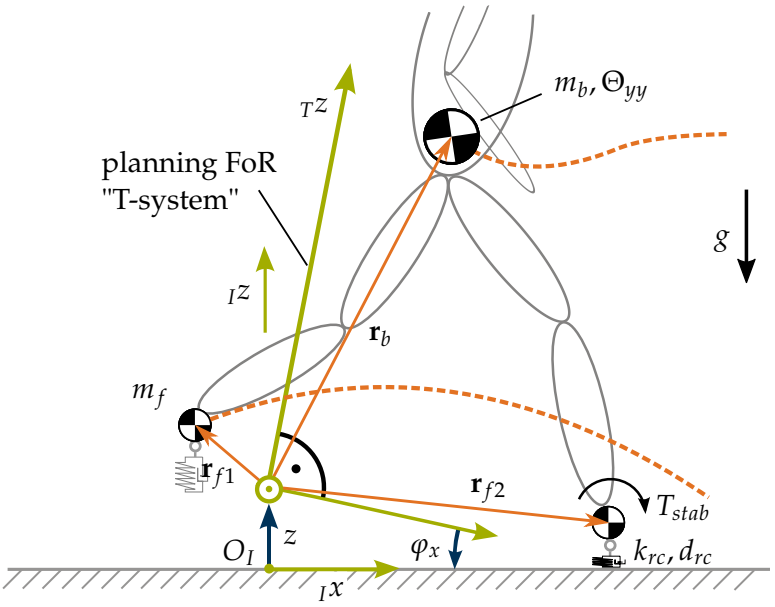


Figure 4.6: Reduced prediction model for the x-z plane.

of the reduced model k_{rc}, d_{rc} are twice of the values used in the full model in order to preserve the overall stiffness per leg. The vertical deformation for the remaining contacts $i \in \{1,2\}$ is

$$\triangle z_{r,i}(\boldsymbol{q},t) = z - x_{fi}(t)\sin\varphi_x + z_{fi}(t)\cos\varphi_x - z_0 \tag{4.21}$$

$$\triangle \dot{z}_{r,i}(\boldsymbol{q},\dot{\boldsymbol{q}},t) = \dot{z} - \dot{x}_{fi}(t)\sin\varphi_x - x_{fi}(t)\dot{\varphi}_x\cos\varphi_x + \dot{z}_{fi}(t)\cos\varphi_x - z_{fi}(t)\dot{\varphi}_x\sin\varphi_x \tag{4.22}$$

and $N_{C,r}$ describes the set of active contacts of the reduced model. The overall EoMs for the reduced model are denoted by

$$\begin{aligned}\boldsymbol{M}(\boldsymbol{q},t)\ddot{\boldsymbol{q}} + \boldsymbol{h}(\boldsymbol{q},\dot{\boldsymbol{q}},t) &= \sum_{i\in N_{C,r}} \left(\frac{\partial \triangle z_{r,i}}{\partial \boldsymbol{q}}\right)^T F_{zr,i} + [0,\ T_{stab}]^T \\ &= \boldsymbol{\lambda}_r(\boldsymbol{q},\dot{\boldsymbol{q}},t) + \boldsymbol{T}_s.\end{aligned} \tag{4.23}$$

The EoM (4.23) can be rewritten again as a set of differential equations of order one with the state vector $\boldsymbol{z}_r = [\boldsymbol{q},\ \dot{\boldsymbol{q}}]^T$

$$\dot{\boldsymbol{z}}_r = \begin{bmatrix} \dot{\boldsymbol{q}} \\ \boldsymbol{M}^{-1}[\boldsymbol{\lambda}_r + \boldsymbol{T}_s - \boldsymbol{h}] \end{bmatrix} := \boldsymbol{f}_r(\boldsymbol{z}_r,\ t). \tag{4.24}$$

4.3.4 Model Verification by Model Order Reduction

The previous sections introduced the prediction model using basic knowledge of the system and performing several assumptions. In the following a verification of those assumptions is shown. The Singular Value Decomposition (SVD) is used to analyze the main components of the motion perturbations computed on a full nonlinear model (Antoulas 2009, pp. 277ff). It will be shown that the two highest singular values of the full MBS during a disturbance in x direction are the absolute inclination and vertical translation between robot and ground. Those are the same as the chosen passive DoFs of the presented prediction models. The results have been presented in Wittmann and Rixen (2016).

A MBS simulation with joint control and imperfect joint angle tracking ($\boldsymbol{q}_{J,d} \neq \boldsymbol{q}_J$) is performed with the robot walking in place and receiving a disturbance force in x direction. The resulting errors in the generalized coordinates $\Delta\boldsymbol{q} \in \mathbb{R}^{n_q}$ are stored as time series data. Those coordinates include the tracking error of each joint angle $\Delta q_{J,i}$ and the error of the absolute pose of the robot wrt. the world. The translation and rotation error $(\Delta\boldsymbol{r}_{w,s}, \Delta\boldsymbol{A}_{w,s})$ between the stance foot and the world is chosen. The IMU rotation angles $(\varphi_x, \varphi_y, \varphi_z)$ (cf. Subsection 3.2.4) are calculated from the rotation error that produces finally a snapshot of the state error vector at time t_i

$$\Delta\boldsymbol{q}(t_i) = [\varphi_x,\ \varphi_y,\ \varphi_z,\ \Delta x_{w,s},\ \Delta y_{w,s},\ \Delta z_{w,s},\ \Delta q_{J,1},\ \ldots,\ \Delta q_{J,24}]^T(t_i). \tag{4.25}$$

The overall time series data is stored during numerical integration in a matrix $\boldsymbol{Y}$ with a discretization of 1 ms which showed to be sufficiently accurate for this purpose. An additional matrix $\boldsymbol{Q}_{\text{norm}}$ is used to convert the vector $\Delta\boldsymbol{q}$ into a dimensionless quantity $\Delta\bar{\boldsymbol{q}} = \boldsymbol{Q}_{\text{norm}}\Delta\boldsymbol{q}$ which is nessessary due to its different measurement units

$$\boldsymbol{Y} = [\Delta\bar{\boldsymbol{q}}(t_0),\ \Delta\bar{\boldsymbol{q}}(t_1),\ \ldots,\ \Delta\bar{\boldsymbol{q}}(t_n)] \quad \in \mathbb{R}^{n_q \times m}. \tag{4.26}$$

In a second step, the SVD[2] is applied to the matrix $\boldsymbol{Y}$

$$\boldsymbol{Y} = \boldsymbol{U}\Sigma\boldsymbol{V}^T = \begin{bmatrix}\boldsymbol{u}_1 & \ldots & \boldsymbol{u}_{n_q}\end{bmatrix} \begin{bmatrix}\sigma_1 & & & \\ & \ddots & & \boldsymbol{O} \\ & & \sigma_{n_q} & \end{bmatrix} \begin{bmatrix}\boldsymbol{v}_1^T \\ \vdots \\ \boldsymbol{v}_{n_q}^T \\ \vdots\end{bmatrix} \tag{4.27}$$

where the singular values σ_i are sorted in descending order. The vectors $\boldsymbol{u}_i$ describe the mapping from σ_i to the physical DoFs of the robot and $\boldsymbol{v}_i^T$ determine the distribution over all time samples. The resulting singular values are depicted in Figure 4.7. It can be seen that the first value is significantly higher than the remaining ones. Following the idea of model order reduction one can say that high singular values describe the most dominant part of the underlying dynamics. In order to connect the σ_i with the DoF of the robot's model the first and second vector $\boldsymbol{u}_1, \boldsymbol{u}_2$ are shown in Figure 4.8. The row number is shown on the x axis and the ordinate represents the value of each row. It can be seen that the first two singular values mainly belong to the absolute inclination error φ_x and vertical translation of the robot for a disturbance in x direction. The time series result φ_x for an approximation using only the first singular value $\boldsymbol{u}_1\sigma_1\boldsymbol{v}_1^T$ is shown in Figure 4.9. It can be seen that it produces an accurate estimate for the inclination angle. The result underlines the assumptions made for the proposed prediction model in the former section. A similar result can be obtained for φ_y during a disturbance in y direction. Consequently, if the aim is to predict the dominant error evolution of the robot's state $\boldsymbol{q}(t)$ one should consider at least the error dynamics for φ_x, φ_y, $\Delta z_{w,s}$ together with the ideal planned motion. Using two planar prediction models of Subsection 4.3.1 for x and y direction fulfills this requirement.

Relation to the Linear Inverted Pendulum Model

In addition to the above stated analysis using SVD, the prediction model (4.23) can be further simplified to receive the commonly in literature used LIPM. This is an important property of the model, because it shows that the prediction model (a) includes the dynamics of the pendulum model and (b) is a refinement thereof. The EoMs of (4.23) with

[2] SVD is computed with MATLAB.

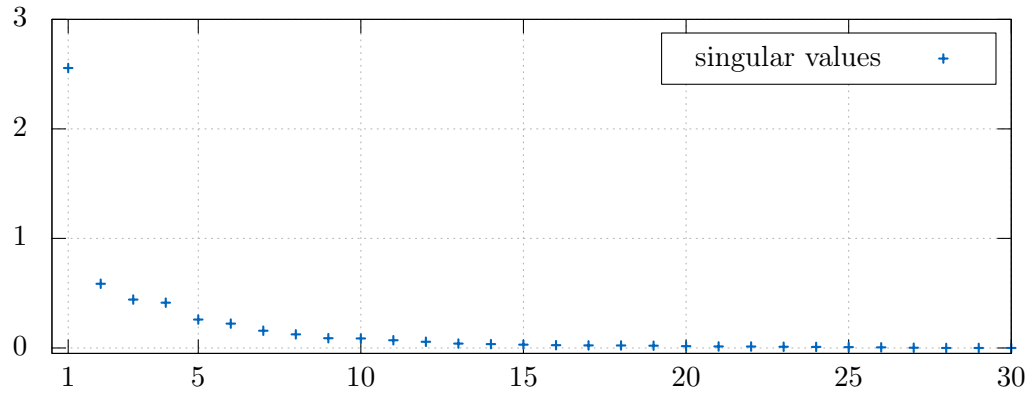


Figure 4.7: Singular values of the multibody time series data.

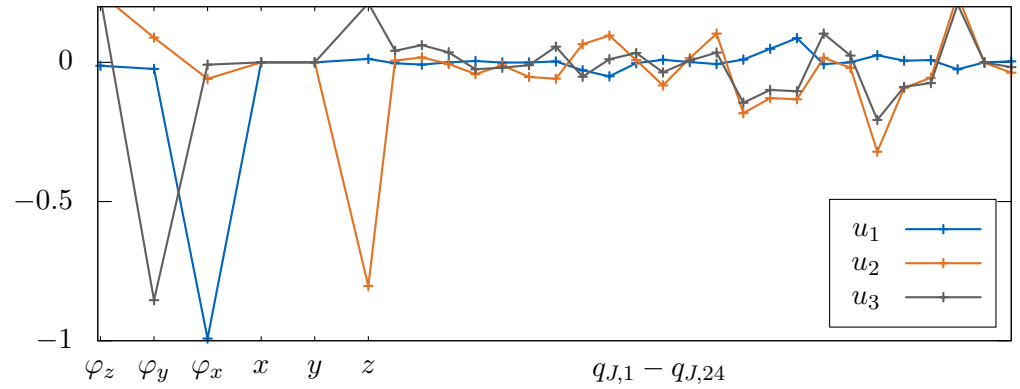


Figure 4.8: First, second and third vectors u_i for the applied SVD.

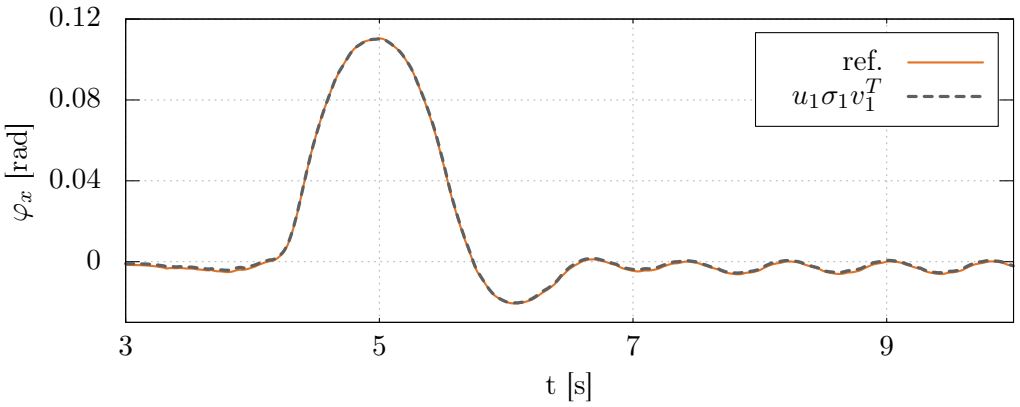


Figure 4.9: Comparison of reference data (ref.) and approximation with first singular value ($\boldsymbol{u}_1\sigma_1\boldsymbol{v}_1^T$) for the inclination φ_x. Only the row of the matrix $\boldsymbol{u}_1\sigma_1\boldsymbol{v}_1^T$ is shown that belongs to φ_x.

massless legs are denoted by

$$\begin{bmatrix} m & -mx_c c_\varphi - mz_c s_\varphi \\ -mx_c c_\varphi - mz_c s_\varphi & mx_c^2 + mz_c^2 + \Theta_{yy} \end{bmatrix} \begin{bmatrix} \ddot{z} \\ \ddot{\varphi}_x \end{bmatrix} + \\ m \begin{bmatrix} g + \ddot{z}_c c_\varphi - \ddot{x}_c s_\varphi - z_c \dot{\varphi}_x^2 c_\varphi + x_c \dot{\varphi}_x^2 s_\varphi - 2\dot{x}_c \dot{\varphi}_x c_\varphi - 2\dot{z}_c \dot{\varphi}_x s_\varphi \\ x_c \ddot{z}_c + z_c \ddot{x}_c + 2x_c \dot{x}_c \dot{\varphi}_x + 2z_c \dot{z}_c \dot{\varphi}_x - x_c g c_\varphi - z_c g s_\varphi \end{bmatrix} = \begin{bmatrix} F_z \\ T_\varphi \end{bmatrix} \tag{4.28}$$

where F_z describes the overall acting vertical force and T_φ the overall acting torque wrt. the defined coordinate origin O_I. Neglecting the rotational inertia Θ_{yy} and setting $z = \dot{z} = \ddot{z} = 0$ simplifies the second row of (4.28) to

$$(x_c^2 + z_c^2)\ddot{\varphi} + x_c \ddot{z}_c + z_c \ddot{x}_c + 2x_c \dot{x}_c \dot{\varphi} + 2z_c \dot{z}_c \dot{\varphi} - x_c g c_\varphi - z_c g s_\varphi = \frac{T_\varphi}{m}. \tag{4.29}$$

The resulting EoM are the same as the *"cart-table model with free rotating joint"* presented in Kajita et al. (2014, p.152). If the CoG height is kept constant and the DoF φ is removed ($\varphi = \dot{\varphi} = \ddot{\varphi} = 0$) the well known EoM of the LIPM arise from (4.29)

$$\frac{z_c}{g}\ddot{x}_c - x_c = \frac{T_\varphi}{mg} \quad \Leftrightarrow \quad \ddot{x}_c = \frac{g}{z_c}(x_c - x_{cop}) \tag{4.30}$$

with the CoP location in x direction

$$x_{cop} = -\frac{T_\varphi}{mg}. \tag{4.31}$$

Revisiting the performed simplifications one can say that the inverted pendulum model is extended by two passive DoF and changed the CoG motion to be predetermined (so far). Further, the assumptions like flat foot walking on a horizontal ground resulting from the ZMP criteria (and also approximate solutions for inclined surfaces, see Sardain and Bessonnet (2004) for a review) are no longer necessary as the contact of the feet is considered explicitly and resolves the resulting contact forces and their effect to the dynamics of the CoG. This is an important property as it enables the model to be easily applied to rough terrain situations or to multi-contact problems (e.g. support with arms on a wall) by modifying the terrain information or adding additional contact points.

4.3.5 Numerical Solution

This section deals with the numerical solution of the above stated EoM of the model. They are nonlinear, time-variant and including changing contact states. Consequently the numerical solution method has to account for those properties and has to provide a sufficiently accurate solution. Sufficient means to be able to predict the trend of the robot's state qualitatively. For this sake the EoM (4.19) and (4.24) are solved with different integrator setups. An explicit Runge-Kutta solver with tight tolerances is used to produce a reference solution. It is compared to the same solver with coarse tolerances and a forward Euler integration scheme with two different but fixed integrator time steps. The detailed setup with the corresponding abbreviation is:

- Explicit Runge-Kutta (2,3) with variable time-step (Bogacki and Shampine 1989) using an absolute tolerance of 10^{-7} and a relative tolerance of 10^{-3} (*ode23-f*)
- Explicit Runge-Kutta (2,3) with variable time-step using an absolute tolerance of $2 \cdot 10^{-4}$ and a relative tolerance of $2 \cdot 10^{-2}$ (*ode23-c*)
- Explicit Euler with fixed integrator step size $\triangle t_e = 10^{-3}$ s (*euler-f*)

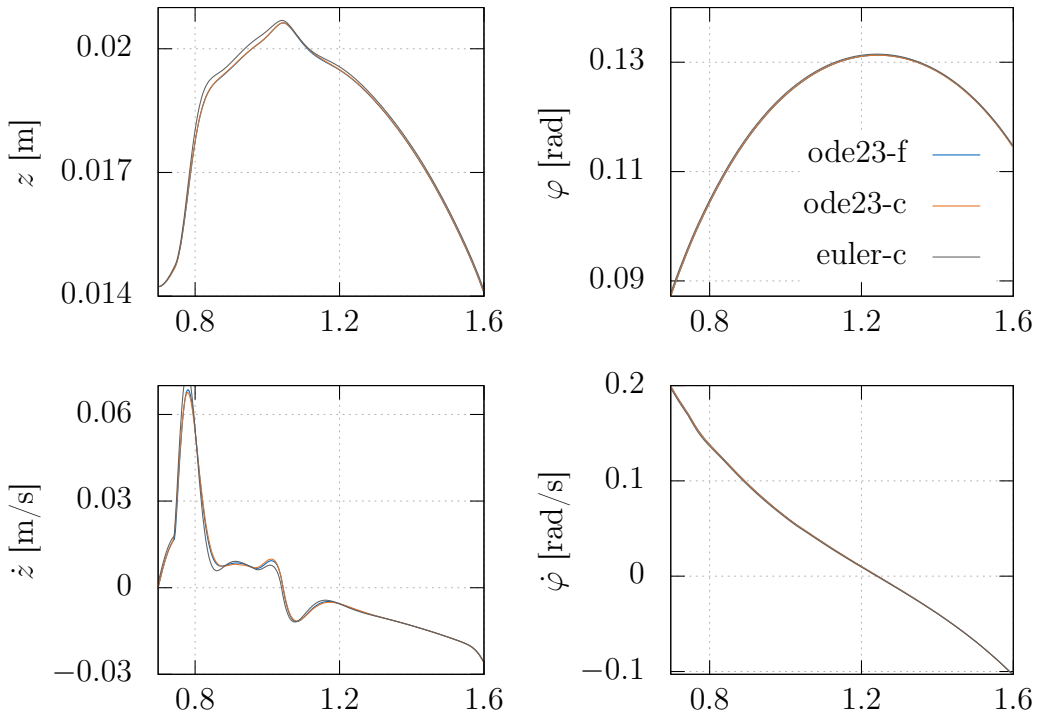


Figure 4.10: State evolution for different integrator setups.

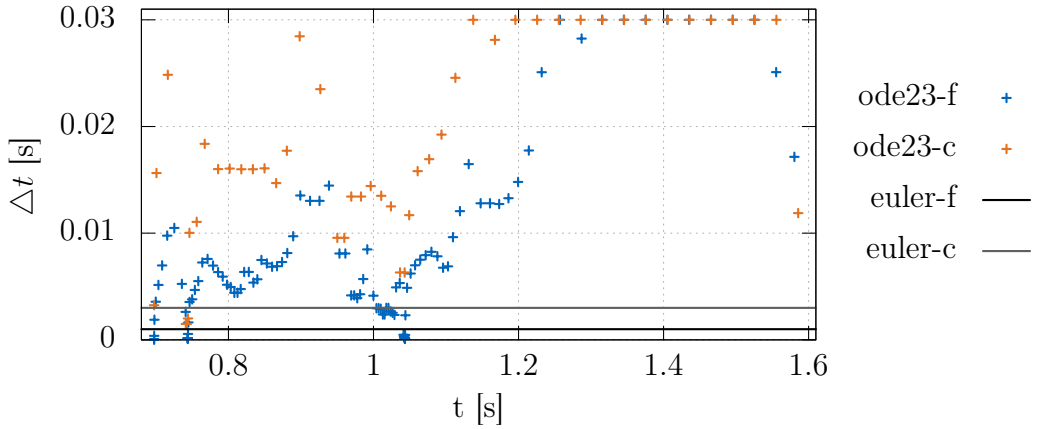


Figure 4.11: Comparison of intergrator time steps $\triangle t$ for fine and coarse explicit Runge-Kutta integrators. The explicit Euler integrator time steps are shown as well.

- Explicit Euler with fixed integrator step size $\triangle t_e = 3 \cdot 10^{-3}$ s (*euler-c*).

These four integrators solve the full model (4.19) for a given disturbed initial state $\boldsymbol{z}(t_0)$ and the results are compared to each other. Ideal trajectories for the CoG and the feet are the same for all integrations and the foot masses are set to zero. The state evolution for the fine and coarse explicit Runge-Kutta and the coarse explicit Euler are depicted in Figure 4.10. It can be seen that the resulting state trajectories are similar, especially for the fine and coarse tolerance setup of the ode23. The coarse explicit Euler shows small deviations primarily in the state $z(t)$. Nevertheless all solvers produce qualitatively the same result.
To have a better comparison between the different results the *ode23-f*-solution is taken as

reference and used for the other solutions to compute the RMSE

$$e_{rmse} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\boldsymbol{q}(t_i) - \boldsymbol{q}_{ref}(t_i))^2 + (\dot{\boldsymbol{q}}(t_i) - \dot{\boldsymbol{q}}_{ref}(t_i))^2}. \tag{4.32}$$

Therefore all solutions are interpolated to match the *euler-c* discretization. The results are listed in Table 4.2. For both models the order from best to worst RMSE is: *ode23-c* > *euler-f* > *euler-c* [3]. Beside to this result we analyzed the integrator step sizes $\triangle t$ for the ode23's which are shown in Figure 4.11. The step size of both Euler integrators are included as well. Comparing them to the adaptive time steps one can see that the *ode23-f* uses only at three time instants smaller time-steps than the *euler-f*. The *euler-c* achieves almost the same tolerances as the *ode23-c*.

Table 4.2: Integration errors compared to *ode23-f* (RMSE).

model	ode23-c	euler-f	euler-c
full	0.000707	0.000708	0.002171
simple	0.007508	0.006175	0.025704

To conclude this analysis it can be said that all four studied numerical integrators produce a similar result especially for the inclination $\varphi(t)$. This is underlined by a comparison of the adaptive and fixed step sizes where the adaptive ones are most of the time higher than the fixed ones. The analysis gives a good indication how to choose a fixed step size for a given tolerance. The result is used to apply the proposed model for real-time model predictive control where computation time is crucial. An explicit Euler integration scheme with a time step size of $\triangle t = 1 - 3\,\text{ms}$ produces sufficient accurate solutions for the model. The computation time for solving the reduced model with a time horizon of 0.8 s is approx. 80 µs[4].

4.3.6 Prediction Accuracy – Results

Predicting the state into the future with either the model (4.19) or (4.24) requires an initial value $\boldsymbol{z}(t_0)$ resp. $\boldsymbol{z}_r(t_0)$ that corresponds to the robot's state. Its absolute inclination and inclination rate are used for the initial values of $\varphi_x(t_0)$, $\dot{\varphi}_x(t_0)$. For the final implementation, a state estimator is used, which is described in Chapter 5 to improve the initial value quality. The values for $\boldsymbol{\alpha}(t_0)$ and the CoG and feet positions are set to current desired values. The absolute vertical translation $z(t_0)$ and its derivative can not be measured at the real system and therefore have to be calculated. The standard approach is to set them in such a way, that the EoM are met. It has been shown that it is sufficient to ensure a static equilibrium which corresponds to the assumption that the robot is in an almost upright pose and the inclination rate produces a velocity that points approximately only in x direction. Additionally changes in the active contact set N_C resp. $N_{C,r}$ have to be considered in the method of finding a $z(t_0)$ for the used full model

$$mg = \sum_{i \in N_C} F_{zi}(t_0) = \sum_{i \in N_C} k_{ci}\Delta z_i(\boldsymbol{q}, t_0) \tag{4.33}$$

or for the reduced model

$$mg = \sum_{i \in N_{C,r}} k_{rci}\Delta z_{r,i}(\boldsymbol{q}, t_0). \tag{4.34}$$

[3] „>“ means in this context „is better than“

[4] Computations are done with an Intel i7-3770K @3.5 GHz and 8 Gb RAM.

Algorithm 1 Model Initialization (Reduced Model)

1: Set $\varphi_x(t_0)$, $\dot{\varphi}_x(t_0)$
2: Set $\boldsymbol{r}_{fi}(t_0)$, $\boldsymbol{r}_b(t_0)$
3: $z(0) \leftarrow z_0 - \frac{mg}{2k_{rc}}$
4: $n_c \leftarrow -1$
5: **while** $n_c \neq n_{c,last}$ **do**
6: $\quad n_{c,last} \leftarrow n_c$
7: $\quad n_c \leftarrow$ # active contacts ($\Delta z_{ri} < 0$?)
8: $\quad z(0) \leftarrow$ eq. (4.34)
9: **end**

After the calculation of $z(t_0)$ with either (4.33) or (4.34) the number of active contacts has to be checked. If they have changed the initial translation in z direction has to be recalculated until the contact states remain the same. The procedure is summarized in Algorithm 1. This simplification drastically decreases computation time which is mandatory for an application in real-time systems as described later. For most cases the algorithm converges after 2-3 iterations.

An analysis of the prediction result is performed with the simulation data already shown in the introductory example of this chapter. The multibody simulation with trajectory planning and stabilization is executed with three different maximum disturbance forces. The current state of the robot at time t_0 is used to initialize the full prediction model with the above prescribed procedure. Then the *euler-f* integrator setup is used to solve the EoM for the defined horizon. Figure 4.12 shows the prediction results for the full prediction model compared to the MBS output and the LIPM output. It can be seen that the prediction model performs better than the LIPM for all three cases and is able to produce a qualitatively similar result as the MBS. The result of the pendulum can be improved by adding a stabilization torque which is computed by a saturated PD-type feedback control. This however turns the model into a nonlinear one and the solution is not as straight forward as before. The EoM can be written as follows

$$\ddot{x}_c = \frac{g}{z_c} x_c + \frac{1}{m z_c} T_{stab}\,, \qquad T_{stab} = \underset{[T_{min}, T_{max}]}{\text{sat}} \left(T_{id} + K_p(x_{c,id} - x_c) + K_d(\dot{x}_{c,id} - \dot{x}_c)\right) \quad (4.35)$$

where sat$(\cdot)$ describes a saturation of the feedback control output according to the lim-

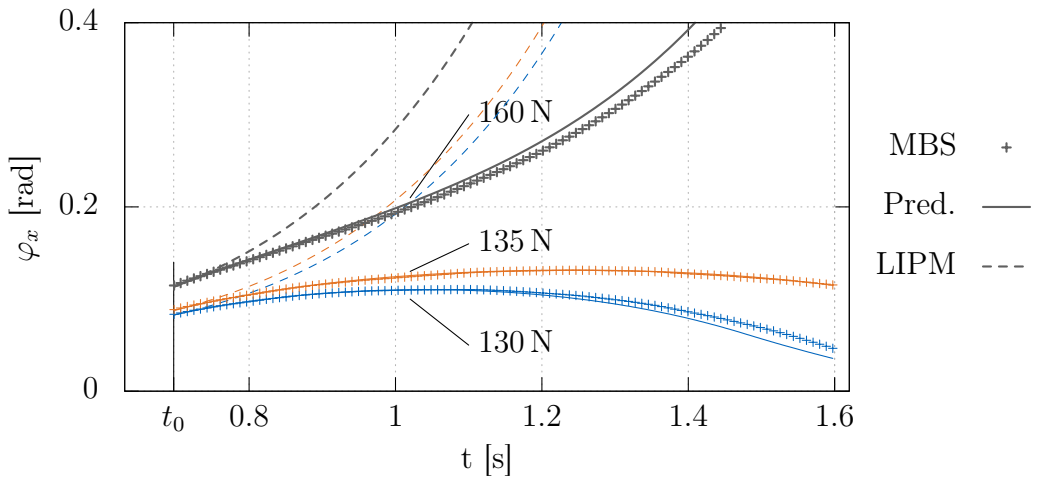


Figure 4.12: Comparison of the prediction result (Pred.) with MBS and LIPM output for three different disturbance cases.

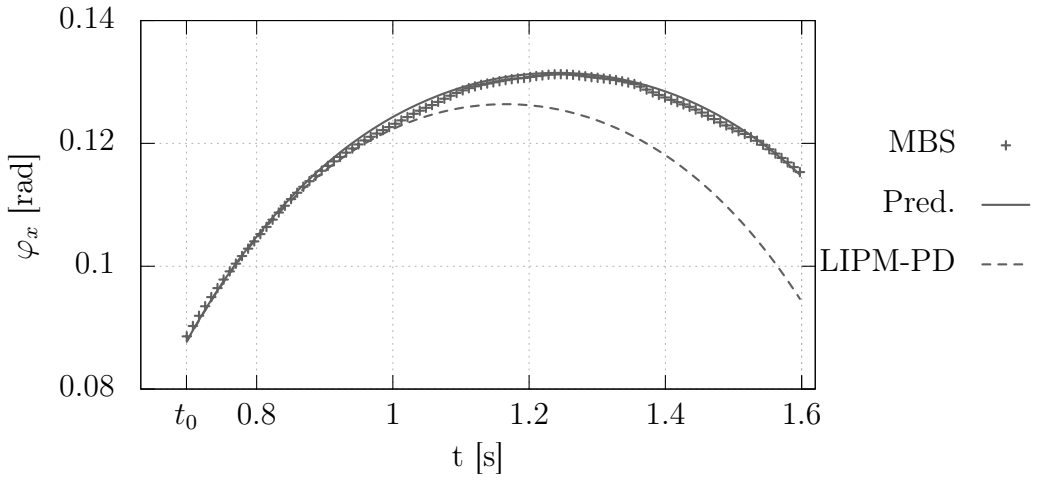


Figure 4.13: Comparison of prediction result with the PD-controlled LIPM.

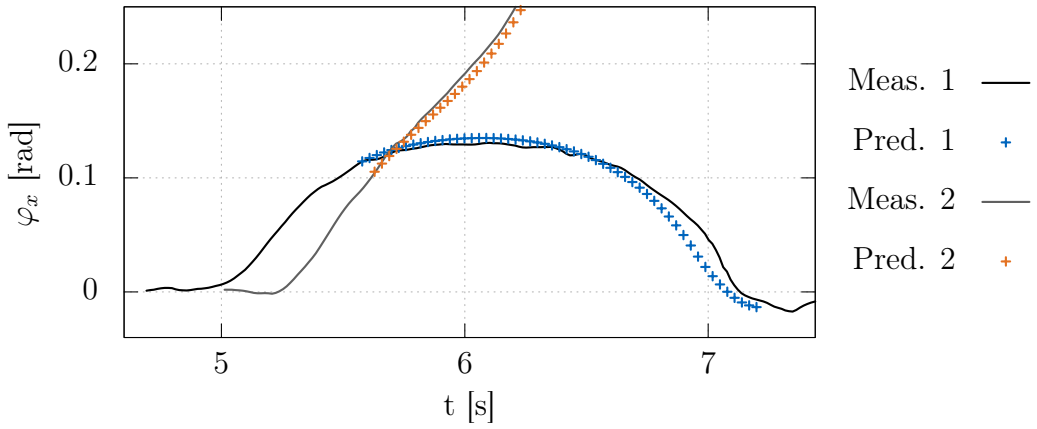


Figure 4.14: Comparison of the prediction result with measurement data of the absolute inclination φ for two different disturbance cases.

its T_{min} and T_{max}. One might recognize that this is just a combination of the linearized pendulum equation (4.5) with the feedback control of the simplified prediction model (4.20). Accordingly the same control gains and saturation limits are used. In Figure 4.13 the result of the improved LIPM is shown and compared to the other models for a peak disturbance force of 135 N. Comparing the result with the plain pendulum model in Figure 4.12 it can be seen that the prediction accuracy is significantly improved but still worse than the proposed prediction model. It should be mentioned that the control gains and saturation limits in (4.35) can be tuned in order to receive a perfect match for one single disturbance case at the expense of worse results for other situations with different disturbances.

A verification of the prediction model with measurements is shown in Figure 4.14. Two experiments are performed where the robot is walking in place and receives a push in x direction. The absolute inclination and the prediction result at a certain time instant for both cases are shown. One can see that the model can also predict the behavior of the real robot sufficiently accurate.

Comparing the results for the reduced and the full prediction model in Figure 4.15 with the simulation output of the robot one can see that the full model performs slightly better than the reduced one. Nevertheless both predict the right trend of the state evolution for φ_x and $\dot{\varphi}_x$. In order to analyze the effect of the additional leg masses, a simulation

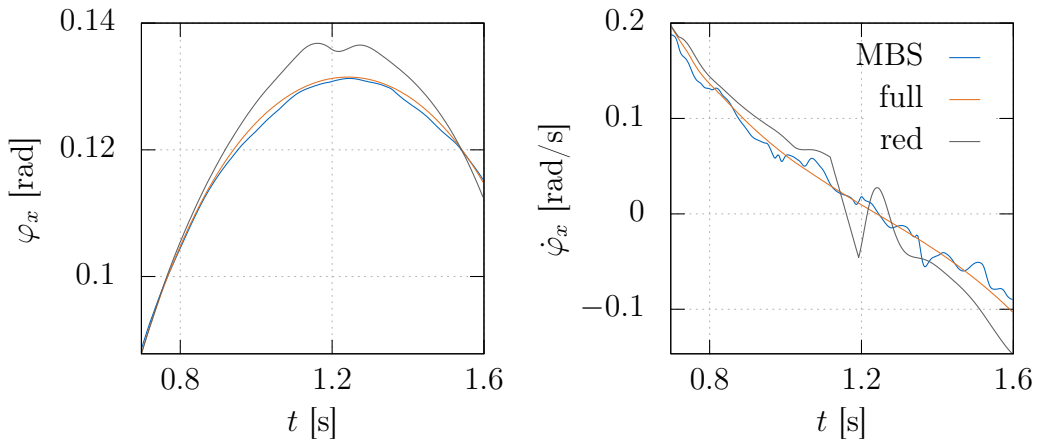


Figure 4.15: Comparison of the prediction result for the full and reduced prediction model with the MBS output.

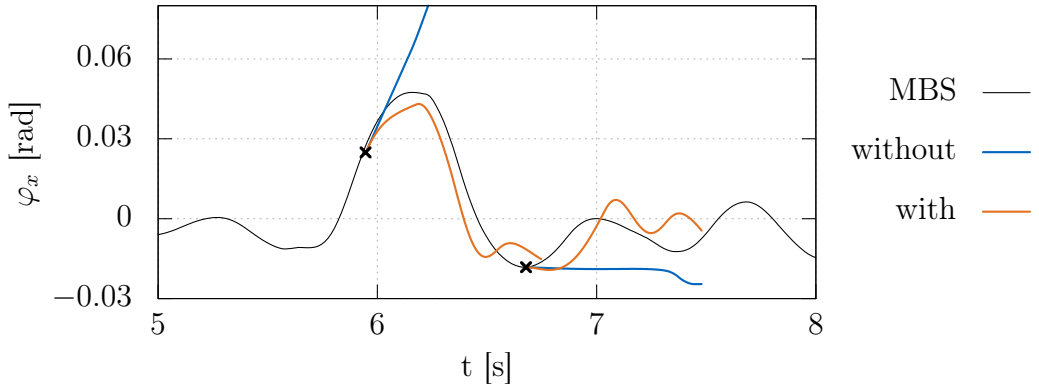


Figure 4.16: Influence of foot masses for the reduced model while the robot is walking forward and receives a disturbance ($m_f = 0.05m$). Black crosses mark the start time t_0 of the prediction.

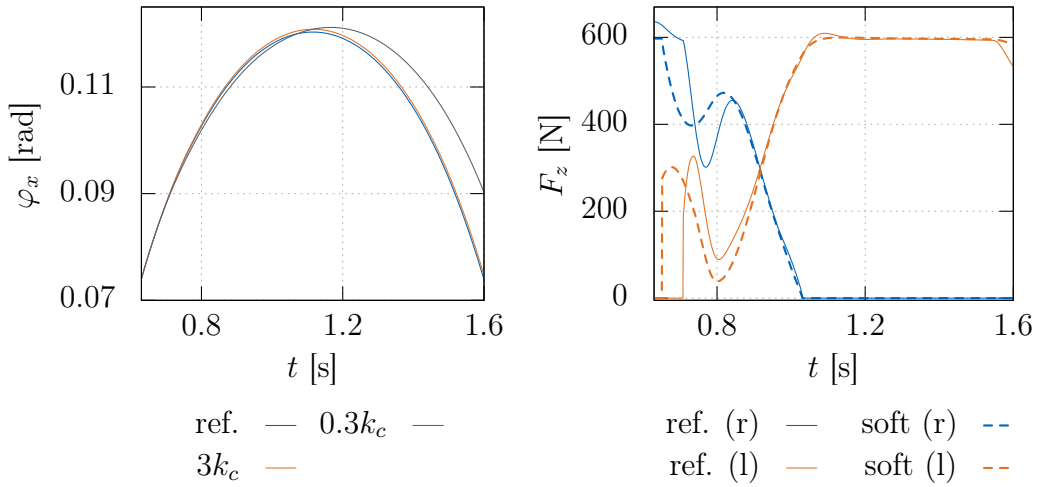


Figure 4.17: Influence of contact stiffness modification to the prediction result for the inclination φ (left) and contact forces F_z (right). The contact forces are shown for the reference (ref.) and the case with $0.3k_c$ (soft) for the right (r) and left (l) leg.

is performed where the robot is walking with 0.5 m/s in x direction and receives a disturbance. Figure 4.16 shows the prediction result of the reduced model for two different time instants (they are marked with black crosses) with and without leg masses. The relative leg mass is chosen to $m_f = 0.05m$. It can be seen that the three-mass model shows a better result for the prediction of φ_x in both cases whereas the difference between with and without leg masses is larger when the state of the robot is disturbed.

Next the effect of a variation of the chosen contact stiffness is discussed. For a given initial state $\boldsymbol{z}(t_0)$ the full model is solved with three different stiffness parameters using the *euler-f* integration setup. One with the reference value $k_c = 1.5 \cdot 10^5$ (ref.) one with a reduced stiffness ($0.3k_c$) and one with a higher stiffness ($3k_c$). Figure 4.17 shows that a reduced stiffness modifies the prediction result more than choosing a too high stiffness. This can be explained with the contact forces shown on the right side of Figure 4.17. For the soft contact the swing foot hits the ground earlier (F_z for the left leg rises earlier) which has a large effect to the inclination $\varphi_x(t)$. Due to the reduced contact stiffness their deformations are higher and this causes an early contact of the swing leg. Extending the model with a compensation of the deformation would decrease this effect.

Overall it can be seen that the model shows a good agreement compared to the multibody model of the robot and also for the real robot. The two chosen passive DoF describe the most important state error, using the right stiffness values and adding foot masses increases the prediction accuracy. It was also shown, that the proposed model outperforms the LIPM and the PD-controlled LIPM. The next part deals with model extensions which enable e.g. the application of this model for trajectory optimization methods.

4.4 Model Motion Adaptations

The following part introduces two extensions to adapt the prediction model's motion, namely a swing-foot modification and a modification of the CoG trajectory. Those can be used as free inputs to adapt the robot's motion according to sensor data. Necessary gradient computations for the model's application are provided as well.

4.4.1 Swing Foot Modification

The swing foot trajectory has two influences to the state evolution of the model. For a positive inclination error φ and with a swing foot trajectory that ends at a height of zero in ${}_Tz$ direction the corresponding contacts will switch to active before the foot reaches its final position. Presented trajectory modifications in Subsection 6.4.3 consider and solve this problem. Consequently a similar mechanism has to be included in the model. The second influence concerns the final swing foot position in ${}_Tx$ resp. ${}_Ty$ direction. Those quantities can modify the lever of the contacts in the contact forces $\boldsymbol{\lambda}$ of the EoM. The overall swing foot trajectory is illustrated in Figure 4.18 and can be written to

$$ {}_T\boldsymbol{r}_{fi} = {}_T\boldsymbol{r}_{fi,id} + \Delta\, {}_T\boldsymbol{r}_{fi} = \begin{bmatrix} x_{fi,id} \\ 0 \\ z_{fi,id} \end{bmatrix} + \begin{bmatrix} \Delta x_{fi} \\ 0 \\ \Delta z_{fi} \end{bmatrix} \tag{4.36} $$

with the ideal ${}_T\boldsymbol{r}_{f1,id}$ and the modification $\Delta\, {}_T\boldsymbol{r}_{f1}$ trajectory. Note that all quantities are written in the robot fixed planning system (T-System).

The Δx_{f1} trajectory is parametrized with quintic polynomials with set points at ΔL_{xi} for the final position of the corresponding swing foot of step i. Figure 4.19 shows an example for one step with the right leg as swing foot and a second step with the left leg being swing foot. The trajectory for each leg consists of three polynomials in order to satisfy the restriction that a leg can only move in SS phase and it is the current swing leg.

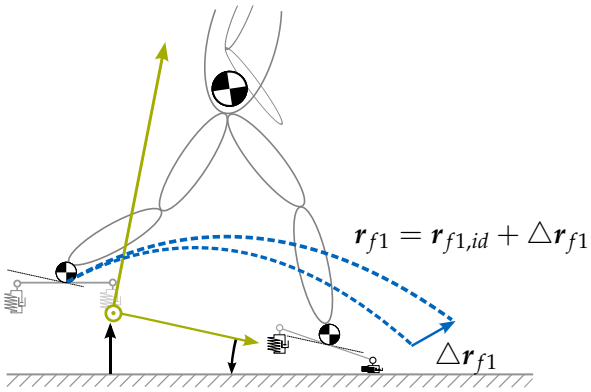


Figure 4.18: Prediction model with additional footstep modification of the swing leg.

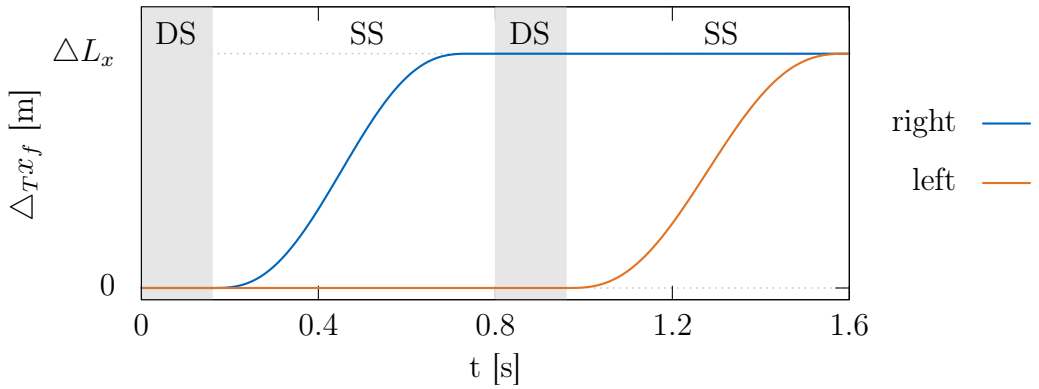


Figure 4.19: Example foot modifications in x direction. DS and SS areas are marked gray and white.

A more detailed description concerning the timing will be given in Chapter 6. To avoid too early or late touchdown of the swing foot the ideal planned z height is modified. The condition is used that the absolute contact distance has to be identical to the ideal planned one during SS phase, i.e. $\Delta z_i = z_{fi,id}$. This leads for the reduced model with relation (4.21) to

$$\Delta z_{fi}(\boldsymbol{q}, t) = \frac{1}{\cos\varphi} \begin{cases} -z + x_{fi}(t)\sin\varphi + z_{fi,id}(t)(1-\cos\varphi) & \text{for } t \in [t_{ds}, t_c] \\ \text{hold value} & \text{otherwise} \end{cases} \tag{4.37}$$

where the modification is calculated according to the state of the model during swing phase (lift off at t_{ds} and touch down at t_c) and holds its current value otherwise. It can be considered as a feedback law which is added to the model in this way. In combination with low-pass filtering the output of (4.37), this ensures that there are no jumps in the swing foot height and each leg changes its desired height only during the corresponding swing phase. For the full model the procedure is similar using the relation (4.10) and taking the value of the contact with the minimum distance of the swing foot. The described swing foot modifications parametrized by final swing foot positions can be used to write the EoM of the model with the additional dependency

$$\dot{\boldsymbol{z}} = \boldsymbol{f}(\boldsymbol{z}, t, \boldsymbol{p}), \qquad \dot{\boldsymbol{z}}_r = \boldsymbol{f}_r(\boldsymbol{z}_r, t, \boldsymbol{p}) \tag{4.38}$$

of the free parameter vector $\boldsymbol{p} = [\Delta L_{x1}, \Delta L_{x2}, \dots]$. They can be chosen heuristically or by a parameter optimization method.

4.4.2 Center of Gravity Modification

Another quantity of the prediction model that can be modified is the body trajectory $\boldsymbol{r}_b$. With the foot masses being $0-5\%$ of the overall mass m the body trajectory has the most significant contribution to the CoG trajectory. This is the reason why this section is called CoG modification even though a modified foot position also changes the CoG slightly. In Chapter 2 it was shown that $\boldsymbol{r}_c$ has to be designed in such a way that conditions resulting from unilateral contacts have to be ensured, slippage has to be avoided and kinematic limits of the robot have to be considered. In this work it is decided to modify only the x trajectory and keep the ideal CoG height. That way the method to calculate the modification only has to ensure that the CoG of the robot does not diverge from the location of the feet in order to have a kinematic feasible solution. As shown in Figure 4.20 a modification similar to (4.36) is added to the ideal trajectory

$$ {}_T\boldsymbol{r}_b = {}_T\boldsymbol{r}_{b,id} + \Delta\, {}_T\boldsymbol{r}_b = \begin{bmatrix} x_{b,id} \\ 0 \\ z_{b,id} \end{bmatrix} + \begin{bmatrix} \Delta x_b \\ 0 \\ 0 \end{bmatrix}. \tag{4.39} $$

In contrast to the swing foot modification $\Delta\, {}_T x_b\,(t)$ is a freely chosen trajectory that has to ensure continuity conditions on velocity or acceleration level. One possibility is to discretize the CoG trajectory and use piecewise constant jerks. Defining the input $u_x = \frac{\mathrm{d}}{\mathrm{d}t}\Delta\ddot{x}_b$ the trajectory is calculated by integrating this input three times

$$ \frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix} \Delta x_b \\ \Delta\dot{x}_b \\ \Delta\ddot{x}_b \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} u_x. \tag{4.40} $$

This results in the same formulation as used in Kajita et al. (2003) and Wieber (2006) but with the difference that the CoG trajectory is coupled with the LIPM in the mentioned works and with the nonlinear model in this work. The input model (4.40) can be combined with the controlled EoM (4.24) by using an extended state vector

$$ \dot{\boldsymbol{z}}_{r,ext} = \frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix} \boldsymbol{q} \\ \dot{\boldsymbol{q}} \\ \Delta x_b \\ \Delta\dot{x}_b \\ \Delta\ddot{x}_b \end{bmatrix} = \begin{bmatrix} \dot{\boldsymbol{q}} \\ \boldsymbol{M}^{-1}[\boldsymbol{\lambda}_r + \boldsymbol{T}_s - \boldsymbol{h}] \\ \Delta\dot{x}_b \\ \Delta\ddot{x}_b \\ u_x \end{bmatrix} = \boldsymbol{f}_{r,ext}\left(\boldsymbol{z}_{r,ext}, t, \boldsymbol{p}, u_x\right). \tag{4.41} $$

The index T for the CoG modification terms are omitted. For the extended first order model several notes and properties can be given

- the mass matrix $\boldsymbol{M}(x_b)$ and vector $\boldsymbol{h}(x_b, \dot{x}_b, \ddot{x}_b)$ have an explicit dependency of the CoG trajectory and therefore of its modification
- the free input $u_x(t)$ is a trajectory and is in this example piecewise constant
- the system is input affine for u_x.
- the previously described footstep modifications (4.36) with the free parameters $\boldsymbol{p}$ can be included in this formulation
- the modifications can be applied to both the full and the reduced model

In order to find an appropriate trajectory for u_x optimal control techniques can be used. In Chapter 6 an indirect approach is given, and it is shown how the problem can be extended by free parameters.

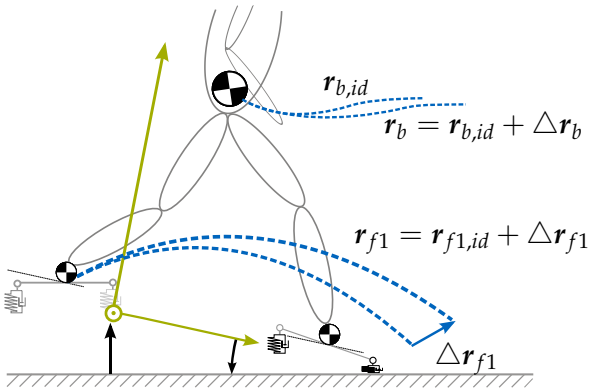


Figure 4.20: Prediction model with footstep modification of the swing leg and CoG trajectory modification.

4.4.3 Gradient Computations

Using the prediction model in a state observer or in a trajectory optimization method requires gradients of the differential equations wrt. the state and other free variables. This section states the computation formula for all necessary gradients while details are given in Appendix B. Necessary gradients are the derivatives of the full and reduced nonlinear state space models wrt. the appropriate state, footstep modification and CoG modification. According to the idea of automatic differentiation (Griewank 2000) the gradient computations exploit the chain rule and are divided into derivatives of smaller parts. This prevents from stating one big formula of the overall gradient which makes the implementation less error-prone and reduces computation time. The following equations consider the reduced prediction model but can be easily adapted for the full model.

State Gradient

The state gradient is the derivative of the model first order differential equation (4.24) wrt. the state z_r

$$\nabla_{z_r} f_r = \frac{\partial f_r}{\partial z_r} = \frac{\partial}{\partial z_r} \begin{bmatrix} \dot{q} \\ \underbrace{M^{-1}[\lambda_r + T_s - h]}_{(*)} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ \frac{\partial (*)}{\partial q} & \frac{\partial (*)}{\partial \dot{q}} \end{bmatrix} \tag{4.42}$$

where the terms $\frac{\partial (*)}{\partial q}$ and $\frac{\partial (*)}{\partial \dot{q}}$ are computed as follows

$$\frac{\partial (*)}{\partial q} = \frac{\partial M^{-1}}{\partial q}[\lambda_r + T_s - h] + M^{-1}\left[\frac{\partial \lambda_r}{\partial q} + \frac{\partial T_s}{\partial q} - \frac{\partial h}{\partial q}\right], \tag{4.43}$$

$$\frac{\partial (*)}{\partial \dot{q}} = \underbrace{\frac{\partial M^{-1}}{\partial \dot{q}}}_{=0}[\lambda_r + T_s - h] + M^{-1}\left[\frac{\partial \lambda_r}{\partial \dot{q}} + \frac{\partial T_s}{\partial \dot{q}} - \frac{\partial h}{\partial \dot{q}}\right]. \tag{4.44}$$

Note that this requires the gradient of the stabilization torque T_s with the non-smooth saturation function. Therefore it is approximated by the inverse tangent function

$$\underset{[T_{min}, T_{max}]}{\text{sat}}(\circ) \approx \frac{1}{2}(T_{min} + T_{max}) + \frac{1}{\pi}(T_{max} - T_{min}) \tan^{-1}(\kappa \circ) \tag{4.45}$$

with the variable κ describing the slope at $\tan^{-1}(0)$. The generalized contact force gradients are summed up over the set of all active contacts N_{cr} of the reduced model and are

computed with

$$\frac{\partial \boldsymbol{\lambda}_r}{\partial \boldsymbol{q}} = \frac{\partial}{\partial \boldsymbol{q}} \sum_{i \in N_{cr}} \left(\frac{\partial \Delta z_{r,i}}{\partial \boldsymbol{q}} \right)^T \underbrace{(-k_{rci} \Delta z_{r,i} - d_{rci} \Delta \dot{z}_{r,i})}_{F_{zr,i}} \tag{4.46}$$

$$= \sum_{i \in N_{cr}} \left(\frac{\partial^2 \Delta z_{r,i}}{\partial \boldsymbol{q}^2} \right)^T F_{zr,i} + \left(\frac{\partial \Delta z_{r,i}}{\partial \boldsymbol{q}} \right)^T \left(-k_{rci} \frac{\partial \Delta z_{r,i}}{\partial \boldsymbol{q}} - d_{rci} \frac{\partial \Delta \dot{z}_{r,i}}{\partial \boldsymbol{q}} \right) \tag{4.47}$$

and

$$\begin{aligned} \frac{\partial \boldsymbol{\lambda}_r}{\partial \dot{\boldsymbol{q}}} &= \frac{\partial}{\partial \dot{\boldsymbol{q}}} \sum_{i \in N_{cr}} \left(\frac{\partial \Delta z_{r,i}}{\partial \boldsymbol{q}} \right)^T \underbrace{(-k_{rci} \Delta z_{r,i} - d_{rci} \Delta \dot{z}_{r,i})}_{F_{zr,i}} \\ &= \sum_{i \in N_{cr}} \left(\frac{\partial^2 \Delta z_{r,i}}{\partial \boldsymbol{q} \partial \dot{\boldsymbol{q}}} \right)^T F_{zr,i} + \left(\frac{\partial \Delta z_{r,i}}{\partial \boldsymbol{q}} \right)^T \left(-k_{rci} \frac{\partial \Delta z_{r,i}}{\partial \dot{\boldsymbol{q}}} - d_{rci} \frac{\partial \Delta \dot{z}_{r,i}}{\partial \dot{\boldsymbol{q}}} \right). \end{aligned} \tag{4.48}$$

Exploiting the properties that the quantities $\boldsymbol{M}, \boldsymbol{h}, \boldsymbol{T}_s$ do not depend on the state z and its derivatives and the contact deformation depends only on the generalized coordinates $\boldsymbol{q}$ saves computation time for the gradient computation.

Center of Gravity Gradient

The derivative of the system model wrt. the CoG trajectory in x direction $x_b(t)$ is for the position, velocity and acceleration level

$$\nabla_{x_b} \boldsymbol{f}_r = \frac{\partial \boldsymbol{f}_r}{\partial x_b} = \frac{\partial \boldsymbol{M}^{-1}}{\partial x_b} [\boldsymbol{\lambda}_r + \boldsymbol{T}_s - \boldsymbol{h}] - \boldsymbol{M}^{-1} \frac{\partial \boldsymbol{h}}{\partial x_b}, \tag{4.49}$$

$$\nabla_{\dot{x}_b} \boldsymbol{f}_r = \frac{\partial \boldsymbol{f}_r}{\partial \dot{x}_b} = -\boldsymbol{M}^{-1} \frac{\partial \boldsymbol{h}}{\partial \dot{x}_b}, \tag{4.50}$$

$$\nabla_{\ddot{x}_b} \boldsymbol{f}_r = \frac{\partial \boldsymbol{f}_r}{\partial \ddot{x}_b} = -\boldsymbol{M}^{-1} \frac{\partial \boldsymbol{h}}{\partial \ddot{x}_b}. \tag{4.51}$$

Gradient terms that are zero are omitted in the above stated equations.

Footstep Gradient

The last gradient shown here is the foot position gradient of the system model. Assume that foot 1 is the current swing foot and its final position is modified with $p = \Delta L_x$. Utilizing that only the contact forces $\boldsymbol{\lambda}_r$ depend on the foot position the gradient is then computed as follows

$$\begin{aligned} \nabla_p \boldsymbol{f}_r &= \frac{\partial \boldsymbol{f}_r}{\partial p} = \boldsymbol{M}^{-1} \frac{\partial \boldsymbol{\lambda}_r}{\partial p} \\ &= \boldsymbol{M}^{-1} \left(\frac{\partial^2 \Delta z_{r,1}}{\partial \boldsymbol{q} \partial p} \right)^T F_{zr,1} + \left(\frac{\partial \Delta z_{r,1}}{\partial \boldsymbol{q}} \right)^T \left(-k_{rc1} \frac{\partial \Delta z_{r,1}}{\partial p} - d_{rc1} \frac{\partial \Delta \dot{z}_{r,1}}{\partial p} \right). \end{aligned} \tag{4.52}$$

Note that the overall gradient (4.52) remains zero until the swing foot hits the ground. For long time horizons of multiple steps it has to be considered that at the next lift off the gradient will become zero again.
There is the possibility to improve the gradient information by including that the state z_r of the model is influenced by the parameter p. This can be done by computing the absolute derivative of the state space model

$$\frac{\mathrm{d} \boldsymbol{f}_r}{\mathrm{d} p} = \frac{\partial \boldsymbol{f}_r}{\partial p} + \frac{\partial \boldsymbol{f}_r}{\partial \boldsymbol{z}_r} \frac{\partial \boldsymbol{z}_r}{\partial p}. \tag{4.53}$$

The term $\frac{\partial z_r}{\partial p}$ can be computed in a recursive way by applying a numerical integration scheme. Suppose that an explicit Euler integrator with time step Δt is used, the state (and its partial derivative) propagates from time step k to $k+1$ with

$$z_{r,k+1} = z_{r,k} + \Delta t \dot{z}_{r,k} \qquad | \frac{\partial(\,)}{\partial p}, \tag{4.54}$$

$$\Rightarrow \frac{\partial z_{r,k+1}}{\partial p} = \frac{\partial z_{r,k}}{\partial p} + \Delta t \frac{\partial f_{r,k}}{\partial p}. \tag{4.55}$$

The relation (4.55) is then used to compute among the state its parameter gradient during numerical integration. The same procedure can be applied for the CoG gradients.

4.4.4 Additional Contact Points - Including Arms

Conceptually, the model can be easily extended to include additional contact points e.g. for arms leaning against a wall. This enables to use it for solving multi-contact problems like the one mentioned here. The following part describes the idea how to extend the model by one arm that can interact with a detected wall (cf. Figure 4.21). The wall is described as surface with an origin, boundary points and a normal vector $\boldsymbol{n}_s$. The position of the arm is described by $\boldsymbol{r}_a$ while the contact between hand and wall is as for the feet an unilateral spring-damper pair (k_a, d_a). If the contact is closed its force is assumed to point always in direction of $\boldsymbol{n}_s$. The force resulting from the new contact yields

$$\boldsymbol{F}_a = \frac{\boldsymbol{n}_s}{|\boldsymbol{n}_s|^2} \boldsymbol{n}_s^T \left(k_a ({}_I\boldsymbol{r}_a - {}_I\boldsymbol{r}_{a,0}) + d_a \dot{\boldsymbol{r}}_a \right) \tag{4.56}$$

with ${}_I\boldsymbol{r}_{a,0}$ being the neutral spring length which can be any point parallel to the surface with a distance of the spring length in $\boldsymbol{n}_s$ direction. The existing models are extended by the new contact point with a projection of (4.56) into the direction of $\boldsymbol{q}$

$$\boldsymbol{\lambda}_a = \frac{\partial\, {}_I\boldsymbol{r}_a}{\partial \boldsymbol{q}} \boldsymbol{F}_a. \tag{4.57}$$

This can be added to the EoM, e.g. for the reduced controlled model (4.23) resulting in

$$\boldsymbol{M}(\boldsymbol{q}, t)\ddot{\boldsymbol{q}} + \boldsymbol{h}(\boldsymbol{q}, \dot{\boldsymbol{q}}, t) = \boldsymbol{\lambda}_r(\boldsymbol{q}, \dot{\boldsymbol{q}}, t) + \boldsymbol{\lambda}_a(\boldsymbol{q}, \dot{\boldsymbol{q}}, t) + \boldsymbol{T}_s \tag{4.58}$$

In future work, this model could be used to plan and adapt an arm motion which is the solution of a multi-contact problem. This might be a way to plan a walking motion with additional arm contacts in real-time. The presented idea exploits that the model does not use a ZMP trajectory to solve the planning problem. Nevertheless this requires a detected surface and a walking pattern planner to determine an ideal position $\boldsymbol{r}_a$ which can be modified and/or optimized with this model. During writing this thesis both are not finally integrated into the control-system of the robot. Another problem that has to be solved is how to determine the contact parameter k_a, d_a.

4.5 Three-Dimensional Model

This sections introduces a three-dimensional model which is based on the planar reduced model (4.24) and which is extended by an additional DoF. The model in x direction is used and extended by the inclination φ_y in y direction. The motivation for this model is to compute the coupled prediction of the biped. This allows to use it for optimizing e.g. step-length modifications in both directions at once. The model properties are equal to the properties presented in Subsection 4.3.3 with the mentioned additional inclination in

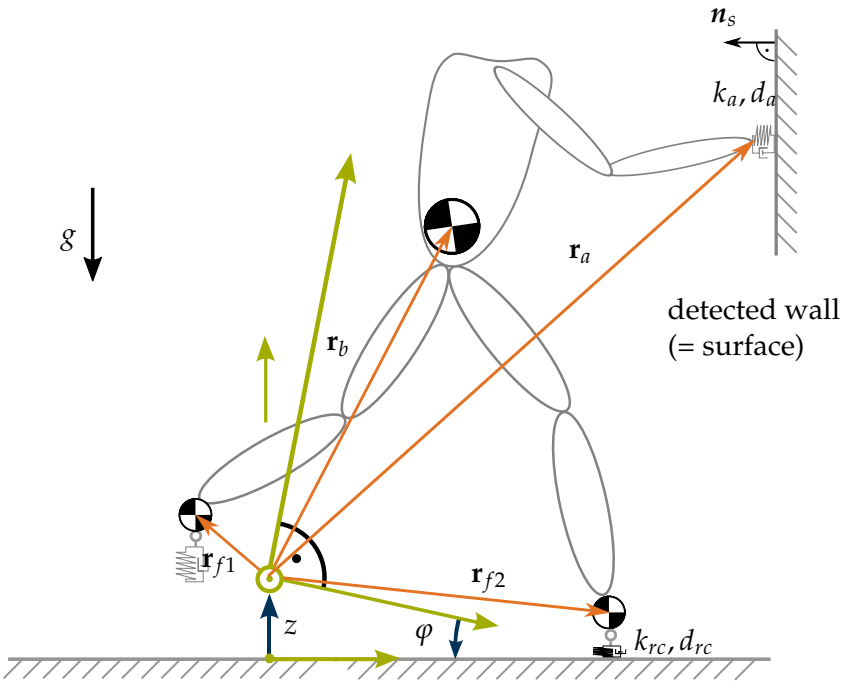


Figure 4.21: Prediction model extended by additional arms and contacts to a detected surface.

y direction. The translation in x and y direction are again neglected. Therefore the vector of generalized coordinates is

$$\boldsymbol{q}_s = [\varphi_x, \varphi_y, z]^T. \tag{4.59}$$

All quantities related to the spatial model (cf. Figure 4.22) are indicated by the index s. The two inclinations are independent variables and describe the angle between the z axis of the T-system and the gravity vector. According to Buschmann (2010) the transformation matrix $\boldsymbol{A}_{IT} = [{}_I\boldsymbol{e}_{x_T}, {}_I\boldsymbol{e}_{y_T}, {}_I\boldsymbol{e}_{z_T}]$ from the robot system (T) to the world system (I) can be stated with the three axis of the T-system described in the I-system

$${}_I\boldsymbol{e}_{x_T} = \begin{bmatrix} \cos\varphi_x \\ 0 \\ -\sin\varphi_x \end{bmatrix}, \quad {}_I\boldsymbol{e}_{y_T} = \begin{bmatrix} 0 \\ \cos\varphi_y \\ \sin\varphi_y \end{bmatrix}, \quad {}_I\boldsymbol{e}_{z_T} = {}_I\boldsymbol{e}_{x_T} \times {}_I\boldsymbol{e}_{y_T}\,. \tag{4.60}$$

where φ_x is positive about the y axis and φ_y a positive rotation about x axis. The overall matrix yields

$$\boldsymbol{A}_{IT} = \begin{bmatrix} \cos\varphi_x & 0 & \sin\varphi_x\cos\varphi_y \\ 0 & \cos\varphi_y & -\cos\varphi_x\sin\varphi_y \\ -\sin\varphi_x & \sin\varphi_y & \cos\varphi_x\cos\varphi_y \end{bmatrix}. \tag{4.61}$$

With the given foot and body trajectories ${}_T\boldsymbol{r}_{f1}(t), {}_T\boldsymbol{r}_{f2}(t), {}_T\boldsymbol{r}_b(t)$ the absolute position for the point masses can be computed with the rotation matrix $\boldsymbol{A}_{IT}(\varphi_x, \varphi_y)$ and the translation z. The transformation is

$${}_I\boldsymbol{r}_j(\boldsymbol{q}_s, t) = \begin{bmatrix} 0 \\ 0 \\ z \end{bmatrix} + \boldsymbol{A}_{IT}(\varphi_x, \varphi_y)\, {}_T\boldsymbol{r}_j(t), \quad \forall j \in N_B \tag{4.62}$$

and in contrast to (4.9) the vector ${}_T\boldsymbol{r}_j = [x_j(t), y_j(t), z_j(t)]^T$ has three entries that are non-zero. The absolute position of the three masses and the inertia about x and y axis Θ_{xx}, Θ_{yy}

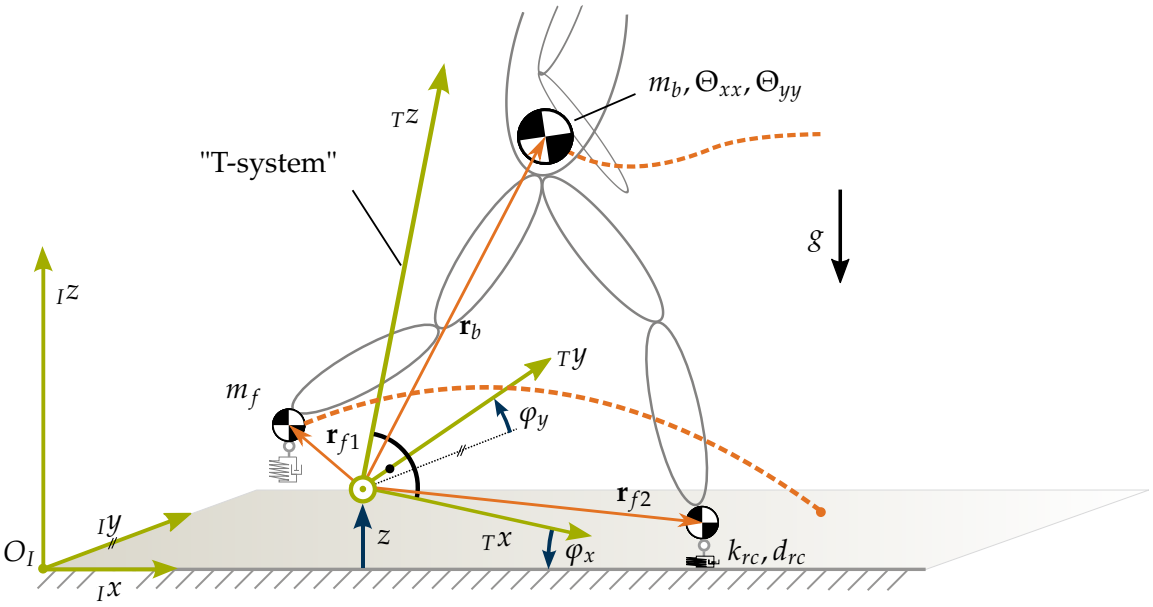


Figure 4.22: Three dimensional prediction model with three unactuated DoF.

can be then used to compute the overall kinetic energy of the system

$$E_{kin} = \sum_{j \in N_B} m_j \, {}_I\dot{\boldsymbol{r}}_j^2 + {}_T\boldsymbol{\omega}^T \, {}_T\boldsymbol{\Theta} \, {}_T\boldsymbol{\omega} \, . \tag{4.63}$$

For the rotational part the angular velocity and inertia of the robot described in the T-system are used

$${}_T\boldsymbol{\omega} = \boldsymbol{A}_{IT}^T \begin{bmatrix} \dot{\varphi}_x \\ \dot{\varphi}_y \\ 0 \end{bmatrix}^T , \quad {}_T\boldsymbol{\Theta} = \begin{bmatrix} \Theta_{xx} & 0 & 0 \\ 0 & \Theta_{yy} & 0 \\ 0 & 0 & 0 \end{bmatrix} . \tag{4.64}$$

The rotational intertia is assumed to be constant and diagonal for the two inclination directions and zero for a rotation about the z axis. The potential energy consists of two parts from the springs (of active contacts) and gravity. Both are determined in the same way shown in (4.12) but using the contact distances of the three DoF model

$$\Delta z_i(\boldsymbol{q}_s, t) = \boldsymbol{e}_z^T \, {}_I\boldsymbol{r}_{fi} - z_0 = z - x_{fi} \sin\varphi_x + y_{fi} \sin\varphi_y + z_{fi} \cos\varphi_x \cos\varphi_y - z_0 \tag{4.65}$$

with ${}_I\boldsymbol{r}_{fi}$ computed from (4.62). The vector of non-conservative forces $Q_{s,NC}$ of the three DoF model is obtained by inserting (4.65) in (4.13). Evaluation of the Lagrange II formalism (4.7) yields finally the EoMs for the spatial model

$$\boldsymbol{M}_s(\boldsymbol{q}_s, t)\ddot{\boldsymbol{q}}_s + \boldsymbol{h}_s(\boldsymbol{q}_s, \dot{\boldsymbol{q}}_s, t) = \boldsymbol{\lambda}_s(\boldsymbol{q}_s, \dot{\boldsymbol{q}}_s, t) + \boldsymbol{T}_s. \tag{4.66}$$

A stabilization torque $\boldsymbol{T}_s = [0, \, T_{stab,x}, \, T_{stab,y}]^T$ as shown in (4.20) is added for both inclinations. The contact force $\boldsymbol{\lambda}_s$ is computed by summing the contribution of all active contacts. In order to set a contact active its deformation Δz_j has to be negative. The model (4.66) can be transformed to a first order differential equation

$$\dot{\boldsymbol{z}}_s = \frac{\mathrm{d}}{\mathrm{d}t} \begin{bmatrix} \boldsymbol{q}_s \\ \dot{\boldsymbol{q}}_s \end{bmatrix} = \begin{bmatrix} \dot{\boldsymbol{q}}_s \\ \boldsymbol{M}_s^{-1}(\boldsymbol{\lambda}_s + \boldsymbol{T}_s - \boldsymbol{h}_s) \end{bmatrix} = \boldsymbol{f}_s(\boldsymbol{z}_s, t) \tag{4.67}$$

and solved with a fixed step forward Euler integration scheme as described for the planar model. With a time discretization of $\Delta t = 3\,\mathrm{ms}$ the computational time for a 0.8 s

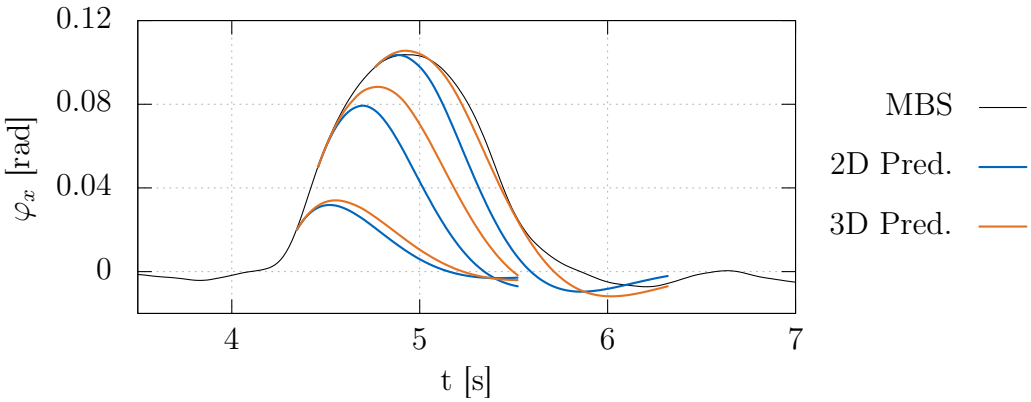


Figure 4.23: Comparison of the prediction result for the two and the three dimensional model with the full simulation model (MBS).

prediction takes approx. 150 µs. This is double the time for solving the planar model for the same time horizon. Nevertheless the planar model has to be solved for both directions and the spatial model only once, i.e. the overall computation time is almost the same. Figure 4.23 shows prediction results of both models for the x direction at several simulation time instants and compares the result with the full simulation model of the robot. Contact parameter, masses and control parameter are the same for the models. It can be seen that both model types (2D and 3D) generate similar predictions whereas the spatial prediction model performs better in this example.

The three DoF model has the advantage that it includes the planned motion in x and y direction. This can be exploited to find an optimal footstep location in one coupled problem. The solution includes the interaction between a modification in sagittal and lateral direction which has been neglected in the planar model. Another advantage of this model is that geometric constraints are easily included. Those constraints result from kinematic restrictions of the robot, e.g. the reachable region (maximum and minimum step lengths) or self collision avoidance. Additionally, it will be shown that the region of feasible modifications is modified according to detected obstacles (Section 6.6).

4.6 Chapter Summary

In this chapter dynamic models of bipedal robots are presented which consider explicitly the underactuation and the resulting stability issues as well as the effect of the robot's local stabilization. It was shown, that the models produce accurate predictions of the state evolution and are applicable in real-time. Desired leg and CoG trajectories are included in the prediction model. In order to use it for trajectory optimization, swing foot and CoG modifications are introduced and their integration into the model are shown. The two applications are:

- State estimation: in the following chapter a state estimator using the prediction model is developed. It is used to fuse IMU measurements, contact force measurements and the robot's walking control output. This is enabled because the model explicitly considers the foot with its unilateral compliant contacts and passive DoF.
- Trajectory adaptation: it will be shown that it is possible to implement powerful trajectory optimization methods with the help of the prediction model. Thanks to its natural inclusion of ideal trajectories, there are different possibilities to apply methods for trajectory adaption in a model predictive way.

Chapter 5

State Estimation

5.1 Introduction

The knowledge of the robot's current state is a necessary requirement to evaluate and, if necessary, adapt its planned motion. This chapter presents state estimators that produce required quantities for the initial state of the models of Chapter 4. As described in Chapter 3 the robot is equipped with an IMU which is located at the upper body incremental encoder at each joint and an FTS at each foot. The mechanical state of the robot consists of its joint angles, the absolute position and orientation of the robot wrt. the world and the according velocities. The joint states can be captured accurately even when the elasticity of the joints is considered by the joint sensors. The necessary quantities of the unactuated DoFs are partly captured by the IMU. This sensor generates measurements of the absolute upper body inclination and inclination rate as well as its acceleration in all directions. A possible solution is, to use the inclination state ($\varphi_{m,x}$ and $\dot{\varphi}_{m,x}$) directly for the initial state of the prediction model. However the sensor signals for the state include high frequent oscillations, e.g. from excitation of the robot's mechanical structure. An example measurement during an external disturbance is shown in Figure 5.2. Especially the inclination rate includes mentioned disturbances. The aim of the estimators is therefore to extract the information that is related to the robot's rigid body motion (and is included in the prediction model) from the IMU data. Note that the presented methods are not used for robot navigation. Consequently the absolute position is not required. The presented estimators are based on a previously published method (Wittmann et al. 2015a) and are extended by a nonlinear model. The classification into the model predictive trajectory adaptation is shown in Figure 5.1. The following part gives an overview of related work.

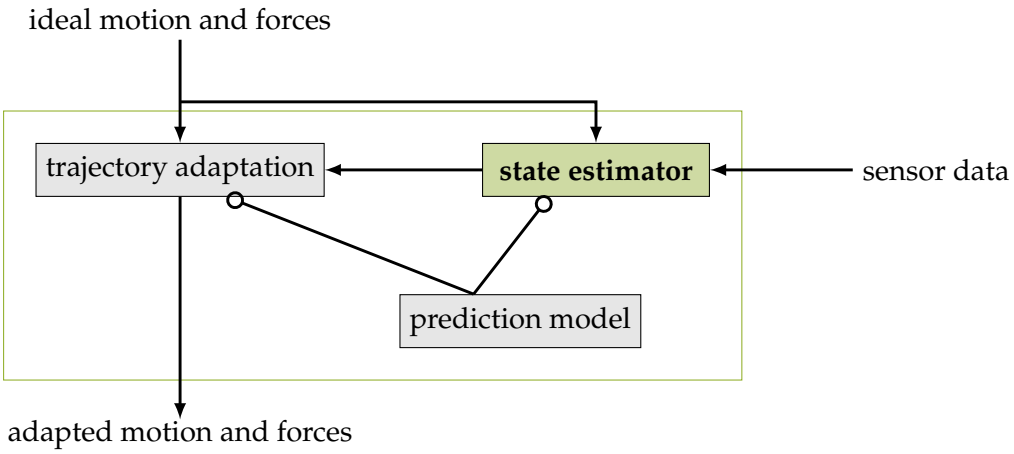


Figure 5.1: Chapter classification into the model predictive trajectory adaptation.

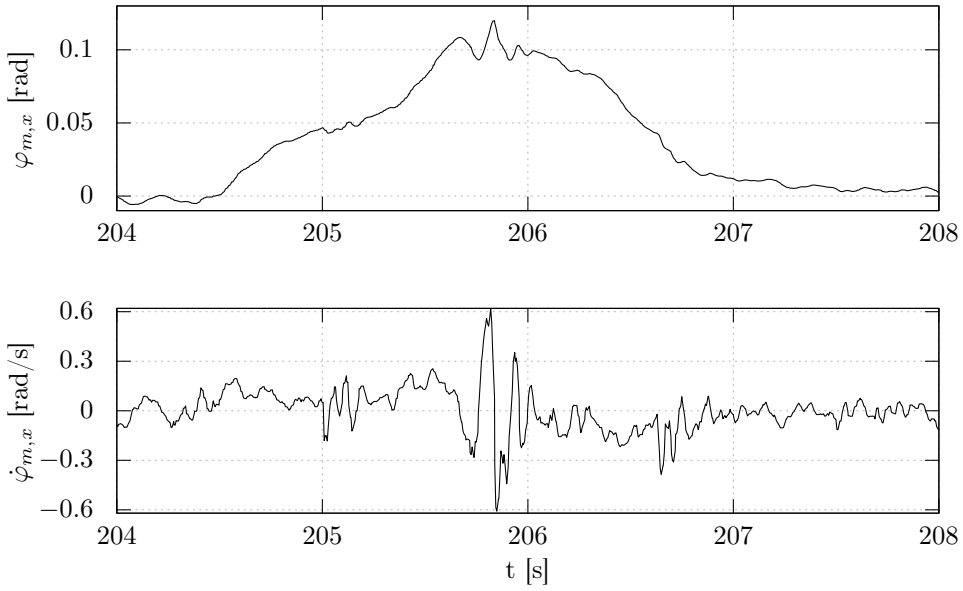


Figure 5.2: Example IMU measurements for inclination and inclination rate in x direction during an external disturbance.

Many methods in literature do not require the overall state of the robot but the state of the CoG for their stabilization algorithms. There are basically two approaches to estimate the CoG state of a walking robot using an IMU. The first one computes the state by evaluating the kinematic chain of the robot including the rotation of the reference frame e. g. Englsberger et al. (2011) and Nishiwaki et al. (2012). Measuring all joint angles enables to compute the transformation from stance leg to the CoG which is then rotated according to IMU measurements around a certain pivot point. Nishiwaki et al. (2012) use the current measured CoP position as pivot. In the second approach, dynamic models are used to fuse the data from different sensors and control inputs. The authors of Kwon (2007) propose a LIPM based Kalman filter for the CoG state. They use the measured CoP position as model input and the CoG state as model output. In Stephens (2011) a similar method is presented which assumes point feet and computes the CoP position only from vertical force measurements. The filter is extended by an un-modeled CoG offset and an external force. They are included as additional state variables. Masuya and Sugihara (2013) propose a dual stage complementary filter using the IMU accelerations in a double integrator. In their method they estimate the point which is used as pivot for the IMU inclination to estimate the trunk position from the robot's kinematics. A similar approach is proposed in Bloesch et al. (2012). In Xinjilefu et al. (2014) and Xinjilefu and Atkeson (2012) the authors propose higher DoF models for their state estimators. This enables the filter to estimate the overall state of the robot using the multibody dynamics. Among others they obtain an estimate of the joint velocities.

In addition to the mentioned approaches that use data from different sensors to improve the robot's estimated state, there are many works that fuse the IMU measurement data (acceleration and inclination rate) e.g. Nishiwaki and Kagami (2007b) and Pongsak et al. (2002). In Subsection 3.2.2 it was stated that there are sophisticated IMU fusion algorithms already available on LOLA. In the following a new method to fuse data from sensors with a dynamic model and a Kalman filter is proposed.

5.2 Extended Kalman Filter based State Estimator

The following parts present the state estimator for the model predictive trajectory adaptation. The extended Kalman filter is introduced briefly. Furthermore details of the plant and the measurement model are provided.

5.2.1 Estimator Overview

State estimators are used to observe states that can not be measured directly or that include measurement errors like noise. For the later case the estimator acts like a filter that extracts relevant data from measurements. This is the case for the presented one. The robot is modeled with the prediction model introduced in Chapter 4. An extended Kalman filter framework (Adamy 2009) is applied since the model is nonlinear. It follows the idea of the Kalman filter (Kalman 1960) and uses a linearization of the nonlinear model at each time instant. As a consequence, the filter gain depends on the state and is time varying. The computation of its filter matrix has to be done online. A brief description of relevant quantities and the filter update is given in the following. The process to be observed is described by the discrete-time nonlinear prediction model

$$\boldsymbol{x}_{k+1} = \boldsymbol{f}(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{w}_k) \tag{5.1}$$

with the state $\boldsymbol{x}_k = \boldsymbol{x}(t_k)$, the control $\boldsymbol{u}_k = \boldsymbol{u}(t_k)$ and the random variable $\boldsymbol{w}_k = \boldsymbol{w}(t_k)$ at time instant k. The measurement output is generated with the model

$$\boldsymbol{y}_k = \boldsymbol{h}(\boldsymbol{x}_k, \boldsymbol{v}_k) \tag{5.2}$$

where $\boldsymbol{v}_k$ is another random variable that represents the measurement noise. The $\boldsymbol{w}_k$ are often called model noise. Both random variables $\boldsymbol{v}_k$ and $\boldsymbol{w}_k$ are assumed to be independent Gaussian white noise with a normal probability distribution

$$p(\boldsymbol{w}) \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{Q}_f), \quad p(\boldsymbol{v}) \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{R}_f), \tag{5.3}$$

with mean zero and the covariance matrices $\boldsymbol{Q}_f, \boldsymbol{R}_f$. Those matrices include the variance for every entry of $\boldsymbol{w}_k$ in the diagonal and the covariance for the side entries. The linear Kalman filter computes an optimal estimate[1] $\hat{\boldsymbol{x}}_{k+1}$ for a linear model. The extended Kalman filter uses the same algorithm with linearized models evaluated at the current state and control input $(\boldsymbol{x}_k, \boldsymbol{u}_k)$. The linearization of (5.1) is

$$\begin{aligned} \boldsymbol{x}_{k+1} &= \frac{\partial \boldsymbol{f}(\boldsymbol{x}_k, \boldsymbol{u}_k, 0)}{\partial \boldsymbol{x}} \boldsymbol{x}_k + \frac{\partial \boldsymbol{f}(\boldsymbol{x}_k, \boldsymbol{u}_k, 0)}{\partial \boldsymbol{u}} \boldsymbol{u}_k + \frac{\partial \boldsymbol{f}(\boldsymbol{x}_k, \boldsymbol{u}_k, 0)}{\partial \boldsymbol{w}} \boldsymbol{w}_k \\ &= \boldsymbol{A}_k \boldsymbol{x}_k + \boldsymbol{B}_k \boldsymbol{u}_k + \boldsymbol{W}_k \boldsymbol{w}_k \end{aligned} \tag{5.4}$$

and of (5.2)

$$\boldsymbol{y}_k = \frac{\partial \boldsymbol{h}(\boldsymbol{x}_k, 0)}{\partial \boldsymbol{x}} \boldsymbol{x}_k + \frac{\partial \boldsymbol{h}(\boldsymbol{x}_k, 0)}{\partial \boldsymbol{v}} \boldsymbol{v}_k = \boldsymbol{Y}_k \boldsymbol{x}_k + \boldsymbol{V}_k \boldsymbol{v}_k \tag{5.5}$$

with the Jacobian abbreviations $\boldsymbol{A}_k, \boldsymbol{B}_k, \boldsymbol{W}_k, \boldsymbol{Y}_k$ and $\boldsymbol{V}_k$. The filter's optimality measure is defined as the vector 2-norm of the squared estimation error (Isermann 2011, pp.540)

$$V_k = E\left(||\hat{\boldsymbol{x}}_k - \boldsymbol{x}_k||_2^2\right) = \mathrm{tr}(\boldsymbol{P}_k) \tag{5.6}$$

at time t_k. The matrix $\boldsymbol{P}_k$ describes the covariance of the estimation error. A filter that satisfies

$$\min_{\hat{x}_k} V_k(\hat{x}_k) \tag{5.7}$$

[1] An estimated variable a is indicated in the following by $\hat{a}$.

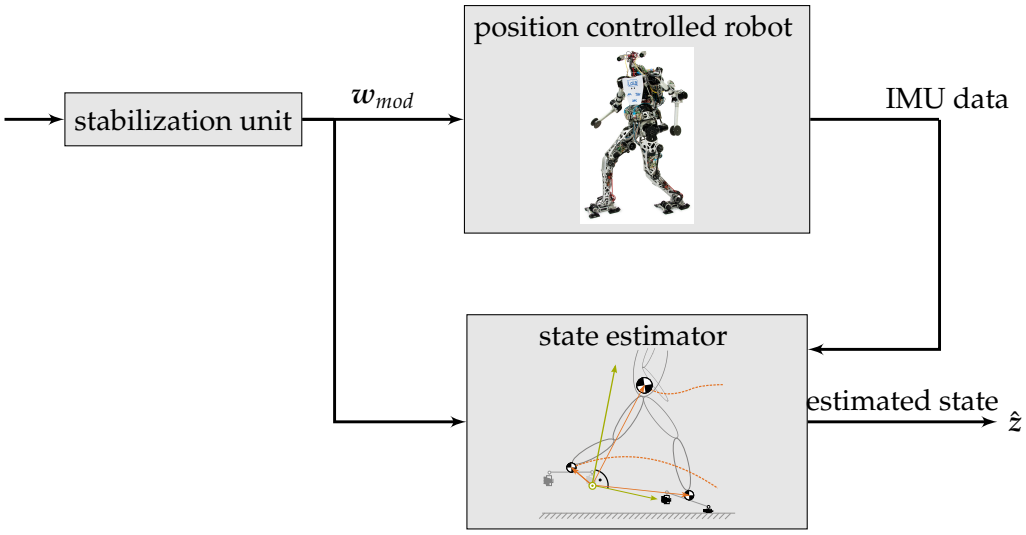


Figure 5.3: Overview of the presented state estimator: control input, model and measurement.

produces an optimal estimate for the input u_k and measurements y_{k+1}. The algorithm for the filter can be stated as a recursive predictor/corrector setting. With the nonlinear model the state $\hat{x}_{k,pred}$ is predicted for the next time step t_k and will be corrected according to the measurements y_k. The recursive algorithm for the extended Kalman filter yields:

Prediction step:

$$\hat{x}_{k,pred} = f(\hat{x}_{k-1}, u_{k-1}, 0), \tag{5.8}$$

$$P_{k,pred} = A_k P_k A_k^T + W_k Q_f W_k, \tag{5.9}$$

Correction step:

$$K_k = P_{k,pred} Y_k^T \left(Y_k P_{k,pred} + V_k R_f V_k \right)^{-1}, \tag{5.10}$$

$$\hat{x}_k = \hat{x}_{k,pred} + K_k \left(y_k - h(\hat{x}_{k,pred}, 0) \right), \tag{5.11}$$

$$P_k = (E - K_k Y_k) P_{k,pred}. \tag{5.12}$$

The filter matrix K_k is used to compute the final state estimate for the current time instant t_k. The algorithm differs from the linear Kalman filter by the time varying Jacobians of the linearization and the nonlinear prediction step. Consequently the filter does not produce an optimal estimate for the nonlinear system. For ill posed linearization points the filter can even diverge. For a successful application the optimality criterion (5.7) should be checked whether it diverges or not.

The estimator used in this thesis includes the dynamic prediction model (4.19) but without its feedback control. The output of the real robot's stabilization unit w_{mod} is used instead. Since the model is nonlinear an extended Kalman filter is used to fuse measurement data with this model. The measurements are obtained from the IMU data. Details will be given in the following sections. An overview of the estimator is shown in Figure 5.3.

5.2.2 Prediction and Measurement Model

The prediction step of the estimator requires a model in form of (5.1). The dynamic prediction model introduced in Chapter 4 will be used and extended in the following. Since the model is planar the estimation process is performed decoupled in x and y direction. The following description refers to the x direction and can be applied the same way for the other direction. The taskspace trajectory which is modified by the hybrid posi-

tion/force control $\boldsymbol{w}_{mod}(t_k), \dot{\boldsymbol{w}}_{mod}(t_k), \ddot{\boldsymbol{w}}_{mod}(t_k)$ at current time instant t_k is used to calculate the vectors for the feet and the upper body $\boldsymbol{r}_{fi}(t_k), \boldsymbol{r}_b(t_k)$ and their derivatives. Note that the taskspace also includes the foot orientation relative to the upper body which is used to determine $\boldsymbol{\alpha}(t_k)$ and $\dot{\boldsymbol{\alpha}}(t_k)$. Including the stabilization output of the real robot in the prediction model is an indirect way to include the FTS data (which is the input for the stabilization). Since disturbances have an effect to the measured forces and torques they are indirectly included in the prediction process. With the state from previous time step $(\boldsymbol{q}_k, \dot{\boldsymbol{q}}_k)$ the EoM

$$\boldsymbol{M}(\boldsymbol{q}_k, t_k)\ddot{\boldsymbol{q}}_k + \boldsymbol{h}(\boldsymbol{q}_k, \dot{\boldsymbol{q}}_k, t_k) = \boldsymbol{\lambda}(\boldsymbol{q}_k, \dot{\boldsymbol{q}}_k, \boldsymbol{\alpha}_k, \dot{\boldsymbol{\alpha}}_k, t_k) + \left(\frac{\partial\, {}_I\dot{\boldsymbol{r}}_b\,(\boldsymbol{q}_k, \dot{\boldsymbol{q}}_k, t_k)}{\partial \dot{\boldsymbol{q}}}\right)^T \boldsymbol{f}_{ext} \tag{5.13}$$

can be used to calculate the accelerations $\ddot{\boldsymbol{q}}_k$. The model is extended by an unknown external force $\boldsymbol{f}_{ext}$ acting on the upper body. The force is assumed to act only in x_I-direction, i.e. $\boldsymbol{f}_{ext} = [f_{ext,x}, 0, 0]^T$. The Jacobian $\boldsymbol{J}_b = \frac{\partial_I \dot{\boldsymbol{r}}_b}{\partial \dot{\boldsymbol{q}}}$ projects it to the directions of the model's generalized coordinates. With this additional state variable it is possible to include unknown disturbances into the prediction process. As a consequence this state enables the estimator to compensate for model and measurement errors, and additionally to estimate imposed external forces. During the prediction process the force is treated as state variable with

$$\dot{f}_{ext,x} = 0. \tag{5.14}$$

Defining the state of the Kalman filter $\boldsymbol{x}_f = [\boldsymbol{q}, \dot{\boldsymbol{q}}, f_{ext,x}]^T$ and using an explicit Euler integration scheme (time step Δt), the first order discrete prediction model of the filter is finally

$$\boldsymbol{x}_{f,k+1} = \boldsymbol{x}_{f,k} + \Delta t \begin{bmatrix} \dot{\boldsymbol{q}} \\ \boldsymbol{M}^{-1}\left(\boldsymbol{\lambda} + \boldsymbol{J}_b^T \boldsymbol{f}_{ext} - \boldsymbol{h}\right) \\ 0 \end{bmatrix}_k + \boldsymbol{w}_k = \boldsymbol{f}_f(\boldsymbol{x}_{f,k}, \boldsymbol{u}_k, \boldsymbol{w}_k). \tag{5.15}$$

For the prediction step with (5.15) the model noise $\boldsymbol{w}_k$ is set to zero. Some further notes to the model are given in the following:

- The contact forces $\boldsymbol{\lambda}$ are due to the unilateral spring only C^0-continuous. Therefore a regularization term has to be used to ensure a continuous gradient $\boldsymbol{A}_k$.
- The model noise is assumed to act on the state with the matrix $\boldsymbol{W} = \boldsymbol{E}$.
- The taskspace trajectories are treated as input vector $\boldsymbol{u}_k = \boldsymbol{w}_{mod}(t_k)$.

The output model is set to full state measurement

$$\boldsymbol{y}_{f,k} = \boldsymbol{x}_{f,k} + \boldsymbol{v}_k \tag{5.16}$$

and direct measurement noise for $\boldsymbol{y}$ ($\boldsymbol{V} = \boldsymbol{E}$). The converted IMU orientation is used as measurement for $\varphi_{x,m}$ and $\dot{\varphi}_{x,m}$. For the state z_m and $\dot{z}_m$ the provided absolute acceleration $\ddot{z}_m$ has to be numerically integrated twice. The noise included in $\ddot{z}_m$ and the error of the numerical integration will cause a drift of the estimated vertical translation and velocity which has to be considered in the filter. The numerical drift can be reduced by using an explicit trapezoidal integration rule using the quantities of the filter output $(\hat{z}_k, \hat{\dot{z}}_k)$

$$\dot{z}_{m,k+1} = \hat{\dot{z}}_k + \frac{\Delta t}{2}(\ddot{z}_{m,k} + \ddot{z}_{m,k-1}), \tag{5.17}$$

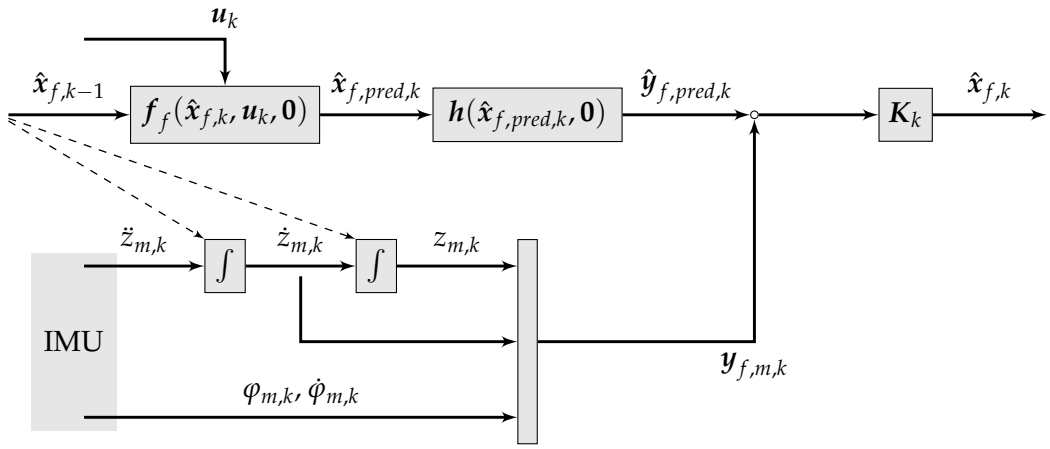


Figure 5.4: Overview of the estimation process for one time step k.

$$z_{m,k+1} = \hat{z}_k + \frac{\Delta t}{2}(\dot{z}_{m,k+1} + \dot{\hat{z}}_k). \tag{5.18}$$

Note that this corresponds to an indirect filter approach which is often used in inertial navigation systems (Maybeck 1979, p.294). The overall estimation process for one time step k is shown in Figure 5.4. The update of the filter gain $\boldsymbol{K}_k$ is not depicted in the scheme. It is computed with the covariance matrices that have in the final implementation only non-zero values for the variances

$$\begin{aligned} \boldsymbol{Q}_f &= \text{diag}(10^{-3}, 10^{-4}, 2 \cdot 10^{-2}, 5 \cdot 10^{-4}, 10), \\ \boldsymbol{R}_f &= \text{diag}(1 \cdot 10^{-2}, 5 \cdot 10^{-4}, 5 \cdot 10^{-2}, 0.25), \end{aligned} \tag{5.19}$$

in the x and y direction. Setting all covariance entries to zero means that a state does not influence other states. The assumption is not exact for the problem but reduces variables that have to be determined.

5.2.3 Observability of the Nonlinear System

The local observability of general nonlinear systems can be verified with a rank condition similar to the one for linear systems (Hermann and Krener 1977). The criterion uses the $n-1$ derivatives of the output $\boldsymbol{y} = \boldsymbol{h}(\boldsymbol{x})$ where n is the dimension of the state $\boldsymbol{x}$. For the continuous-time system $\dot{\boldsymbol{x}} = \boldsymbol{f}_c(\boldsymbol{x}, \boldsymbol{u})$ these derivatives are

$$\dot{\boldsymbol{y}} = \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}} \boldsymbol{f}_c = \boldsymbol{h}_1(\boldsymbol{x}, \boldsymbol{u}) \tag{5.20}$$

$$\ddot{\boldsymbol{y}} = \frac{\partial \boldsymbol{h}_1}{\partial \boldsymbol{x}} \boldsymbol{f}_c + \frac{\partial \boldsymbol{h}_1}{\partial \boldsymbol{u}} \dot{\boldsymbol{u}} = \boldsymbol{h}_2(\boldsymbol{x}, \boldsymbol{u}, \dot{\boldsymbol{u}}) \tag{5.21}$$

$$\dddot{\boldsymbol{y}} = \frac{\partial \boldsymbol{h}_2}{\partial \boldsymbol{x}} \boldsymbol{f}_c + \frac{\partial \boldsymbol{h}_2}{\partial \boldsymbol{u}} \dot{\boldsymbol{u}} + \frac{\partial \boldsymbol{h}_2}{\partial \dot{\boldsymbol{u}}} \ddot{\boldsymbol{u}} = \boldsymbol{h}_3(\boldsymbol{x}, \boldsymbol{u}, \dot{\boldsymbol{u}}, \ddot{\boldsymbol{u}}) \tag{5.22}$$

$$\vdots$$

The local observability condition in the region of a set point $(\boldsymbol{x}_a, \boldsymbol{u}_a, \dot{\boldsymbol{u}}_a, \dots)$ is then verified with the rank of the matrix

$$\text{rank} \begin{bmatrix} \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}} \\ \frac{\partial \boldsymbol{h}_1}{\partial \boldsymbol{x}} \\ \vdots \\ \frac{\partial \boldsymbol{h}_{n-1}}{\partial \boldsymbol{x}} \end{bmatrix}_{\boldsymbol{x}=\boldsymbol{x}_a, \boldsymbol{u}=\boldsymbol{u}_a, \dots} = n. \tag{5.23}$$

For the state $\boldsymbol{q} = \dot{\boldsymbol{q}} = \mathbf{0}$ and a static taskspace $\boldsymbol{w}_{mod}$ for standing with both feet on the ground, (5.23) is fulfilled. Other definitions of the output (e.g. $\boldsymbol{y}_f = [\varphi, \dot{\varphi}, \ddot{z}]^T$) would violate the criterion and cause the filter to diverge.

5.3 Model Error Compensation

This section adopts the idea to calculate a measure of the modeling error that occurs in the CoG planning due to the use of simplified models. It was initially proposed for motion planning by Kajita et al. (2003) and Takenaka et al. (2009b). It was shown in Chapter 2 that popular planning methods use the ZMP trajectory as input to determine a CoG trajectory with the LIPM. However this produces a different ZMP trajectory when replaying the resulting CoG motion with the full multibody model of the robot. The difference between planned and resulting ZMP can then be used to calculate a force driving the CoG in the right direction and reduce the discrepancy between CoG and ZMP for the full model. In the following this idea is transferred to the state estimation problem. For a given robot configuration $\boldsymbol{q}$ and its derivatives $\dot{\boldsymbol{q}}$, $\ddot{\boldsymbol{q}}$, one can compute the inverse dynamics of the full robot model which results in the total external forces/moments $\boldsymbol{\Lambda}$ and joint torques $\boldsymbol{\tau}$ acting on the robot. The resulting $\boldsymbol{\Lambda}$ can then be used to compute the overall force vector $\mathbf{F}_{total}$ and the moment $\mathbf{T}_{total} = [T_x, T_y, T_z]^T$ acting on the robot's current stance foot. Note that this computation is done for the planned motion of the robot which includes also the planned force distribution of the legs. A comparison of the horizontal contact moments T_x and T_y with the ideal ones ($T_{x,ideal}$, $T_{y,ideal}$) from the CoG planning produces a good measure of the dynamics error

$$\Delta T_x = T_{x,ideal} - T_x, \tag{5.24}$$

$$\Delta T_y = T_{y,ideal} - T_y. \tag{5.25}$$

Figure 5.5 shows resulting dynamics error trajectories. This error can be calculated in every time step for the desired trajectories and contact moments and fed in the state estimator as additional input ΔT_y to the model

$$\boldsymbol{M}\ddot{\boldsymbol{q}} + \boldsymbol{h} = \boldsymbol{\lambda} + \boldsymbol{J}_b^T \boldsymbol{f}_{ext} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \Delta T_y. \tag{5.26}$$

In order to save computational time, the output of the inverse kinematics of the local control is used and the accelerations $\ddot{\boldsymbol{q}}$ are generated with a low-pass filtered derivative (DT1-filter) of the form

$$G_{DT1}(s) = \frac{s}{Ts + 1}. \tag{5.27}$$

They are finally used to compute the inverse dynamics. In the next control cycle it can be used for the estimator. However, this introduces a time delay of one control cycle in $\Delta T_x(t)$ and $\Delta T_y(t)$. It is possible to include additional information of the multibody dynamics that acts on acceleration level with this extended input vector. One of the main advantages of including them in this way is that the filter's state and EoM dimensions remain small. A higher number of states increases the complexity of the filter and increases the difficulty to set appropriate entries in the covariance matrices $\boldsymbol{Q}_f$ and $\boldsymbol{R}_f$.

5.4 LIPM Based State Estimator

In Chapter 2 it was shown that considering the biped's multibody dynamics in lateral and sagittal direction as decoupled, keeping the CoG height at a constant value and linearizing, results in the well-known LIPM as an approximate model for the humanoid's

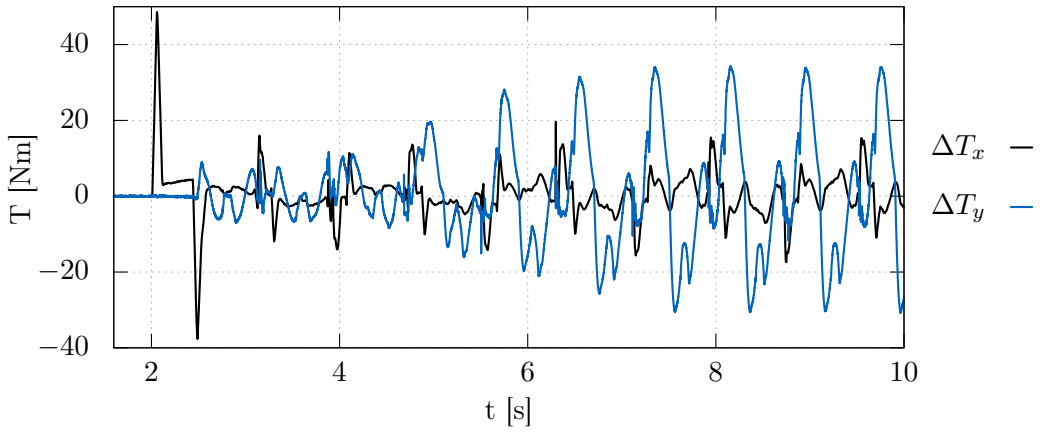


Figure 5.5: Resulting dynamics errors in both directions while the robot accelerates from standing to walking with 0.5 m/s.

dynamics. Restating the EoM in sagittal direction yields

$$\ddot{x}_c = \omega^2 \left(x_c - x_{zmp}\right) + \frac{1}{m} f_x + \frac{1}{mh} \Delta T_y. \tag{5.28}$$

Similar to the extended Kalman filter, the model (5.28) is extended by an external force f_x to include disturbances into the prediction process. The model error compensation with ΔT_y is also added. It is used as input $\boldsymbol{u}$ together with the measured ZMP position. The CoG position and velocity $\boldsymbol{y} = [x_c, \dot{x}_c]^T$ are the output. With the state vector $\boldsymbol{z}_l = [x_c, \dot{x}_c, f_x]^T$ the system can be written in statespace form

$$\dot{\boldsymbol{z}}_l = \begin{bmatrix} 0 & 1 & 0 \\ \omega^2 & 0 & 1/m \\ 0 & 0 & 0 \end{bmatrix} \boldsymbol{z}_l + \begin{bmatrix} 0 & 0 \\ -\omega^2 & -1/(mh) \\ 0 & 0 \end{bmatrix} \boldsymbol{u} = \boldsymbol{A}_l \boldsymbol{z}_l + \boldsymbol{B}_l \boldsymbol{u}, \tag{5.29}$$

$$\boldsymbol{y}_l = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \boldsymbol{z}_l = \boldsymbol{C}_l \boldsymbol{z}_l \tag{5.30}$$

where the input is defined as $\boldsymbol{u} = [x_{zmp}, \Delta T_y]^T$. The system (5.29) is observable. Using the ZMP position introduces the possibility to include disturbances from measurements at acceleration level in the filter. This improves the prediction and reduces a possible time-delay. Taking into account the absolute position and orientation of the feet, measured forces and torques are used to calculate the ZMP position of the robot ${}_I\boldsymbol{r}_{zmp} = [x_{zmp}, y_{zmp}, 0]^T$. The calculation is done in world coordinates using the FTS information of both feet and exploiting the full kinematic model of the robot. First, the force and torque vector of each foot ($\boldsymbol{F}_i$, $\boldsymbol{T}_i$) is transformed into the world FoR. These vectors are then used in a second step to calculate the overall torque wrt. the planning FoR origin $\boldsymbol{O}$ which is then used to calculate the ZMP position:

$${}_I\boldsymbol{F}_O = \sum_{i=0}^{1} {}_I\boldsymbol{F}_i\,, \qquad {}_I\boldsymbol{T}_O = \sum_{i=0}^{1} {}_I\boldsymbol{T}_i + {}_I\boldsymbol{r}_{Oi} \times {}_I\boldsymbol{F}_i\,, \tag{5.31}$$

$$\boldsymbol{O} = {}_I\boldsymbol{T}_O + {}_I\boldsymbol{r}_{0,ZMP} \times {}_I\boldsymbol{F}_O \Rightarrow {}_I\boldsymbol{r}_{O,ZMP}\,. \tag{5.32}$$

A plot of the ZMP y position calculated from measurements while the robot starts walking with a step size of 0.4 m and step time of 0.8 s is shown in Figure 5.7. This is different to the method presented in Stephens (2011) that uses only measured external vertical forces to calculate a force distribution between the two feet that are modeled as point

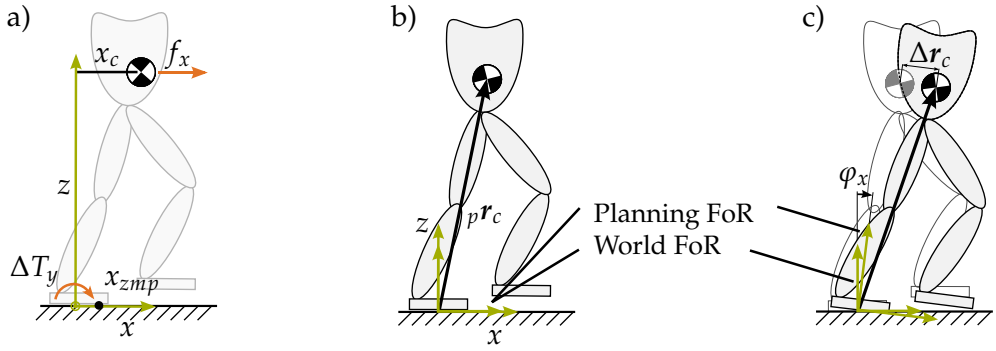


Figure 5.6: LIPM with external force and additional model error compensation ΔT_y (a). Planning and world FoR in the ideal case (b) and with an inclination error $\Delta\varphi_x$ (c) which results in a CoG error $\Delta \boldsymbol{r}_c$.

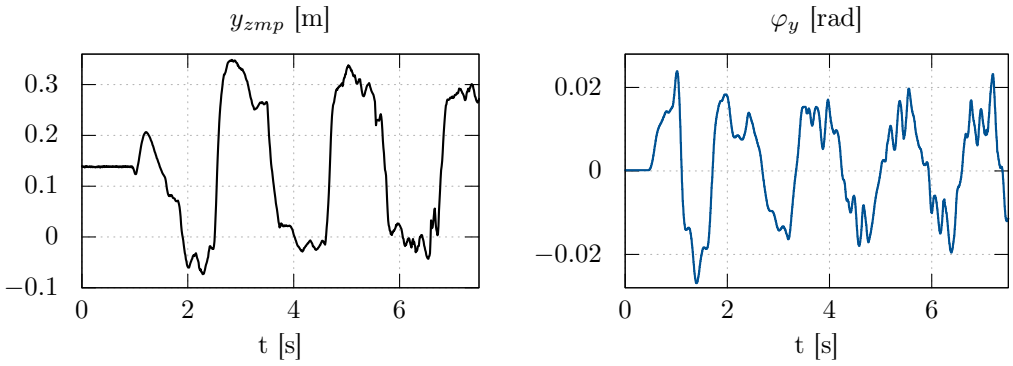


Figure 5.7: Example measurement of ZMP position and inclination error in lateral direction (unfiltered data).

contacts. This means that their computed ZMP position can not move when only one leg is on the ground while the position computed from (5.31) and (5.32) moves continuously and provides more information.

The filter additionally requires a measurement of $\boldsymbol{y}_l$ which consists of the absolute CoG position and velocity. These values are calculated with the kinematic assumption that the stance foot doesn't slip and the measured inclination errors from the IMU result from a rotation about the middle of the stance foot. The CoG position in world coordinates ${}_0\mathbf{r}_c$ is obtained from the CoG position described in the planning FoR ${}_T\boldsymbol{r}_c$ and the inclination errors φ_x, φ_y

$$ {}_I\boldsymbol{r}_c = \mathbf{A}_{IT}(\varphi_x, \varphi_y)\, {}_T\boldsymbol{r}_c\,(\boldsymbol{q}_J). \tag{5.33} $$

Here, the matrix $\mathbf{A}_{IT}$ denotes a rotation matrix due to φ_x, φ_y and transforms from planning to world FoR (cf. Section 4.5). The transformation is also depicted in Figure 5.6. $\boldsymbol{q}_J$ describes the robot's current joint configuration. The CoG velocity is calculated by differentiating (5.33) wrt. time and using the measured inclination rates $\dot{\varphi}_x$, $\dot{\varphi}_y$

$$ \begin{aligned} {}_I\dot{\boldsymbol{r}}_c =& \dot{\boldsymbol{A}}_{IT}\, {}_T\boldsymbol{r}_c + \mathbf{A}_{IT}\, {}_T\dot{\boldsymbol{r}}_c\,(\boldsymbol{q}_J, \dot{\boldsymbol{q}}_J) \\ =& \left(\frac{\partial \mathbf{A}_{IT}}{\partial \varphi_x}\dot{\varphi}_x + \frac{\partial \mathbf{A}_{IT}}{\partial \varphi_y}\dot{\varphi}_y\right) {}_T\boldsymbol{r}_c + \mathbf{A}_{IT}\, {}_T\dot{\boldsymbol{r}}_c\,(\boldsymbol{q}_J, \dot{\boldsymbol{q}}_J). \end{aligned} \tag{5.34} $$

The quantities from (5.33) and (5.34) determine the measurement output $\boldsymbol{y}_l$ completely. Finally a Kalman filter with the linear state space model (5.29) and (5.30) is used to produce the state estimate $\hat{\boldsymbol{z}}_l$. The covariance matrices are set to

$$ \boldsymbol{Q}_f = \mathrm{diag}(10^{-5}, 2\cdot 10^{-5}, 40), \quad \boldsymbol{R}_f = \mathrm{diag}(3\cdot 10^{-3}, 3\cdot 10^{-2}). $$

The resulting predicted CoG state ($\hat{x}_c$, $\dot{\hat{x}}_c$) is transformed back to a CoG state error $\Delta\hat{x}_c$, $\Delta\dot{\hat{x}}_c$ which is then used to compute predicted values for the inclination and inclination rates.

5.5 Comparison and Analysis

5.5.1 Filter Performance

To show the performance of the developed nonlinear state estimator of Section 5.2, a simulation experiment is performed. The robot accelerates in this experiment to a walking speed of 0.5 m/s and receives an external disturbance in x direction. The peak force is 100 N. Figure 5.9 shows the resulting measurement and filter output of the state. There are basically three main characteristics that can be seen in these plots: (1) the estimated vertical translation $\hat{z}$ and its velocity do not drift. (2) the filter follows the measurement of $\dot{\hat{z}}$ and $\hat{\varphi}_x$ almost perfectly and (3) the trajectory of the inclination rate is smoothed. The last point was the key goal of the proposed state estimator. A comparison of the results with a naive implementation which uses a butterworth low-pass filter (10 Hz cutoff frequency) for the inclination rate is shown in Figure 5.8. It can be seen that the estimator filters oscillations better and has less time delay compared to the low-pass filter. Figure 5.10 shows the additionally obtained estimated external force $\hat{f}_{ext,x}$. The estimated force continuously changes for the filter which is affected due to model errors, contact dynamics and joint errors. Nevertheless, using this state the filter is able to compensate for those effects. Additionally it is able to detect an unusual high error and react to it in the right way when the external disturbance occurs (with some time delay). Note that the focus is not to estimate the correct external force and not to use it in the controller. The time evolution of the filter cost function (5.6) is depicted in Figure 5.10 on the right hand side.

Figure 5.11 shows the estimation result for the example measurement of the introduction. The result is a smoothed trajectory. Nevertheless, the oscillations right before $t = 206$ s are unwanted. The robot has a positive inclination with almost constant mean value in a global point of view. The local position variations cause strong oscillations of the inclination rate. This can effect undesired behavior when the estimated state is used to adapt the robot's motion.

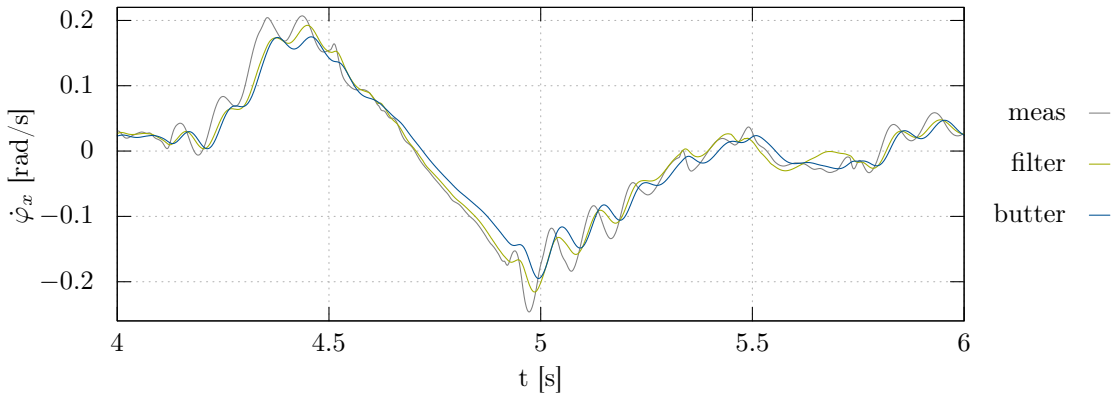


Figure 5.8: Estimator result in simulation: comparison of estimated inclination rate $\dot{\varphi}_x$ (filter) and low-pass filter result (butter).

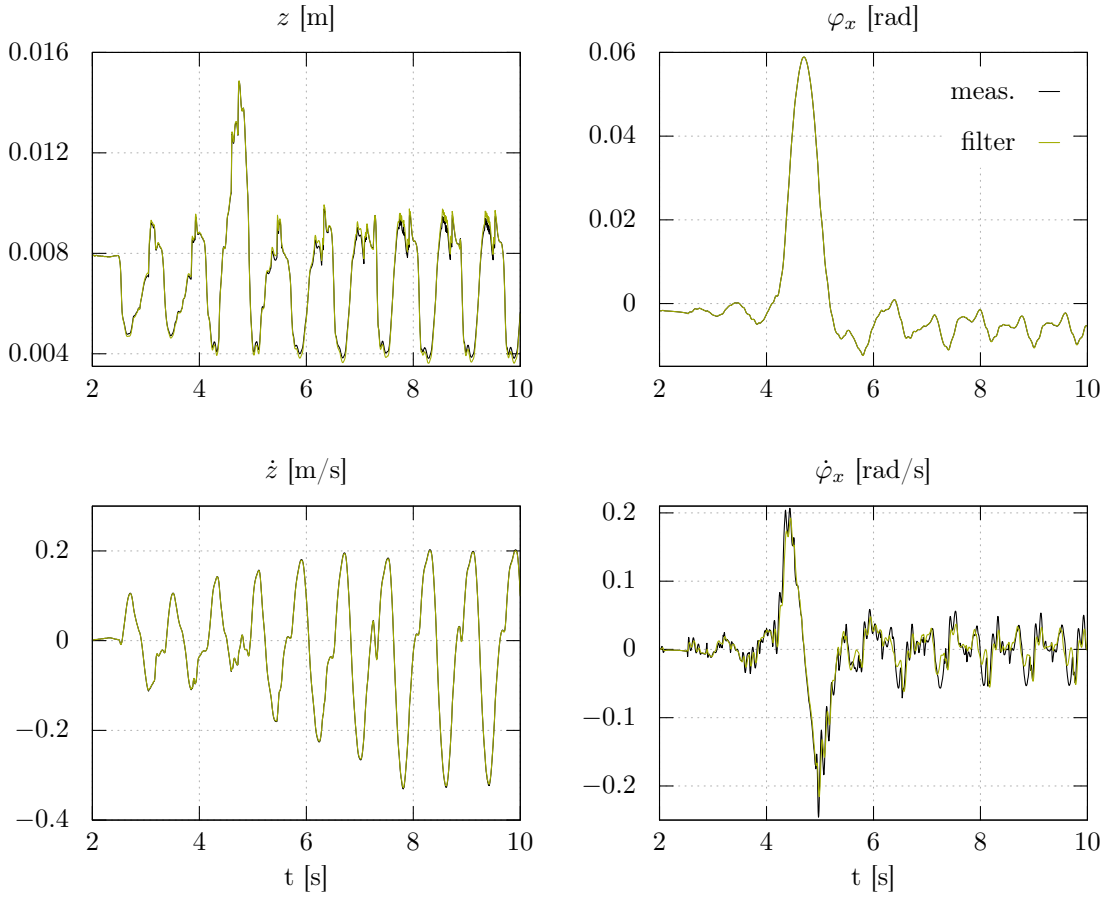


Figure 5.9: Estimation result in simulation: state measurement (meas) and filter output (filter).

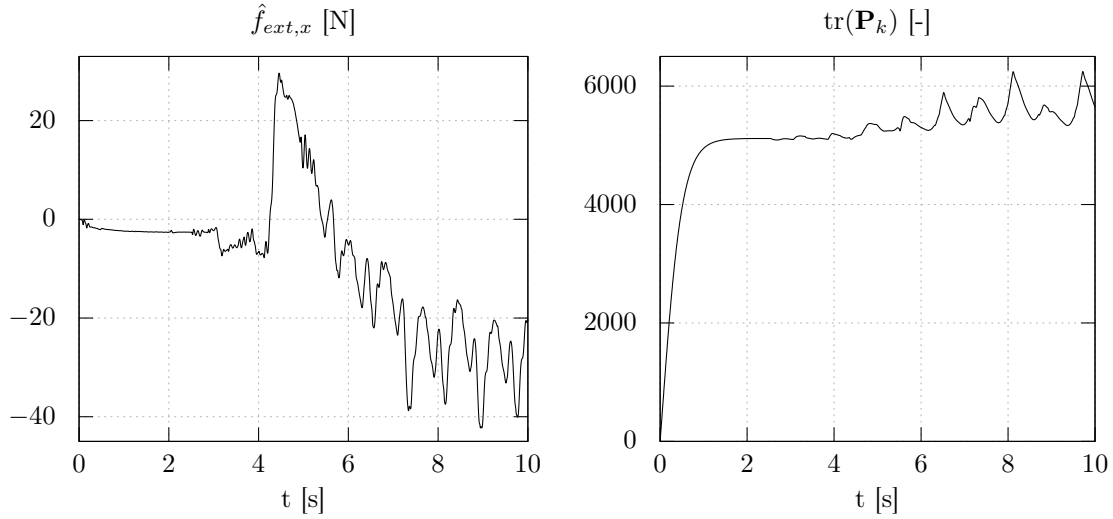


Figure 5.10: Estimation result in simulation: predicted external force (left) and trace of the estimation error covariance (right).

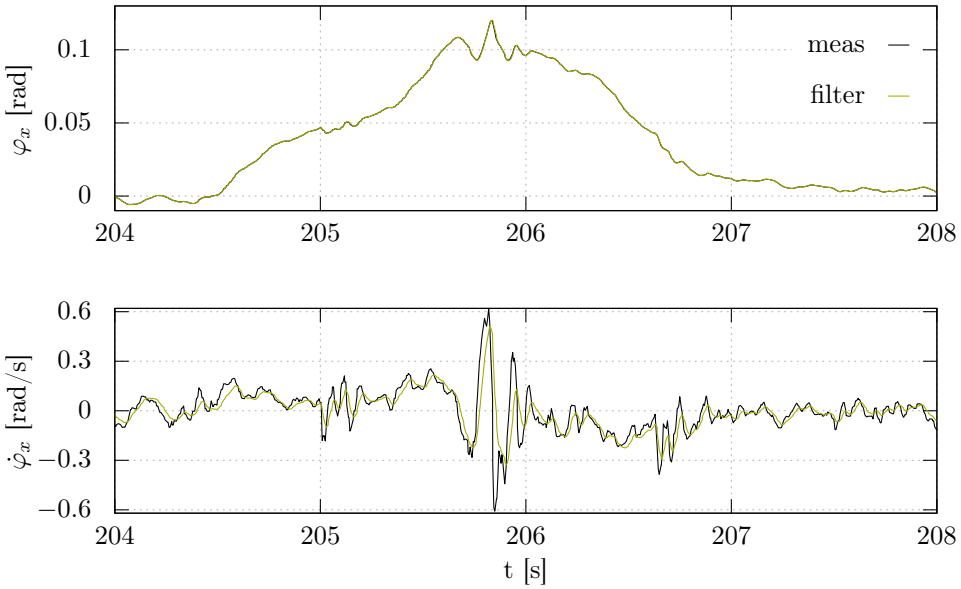


Figure 5.11: Estimator result for measurement data: inclination measurement (meas) and filter output (filter).

5.5.2 Error Analysis

Several different simulation scenarios are performed to analyze proposed state estimators. The different setups include cases for model errors and unknown disturbances. All filters are implemented in the walking control system of LOLA and are executed with 1 kHz. They are the presented extended Kalman filter without model error compensation (Ext.) and with model error compensation (Ext.Model) as well as the LIPM based filter (LIPM). The results are additionally compared with a simple implementation which uses a butterworth low-pass filter (10 Hz cutoff frequency, second order) for the inclination rates $\dot{\varphi}_x$, $\dot{\varphi}_y$ (Simple). All simulation tests are done with the multibody simulation of LOLA which considers compliant unilateral contacts and the motor dynamics combined with the joint control loop. Adequate sensor models enable a realistic simulation behavior. The robot is walking with a step length of 0.4 m and a step time of 0.8 s in all cases (except the last one). The following setups have been analyzed:

- walking undisturbed without errors (std)
- disturbance with a push in the x direction and a maximum force of 70 N that corresponds to an 10 Ns impulse at $t = 5.8$ s ($f_x = 70$)
- disturbance with a push in the y direction and a maximum force of 70 N that corresponds to an 10 Ns impulse at $t = 5.8$ s ($f_y = 70$)
- robot steps on an unknown board that is 0.04 m high (board)
- filter model mass error of 5% (Δm)
- walking with a maximum step length of 0.6 m.

Figure 5.12 shows the RMSE of the four different filter methods within these simulations. The butterworth filter is outperformed in every scenario for the x direction whereas the LIPM based and the extended Kalman filter with model error compensation perform similarly. For the y direction there is no clear preference visible. For the unknown push in y direction and the board scenario the low-pass filter is worse than the Kalman filters

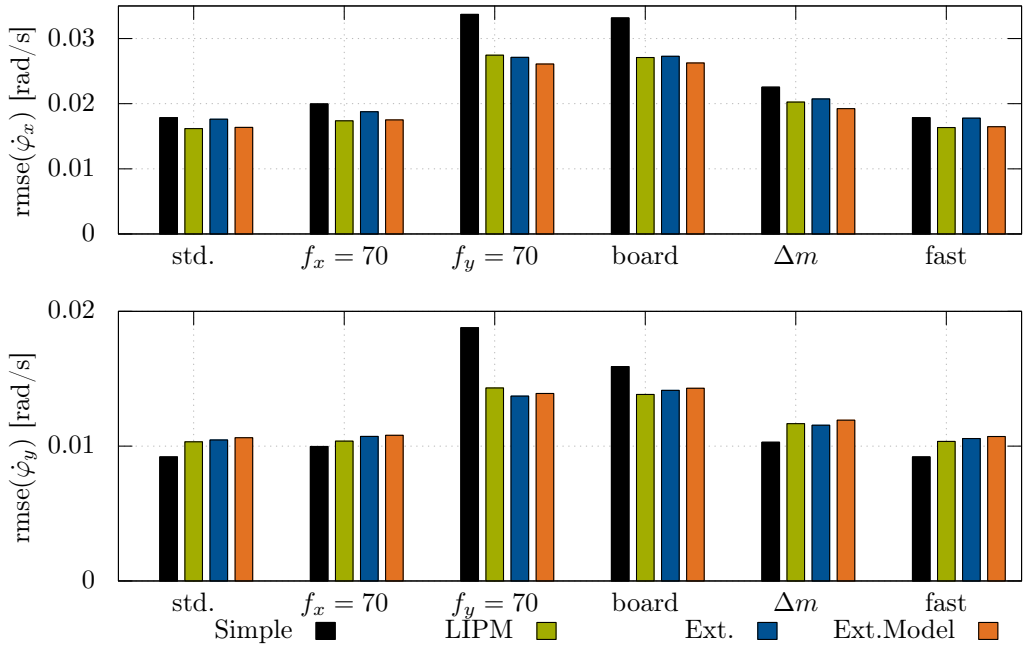


Figure 5.12: RMSE of the estimated inclination rates for different disturbance cases and different filters.

while for other cases it seems to be the best choice. However this is also related to the tuning of the covariance matrices $\boldsymbol{Q}_f$ and $\boldsymbol{R}_f$. For final experiments the extended Kalman filter with model error compensation will be used.

5.6 Chapter Summary

The outcome of this chapter are different state estimation methods for humanoid robots. They aim to extract information of the robot's global trend from IMU data and desired motion. The main goal is to remove high frequent oscillations that occur in the inclination rates of the upper body. A filter based on the extended Kalman filter is presented. It uses the model of Chapter 4 and extends it by an unknown external force acting at the robot's CoG. This external force is regarded as a "trash" variable for unknown errors. It was shown that the filter outperforms a low-pass filter in terms of smoothing and while maintaining a small time delay.

A second filter is presented and compared to the nonlinear one which is based on the LIPM. It shows equivalent results for the presented simulation cases. However the nonlinear filter provides an additional estimate for the absolute vertical translation of the robot. Even though it is not used so far, it might be useful for example for an adaptive contact model. The final state estimate for the inclination and inclination rate is the input for trajectory adaptation methods of the following chapter.

Chapter 6

Model Predictive Trajectory Adaptation

6.1 Introduction

This chapter covers the usage of sensor data in trajectory generation. It is an established way in robotic systems to react to unknown situations when acting in a dynamic environment. Adapting the overall trajectories requires a predictive behavior since they describe future motion. Examples for necessary situations are newly detected environment information obtained from a vision system or the current (disturbed) state of the robot. A disturbed state of the robot means that it does not match the ideal planned motion which is caused by e.g. an external disturbance. The trajectory generation methods in this chapter consider the later problem. One important property of such situations is that they can occur spontaneously and the robotic system has to react to within a defined time. This means that trajectories have to be recalculated in *real-time*. The planning system has to provide updated trajectories with a certain frequency for a given time horizon. In the case of unknown disturbances this introduces the following challenges for suitable planning methods:

- The computation time of the trajectory planning method has to be sufficiently fast in order to meet the real-time requirement.
- The usage of disturbed states as initial values often introduces discontinuities in the commanded trajectories that can produce problems.

The chapter's classification into the model predictive trajectory adaptation is shown in Figure 6.1.

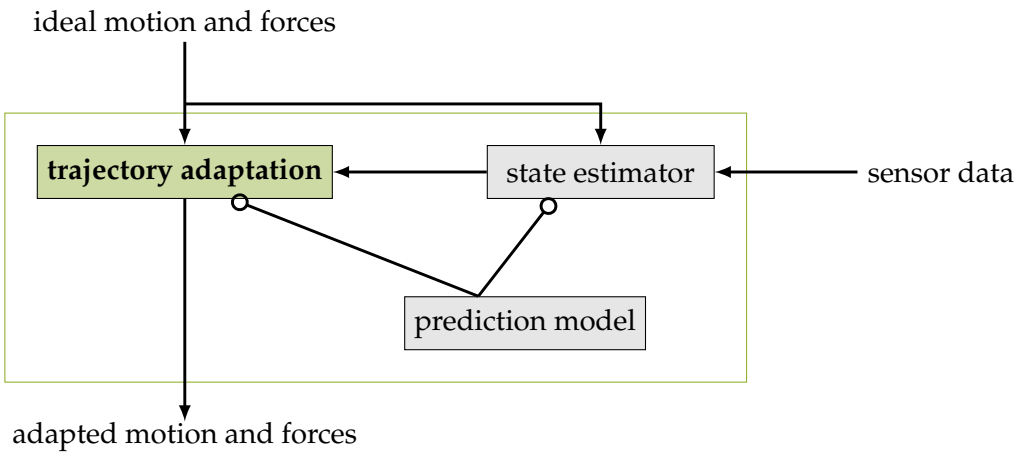


Figure 6.1: Chapter classification into the model predictive trajectory adaptation.

6.2 Related Work

For a literature overview of general online trajectory generation the reader is referred to Kröger (2010). Depending on the complexity of the used model, state and control constraints and the planning time horizon, the real-time requirement is a critical point. In the case of humanoid robots, the model has about 20-40 DoFs, switching contact states and constraints for contact forces. A common planning horizon is between 1-2 s while the available time to solve the planning problem is in the magnitude of milliseconds. Subsection 2.2.2 presented some basic approaches in the field of humanoid robots.

In many cases the trajectory planning problem is formulated as optimization where the robot's motion is designed such that a predefined cost function (optimization criterion) is minimized. Nishiwaki and Kagami (2006) state the CoG trajectory planning problem for the LIPM with the current measured state as optimal control problem. They generated a parameter database of the solution for all possible model parameters (they use different CoG heights). This database is then used during online trajectory generation with a moderate frequency of 50 Hz for the current measured CoG state as initial value. This rather pragmatic solution showed very promising results in experiments with the robot HRP2. The method presented in Wieber (2006) solves the optimal control problem for the CoG trajectory and next foot positions with a direct approach. In simulations they show that a continuous recomputation of the trajectories with the current state as initial value increases the stability of the robot while it receives an external disturbance.

The work in Dimitrov et al. (2008) discusses implementation details for such a method and how state constraints are considered in the online trajectory planning method. Recently, Naveau et al. (2017) published a method to extend this walking controller by taking into account convex obstacles as additional constraints. The robot *HRP2* can adjust footholds locally to avoid circular obstacles in experiments using this controller. They also proved their concept in simulation to reject disturbances. To the best of the author's knowledge, the combination of perturbations and obstacles has not been shown. Integration into a whole planning framework including a footstep planner is pending. Another approach is presented by Tassa et al. (2012) where the nonlinear optimal control problem for a multibody-model of a bipedal robot is solved almost in real-time. They solve their trajectory optimization with a Differential Dynamic Programming (DDP) like method (Li et al. 2004) that they call iLGQ and uses a simplified Hessian approximation. Their algorithm is similar to gradient methods for trajectory optimization (Graichen and Käpernick 2011) with the difference that they use second order derivative information. Nevertheless, the time to compute second-order derivatives increases drastically the overall computation time for the method. This can be avoided by using conjugate gradient search directions (Lasdon et al. 1967). They try to improve the quality of first-order gradient methods while the computation time is almost the same as for steepest descent methods. A comparison of mentioned indirect trajectory optimization methods can be found in Bryson (1969).

In contrast to above mentioned online-methods there is also a huge variety of offline trajectory optimization methods. In the following only a few concepts that were applied to humanoid robots are mentioned. Buschmann et al. (2005) presents a nonlinear parameter optimization method to calculate a walking pattern for bipedal robots. Two different optimization criteria for minimal taskspace accelerations and minimal joint torques are presented. They use the overall controlled multibody model of the biped JOHNNIE. The authors of Bessonnet (2004) formulate the walking pattern generation problem as dynamic trajectory optimization with a planar multibody model. Using Pontryagin's Minimum Principle they obtain a two-point BVP which they solve under several state and control constraints. There is also the possibility to consider only the overall linear

and angular momentum of the biped (Kuindersma et al. 2015) in order to calculate a feasible set of CoG motion and contact forces. The method has the advantage that multi-body effects are included while the decision variables are kept small. Unfortunately this requires to solve the inverse kinematics problem of the overall biped and rises the computation time drastically. A complex trajectory optimization that focuses on the solution for the right contact states is presented in Posa et al. (2013). Some research groups try to tackle the problem of too high computation times by database concepts (Koch et al. 2015). The walking parameters are discretized by few values and an optimal control problem for minimal torque motion is solved. Movement primitives are computed with machine learning techniques from this training data and are afterwards used to interpolate for walking parameters in between the discretization. To this date and the author's knowledge only forward walking is considered. The problem of the increasing database size which will be necessary to cover all walking situations is not yet discussed in detail.

The concepts presented in this chapter do not try to solve the overall motion generation in one huge problem. Instead simplified models are used which allow to solve the optimization problem in real-time.

6.3 Problem Description

The main goal of the presented methods is to take the robot's current state into account when trajectories are planned. This has to be done with special consideration to the stability of the system, i.e. falling of the robot is avoided. Underactuation and constraints for the contact forces have to be considered in order to obtain a feasible motion that stabilizes the robot. The input is the estimated current state of the robot, namely the absolute inclination and inclination rate (see Chapter 5). To obtain a stabilizing effect one has to choose parts of the robot's motion that will be adapted. As one can see in (2.1) the unactuated DoFs $\boldsymbol{q}_T$ are mainly controlled via the contact forces which can be modified by changing the motion of the joints $\boldsymbol{q}_J$. Next to the existing force control that modifies the stance foot's orientation and vertical position, a modification of future stance foot horizontal positions ($\Delta \boldsymbol{L} = [\Delta L_x, \Delta L_y]^T$) and a modification of the overall planned horizontal CoG trajectories ($\Delta x_b(t), \Delta y_b(t)$) will be added. A modification of next footstep positions changes the lever arm of the contact forces and shifts consequently the limits for feasible forces. Moving it in the right way can be used to stabilize the robot even if it rotates around one edge of the foot. The CoG trajectories directly influence the magnitude of the contact forces $\boldsymbol{\Lambda}$ which can be also used to control $\boldsymbol{q}_T$. In order to compensate for inclination errors and avoid too early or late contacts of the feet three additional variables are introduced. Those are the final swing foot height Δz_{sw} and orientation $\Delta c_{\varphi_x}, \Delta c_{\varphi_y}$. They are determined using geometric considerations. An overview of the introduced trajectory modifications is shown in Figure 6.2.

The models presented in Chapter 4 allow to produce a prediction for the inclinations. This enables to formulate the trajectory generation such that the absolute inclination error of the robot is minimized for a defined time horizon. The minimization is done wrt. an ideal pose. Following this approach all disturbances are treated as initial inclination errors. As a consequence this allows to stabilize only disturbances that cause a deviation of this feedback variable. For rough terrain locomotion and external pushes this showed to be sufficient information. In situations that would require more information a resolution of the foot contacts or an estimation of the remaining part of $\boldsymbol{q}_T$ could be added.

Two different problems are defined where the first includes only modifications of foot trajectories while the second one extends it by the CoG modification. This results finally in two mathematical problems that are described in the following. They are solved decoupled for x and y direction with planar models and can be easily applied for the x

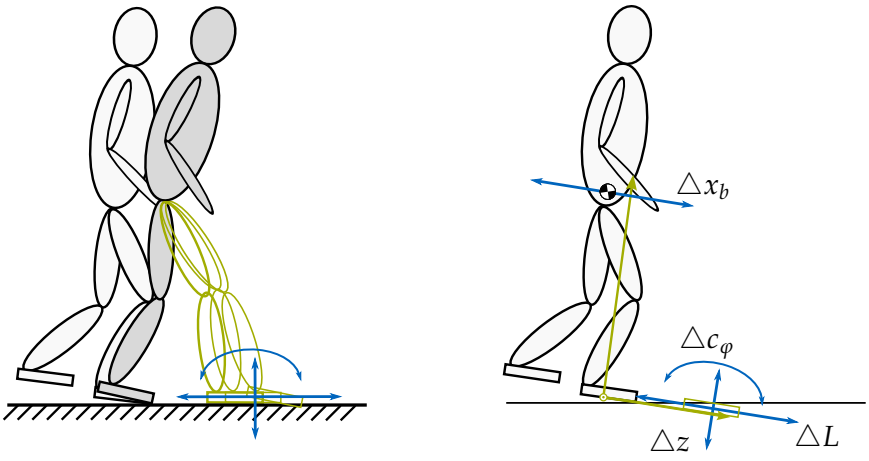


Figure 6.2: Overview of introduced trajectory modifications. On the right hand side all swing foot trajectory variables and the CoG modification are shown in the planning FoR.

and y directions by using the according indices.

6.3.1 Problem A

Problem A includes the next horizontal position where the swing foot will land as free variable $p = \Delta L$. The free variables for the final swing foot height Δz_{sw} and orientation $\Delta c_{\varphi_x}, \Delta c_{\varphi_y}$ are also included. The state prediction $\boldsymbol{z}_r(t)$ is calculated with the reduced prediction model (6.2). The goal of the variable p is the minimization of a quadratic cost function $J_A(p, \boldsymbol{z}_r)$. It depends on the state prediction for a given time horizon $[t_a,\, t_e]$ and has the (positive definite) weighting matrices $\boldsymbol{S}_z$, $\boldsymbol{S}_p$, $\boldsymbol{Q}$. Those matrizes are chosen such, that the inclination errors φ are mainly penalized. The initial value $\boldsymbol{z}_r(t_a)$ is given and the final state value is not fixed but included via final costs in J_A. End conditions are omitted in the formulation due to the fact that the problem has to be kept small to satisfy real–time requirements. Another reason is that a desired final value can not be specified easily since it is not mandatory or possible to come back to an undisturbed state after only one step. If this is the case and the robot can not come back to an undisturbed state after only one step, next footsteps will be modified until the robot reaches the ideal state. The variables are bounded to kinematically feasible values described by the set $\mathcal{P}$. The determination of this set is part of Section 6.6. The resulting optimization problem states as follows:

$$\min_p J_A(\boldsymbol{z}_r, p) = \min_p \Bigg[\underbrace{\frac{1}{2}\boldsymbol{z}_r^T(t_e)\boldsymbol{S}_z\boldsymbol{z}_r(t_e) + \frac{1}{2}p^T\boldsymbol{S}_p p}_{\bar{s}(\boldsymbol{z}_r(t_e),p)} + \int_{t_a}^{t_e} \underbrace{\frac{1}{2}\boldsymbol{z}_r^T\boldsymbol{Q}\boldsymbol{z}_r}_{\bar{h}(\boldsymbol{z}_r,p,t)} \,\mathrm{d}t \Bigg] \tag{6.1}$$

$$\text{s.t.} \quad \dot{\boldsymbol{z}}_r = \boldsymbol{f}_r(\boldsymbol{z}_r, p, t)\,, \tag{6.2}$$

$$\boldsymbol{z}_r(t_a) = \boldsymbol{z}_{r,a}\,, \tag{6.3}$$

$$p \in \mathcal{P}. \tag{6.4}$$

6.3.2 Problem B

Problem B includes the next horizontal position where the swing foot will land ($p = \Delta L$) as well as the CoG modification trajectory $\Delta x_b(t) = f(u(t))$. The free variables for the final swing foot height Δz_{sw} and orientation $\Delta c_{\varphi_x}, \Delta c_{\varphi_y}$ are included as well. The state prediction $\boldsymbol{z}_{r,ext}(t)$ is calculated with the reduced prediction model that includes the trajec-

tory u (6.7). The goal of p and $u(t)$ is to minimize a quadratic cost function $J_B(p, \boldsymbol{z}_{r,ext}, u)$ that depends on the state prediction for a given time horizon $[t_a,\ t_e]$ and has the (positive definite) weighting matrices $\boldsymbol{S}_z$, $\boldsymbol{Q}$, $\boldsymbol{R}$. The cost function is chosen such, that it mainly penalizes the inclination errors φ. The initial value $\boldsymbol{z}_{r,ext}(t_a)$ is given and the final state value is free but it is included via final costs in J. The variables and trajectory u are bounded to kinematically feasible values described by $\mathcal{P}$ and $\mathcal{U}$. The resulting dynamic optimization problem yields:

$$\min_{p,u} J_B(\boldsymbol{z}_{r,ext}, p, u) = \min_{p,u} \Bigg[\underbrace{\frac{1}{2}\Delta \boldsymbol{z}_{r,ext}^T(t_e)\boldsymbol{S}_z\Delta \boldsymbol{z}_{r,ext}(t_e)}_{s(\boldsymbol{z}_{r,ext}(t_e),p)} + \int_{t_a}^{t_e} \underbrace{\frac{1}{2}\Delta \boldsymbol{z}_{r,ext}^T\boldsymbol{Q}\Delta \boldsymbol{z}_{r,ext} + \frac{1}{2}u^T\boldsymbol{R}u}_{h(\boldsymbol{z}_{r,ext},p,u,t)} \,\mathrm{d}t \Bigg] \tag{6.5}$$

$$\text{s.t.} \quad \dot{\boldsymbol{z}}_{r,ext} = \boldsymbol{f}_{r,ext}(\boldsymbol{z}_{r,ext}, p, u, t)\,, \tag{6.6}$$

$$\boldsymbol{z}_{r,ext}(t_a) = \boldsymbol{z}_{r,ext,a}\,, \tag{6.7}$$

$$p \in \mathcal{P}, \tag{6.8}$$

$$u \in \mathcal{U}. \tag{6.9}$$

6.4 Foot Trajectory Modifications

This section discusses an approach to solve problem A for the bipedal robot LOLA and is based on Wittmann et al. (2015b). A solution for the foot position p is presented as well as the determination of the remaining swing foot height and orientation. A continuous replanning of the modification trajectories with the resulting parameters is finally described.

6.4.1 Foot Position Optimization

A solution for the foot position p of the optimization problem (6.1) to (6.4) is computed with a direct shooting method (Betts 1998). It is a widely used and easy to implement method which can be an effective procedure for problems that have only few optimization variables. There are many successful implementations in space applications like launch and orbit transfer problems. The direct shooting optimization uses the following procedure: The initial value problem (6.2) and (6.3) is solved for an initial guess of p^0. This has to be done by numerical integration of the first order Ordinary Differential Equation (ODE) as described in Subsection 4.3.5. The resulting trajectory $\boldsymbol{z}_r(t, p^0)$ can then be used to compute the cost function $J_A(\boldsymbol{z}_r(t, p^0), p^0)$. This way problem A is converted into the static optimization problem

$$\min_{p\in\mathcal{P}} J_A(p) \tag{6.10}$$

which is unconstrained as long as the variable p remains inside the allowable set $\mathcal{P}$. The minimum can be recursively computed with nonlinear programming methods like steepest descent or Newton's method (Nocedal and Wright 2004). In this work Newton's method will be used to find the optimal p for (6.10). In the unconstrained case the update rule is

$$p^{k+1} = p^k - \left(\nabla_p^2 J_A(p^k)\right)^{-1} \nabla_p J_A(p^k) = p^k + \Delta p^k \tag{6.11}$$

where $\nabla_p J_A(p^k)$ and $\nabla_p^2 J_A(p^k)$ describe the gradient and Hessian of the cost function. In the following the search direction of optimization iteration k is abbreviated with Δp^k. To compute p^k, first and second order derivative information is required. The problem is formulated in Lagrange form (the cost function involves an integral) and the system equation includes discontinuous dynamics. Hence numerical derivatives with the central difference formula are used:

$$\nabla_p J_A(p^k) \approx \frac{J_A(p^k + \delta) - J_A(p^k - \delta)}{2\delta}, \tag{6.12}$$

$$\nabla_p^2 J_A(p^k) \approx \frac{J_A(p^k + \delta) - 2J_A(p^k) + J_A(p^k - \delta)}{\delta^2}. \tag{6.13}$$

This requires three integrations of (6.2) and (6.3) for each iteration of the shooting method. To reduce the computation time, the ideal trajectories $\boldsymbol{r}_b(t), \boldsymbol{r}_{f1}(t), \boldsymbol{r}_{f2}(t)$ are evaluated at the time steps for the prediction horizon in advance and are stored into an array which is then used in the integration of the prediction model. Those are known prior to the optimization because an integration scheme with fixed time steps is used. Note that positive definiteness of the hessian has to be ensured in each iteration. The term $\frac{1}{2} p^T \boldsymbol{S}_p p$ has a positive influence to this restriction since its second derivative is $\boldsymbol{S}_p$ which is a positive definite matrix.

To ensure sufficient decrease and no increase of the cost function the computed search direction Δp^k is multiplied with a scalar factor α^k. This factor is determined in a one-dimensional line-search algorithm that tries to fulfill the Wolfe condition (Nocedal and Wright 2004, pp.33ff)

$$J_A(p^k + \alpha^k \Delta p^k) \leq J_A(p^k) + c\alpha^k \left(\nabla_p J_A(p^k)\right)^T \Delta p^k \tag{6.14}$$

with the contraction factor $c \in [0, 1]$. In combination with Newton's method one popular algorithm to find a feasible α^k for (6.14) is the Backtracking line-search. Starting with $\alpha^k = \alpha_{init}$ it is decreased with a factor $\rho \in [0, 1]$ until (6.14) is fulfilled or a maximum number of iterations is reached (cf. Algorithm 2). This is essential as every iteration requires an additional integration of the system in order to evaluate the cost function. The adapted update rule for the variable p is

$$p^{k+1} = p^k + \alpha^k \Delta p^k. \tag{6.15}$$

The above described procedure requires a maximum of six evaluations of the prediction model (6.15) over the time horizon $[t_a, t_e]$ for each iteration. This underlines the importance of a good initial solution p^0 if the maximum number of optimization iterations has to be kept small to limit the computational time.

Algorithm Initialization

The initialization tackles the mentioned issue first by providing a good initial value p^0 and second by shortening the time horizon $[t_a, t_e]$ as much as possible. Supposing that a chang of the swing foot trajectory has little influence to the prediction model's state $\boldsymbol{z}_r$ until it touches the ground it is possible to neglect this deviations during one optimization. Current implementation integrates the model with the ideal foot trajectory from current time t_0 until the end of the actual step t_1 and uses the predicted state $\boldsymbol{z}_r(t_1)$ as initial value of the optimization problem, i.e. $t_a = t_1$ and $\boldsymbol{z}_r(t_a) = \boldsymbol{z}_r(t_1)$. The final time t_e is set to the end of next step (t_2). This way, one obtains a constant time horizon of the optimization and considers only the part with the modified stance foot on the ground. The predicted state at t_1 is additionally used to calculate an initial solution Δp^0. The initial step length is

Algorithm 2 Backtracking Line-Search

function BACKTRACKING($J(p)$, Δp, $\nabla_p J(p)$, c, ρ)
  j $\leftarrow 0$
  initialize $\alpha_j = \alpha_{init}$
  repeat
    solve $z_r(p + \alpha_j \Delta p)$
    evaluate $J(p + \alpha_j \Delta p)$
    converged $\leftarrow \left(J(p + \alpha_j \Delta p) < J(p) + c\alpha_j \left(\nabla_p J(p) \right)^T \Delta p \right)$
    if (converged = false) **then**
      $\alpha_{j+1} \leftarrow \rho \alpha_j$
      $j \leftarrow j + 1$
  until (converged = true **or** j $>$ max iterations)
  return α_j

computed using the predicted error of the absolute CoG position $\Delta\, {}_I x_c\,(t_1)$ and velocity $\Delta\, {}_I \dot{x}_c\,(t_1)$ in a linear heuristic

$$p^0 = k_{step,a} \left(\Delta\, {}_I x_c\,(t_1) + \frac{1}{\omega} \Delta\, {}_I \dot{x}_c\,(t_1) \right) \tag{6.16}$$

with a manually tuned factor $k_{step,a} \in [0, 1]$. The CoG errors are determined with the relation (4.9) for all three masses

$$\Delta\, {}_I x_c\,(t_1) = \frac{c_\varphi(t_1) - 1}{m} \sum_{j \in N_B} m_j\, {}_T x_j\,(t_1) + \frac{s_\varphi(t_1)}{m} \sum_{j \in N_B} m_j\, {}_T z_j\,(t_1), \tag{6.17}$$

$$\begin{aligned} \Delta\, {}_I \dot{x}_c\,(t_1) = {} & \frac{c_\varphi(t_1) - 1}{m} \sum_{j \in N_B} m_j\, {}_T \dot{x}_j\,(t_1) - \frac{(s_\varphi(t_1) \dot{\varphi}_x}{m} \sum_{j \in N_B} m_j\, {}_T x_j\,(t_1) \\ & + \frac{s_\varphi(t_1)}{m} \sum_{j \in N_B} m_j\, {}_T \dot{z}_j\,(t_1) + \frac{c_\varphi(t_1) \dot{\varphi}_x}{m} \sum_{j \in N_B} m_j\, {}_T z_j\,(t_1). \end{aligned} \tag{6.18}$$

This heuristic is motivated by the unstable respectively divergent solution of the LIPM (Matsumoto et al. 2004). It is computed with the idea presented in Subsection 2.3.3 to use the predicted value $q(t_1)$ at the end of current step which is additionally scaled with $k_{step,a}$ in order to account for the stabilizing effect of the local walking control. The divergent solution is obtained from the LIPM without a feedback control. Consequently the predicted value is too conservative for the real robot's behavior.

Example Optimization

This part presents details of an optimization example to illustrate the optimization and its procedure. A case is chosen in which the robot is walking in place while it is pushed by an external force. Figure 6.3 shows at one time instant the resulting footstep modification ΔL_x for 4 iterations and the evaluated cost function for these values. For visualization the cost landscape is sampled for discrete values of ΔL_x with the cost function weights shown in Table 6.1. As one can see, the optimization is able to converge almost to the minimum of J_A in these iterations. The predicted evolution of the absolute inclination φ_x for the corresponding iterations is shown in Figure 6.4. As mentioned before, an initial integration is performed from t_0 to t_1 and the optimization considers the time interval $[t_a, t_e] = [t_1, t_2]$. The predicted inclination approaches its stable and desired value with increasing iteration number. Nevertheless it is clear that without an imposed terminal

constraint, it will never reach exactly zero (which could be also not the best solution in terms of stability). Increasing the time horizon for the optimization could reduce the state error but the cost gradients are much steeper. This can cause numerical problems. The progress of the initial and the corresponding optimized costs is shown in Figure 6.5. In this example the robot is walking in place while it is pushed from behind with the force trajectory F_{push} shown in Figure 6.5. There are two characteristics observable: in the undisturbed case initial and optimized costs are close to zero, while the costs are significantly reduced by the optimization right after the disturbance.

Table 6.1: Parameter values for example optimization of next footstep position.

parameter	value
$\boldsymbol{S}_z$	$\mathbf{0}$
$\boldsymbol{S}_p$	1.3
$\boldsymbol{Q}$	$\mathrm{diag}(0.001,\, 3,\, 0.001,\, 0.1)$
$k_{step,a}$	0.5
δ	0.01
α_{init}	1.0
c	0.01
ρ	0.2

6.4.2 Coupled 2D Foot Position Optimization

The foot position in the x and y direction can be optimized in one coupled problem using the spatial model introduced in Section 4.5. This can be necessary for situations where the interaction between the two directions is not negligible. The resulting problem is similar to problem A but with the optimization variable $\boldsymbol{p} = [\Delta L_x,\, \Delta L_y]^T$ and using the three DoF state equations

$$\min_{\boldsymbol{p}} J(\boldsymbol{z}_s, \boldsymbol{p}) = \min_{\boldsymbol{p}} \left[\frac{1}{2} \boldsymbol{p}^T \boldsymbol{S}_{s,p} \boldsymbol{p} + \int_{t_a}^{t_e} \frac{1}{2} \boldsymbol{z}_s^T \boldsymbol{Q}_s \boldsymbol{z}_s \mathrm{d}t \right], \tag{6.19}$$

$$\text{s.t.} \quad \dot{\boldsymbol{z}}_s = \boldsymbol{f}_s(\boldsymbol{z}_s, \boldsymbol{p}, t)\,, \tag{6.20}$$

$$\boldsymbol{z}_s(t_a) = \boldsymbol{z}_{s,a}\,, \tag{6.21}$$

$$\boldsymbol{p} \in \mathcal{P}. \tag{6.22}$$

Additional quintic polynomials for both horizontal foot position modifications are added to the prediction model. The values of $\boldsymbol{p}$ are used as final values of the swing leg. The optimization (6.19) to (6.22) is solved as the scalar problem with the direct shooting method. The solution of the initial value problem $\boldsymbol{z}_s(\boldsymbol{p}, t)$ with a given parameter vector $\boldsymbol{p}$ can be used to transform the dynamic optimization problem into the static one

$$\min_{\boldsymbol{p} \in \mathcal{P}} J(\boldsymbol{p}). \tag{6.23}$$

The minimum of (6.23) is approximately solved by using Newton's method. Computing the gradient vector and Hessian matrix with finite differences increases the number of numerical integrations drastically. The Hessian for $J(\boldsymbol{p}^k)$ with central difference approximation yields

$$\nabla_p^2 J(\boldsymbol{p}^k) = \begin{bmatrix} \frac{1}{\delta^2}\left(J_{x^+,y} - 2J_{x,y} + J_{x^-,y}\right) & \frac{1}{4\delta^2}\left(J_{x^+,y^+} + J_{x^-,y^-} - J_{x^+,y^-} - J_{x^-,y^+}\right) \\ \text{sym.} & \frac{1}{\delta^2}\left(J_{x,y^+} - 2J_{x,y} + J_{x,y^-}\right) \end{bmatrix} \tag{6.24}$$

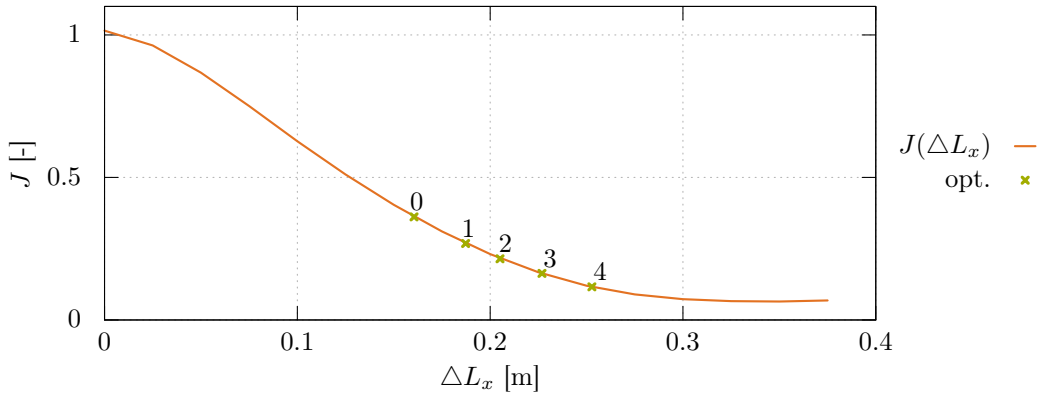


Figure 6.3: Optimization results: Initial costs (0) and four iterations (1-4).

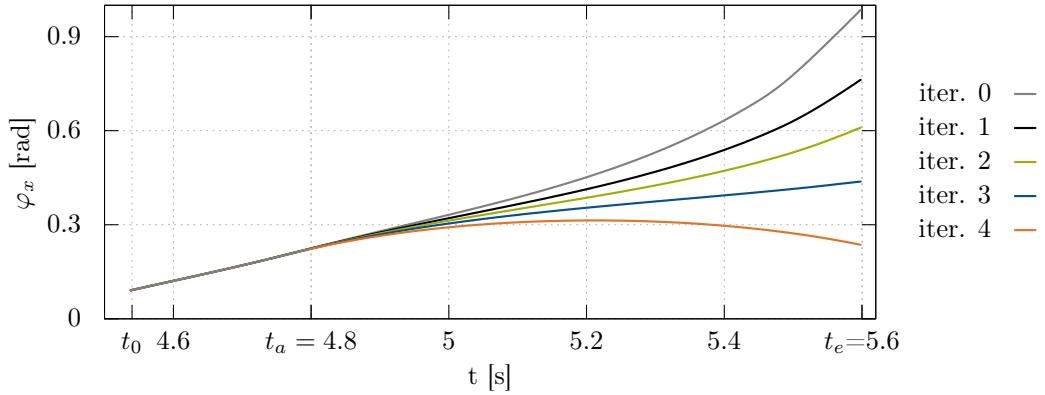


Figure 6.4: Optimization results: predicted trajectory for the inclination error φ_x for the first four optimization iterations.

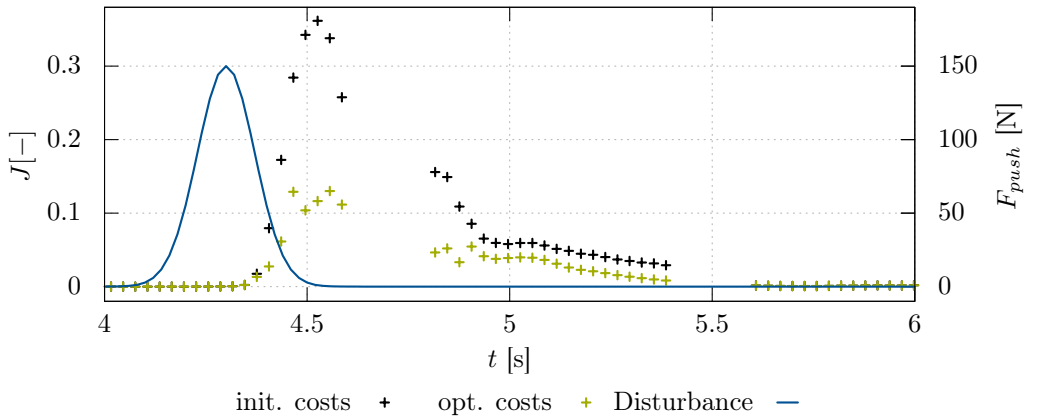


Figure 6.5: Optimization results: initial and optimized costs at different robot simulation time instants after an external disturbance.

where

$$J_{x^{\pm},y} = J(\Delta L_x \pm \delta,\, \Delta L_y), \qquad J_{x,y^{\pm}} = J(\Delta L_x,\, \Delta L_y \pm \delta),$$
$$J_{x^{\pm},y^{\pm}} = J(\Delta L_x \pm \delta,\, \Delta L_y \pm \delta), \qquad J_{x,y} = J(\Delta L_x,\, \Delta L_y).$$

Each optimization update requires nine solutions of the initial value problem. The overall time for one optimization iteration is more than 2 ms with 1-3 additional evaluations due to the backtracking method and a computation time of 150 µs for one evaluation. The computation time can be reduced by using quasi-Newton methods like the BFGS formula (Nocedal and Wright 2004, pp.136). They also avoid that the Hessian might be not positive definite which is a necessary requirement for the method. In current implementation the steepest descent is used for such cases. The iterative computation of the parameter vector

$$\boldsymbol{p}^{k+1} = \boldsymbol{p}^k - \alpha^k \left(\nabla_p^2 J(\boldsymbol{p}^k)\right)^{-1} \nabla_p J(\boldsymbol{p}^k) \tag{6.25}$$

also uses the scaling factor α^k which is computed with the backtracking method (Algorithm 2). The initial value $\boldsymbol{p}^0$ is provided by the heuristic of the planar case (6.16) in both directions. The error of the absolute CoG of the model is computed with:

$$\Delta_I \boldsymbol{r}_c\,(t_1) = \left[\boldsymbol{A}_{IT}(\varphi_x(t_1), \varphi_y(t_1)) - \boldsymbol{E}\right] \frac{1}{m} \sum_{j \in N_B} m_j\, {}_T\boldsymbol{r}_j\,(t_1). \tag{6.26}$$

Here $\boldsymbol{A}_{IT}$ is the transformation matrix shown in (4.61) which uses the inclination of the model at t_1 and $\boldsymbol{E}$ is the identity matrix. The velocity error is obtained by differentiating (6.26) wrt. time.

Example Optimization

The following part shows results from one exemplary optimization. The model is initialized with the robot's state right after a disturbance. Figure 6.6 shows the landscape of the cost function which is obtained with the cost function parameter of Table 6.2. It also shows the initial solution and the optimization result for three iterations. It can be seen that the minimum is almost reached within these iterations. Nevertheless one problem is that the cost function is very flat in the region around the minimum.

Table 6.2: Cost function weights for optimizing footstep positions with the spatial model.

parameter	value
$\boldsymbol{S}_z$	$\boldsymbol{0}$
$\boldsymbol{S}_p$	diag(1.5, 1.5)
$\boldsymbol{Q}$	diag(0.001, 3, 3, 0.001, 0.01, 0.01)

6.4.3 Predictive Inclination Compensation

The aim of the inclination compensation is to avoid early or late contact of the swing foot. The basic idea is adopted from the methods presented in Buschmann et al. (2011) and Nishiwaki and Kagami (2007b). The definition of the planning FoR in Subsection 3.2.4 says that it rotates with the upper body of the robot. All taskspace trajectories are written in this FoR including the swing foot motion. Consequently a non-zero inclination error of the upper body at ground contact of the swing foot leads to a posture and position

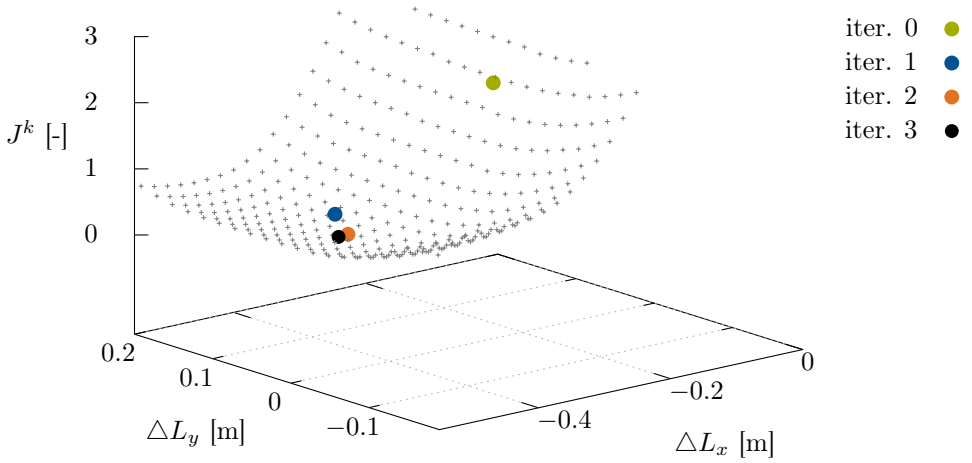


Figure 6.6: Optimization results for the coupled problem: Initial solution (0) and three iterations.

error of the foot wrt. the world. For flat ground walking this means that the swing foot is not aligned horizontally and will end above or below ground level. The compensation method modifies final swing foot horizontal orientation $(\Delta c_{\varphi_x}, \Delta c_{\varphi_y})$ and vertical position Δz_{sw} in order to avoid those issues (cf. Figure 6.7).

In contrast to Nishiwaki and Kagami (2007b) the proposed method does not use current posture error of the robot but the predicted error at the end of current step t_1. The inclination errors are already available due to the initial integration of the prediction models in the x and y direction from t_0 to t_1 for the optimization initialization. The rotation matrix from world to planning FoR $\boldsymbol{A}_{TI}(\hat{\boldsymbol{\varphi}}(t_1))$ can be computed with the predicted values $\hat{\boldsymbol{\varphi}}(t_1) = [\hat{\varphi}_x(t_1), \hat{\varphi}_y(t_1)]^T$. The posture error is obtained by a multiplication with the ideal planned rotation matrix $\boldsymbol{A}_{IT,id}(\boldsymbol{\varphi}_{id})$ of the robot

$$\Delta \boldsymbol{A} = \boldsymbol{A}_{IT,id} \boldsymbol{A}_{TI}(\hat{\boldsymbol{\varphi}}(t_1)). \tag{6.27}$$

The overall swing foot modification parameters are then determined with the ideal step lengths $L_{x,id}, L_{y,id}$ and the optimization results $\Delta L_x, \Delta L_y$ as follows

$$\Delta c_{\varphi_x} = \boldsymbol{e}_z^T \Delta \boldsymbol{A} \boldsymbol{e}_x, \tag{6.28}$$

$$\Delta c_{\varphi_y} = \boldsymbol{e}_z^T \Delta \boldsymbol{A} \boldsymbol{e}_y, \tag{6.29}$$

$$\Delta z_{st} = \begin{cases} 0 & \text{if} \quad \Delta z_{sw,last} < \Delta z_{min} \\ 0.5 \Delta z_{sw,last} & \text{else} \end{cases}, \tag{6.30}$$

$$\Delta z_{sw} = \boldsymbol{e}_z^T \Delta \boldsymbol{A} \begin{bmatrix} L_{x,id} + \Delta L_x \\ L_{y,id} + \Delta L_y \\ 0 \end{bmatrix} + \Delta z_{st}. \tag{6.31}$$

During the stance phase, the modified foot rotation is kept constant while the modified final stance foot height Δz_{st} is planned back to zero if $\Delta z_{sw,last} < \Delta z_{min}$ and otherwise to half of its value $\Delta z_{sw,last}$. This value has to be added to the final swing foot height. Changing the stance foot height trajectory changes the effective CoG height which may cause problems with the ideal generated horizontal CoG trajectories. Nevertheless this is necessary to bring the commanded configuration back towards the ideal values and avoid the effect of "the robot running into the ground" or "stretched knees".

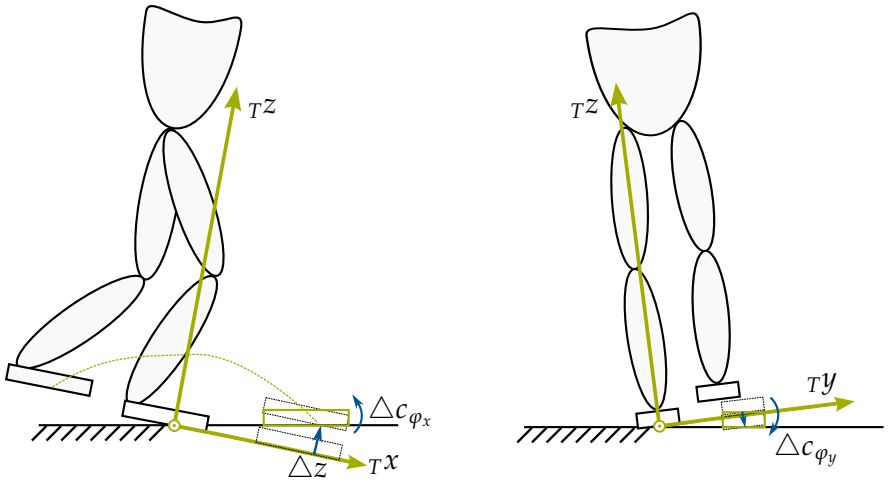


Figure 6.7: taskspace modifications to compensate for upper body inclination errors in x and y direction.

6.4.4 Continuous Trajectory Replanning

The final step of the method is the generation of continuous trajectories from the above calculated variables. All mentioned variables can be summarized in the vector

$$\mathbf{\Gamma} = [\Delta L_x, \Delta L_y, \Delta c_{\varphi_x}, \Delta c_{\varphi_y}, \Delta z_{sw}, \Delta z_{st}] \tag{6.32}$$

which is used to calculate a set of fifth-order polynomials for the taskspace modifications of the legs

$$\triangle \boldsymbol{w}_{legs}(t) = \boldsymbol{f}(t, \mathbf{\Gamma}). \tag{6.33}$$

This has to be done continuously every time new values $\mathbf{\Gamma}$ are generated. In order to ensure C^2-continuity of the commanded motion, values from last time step for position $\Delta \boldsymbol{w}_{legs}(t_{k-1})$, velocity $\Delta \dot{\boldsymbol{w}}_{legs}(t_{k-1})$ and acceleration $\Delta \ddot{\boldsymbol{w}}_{legs}(t_{k-1})$ are used as initial values for the fifth order polynomials. This is important because there may be non-zero values if the trajectories are updated during swing phase. $\mathbf{\Gamma}$ are set as final position values for the polynomials with zero velocity and acceleration. Additionally the horizontal swing leg motion ends at an earlier time $t_{lead,h}$ than the other values with t_{lead}. Figure 6.8 visualizes example trajectories for x and z direction. If the current time is during DS phase (where the feet should not move) a constant polynomial is added until the end of the DS phase t_{ds}.

The calculation of compensating motions is controlled by the finite state machine of the pattern generator. Note that the method is also triggered by the early contact event which is part of the sensor feedback of the finite state machine. If the detected contact state of the current swing leg switches to closed before the planned step time is passed, the next double support phase is initiated. When this event occurs, the motion of the current swing leg is stopped in a few timesteps. The connection to the early contact event is an important feature because there are cases in which the predicted final state may differ from the real state of the robot. Then the desired motion has to adapt to this unexpected event in order to stabilize the overall system. The final adapted taskspace position for a time instant t_k is computed via

$$\boldsymbol{w}_d(t_k) = \boldsymbol{w}_{id}(t_k) + \begin{bmatrix} \Delta \boldsymbol{w}_{legs}(t_k) \\ \boldsymbol{O} \end{bmatrix} = \boldsymbol{w}_{id}(t_k) + \triangle \boldsymbol{w}(t_k) \tag{6.34}$$

and analogous for the taskspace velocity $\dot{\boldsymbol{w}}_d(t_k)$. The overall method is summarized in Figure 6.9. The state observer determines estimates for current inclination and inclination

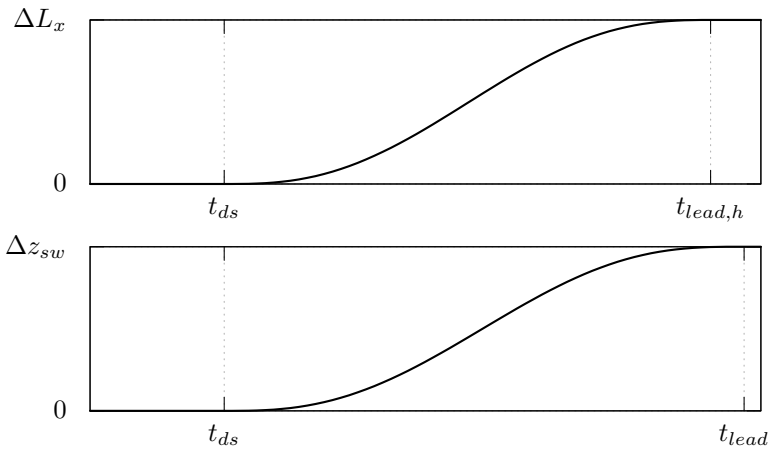


Figure 6.8: Example modification trajectories in x and z direction.

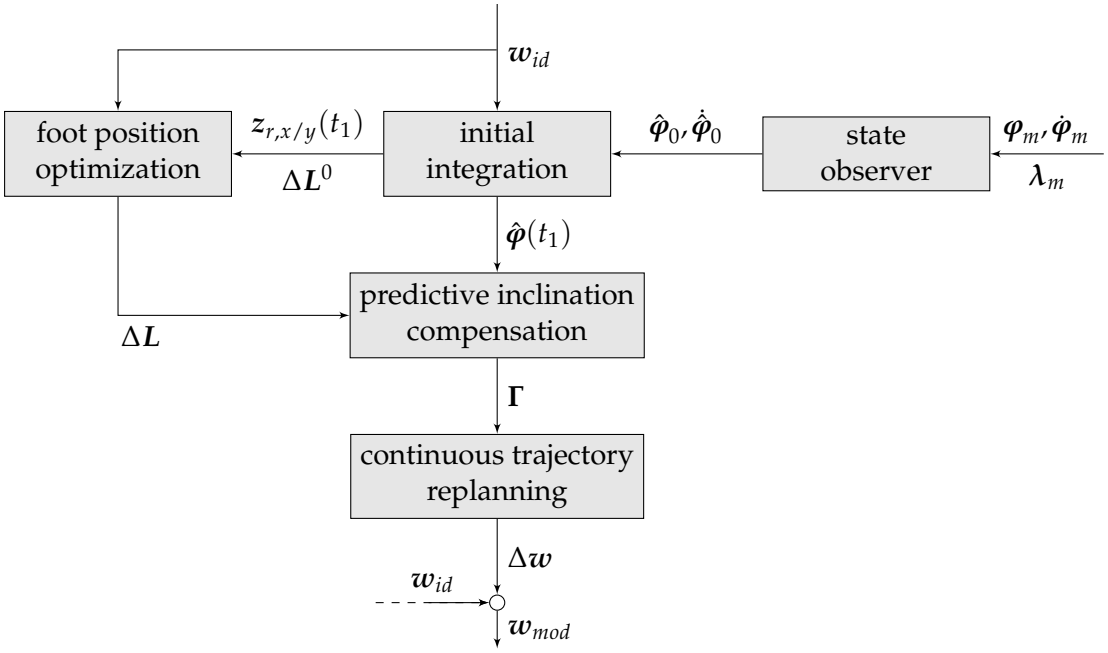


Figure 6.9: Overview of the sensor feedback framework for problem A. The state observer, initial integration and foot position optimization are performed in x and y direction independently.

rate $\hat{\boldsymbol{\varphi}}_0, \dot{\hat{\boldsymbol{\varphi}}}_0$ and provides these to the initial integration. The resulting initial state at time t_1 in x and y direction $z_{r,x/y}(t_1)$ and initial parameter $\Delta \boldsymbol{L}^0$ is used in the foot position optimization. These steps are performed decoupled for both walking directions. The resulting $\Delta \boldsymbol{L}$ and $\hat{\boldsymbol{\varphi}}(t_1)$ are finally used to recalculate the remaining foot parameters and the corresponding trajectories.

6.5 Center of Gravity Modification

This section discusses an approach to optimize the CoG trajectory and solves problem B for the bipedal robot LOLA. It is based on the paper Wittmann et al. (2016). Two different methods are presented: the first one solves the CoG optimization and uses an heuristics for the footstep position. The second one solves both quantities in one optimization problem.

6.5.1 Center of Gravity Trajectory Optimization

The problem to compute an optimal CoG trajectory for the humanoid robot is solved with an indirect approach. This section does not treat the whole problem defined in Subsection 6.3.2 but a slightly simplified version where the parameter p is determined in advance via a heuristic. The prediction model with footstep and CoG modification presented in Subsection 4.4.2 is used. The only difference is the definition of the input $u(t) = \frac{\mathrm{d}}{\mathrm{d}t}\Delta\dot{x}_b(t)$ which is then integrated twice in order to obtain the trajectory $\Delta x_b(t)$. In optimization runs it showed that the acceleration input performs better than the jerk input. However $\mathcal{C}^1$-continuity is achieved for $\Delta x_b(t)$. The overall model with the integration of the input yields

$$\dot{\boldsymbol{z}}_{r,e} = \frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix}\boldsymbol{q}\\ \dot{\boldsymbol{q}}\\ \triangle x_b\\ \triangle \dot{x}_b\end{bmatrix} = \begin{bmatrix}\dot{\boldsymbol{q}}\\ \boldsymbol{M}^{-1}[\boldsymbol{\lambda}_r + \boldsymbol{T}_s - \boldsymbol{h}]\\ \triangle \dot{x}_b\\ u\end{bmatrix} = \boldsymbol{f}_{r,e}\left(\boldsymbol{z}_{r,e}, t, p, u\right). \tag{6.35}$$

For a given initial value $\boldsymbol{z}_{r,e,a}$, parameter p and control trajectory $u(t)$ the first order differential equation (6.35) can be solved to obtain $\boldsymbol{z}_{r,e}(t, p, u(t))$. In problem B it was stated that the goal is to find $u(t) \in \mathcal{U}$ such that the cost function

$$J_B(u(t)) = \underbrace{\frac{1}{2}\Delta\boldsymbol{z}_{r,e}^T(t_e)\boldsymbol{S}_z\Delta\boldsymbol{z}_{r,e}(t_e)}_{s(\boldsymbol{z}_{r,e}(t_e))} + \int_{t_a}^{t_e} \underbrace{\frac{1}{2}\Delta\boldsymbol{z}_{r,e}^T\boldsymbol{Q}\Delta\boldsymbol{z}_{r,e} + \frac{1}{2}u\boldsymbol{R}u}_{h(\boldsymbol{z}_{r,e},u,t)}\,\mathrm{d}t \tag{6.36}$$

is minimized. The control trajectory and the parameter are restricted to the valid regions $\mathcal{P}$ and $\mathcal{U}$. The state error $\Delta\boldsymbol{z}_{r,e}$ is defined as follows

$$\Delta\boldsymbol{z}_{r,e} = \boldsymbol{z}_{r,e} - [0, 0, 0, 0, p, 0]^T \tag{6.37}$$

which has the effect that a modification of the CoG position is penalized only wrt. the footstep modification. This creates a behavior where only the difference $\Delta x_b(t_k) - p$ is penalized. It is a consequential definition because solutions where the footstep is modified by 10 cm and the final CoG position shows the same final modification is preferable and should produce no costs. For a given p, given initial value $\boldsymbol{z}_{r,e}(t_a)$ and fixed final time t_e with free final state $\boldsymbol{z}_{r,e}(t_e)$, the optimality conditions from Pontryagin's minimum principle (Geering 2007) are well known. Introducing the costate $\boldsymbol{\psi}(t)$ and the Hamiltonian $H(\boldsymbol{z}_{r,e}, \boldsymbol{\psi}, u, t) = h(\boldsymbol{z}_{r,e}, u, t) + \boldsymbol{\psi}^T\boldsymbol{f}_{r,e}(\boldsymbol{z}_{r,e}, u, t)$, the optimality conditions for the dynamic problem (6.35) and (6.36) are

$$\dot{\boldsymbol{\psi}} = -\left(\frac{\partial H}{\partial \boldsymbol{z}_{r,e}}\right)^T = -\left(\frac{\partial h}{\partial \boldsymbol{z}_{r,e}}\right)^T - \left(\frac{\partial \boldsymbol{f}_{r,e}}{\partial \boldsymbol{z}_{r,e}}\right)^T \boldsymbol{\psi}, \tag{6.38}$$

$$\dot{\boldsymbol{z}}_{r,e} = \left(\frac{\partial H}{\partial \boldsymbol{\psi}}\right)^T = \boldsymbol{f}_{r,e}, \tag{6.39}$$

$$\boldsymbol{\psi}(t_e) = \left(\frac{\partial s}{\partial \boldsymbol{z}_{r,e}}\bigg|_{t_e}\right)^T, \tag{6.40}$$

$$\boldsymbol{z}_{r,e,a} = \boldsymbol{z}_{r,e}(t_a) \tag{6.41}$$

where the Hamiltonian has a global minimum wrt. $u \in \mathcal{U}$:

$$u = \arg\min_{u\in\mathcal{U}} H(\boldsymbol{z}_{r,e}, \boldsymbol{\psi}, u, t). \tag{6.42}$$

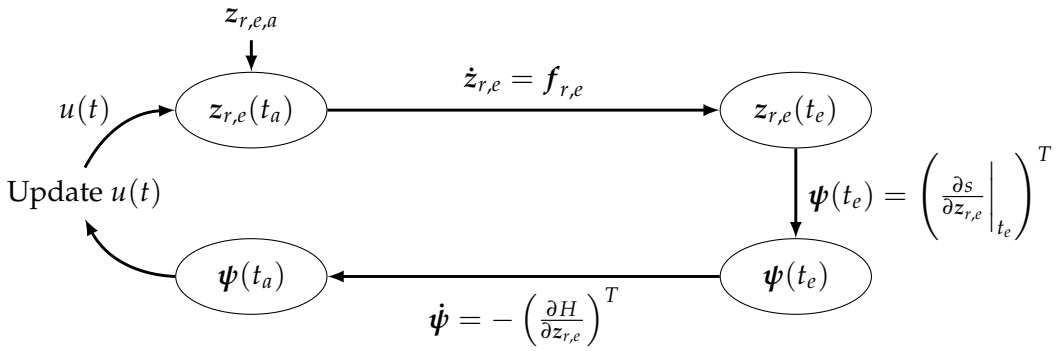


Figure 6.10: Graphical overview of the iterative gradient algorithm.

If u is not on the boundary of $\mathcal{U}$, relation (6.42) reduces to the first order necessary condition

$$\frac{\partial H}{\partial u} = \frac{\partial h}{\partial u} + \boldsymbol{\psi}^T \frac{\partial \boldsymbol{f}_{r,e}}{\partial u} = 0. \tag{6.43}$$

The resulting BVP has decoupled boundary conditions where the left one is for the state $z_{r,e}(t_a)$ while the right one restricts the costate $\boldsymbol{\psi}(t_e)$. The gradient method exploits this property to iteratively compute the solution for (6.38) to (6.42). For a given $u(t)$ the state $z_{r,e}$ is integrated forward with (6.39) from $t_a \rightarrow t_e$ and $\boldsymbol{\psi}(t_e)$ can be computed with (6.40). Afterwards $\boldsymbol{\psi}$ is integrated with (6.38) backward in time. The obtained trajectories $z_{r,e}(t)$ and $\boldsymbol{\psi}(t)$ are than used to compute an update for the input trajectory with (6.42) resp. (6.43). The procedure is visualized in Figure 6.10. The gradient method with steepest descent is known for its slow convergence. Therefore the conjugate gradient is used in this work as proposed in Lasdon et al. (1967) and Schuetz et al. (2014). The iterative calculation of the input trajectory can be stated to

$$\begin{aligned} u^{k+1} = u^k + \alpha^k s_u^k, \quad s_u^k(t) &= -g_u^k(t) + \beta_u^k \boldsymbol{s}_u^{k-1}(t), \\ g_u^k(t) &= \frac{\partial H(\boldsymbol{z}_{r,e}^k, \boldsymbol{\psi}^k, u^k, t)}{\partial u} \end{aligned} \tag{6.44}$$

with the abbreviations $\boldsymbol{z}_{r,e}^k = \boldsymbol{z}_{r,e}(u^k)$ and $\boldsymbol{\psi}^k = \boldsymbol{\psi}(u^k)$. The factor β_u is determined with the Fletcher-Reeves method

$$\beta_u^k = \frac{\left(g_u^k, g_u^k\right)}{\left(g_u^{k-1}, g_u^{k-1}\right)} \quad \text{where } \left(g_u^k, g_u^k\right) := \int g_u^k(t) g_u^k(t) \mathrm{d}t. \tag{6.45}$$

The step size α^k is computed in an underlying line-search problem with the goal to achieve sufficient decrease of the cost function (6.36) in each iteration. It is obtained by solving the one-dimensional optimization

$$\alpha^k = \arg\min_{\alpha > 0} J_B \left(\boldsymbol{z}_{r,e}(u^k + \alpha s_u^k), u^k + \alpha s_u^k\right) \tag{6.46}$$

approximately. This is solved in an analogous manner as for the foot position optimization with the backtracking method (Algorithm 2).

At the beginning of this paragraph it was assumed that the parameter p for the foot position is determined in advance. The same linear heuristic as for the initial value of the footstep optimization is used. During the first iteration's numerical integration of $z_{r,e}(t)^0$ the absolute CoG position and velocity error at t_1 is stored and used to calculate the parameter

$$p = k_{step,b}(\Delta_I x_c(t_1) + \frac{1}{\omega} \Delta_I \dot{x}_c(t_1)) \tag{6.47}$$

with the scalar factor $k_{step,b} \in [0, 1]$. This approximate solution for problem B works well even though the parameter is not the optimal one. A reason is that the foot has to be particularly modified in the right direction. By penalizing only the difference to the modified CoG position in the cost function, the method produces the desired behavior. During an external disturbance e.g. an impulse in walking direction, the robot steps forward and comes to rest somewhere in this direction. This is also visible regarding the chosen weights for $(\Delta x_b - p)$ in $\boldsymbol{Q}$ and $\boldsymbol{S}_z$.

Table 6.3: Cost function weights and heuristic factor for the CoG trajectory optimization.

$$\boldsymbol{Q} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1000 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2500 & 0 \\ 0 & 0 & 0 & 0 & 0 & 3000 \end{bmatrix}, \qquad \boldsymbol{S}_z = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1500 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1000 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2500 & 0 \\ 0 & 0 & 0 & 0 & 0 & 5000 \end{bmatrix},$$

$$\boldsymbol{R} = 1, \qquad k_{step,b} = 0.75.$$

Example Optimization

In the following an exemplary optimization with the chosen weights of Table 6.3 is discussed. The model in x direction is initialized with an inclination error $\Delta\varphi_x(t_a) = 0.15\,\text{rad}$ and inclination rate error $\Delta\dot{\varphi}_x(t_a) = 0.35\,\text{rad/s}$. The initial value for $z_{r,e}(t_a)$ is computed with Algorithm 1 while the initial CoG modification and the control trajectory $u^0(t)$ are set to zero. The optimization horizon is set to the step time $t_e - t_a = T_{step}$ which is 0.8 s in this example. The time discretization for numerical integration is $\Delta t = 4\,\text{ms}$. Figure 6.11 shows the costs $J_B(u^k(t))$ for 15 iterations. The optimization was performed once with the steepest descent search direction and once with the conjugate gradient search direction. It can be seen that except for the first iteration the conjugate gradient method is superior. It reaches the optimum in about 6 iterations. Figure 6.12 shows the resulting control trajectories $u^k(t)$ and Figure 6.13 the evolution of the inclination error and the CoG modification for the conjugate gradient method. The parameter value was set to $p = 0.255\,\text{m}$ in iteration 0. Note that the modification Δx_b is performed in the FoR of the model rotated by φ_x.

6.5.2 Pontryagin's Minimum Principle with Additional Parameters

This section derives an algorithm to optimize a trajectory $u(t)$ and additional free parameters at the same time with an indirect shooting method. It is based on a conjugate gradient algorithm from trajectory generation with an additional update law for the parameter, see Boček (1980). There are two different ways to derive the optimality conditions, one that transforms the problem to an ordinary trajectory optimization problem and one that derives Pontryagin's Minimum Principle directly. The first way is described in this section while the second one is included in Appendix C.

The transformation is performed as follows: the unknown parameter will be treated as additional state with a static behavior $\dot{p} = 0$ and unknown initial value $p(t_a)$. Problem B can be rewritten with the augmented state vector $\boldsymbol{\xi} = [\boldsymbol{z}_{r,e}^T, p]^T$ as follows

$$\min_{p,u} J_B(\boldsymbol{\xi}, u) = \min_{p,u} \left[s(\boldsymbol{\xi}(t_e)) + \int_{t_a}^{t_e} h(\boldsymbol{\xi}, u, t) \mathrm{d}t \right] \tag{6.48}$$

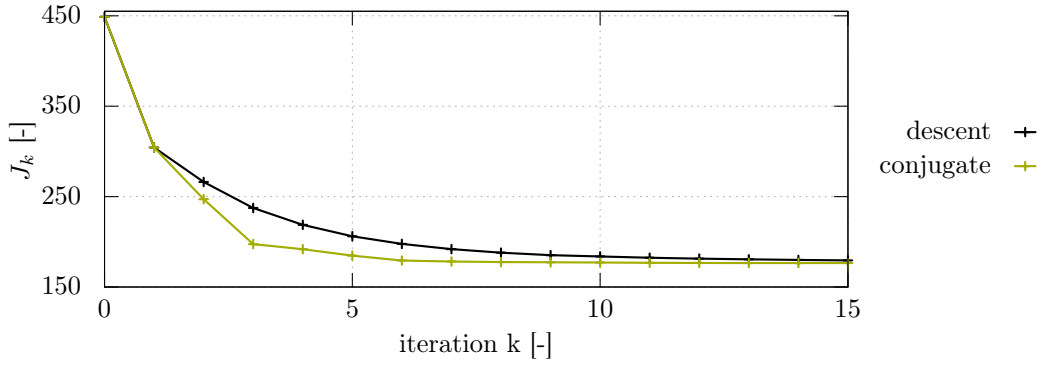


Figure 6.11: Example optimization with the gradient method: Cost over iterations for steepest descent and conjugate gradient search direction.

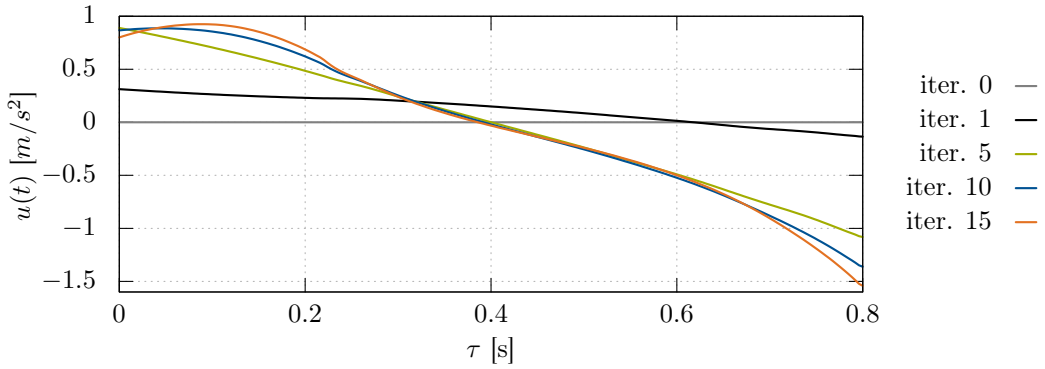


Figure 6.12: Example optimization with the conjugate gradient method: Input trajectories $u^k(t)$.

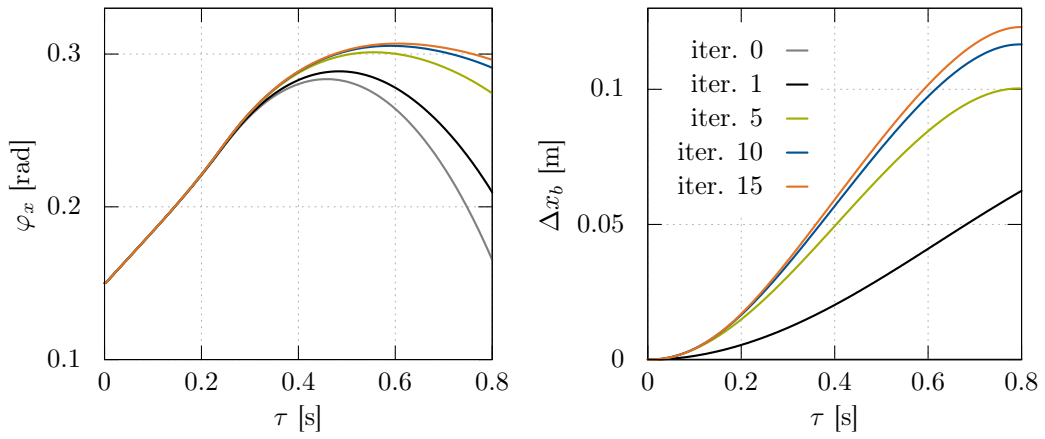


Figure 6.13: Example optimization with the conjugate gradient method: Predicted inclination error $\varphi_x(t)$ and CoG modification trajectory $\Delta x_b(t)$.

$$\text{s.t.} \quad \dot{\boldsymbol{\xi}} = \begin{bmatrix} \dot{\boldsymbol{z}}_{r,e} \\ \dot{p} \end{bmatrix} = \begin{bmatrix} \boldsymbol{f}_{r,e}(\boldsymbol{\xi}, u, t) \\ 0 \end{bmatrix} = \boldsymbol{f}_{\xi}(\boldsymbol{\xi}, u, t)\,, \tag{6.49}$$

$$\boldsymbol{z}_{r,e}(t_a) = \boldsymbol{z}_{r,e,a}\,, \tag{6.50}$$

$$p \in \mathcal{P}\,, \tag{6.51}$$

$$u \in \mathcal{U}\,. \tag{6.52}$$

Note that the BVP will be no longer decoupled as it is the case for fixed initial and free final state due to the partly unknown initial value.

The derivation of the optimality conditions for the present problem will be done similar to Geering (2007). The constraints can be adjoined to the cost function by introducing the Lagrange multipliers $\boldsymbol{\psi}_{z,a}$ and $\boldsymbol{\psi}(t) = [\boldsymbol{\psi}_z^T(t), \psi_p(t)]^T$ with the same dimension as the augmented state $\boldsymbol{\xi}$. The resulting augmented cost function is

$$\begin{aligned} \bar{J} &= s(\boldsymbol{\xi}(t_e)) + \boldsymbol{\psi}_{z,a}^T \left[\boldsymbol{z}_{r,e}(t_a) - \boldsymbol{z}_{r,e,a}\right] + \int_{t_a}^{t_e} h(\boldsymbol{\xi}, u, t) + \boldsymbol{\psi}^T(t) \left[\boldsymbol{f}_{\xi}(\boldsymbol{\xi}, u, t) - \dot{\boldsymbol{\xi}}\right] \mathrm{d}t \\ &= s(\boldsymbol{\xi}(t_e)) + \boldsymbol{\psi}_{z,a}^T \left[\boldsymbol{z}_{r,e}(t_a) - \boldsymbol{z}_{r,e,a}\right] + \int_{t_a}^{t_e} H(\boldsymbol{\xi}, u, \boldsymbol{\psi}, t) - \boldsymbol{\psi}^T(t)\dot{\boldsymbol{\xi}}\mathrm{d}t \end{aligned} \tag{6.53}$$

with the Hamiltonian $H(\boldsymbol{\xi}, u, \boldsymbol{\psi}, t) = h(\boldsymbol{\xi}, u, t) + \boldsymbol{\psi}^T(t)\boldsymbol{f}_{\xi}(\boldsymbol{\xi}, u, t)$. Omitting all function dependencies the first variation of (6.53) around the optimal solution $\boldsymbol{\xi}^*, u^*$ can be written

$$\begin{aligned} \delta\bar{J} =& \left.\frac{\partial s}{\partial \boldsymbol{\xi}}\right|_{t_e} \delta\boldsymbol{\xi}(t_e) + \delta\boldsymbol{\psi}_{z,a}^T \left[\boldsymbol{z}_{r,e}(t_a) - \boldsymbol{z}_{r,e,a}\right] \\ &+ \int_{t_a}^{t_e} \frac{\partial H}{\partial \boldsymbol{\xi}}\delta\boldsymbol{\xi} + \frac{\partial H}{\partial u}\delta u + \frac{\partial H}{\partial \boldsymbol{\psi}}\delta\boldsymbol{\psi} - \delta\boldsymbol{\psi}^T\dot{\boldsymbol{\xi}} - \boldsymbol{\psi}^T\delta\dot{\boldsymbol{\xi}}\mathrm{d}t. \end{aligned} \tag{6.54}$$

Integration by parts of the term $\int \boldsymbol{\psi}^T\delta\dot{\boldsymbol{\xi}}\mathrm{d}t$ and rearranging yields

$$\begin{aligned} \delta\bar{J} =& \left.\frac{\partial s}{\partial \boldsymbol{\xi}}\right|_{t_e} \delta\boldsymbol{\xi}(t_e) + \delta\boldsymbol{\psi}_{z,a}^T \left[\boldsymbol{z}_{r,e}(t_a) - \boldsymbol{z}_{r,e,a}\right] - \boldsymbol{\psi}^T(t_e)\delta\boldsymbol{\xi}(t_e) + \boldsymbol{\psi}^T(t_a)\delta\boldsymbol{\xi}(t_a) \\ &+ \int_{t_a}^{t_e} \left(\frac{\partial H}{\partial \boldsymbol{\xi}} + \dot{\boldsymbol{\psi}}^T\right)\delta\boldsymbol{\xi} + \left(\frac{\partial H}{\partial \boldsymbol{\psi}} - \dot{\boldsymbol{\xi}}^T\right)\delta\boldsymbol{\psi} + \frac{\partial H}{\partial u}\delta u\mathrm{d}t. \end{aligned} \tag{6.55}$$

The not completely known initial state $\boldsymbol{\xi}(t_a)$ causes the product $\boldsymbol{\psi}^T(t_a)\delta\boldsymbol{\xi}(t_a)$ to not completely vanish. It can be simplified by splitting the augmented state and the according Lagrange multipliers

$$\boldsymbol{\psi}^T(t_a)\delta\boldsymbol{\xi}(t_a) = \begin{bmatrix} \boldsymbol{\psi}_z^T(t_a)\ \psi_p(t_a) \end{bmatrix} \begin{bmatrix} \delta\boldsymbol{z}_{r,e}(t_a) \\ \delta p(t_a) \end{bmatrix} = \cancelto{0}{\boldsymbol{\psi}_z^T(t_a)\delta\boldsymbol{z}_{r,e}(t_a)} + \psi_p(t_a)\delta p(t_a). \tag{6.56}$$

The variation of $\boldsymbol{z}_{r,e}$ at time t_a is equal to zero because its initial value is included in the constraints. The variation of p is arbitrary in contrast. The optimal solution of (6.55) has to fulfill $\delta\bar{J} = 0$ for any variation $\delta\boldsymbol{\xi}, \delta\boldsymbol{\psi}_{z,a}, \delta\boldsymbol{\xi}, \delta u$. This delivers the known optimality conditions for problems with free final state and fixed final time extended by one additional condition for $\psi_p(t_a)$

$$\dot{\boldsymbol{\psi}} = -\left(\frac{\partial H}{\partial \boldsymbol{\xi}}\right)^T, \tag{6.57}$$

$$\dot{\boldsymbol{\xi}} = \boldsymbol{f}_{\xi}, \tag{6.58}$$

$$\boldsymbol{\psi}(t_e) = \left(\frac{\partial s}{\partial \boldsymbol{\xi}}\bigg|_{t_e} \right)^T, \tag{6.59}$$

$$\psi_p(t_a) = 0, \tag{6.60}$$

$$u^* = \arg\min_u \; H(\boldsymbol{\psi}^*, \boldsymbol{\xi}^*, u, t). \tag{6.61}$$

Condition (6.60) can be rewritten when the solution of the costate is evaluated by using the given final value at t_e and integrating (6.57) backward in time

$$\boldsymbol{\psi}(t) = \frac{\partial s}{\partial \boldsymbol{\xi}}\bigg|_{t_e} + \int_t^{t_e} \frac{\partial H}{\partial \boldsymbol{\xi}} \mathrm{d}t. \tag{6.62}$$

The last row of (6.62) evaluated at $t = t_a$ results in the condition

$$\psi_p(t_a) = \frac{\partial s}{\partial p}\bigg|_{t_e} + \int_{t_a}^{t_e} \frac{\partial H}{\partial p} \mathrm{d}t \stackrel{!}{=} 0. \tag{6.63}$$

The result represents the interpretation of the costate (or adjoint variables) $\boldsymbol{\psi}(t)$ that they are a measure of the sensitivity of the cost function concerning variations in the state $\boldsymbol{\xi}(t)$ (Graichen 2015, p.100). Consequently the residual of (6.63) can be used to modify the initial value $p(t_a)$ in order to improve the solution. This can be added to the gradient method for dynamic trajectory optimization to include additional parameters. The iterative update laws for the control trajectory and the parameter are in the unconstrained case with steepest descent

$$u^{k+1} = u^k - \alpha^k \frac{\partial H}{\partial u} = u^k - \alpha^k g_u^k \tag{6.64}$$

$$p^{k+1} = p^k - \alpha^k \left(\frac{\partial s}{\partial p}\bigg|_{t_e} + \int_{t_a}^{t_e} \frac{\partial H}{\partial p} \mathrm{d}t \right) = p^k - \alpha^k g_p^k \tag{6.65}$$

where α^k can be determined by some step-length selection algorithm, e.g. backtracking or quadratic interpolation method (Nocedal and Wright 2004). Those methods solve the one-dimensional problem

$$\alpha^k = \min_{\alpha} J_B(u^k + \alpha g_u^k, p^k + \alpha g_p^k) \tag{6.66}$$

approximately. Note that the factor α^k is the same for both update laws (6.64) and (6.65) which correlates with the condition that the u and p have to be chosen such that the cost function $J_B(z_{r,e}, p, u)$ has a minimum.

The search directions determined by steepest descents $s_u = -g_u$ and $s_p = -g_p$ can be replaced by conjugate gradients in order to improve the overall convergence rate of the algorithm, e.g. using again the Fletcher-Reeves adaptation

$$s_u^k(t) = -g_u^k(t) + \beta_u^k s_u^{k-1}(t), \qquad \beta_u^k = \frac{\left(g_u^k, g_u^k\right)}{\left(g_u^{k-1}, g_u^{k-1}\right)}, \tag{6.67}$$

$$s_p^k = -g_p^k + \beta_p^k s_p^{k-1}, \qquad \beta_p^k = \frac{g_p^k g_p^k}{g_p^{k-1} g_p^{k-1}}. \tag{6.68}$$

The overall procedure of the gradient method for trajectory optimization with additional parameters is summarized in Algorithm 3. The unknown initial value is replaced with the unknown parameter p for sake of clarity.

Algorithm 3 Gradient Method with Parameter

Determine $\hat{\boldsymbol{z}}_{r,e,a}$
initialize p^0
initialize $u^0(t) \leftarrow u^*_{previous}(t)$
k $\leftarrow 0$
repeat
  solve $\tilde{\boldsymbol{\xi}}^k(u^k, t)$ from (6.58)
  solve $\boldsymbol{\psi}^k(u^k, \tilde{\boldsymbol{\xi}}^k, t)$ from (6.57)
  compute s_u^k, s_p^k e.g. with (6.67) and (6.68)
  $\alpha^k \leftarrow$ solve line search problem (6.66)
  update u^{k+1}, p^{k+1}
  $k \leftarrow k+1$
until (converged **or** k $>$ max iterations)
end

Optimization Example

This paragraph presents an optimization example with a simple optimization problem to show the behavior of Algorithm 3. The following scalar linear problem is considered

$$\min_{p,u(t)} J = \min_{p,u(t)} \frac{1}{2} s_p p^2 + \frac{1}{2} s_x x^2(T) + \frac{1}{2} \int_0^T r u^2(t) \mathrm{d}t, \tag{6.69}$$

$$\text{s.t.} \quad \dot{x} = p + u(t), \tag{6.70}$$

$$x(0) = x_0 \tag{6.71}$$

with the state $x(t)$, its given initial value x_0 and the unknown control trajectory $u(t)$ and parameter p. The optimization horizon is $t \in [0,\, T]$. The optimality conditions are

$$\begin{bmatrix} \dot{x} \\ \dot{p} \end{bmatrix} = \begin{bmatrix} p+u \\ 0 \end{bmatrix}, \quad \dot{\boldsymbol{\psi}} = \begin{bmatrix} 0 \\ -\psi_1 \end{bmatrix}, \quad \boldsymbol{\psi}(T) = \begin{bmatrix} s_x x(T) \\ s_p p \end{bmatrix}, \quad \frac{\partial H}{\partial u} = ru + \psi_1 = 0 \tag{6.72}$$

with the costate $\boldsymbol{\psi} = [\psi_1, \psi_2]^T$, the Hamiltonian $H = \frac{1}{2} r u^2 + \psi_1 (p+u)$ and the boundary conditions at $t = 0$

$$x(0) = x_0, \quad \psi_2(0) = 0. \tag{6.73}$$

The analytic solution for this example yields

$$p^* = \frac{-s_x x_0}{\frac{s_p}{T} + \frac{s_x s_p}{r} + s_x T}, \tag{6.74}$$

$$u^*(t) = \frac{s_p p}{Tr}, \tag{6.75}$$

$$x^*(t) = \left(p + \frac{s_p p}{Tr} \right) t + x_0 . \tag{6.76}$$

This solution is used as reference for comparison with the numerical optimization. Algorithm 3 is used once with normal gradient information and once with conjugate gradient information to solve (6.69) to (6.71) numerically. The results for 30 iterations are depicted in Figure 6.14 where the values of Table 6.4 are used. The solution for the input trajectory $u(t)$ is constant which is the reason that the figure shows only the resulting value. The scaling factor for the search directions is set to the fixed value of $\alpha = 0.15$

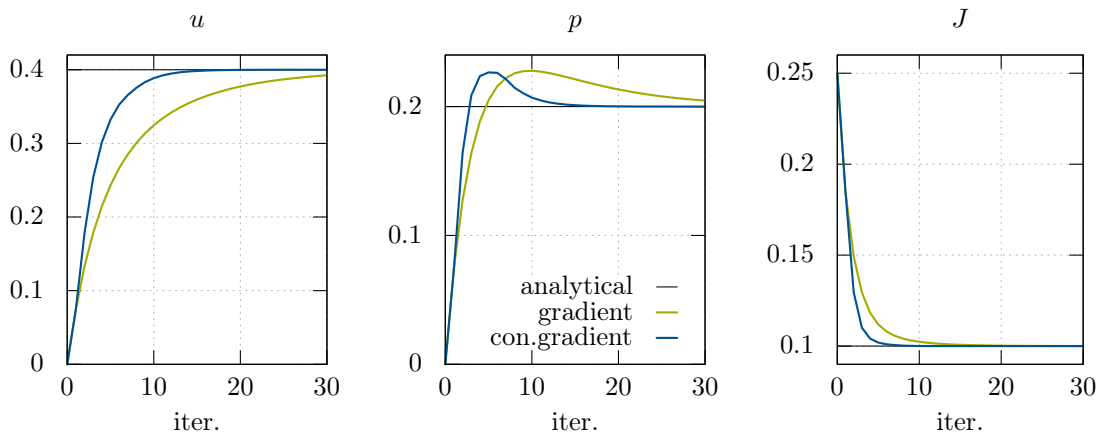


Figure 6.14: Comparison of optimization results for 30 iterations.

for both cases (gradient, conjugate gradient). It was determined manually such that the optimizations do not diverge. State and costate trajectories are solved with an explicit Euler integration scheme with the time discretization Δt. Both variants of Algorithm 3 converge to the analytical solution but as supposed the convergence of the conjugate gradient is faster. The overshooting of the parameter value p may be caused by the fact that a constant factor α for the line-search is used.

Table 6.4: Parameter values for example optimization (6.69) to (6.71).

parameter	value	parameter	value
s_p	1.0	x_0	-1.0
s_x	0.5	p_{init}	0.0
r	0.5	$u_{init}(t)$	0.0
T	1.0	Δt	0.005

6.5.3 Center of Gravity and Footstep Optimization

The algorithm presented above is applied for solving Problem B in this section. It uses the augmented state as introduced in the problem statement (6.48) to (6.51). Therefore the cost function (6.5) defined at the beginning has to be transformed to fit to the new state vector definition $\boldsymbol{\xi} = [\boldsymbol{z}_{r,e}^T, p]^T$:

$$\begin{aligned} J_B(\boldsymbol{\xi}, u) &= s(\boldsymbol{\xi}(t_e)) + \int_{t_a}^{t_e} h(\boldsymbol{\xi}, u, t)\mathrm{d}t = \frac{1}{2}\boldsymbol{\xi}^T(t_e)\boldsymbol{S}_{\xi}\boldsymbol{\xi}(t_e) + \frac{1}{2}\int_{t_a}^{t_e} \boldsymbol{\xi}^T\boldsymbol{Q}_{\xi}\boldsymbol{\xi} + \boldsymbol{R}u^2\mathrm{d}t \\ &\overset{!}{=} \frac{1}{2}\Delta\boldsymbol{z}_{r,e}^T(t_e)\boldsymbol{S}_z\Delta\boldsymbol{z}_{r,e}(t_e) + \frac{1}{2}\int_{t_a}^{t_e} \Delta\boldsymbol{z}_{r,e}r^T\boldsymbol{Q}\Delta\boldsymbol{z}_{r,e} + \boldsymbol{R}u^2\mathrm{d}t. \end{aligned} \tag{6.77}$$

To obtain the same optimization criterion the cost function matrices are set to

$$\boldsymbol{Q}_{\xi} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1000 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2500 & 0 & -2500 \\ 0 & 0 & 0 & 0 & 0 & 3000 & 0 \\ 0 & 0 & 0 & 0 & -2500 & 0 & 2500 \end{bmatrix}, \tag{6.78}$$

$$S_\xi = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1500 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1000 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2500 & 0 & -2500 \\ 0 & 0 & 0 & 0 & 0 & 5000 & 0 \\ 0 & 0 & 0 & 0 & -2500 & 0 & 2500 \end{bmatrix}. \tag{6.79}$$

In simulation it showed that the search direction for the parameter has to be improved. Using only the residual of the condition (6.63) gave bad optimization results. A revised algorithm is developed that solves condition (6.63) via linearization (Newton's method). The parameter update Δp^k consequently yields

$$\Delta p^k = -\left(\frac{\partial \psi_p}{\partial p}(t_a)\right)^{-1} \psi_p^k(t_a) \tag{6.80}$$

with the gradient of the costate $\psi_p(t_a)$ approximately determined by finite differences. An example optimization is considered with the same setup as presented on page 94. This includes also the initial solution for $u^0(t)$ and the parameter p^0. The results can be seen in Figures 6.15 and 6.16. The cost function evaluation over the iteration is compared for the CoG optimization with and without step position optimization. Both methods use the same heuristic (6.47) for the initial solution of p^0. The additional step optimization shows little improvement of the overall cost evaluation. The residual of the additional optimality condition in Figure 6.16 is not zero. During the optimization it even increases. Reasons are that an update for u has an influence to ψ_p in the optimization and ψ_p is not directly included in the cost function. In simulation the overall method performed similar compared to only optimizing the CoG trajectory. The additional evaluations for determining the search direction of ψ_p with finite differences increase the computational time. Therefore, the final implementation uses only the heuristic for the footstep modification. However the overall time for both directions is still within the admissible maximal value (20 ms) and can be compensated for by an increased FIFO buffer size (see Subsection 3.3.4).

6.5.4 System Integration Details

This paragraph provides further details to the real-time implementation of the CoG modification and its integration into the overall walking control system of LOLA. One important point is the initial solution for the control trajectory u^0 which uses the solution from previous optimization $u_{previous}$. It is generated following the model predictive control principle (Diehl et al. 2009) and by using a moving horizon $T_h = t_e - t_a$. The control initialization is

$$u^0(\tau) = u_{previous}(\tau + \Delta t_{mpc}) \quad \text{where } \tau \in [0, T_h - \Delta t_{mpc}],\ t = \tau + t_a \tag{6.81}$$

with Δt_{mpc} being the time between two successive optimization runs. The new part of the time horizon $\tau \in [T_h - \Delta t_{mpc}, T_h]$ is filled with zeros.

Another strategy could be a linear function from $u^0(T_h - \Delta t_{mpc})$ to zero during the new time horizon. Figure 6.17 visualizes the initialization between two consecutive optimization runs. Former results for the predicted trajectories $\boldsymbol{z}_{r,e}(t)$, $\boldsymbol{\psi}(t)$ and the conjugate gradient information s_u are discarded for a new optimization run. This showed the best results for experiments because the predicted state trajectory and consequently the costate trajectory can change during Δt_{mpc} drastically.

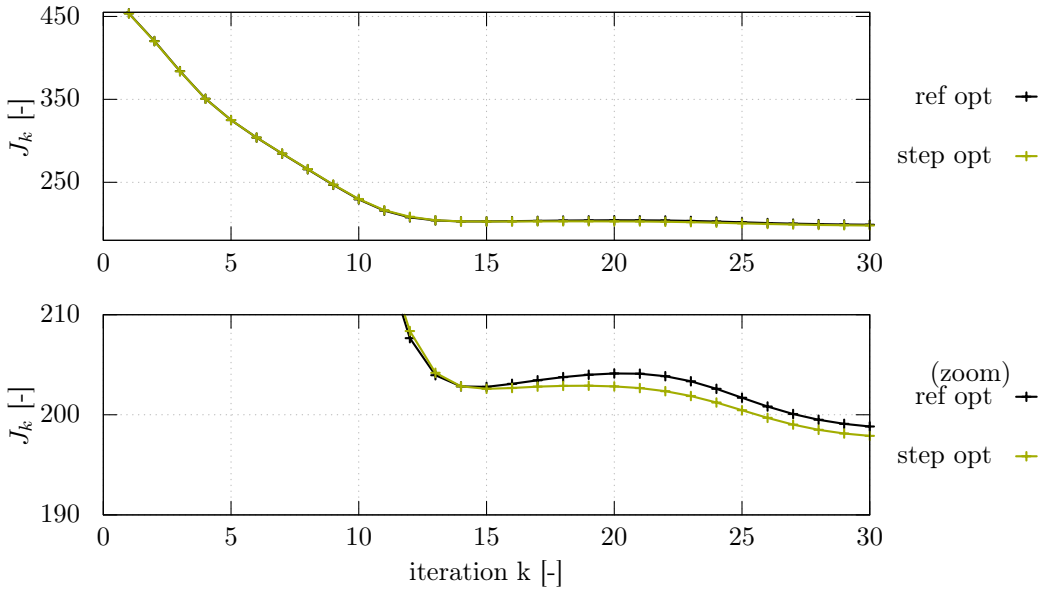


Figure 6.15: Optimization results for CoG optimization without step optimization (ref opt) and with step optimization (step opt). On the bottom a zoomed plot is shown.

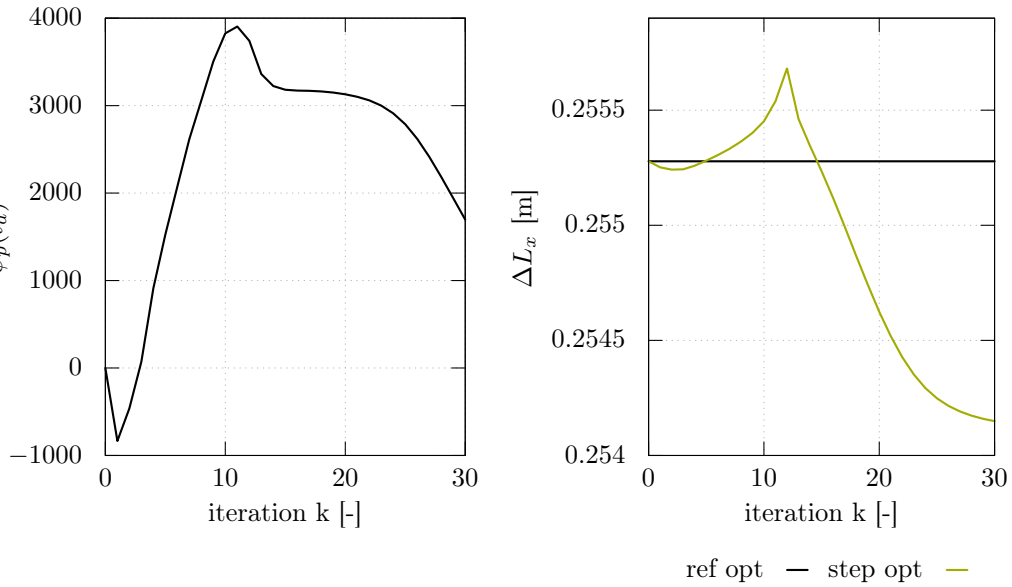


Figure 6.16: Evolution of the residual and optimized parameter.

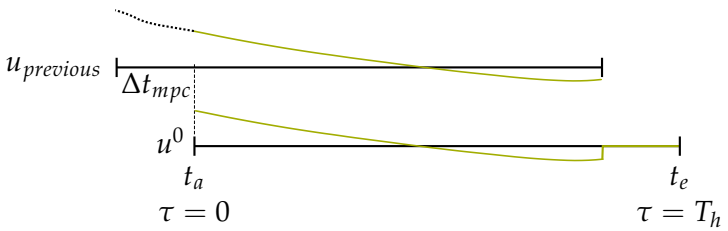


Figure 6.17: Control trajectory initialization with previous solution.

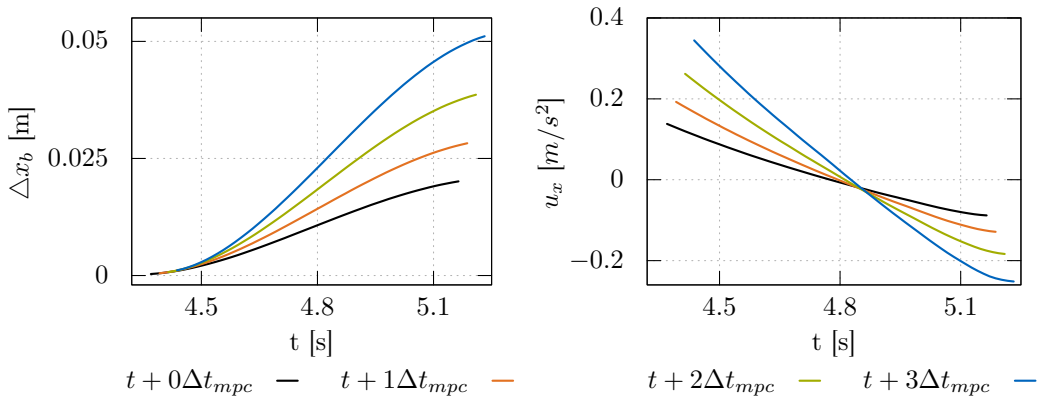


Figure 6.18: Resulting trajectories from four optimization runs: continuous CoG trajectory Δx_b and discontinuous input trajectory u_x.

Each Δt_{mpc} the optimization is called and performs three iterations. The resulting computational time of the algorithm is maximal 3.5 ms for one direction. Consequently the implementation fulfills the real-time requirement that the overall computational time is below Δt_{mpc} between two consecutive optimization runs. A low limit for the number of iterations produces a suboptimal solution. Nevertheless the method can converge to the optimum by using the solution from previous optimization as initial guess.

The resulting $u_x(t)$ and $u_y(t)$ are applied to the robot's motion until an updated trajectory is available. Only $u_x(t)$, $u_y(t)$ are used and numerically integrated separately to obtain the trajectory $\Delta x_b(t), \Delta y_b(t)$. The overall modification of the CoG motion $\Delta \dot{\boldsymbol{w}}_{cog}(t_k)$ and $\Delta \boldsymbol{w}_{cog}(t_k)$ at time t_k is computed via

$$\Delta \dot{\boldsymbol{w}}_{cog}(t_k) = \begin{bmatrix} \Delta \dot{x}_b(t_k) \\ \Delta \dot{y}_b(t_k) \end{bmatrix} = \begin{bmatrix} \Delta \dot{x}_b(t_{k-1}) + t_c u_x(t_k - t_a) \\ \Delta \dot{y}_b(t_{k-1}) + t_c u_y(t_k - t_a) \end{bmatrix}, \tag{6.82}$$

$$\Delta \boldsymbol{w}_{cog}(t_k) = \begin{bmatrix} \Delta x_b(t_k) \\ \Delta y_b(t_k) \end{bmatrix} = \begin{bmatrix} \Delta x_b(t_{k-1}) + t_c \Delta \dot{x}_b(t_k) \\ \Delta y_b(t_{k-1}) + t_c \Delta \dot{y}_b(t_k) \end{bmatrix}. \tag{6.83}$$

The problem of generating discontinuous desired trajectories (cf. Figure 6.18) is avoided by using only the computed accelerations. The values for the parameter in the x and y direction p_x and p_y are directly used as footstep modification values ΔL_x and ΔL_y respectively. The predictive inclination compensation (Subsection 6.4.3) and the continuous replanning of the foot trajectories (Subsection 6.4.4) can be used in the same way as it was shown for problem A. The final adapted taskspace position for a time instant t_k yields

$$\boldsymbol{w}_d(t_k) = \boldsymbol{w}_{id}(t_k) + \begin{bmatrix} \Delta \boldsymbol{w}_{legs}(t_k) \\ \Delta \boldsymbol{w}_{cog}(t_k) \\ \mathbf{0} \end{bmatrix} \tag{6.84}$$

and analogous for the taskspace velocity $\dot{\boldsymbol{w}}_d(t_k)$. The overall method is summarized in Figure 6.19. State observer, initial integration and CoG trajectory optimization are performed in the x and y direction independently.

6.6 Constraints from Obstacle Avoidance

The methods presented above have to be combined with collision avoidance methods to obtain *flexible and robust* walking simultaneously. The following paragraph is submitted in Hildebrandt et al. (2017) and developed methods result from joint work with Arne-Christoph Hildebrandt. In keeping with the hierarchical control system structure,

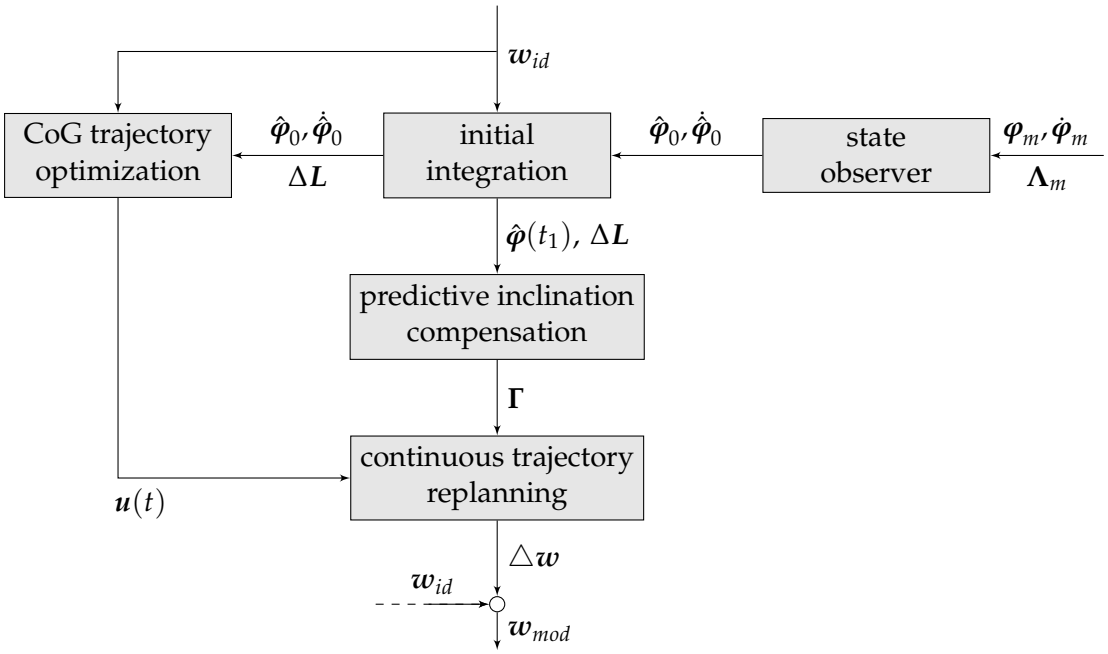


Figure 6.19: Overview of the sensor feedback framework for problem B.

a collision free motion that is then adapted for disturbance rejection based on sensor data is determined first. Possible solutions of (6.10) are restricted to reachable and obstacle free regions. Figure 6.20 shows a sample situation that is treated in the following. Starting point is a valid final swing foot location $\boldsymbol{L}_{id}$, determined by the step planner. The trajectory adaptation then modifies this position by $\Delta \boldsymbol{L} = [\Delta L_x, \Delta L_y]^T$ to stabilize the robot. The resulting final foothold position would cause a collision. The main question is how such constraints can be described and accounted for in a real-time optimization procedure. Short computation time is crucial since the method is used with sensor feedback and it has to react instantaneously to unknown disturbances. The desired output is a modified yet collision free foot position

$$\boldsymbol{L}_m^* = \boldsymbol{L}_{id} + \Delta \boldsymbol{L}^*. \tag{6.85}$$

with the feasible modification $\Delta \boldsymbol{L}^*$. A summary of the method is shown in Figure 6.21. In Subsection 3.3.2 two approaches to handle such situations were stated: the first is to determine an optimal step length modification without constraints and project the final solution onto the cone of the feasible set (which is determined by the constraints). The second accounts for the constraints during optimization: however, this requires an optimization for the step length modifications in both directions since the feasible set is at least two-dimensional and the boundaries for ΔL_x and ΔL_y are coupled. For this reason the spatial prediction model with the 2D footstep optimization is used. The problem description is summarized as follows:

- efficient calculation time ($\ll 1\,\text{ms}$)
- several arbitrary shaped obstacles
- over-stepping of obstacles should be possible
- kinematic limits have to be included
- solution must be generated reliably.

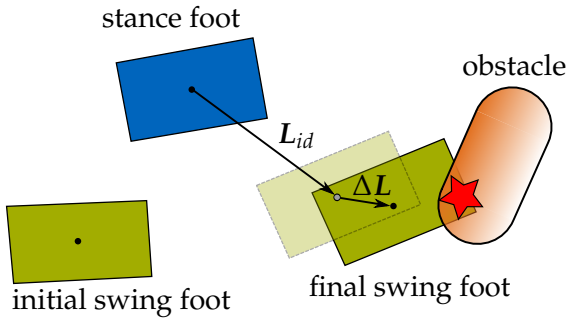


Figure 6.20: Example for problem description of constraints from obstacle avoidance. Ideal collision free final swing foot position and the modified invalid position.

The physical constraints and the mathematical description for such situations are given below. Projection of an invalid point onto a feasible region is discussed in the subsequent part. This projection method is then applied to the stated problem and the two solution strategies are presented.

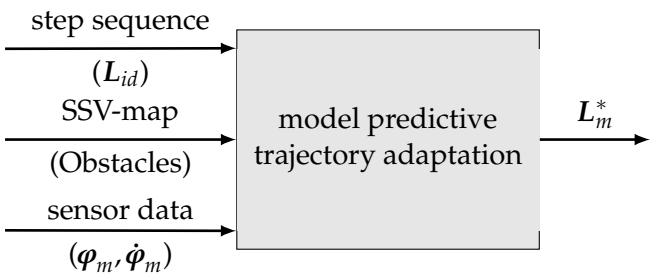


Figure 6.21: Overview of the main input and output data for the sensor based trajectory adaptation with additional obstacle avoidance.

6.6.1 Geometric Constraints

The key point when taking into account the geometrical constraints imposed on walking stabilization, namely kinematic constraints and obstacles, is their consistent and compact representation. To allow fast calculations convex polytopes are chosen to represent both kinematic constraints and obstacles. They allow a fast projection to a feasible region. The resulting n_{eq} linear inequalities for a vector $\boldsymbol{x}$ are described by

$$\boldsymbol{c}_{eq,j} := \{\boldsymbol{x} \in \mathbb{R}^2 | \boldsymbol{a}_j^T \boldsymbol{x} > b_j\}. \tag{6.86}$$

A set of n_{C_I} linear inequalities describes one convex polytope, $\mathcal{C}_I$, which is an *invalid region*. In total, the *invalid area* $\bar{\mathcal{A}}_S$ consists of n_p polytopes and is defined as follows:

$$\bar{\mathcal{A}}_S := \bigcup_{i=1}^{n_p} \mathcal{C}_{I_i}. \tag{6.87}$$

The corresponding *valid area*, $\mathcal{A}_S$, is formally defined as

$$\mathcal{A}_S := \mathcal{A} - \bar{\mathcal{A}}_S, \tag{6.88}$$

based on the total search area $\mathcal{A}$. The determination of $\bar{\mathcal{A}}_S$ is composed of the following parts:

- *Kinematic Limits*: Starting with the current stance foot the kinematically reachable area is approximated by the polytope as depicted in Figure 6.22a. The unreachable area results in the system of inequalities

$$\mathcal{C}_{I,kin} = \left\{ \boldsymbol{x} | \exists i \in \{1, ..., n_{kin}\} : \boldsymbol{a}_{kin,i}^T \boldsymbol{x} > b_{kin,i} \right\}. \tag{6.89}$$

- *Obstacles*: As described in Subsection 3.3.2, 3D segments approximate obstacles and areas of the environment the robot can not step onto. The former consist of n_{SSV} convex SSV-objects to allow for a detailed approximation of detected objects. The SSV-objects are reduced to 2D polytopes to comply with the hard timing constraints (cf. Figure 6.22b). First, the SSV-objects are projected on the ground, and then the three types of SSV-Objects (sphere, line and triangle SSV) are represented as polytopes. Each object j results in a convex hull:

$$\mathcal{C}_{I,SSV,j} = \left\{ \boldsymbol{x} | \exists i \in \{1, ..., n_{eq,SSV,j}\} : \boldsymbol{a}_{SSV,j,i}^T \boldsymbol{x} > b_{SSV,j,i} \right\}. \tag{6.90}$$

  The valid area for a step is already restricted by kinematic limits, therefore only obstacles within this kinematically reachable area are considered. This approach highly reduces the computational costs for the inequality constraints.

- *Foot Geometry*: The sophisticated 3D representation of the foot and lower leg geometry used in the step planner helps to give a better approximation of the robot's whole kinematic movement. The obstacles are enlarged by the foot geometry as shown in Figure 6.22c taking into account the desired foot rotation α. Thus only one point, which describes the foothold position, can be used to analyze the geometric constraints.

- *Large Obstacles*: Considering the robot's whole kinematic movement, obstacle-free regions are not necessarily steppable or kinematically reachable. Large obstacles, which the robot can not over-step, make adjacent obstacle-free regions inaccessible due to kinematic constraints. These large obstacles are already considered in the environment representation to avoid repeated checks for obstacle-free regions and inaccessible regions. Instead of introducing a polytope for the inaccessible region, the obstacle's representation is enlarged (compare Figure 6.22d). That way the obstacle approximation as well as the representation of the inaccessible region remains convex.

6.6.2 Finding Safe Regions

In the following the question initially posed, is answered: how can the valid foothold position closest to a given modified but invalid one be determined (cf. Figure 6.20)? There are several methods to compute a solution. They can be basically divided into two different approaches. The first starts with an initially invalid solution $\boldsymbol{L}_m$ and tries to find a valid point $\boldsymbol{L}_m^*$. Variants are sampling based methods or geometric testing of all boundaries.

Instead of finding the closest valid point, the problem can be inverted and $\mathcal{A}_S$ is divided into a set of convex valid regions C_V. In each of these valid and convex regions, the closest point to the invalid point $\boldsymbol{L}_m$ can be determined separately and, because of the regions' convexity, very efficiently. In contrast to the *geometric testing* method, intersections of the valid convex regions do not pose a problem, because the search is applied on a set of valid regions. The solutions are consequently independent of each other. The best one is chosen based on the calculated set of closest points. This procedure of searching

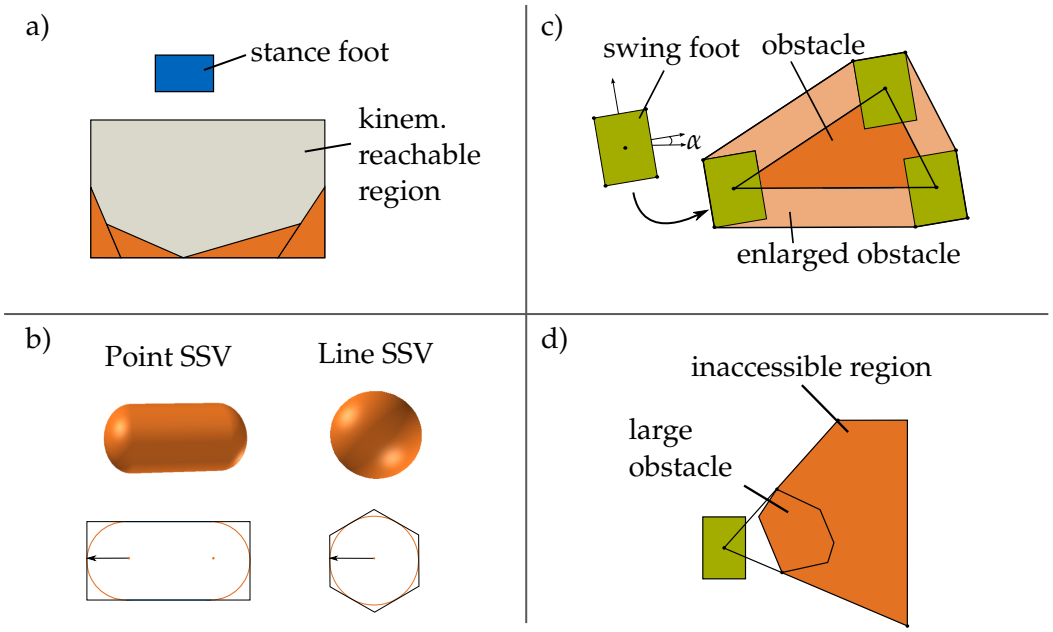


Figure 6.22: Geometric constraints: a) kinematic limits, b) obstacle representations, c) enlargement by foot geometry and d) large obstacle modification.

valid convex regions instead of only considering the invalid regions is largely inspired by Chestnutt and Takaoka (2010). They presented a method to calculate a valid convex area around a valid starting point. The algorithm starts with a convex area. It iterates around the starting point and it removes invalid parts of the initial convex area. A drawback of the implementation is that only one convex area around the starting point is found. Deits and Tedrake (2015) presented a powerful open-source tool, called *IRIS*, which has already been applied to step planning (Deits and Tedrake 2014). *IRIS* uses the corner points of invalid convex regions as input and calculates the corresponding inequalities. It also determines the largest valid convex region which is closest to a starting point. Although it seems to be well suited for the present problem, it exhibits some shortcomings. Those are the requirement of a predetermined search area, the computation of only one convex area and stability issues. Liu et al. (2010) and Sarmiento et al. (2005) present methods to divide arbitrary areas in convex regions. On the one hand, neither method is restricted to convex invalid regions; however, neither benefits from reduced computational costs for convex problems. In this application, only convex invalid regions are used, to gain the benefit in calculation times. Similar to (Liu et al. 2010; Sarmiento et al. 2005), the valid area is tried to cover by multiple convex regions, which may intersect each other. To meet timing limitation, the following characteristics of this mathematical problem are exploited:

- $\mathcal{C}_{I,i}$ are all convex.
- The starting point of the algorithm, which is the ideal step location calculated by the step planner $\boldsymbol{L}_{id}$, lies always inside $\mathcal{A}_S$.
- $\mathcal{A}$ is limited due to the kinematic constraints.

The algorithm follows an iterative approach. It is started with a grid of seed points over $\mathcal{A}$. The seed points, which lie in $\mathcal{A}_S$ are successively becoming the starting points for searching a convex region around it. Once the convex region around a seed point is determined, it is removed from the remaining valid area to avoid repetitive searches. As a first seed point the ideal footstep location is used, since it is valid per definition. This

Algorithm 4 Dividing A_S in convex regions

function FIND-CONVEX-REGIONS($\mathcal{A}_S$, $\bar{\mathcal{A}}_S$)
  initialize set of seed points $\mathcal{P}_S$
  $\mathcal{C}_V = \{\}$
  for all $p_k \in \mathcal{P}_S$ **do**
    verify p_k valid?
    j $\leftarrow 0$
    repeat
      Calculate closest boundary line l_j of $\bar{\mathcal{A}}_S$
      Add l_j as inequality to boundaries of $\mathcal{C}_{V_k}$
      Remove all inactive boundaries of $\bar{\mathcal{A}}_S$
      $j \leftarrow j + 1$
    until no more active boundaries
    remove $\mathcal{C}_{V_k}$ from $\mathcal{A}_S$
    update $\mathcal{C}_V = \mathcal{C}_V \cup \mathcal{C}_{V_k}$
  Output: $\mathcal{C}_V$

helps to reduce the computational costs. For one seed point the search for a valid convex region can be summarized as follows: The closest boundaries to the seed point are determined iteratively. The boundaries are therefore considered to be line segments with start and end points. Once the closest boundary, l_j, is found, it is added as a linear inequality to the valid region $\mathcal{C}_{V_k}$. Here, the convexity of the invalid region is used to efficiently find the closest boundaries. All boundaries outside $\mathcal{C}_{V_k}$ are skipped and considered inactive for the following steps. The search stops when there are no active boundaries left. The algorithm is summarized in Algorithm 4. The calculation of $\mathcal{C}_V$ has to be done only once before each of the robot's physical steps. The kinematically reachable area lies outside the camera's field of view. Therefore, the representation of the environment does not change during execution of one step.

6.6.3 Footstep Modification with Geometric Constraints

This section finally describes how the algorithm for finding a point in the safe regions can be combined with the optimization of next footsteps for stabilizing the robot.

Restrict Optimization Result

One straight forward solution for considering safe regions is to project the optimized quantities $\Delta \boldsymbol{L} = [\Delta L_x, \Delta L_y]^T$ onto the safe regions. This way optimization for the step length modifications can be done separately in the x and y directions. The optimization results are projected onto the set $\mathcal{P} = \mathcal{A}_S$. Two different criteria are tested to find the point that is closest to the optimal solution: the geometric distance between $\Delta \boldsymbol{L}$ and $\Delta \boldsymbol{L}^*$ and the point with the best (lowest) costs determined using (6.1). For the second criteria several candidate points are generated all lying on the cone of $\mathcal{A}_S$. Nevertheless this requires additional time consuming evaluations of the EoM (6.2) and the cost function. Both methods perform similar and consequently the closest distance criteria is chosen.

Optimization with inequality constraints

The mathematically correct way includes the inequality constraints in the optimization. This can be realized with hard constraints or using a penalty function. The later has the advantage that the problem is again unrestricted and exhibits better computational time.

The cost function for the coupled optimization (6.19) is rewritten for the spatial model (6.20) as

$$J_s = \boldsymbol{z}_s^T \boldsymbol{S}_z \boldsymbol{z}_s + \Delta \boldsymbol{L}^T \boldsymbol{S}_p \Delta \boldsymbol{L} + \int_{t_0}^{t_f} \boldsymbol{z}_s^T \boldsymbol{Q} \boldsymbol{z}_s \mathrm{d}t + h(\Delta \boldsymbol{L}) \tag{6.91}$$

and extended by an additional penalty term

$$h(\Delta \boldsymbol{L}) = \begin{cases} \beta \left(\boldsymbol{L}_m - \boldsymbol{L}_p \right)^2 & \boldsymbol{L}_m \in \bar{\mathcal{A}}_S \\ 0 & \boldsymbol{L}_m \in \mathcal{A}_S \end{cases} \tag{6.92}$$

that includes the distance to the closest valid point $\boldsymbol{L}_p$ at the cone of $\mathcal{A}_S$. The additional weight β is set to a value higher than all other weighting matrix entries. The optimization result obtained from (6.91) is not necessarily valid since invalid solutions are penalized but not completely avoided. Nonetheless, the solution is at least close to $\mathcal{A}_S$. Consequently the optimization result $\boldsymbol{L}_m$ will be verified whether it is valid or not and if necessary projected onto $\mathcal{A}_S$ as described in Subsection 6.6.3. In simulation experiments this strategy shows the ability to find solutions that require over-stepping of obstacles to maintain the robot's balance. This is one common advantage with the method to restrict the optimization result and is possible because a set of safe regions is used.

6.6.4 Implementation Details

The following paragraph provides details about the implementation and real-time realization of the presented framework. After the A*-search and before a new step k begins the ideal motion is planned (*Ideal*) and the set of valid regions $\mathcal{A}_S$ is computed from the SSV-map. The map is continuously updated with data from the vision-system (*Upd.SSV*). $\mathcal{A}_s$ is computed only once before a new physical step. This is sufficient since obstacles that are within the kinematic limit of one step are not visible with the vision system mounted on the head. *Reactive collision avoidance* prevents collisions between the swing foot and the obstacle that a footstep adaptation might cause. This may occur since only the final position is checked and the swing foot height is adapted via a heuristic.

Table 6.5: Computational time summary for maximal 4 obstacles. Runtimes are obtained from the real-time QNX computer.

method	avg. [µs]	max. [µs]
Comp.$\mathcal{A}_S$	600	2000
Find closest point	4	250
Adapt (total)	1000	2500

6.7 Chapter Summary

In this chapter different methods to include sensor feedback in trajectory generation were presented. The methods utilize the former introduced dynamic prediction model together with the state estimate. According to this state foot trajectories and the CoG trajectories are adapted. Two different optimization algorithms are developed. The first determines next horizontal footstep position with a direct shooting algorithm. The second solves the dynamic trajectory planning problem for the CoG trajectory by a conjugate gradient algorithm. Finally an algorithm to combine both optimizations is presented.

All variants include an inclination compensation. Based on geometric considerations, height and orientation of the feet are adapted. The aim is preventing early or late contacts of the feet. The last part of the chapter covers methods to combine trajectory modifications due to unknown disturbances with collision avoidance methods. The basic idea preserves the hierarchical structure and obstacles are included as additional inequality constraints for the foot position adaptation. That way flexible and robust walking is enabled. All presented methods include implementation details since they are applied to a real robot. The next chapter presents experimental results of the bipedal robot LOLA and the performance of the overall model predictive trajectory adaptation will be discussed.

Chapter 7

Experimental Results

This chapter presents selected experimental results with the bipedal robot LOLA. The robot is subjected to unknown disturbances in different situations to show its increased robustness by the model predictive trajectory adaptation. The following parts present different situations with the developed methods running on the real robot. A comparison of the walking performance without the new presented methods is not shown since creating the same push at the same relative time is almost impossible.

7.1 Walking on the Spot with Disturbances

7.1.1 Footstep Optimization (Experiment 1a)

In the first experiment the robot is walking on the spot while it receives pushes at the upper body in arbitrary directions. The step time is chosen to $T_{step} = 0.8\,\text{s}$. The sensor based trajectory adaptation with decoupled footstep optimization is used in this experiment. The robot is pushed with a bar which is equipped with a force transducer (model HBM U9B, 1 kN nominal rated force) in order to measure the transferred impulse. The analog signal is read with the GPIO/AD-slave. The force acts at heights between 1.0-1.5 m from the ground. The measured force and the resulting inclination errors are shown in Figure 7.1. The control system stabilizes the robot with the modifications of Figure 7.2. This is realized among others by the filtered inclination rates (Figure 7.3). Using the raw data would cause a decrease of the effectiveness of the method.

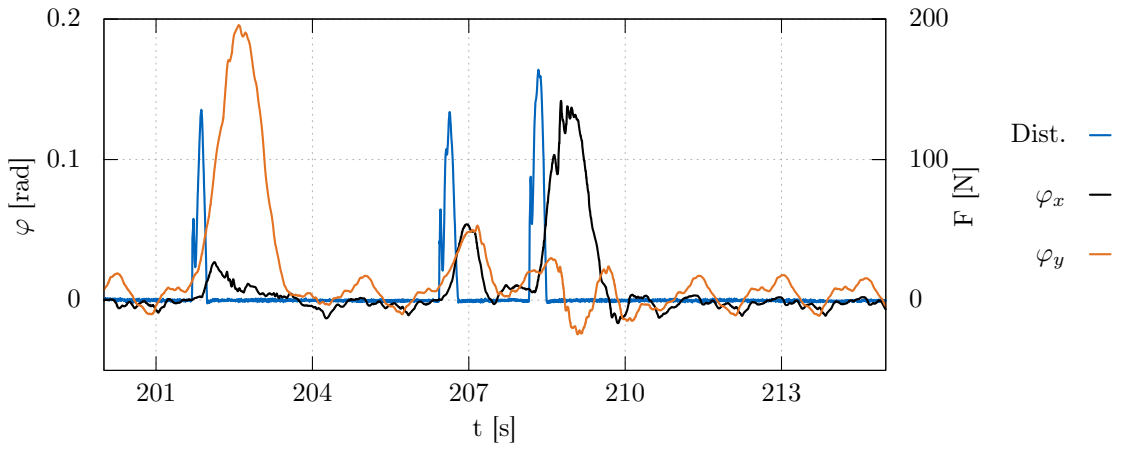


Figure 7.1: Experiment 1a: Disturbance force measurement and resulting inclination errors.

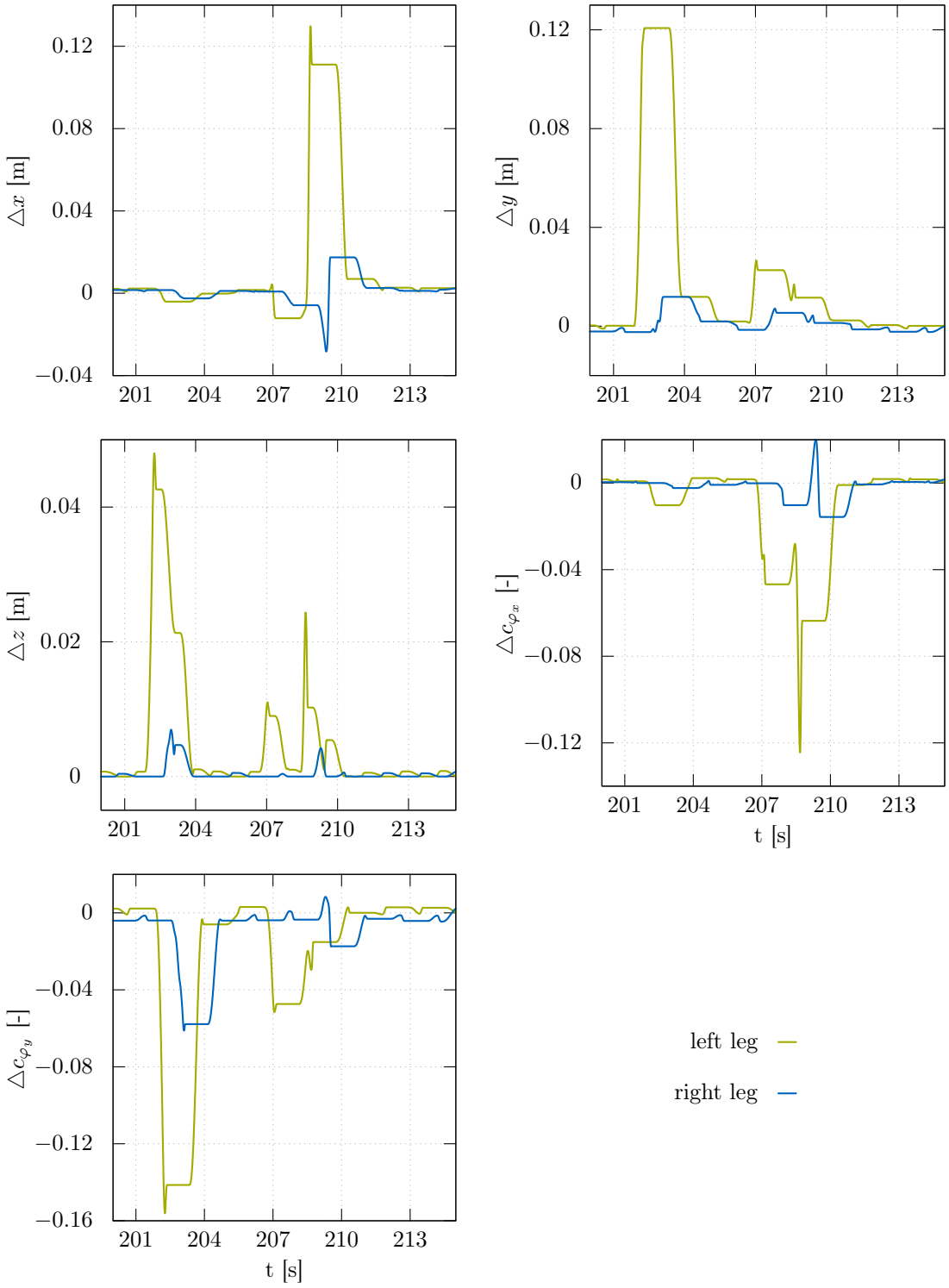


Figure 7.2: Experiment 1a: Resulting foot position and orientation trajectory modifications.

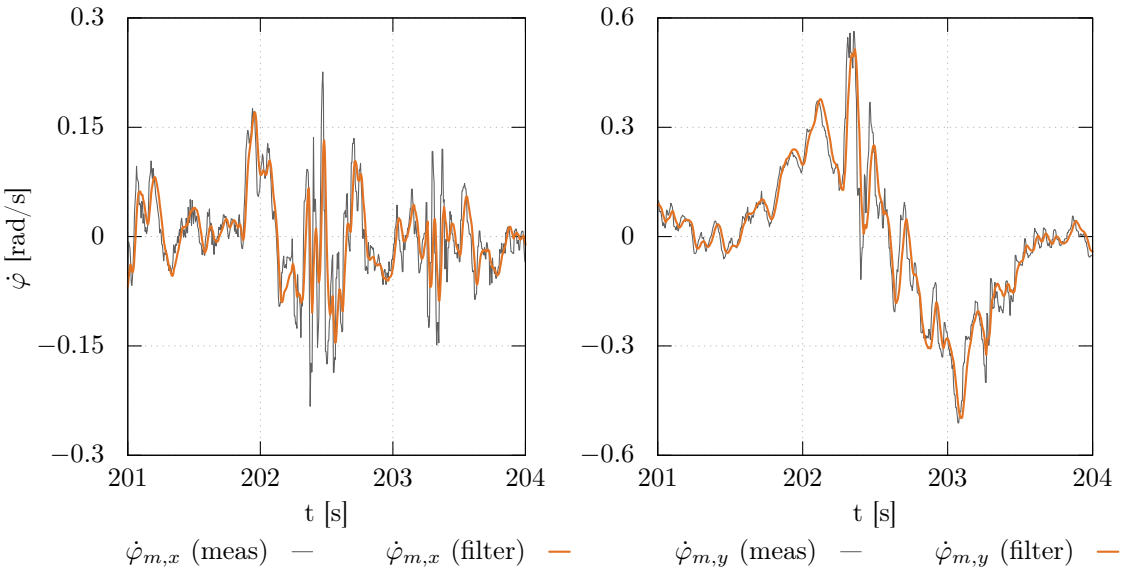


Figure 7.3: Experiment 1a: Filtered inclination rates during an external disturbance.

7.1.2 Center of Gravity Optimization (Experiment 1b)

The same experiment as described above is performed but with the optimization of the CoG. Footstep modifications are determined by the heuristic. The measured force and the resulting inclination errors are shown in Figure 7.4. The control system stabilizes the robot with the modifications depicted in Figure 7.6. Only foot modifications in the x and y direction are shown. A sequence of photographs of the robot for the second push ($t \approx 27.5\,\text{s}$) can be seen in Figure 7.5. In the first picture it is pushed at the left arm, in the next step its motion is adapted according to the disturbed state (robot is inclined to the right) and the third picture shows it again in an upright posture. The according control trajectories u_x and u_y (Figure 7.7) are discontinuous. Note that they are limited to the interval $[-3, 3]$.

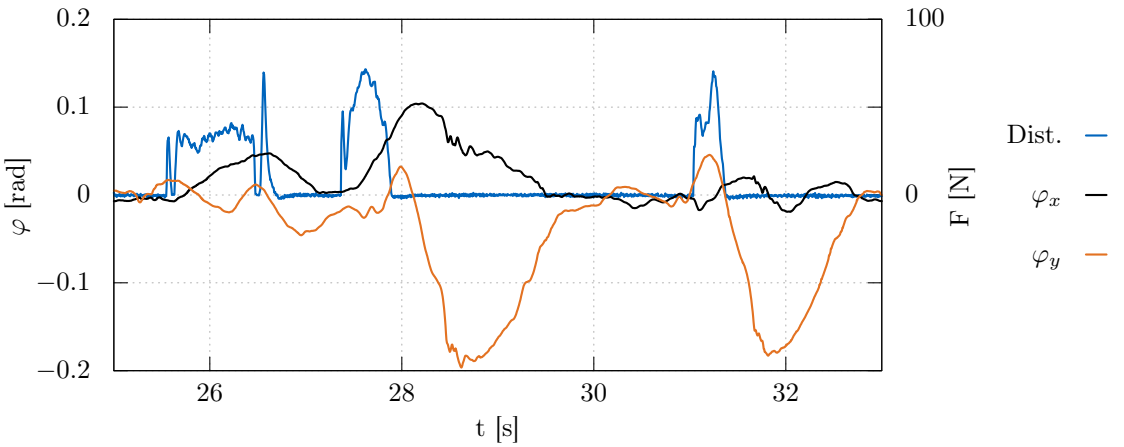


Figure 7.4: Experiment 1b: Resulting inclination errors and disturbance force measurement.

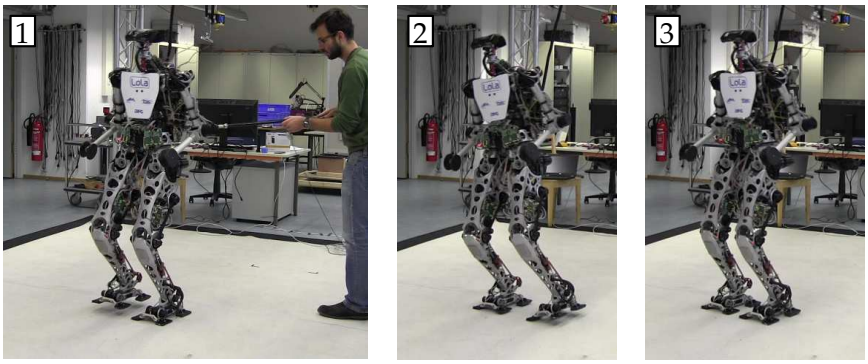


Figure 7.5: Experiment 1b: robot stabilizing after a disturbance.

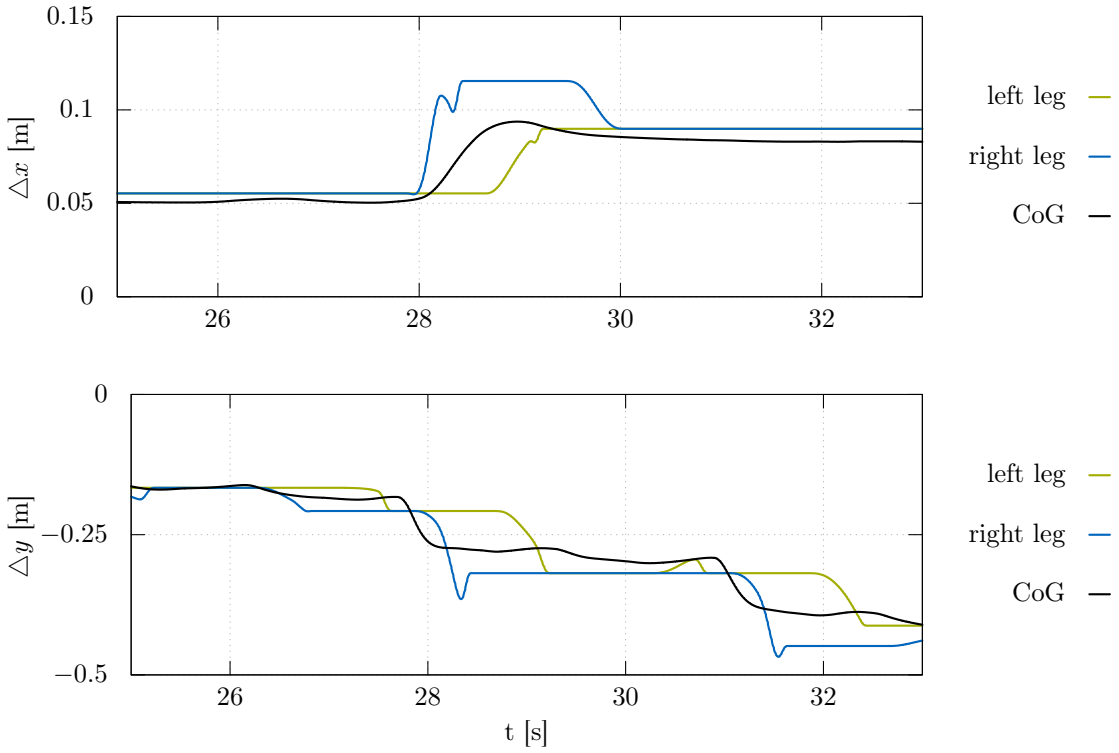


Figure 7.6: Experiment 1b: Foot and CoG position trajectory modifications.

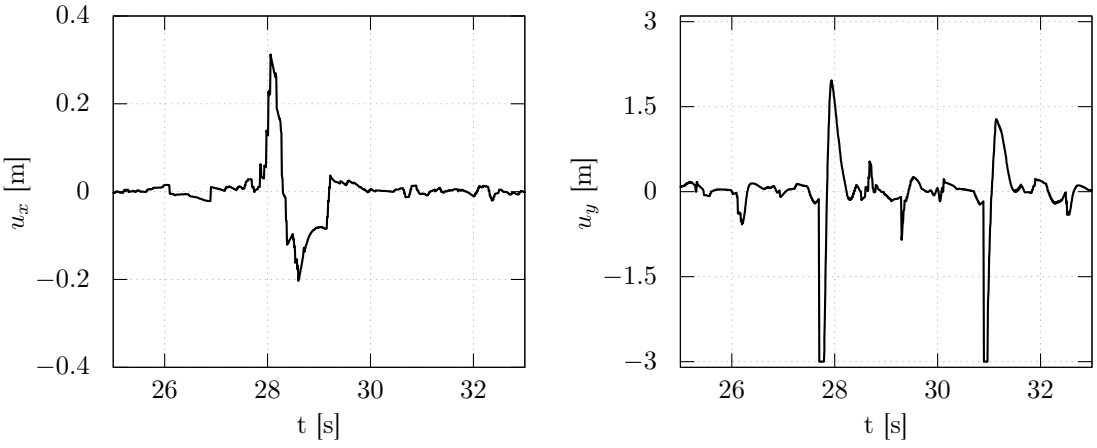


Figure 7.7: Experiment 1b: control trajectories from conjugate gradient algorithm.

7.2 Forward Walking with Disturbances (Experiment 2)

The following experiment is conducted with the decoupled footstep optimization method presented in Section 6.4. The results are already presented in Wittmann et al. (2015b). The humanoid is commanded to walk with a step length of 0.4 m and a step time of 0.8 s forward for 4.5 m. The robot is pushed twice during walking. The corresponding external forces are shown in Figure 7.10. The ideal and the modified step length for this experiment are depicted in Figure 7.8. The approximate moments when the pushes occur are visualized with red arrows. The corresponding trajectory modifications for the foot positions are shown in Figure 7.11, modification of the orientation of the feet is omitted. Photographs of the experiment are shown in Figure 7.9.

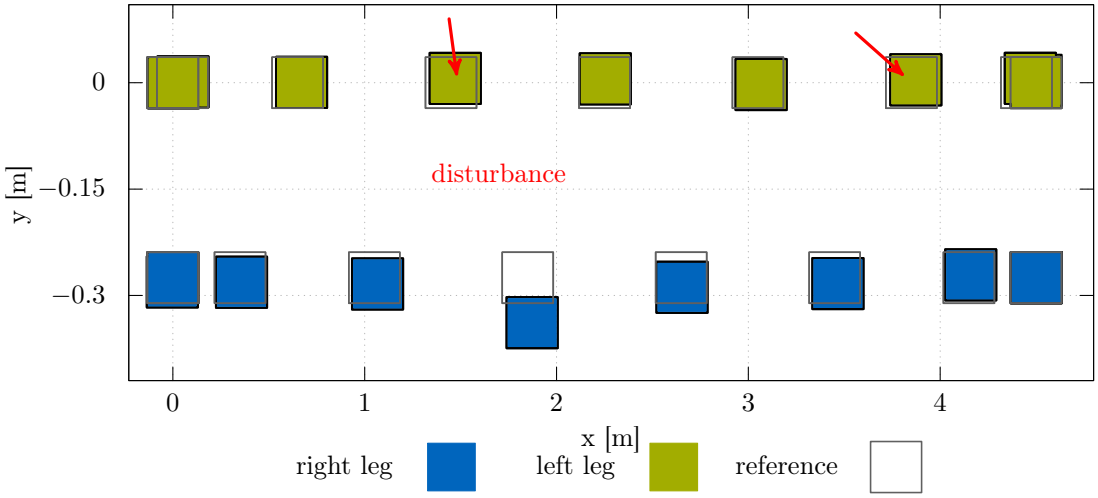


Figure 7.8: Experiment 2: Ideal and modified foot steps of the robot. It is walking with approx. 0.5 m/s while it is pushed twice (adopted from Wittmann et al. (2015b)).

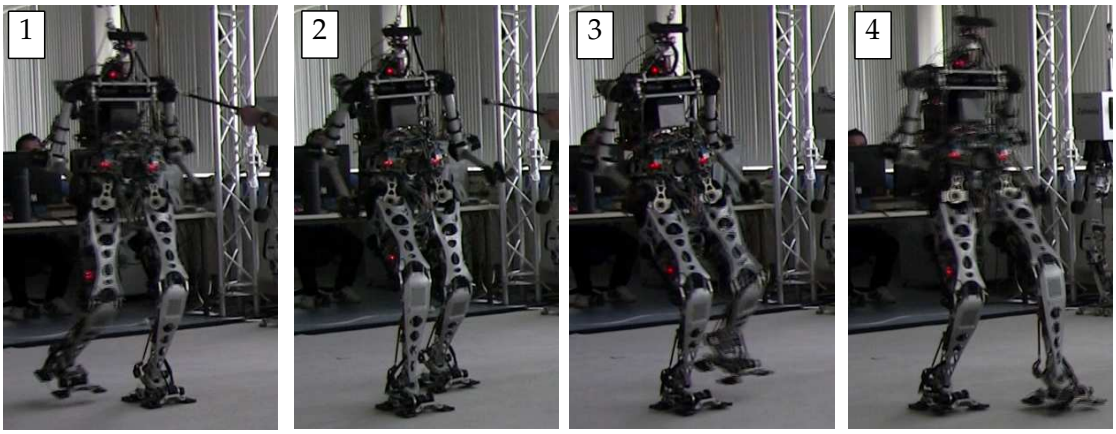


Figure 7.9: Experiment 2: The robot is pushed in the first picture and stabilizes itself (adopted from Wittmann et al. (2015b)).

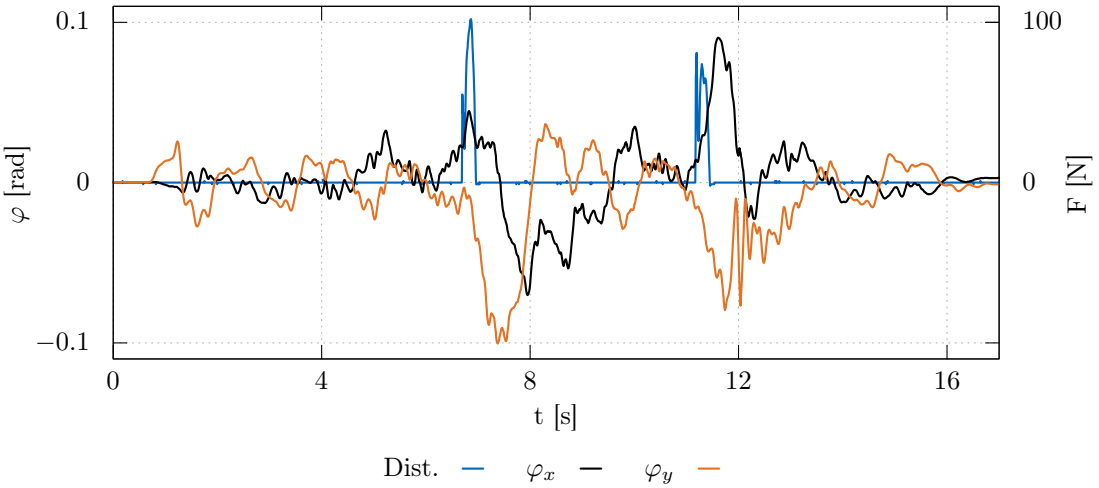


Figure 7.10: Experiment 2: Upper body inclination errors and disturbance force measurement (with external device, adopted from Wittmann et al. (2015b)).

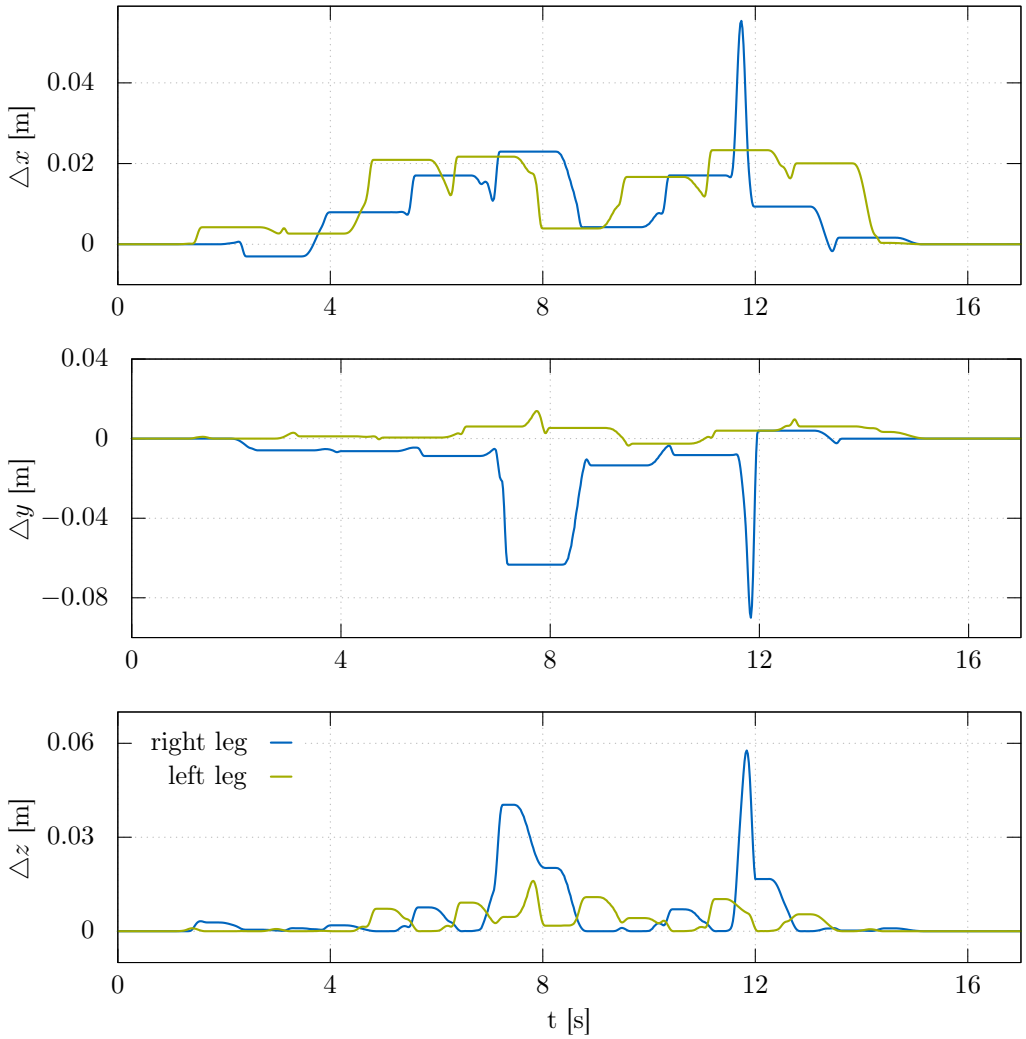


Figure 7.11: Experiment 2: Foot position trajectory modifications (adopted from Wittmann et al. (2015b)).

7.3 Rough Terrain Walking (Experiment 3)

This experiment shows results from walking over unknown rough terrain that is not sensed by an vision system. The scene is depicted in Figure 7.12. The robot is commanded to walk with 30 cm steps and a step time of 0.8 s forward. The method for decoupled footstep optimization is used in this experiment. Figure 7.14 shows the resulting upper body inclinations and Figure 7.13 the trajectory modifications for the foot positions. Foot orientation modifications are not shown. It can be observed that the modifications are quite small. A reason is that the force control compensates for most of the disturbances and the prediction model accounts for this effect.

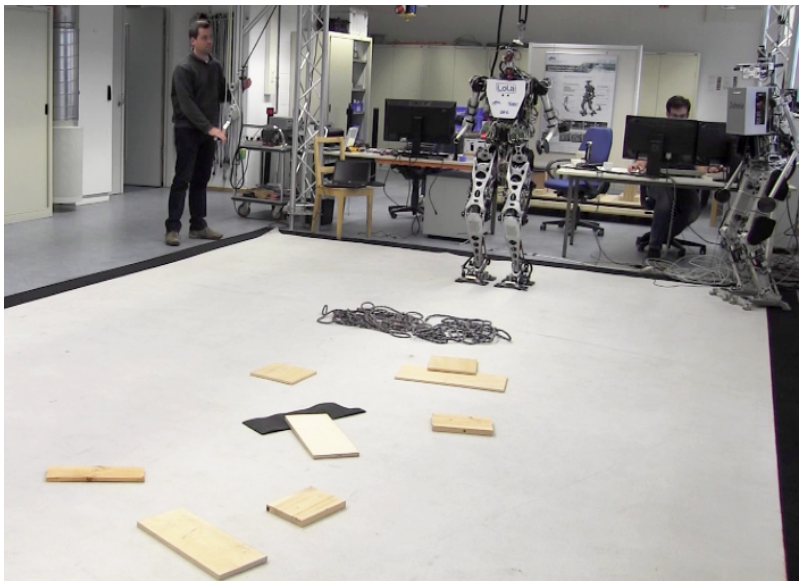


Figure 7.12: Experiment 3: scene of the unknown terrain for the walking experiment.

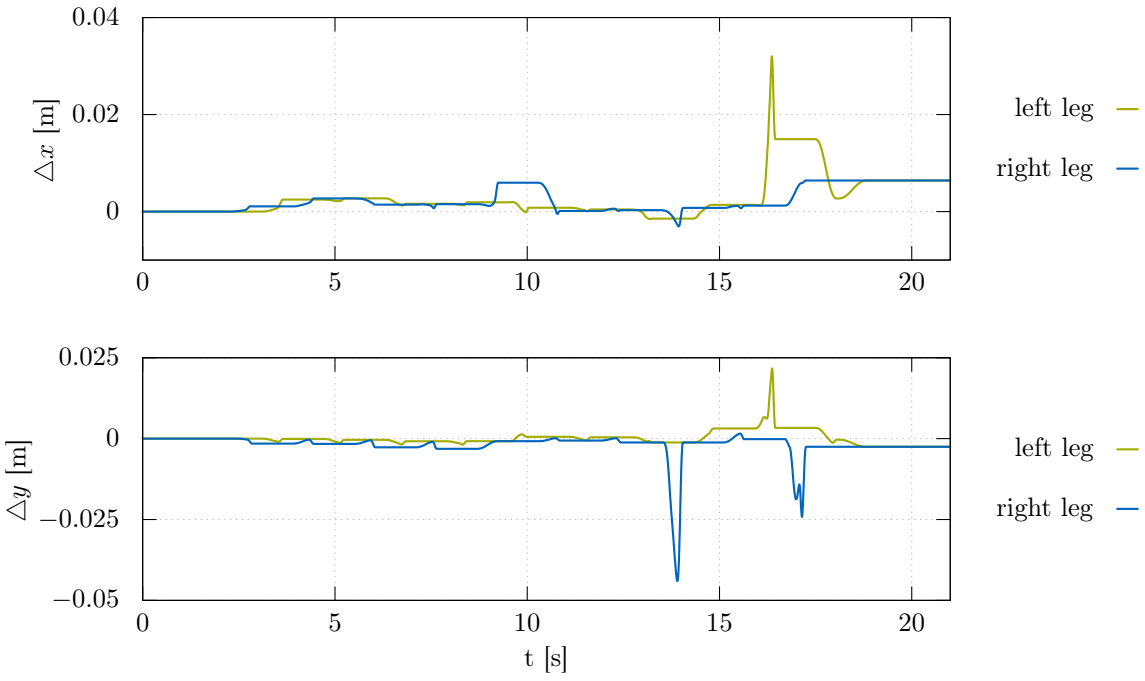


Figure 7.13: Experiment 3: Foot position trajectory modifications.

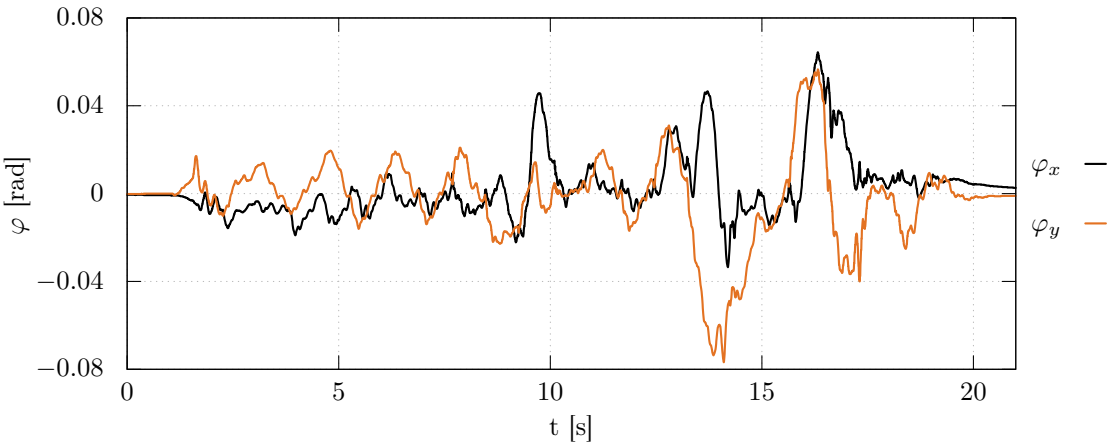


Figure 7.14: Experiment 3: Upper body inclination errors.

7.4 Disturbances with Obstacles

Two experiments will be presented to show the footstep modification with additional constraints due to obstacles under real-world conditions.

7.4.1 Synthetic Case (Experiment 4a)

In the first experiment, the humanoid is commanded to walk in place while subjected to a disturbance force in its walking direction (x-direction). One obstacle is placed close to the robot to limit feasible footstep modifications. It is sent manually to the robot without vision system. The vision system is not used for two reasons: (1) According to the described setup, the obstacle is not in the system's camera field of view. (2) This experiment should examine only the procedure for the method to consider obstacles during disturbance rejection without having the uncertainties of a running vision system. Figure 7.16 shows the resulting inclination errors for the decoupled method. The robot is still able to stabilize itself with the limited foot positions (Figure 7.17). A snapshot at each physical step of the robot and the 2D-polytopes is shown in Figure 7.15.

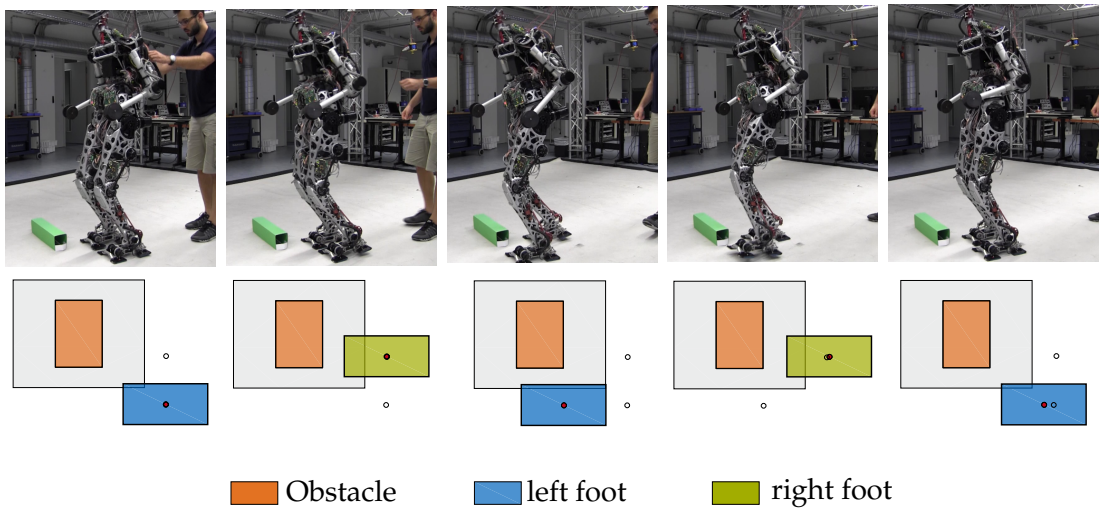


Figure 7.15: Experiment 4a: Snapshots at different time instants from robot and polytopes (adopted from Hildebrandt et al. (2017)).

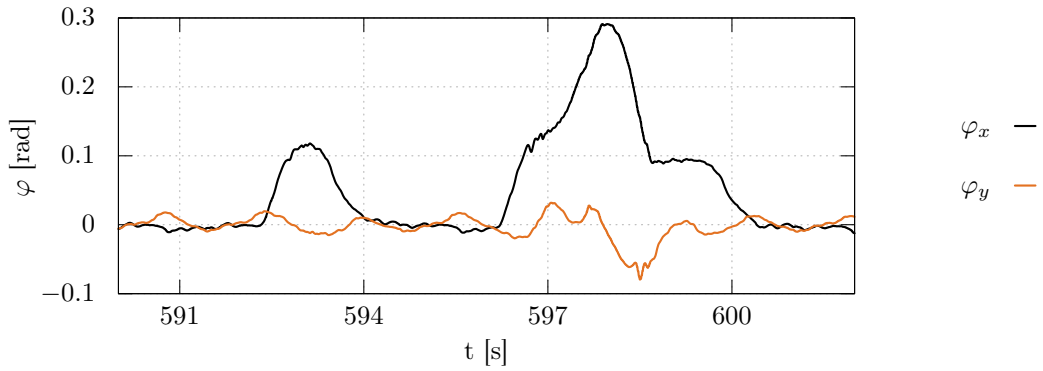


Figure 7.16: Experiment 4a: Resulting inclination errors (adopted from Hildebrandt et al. (2017)).

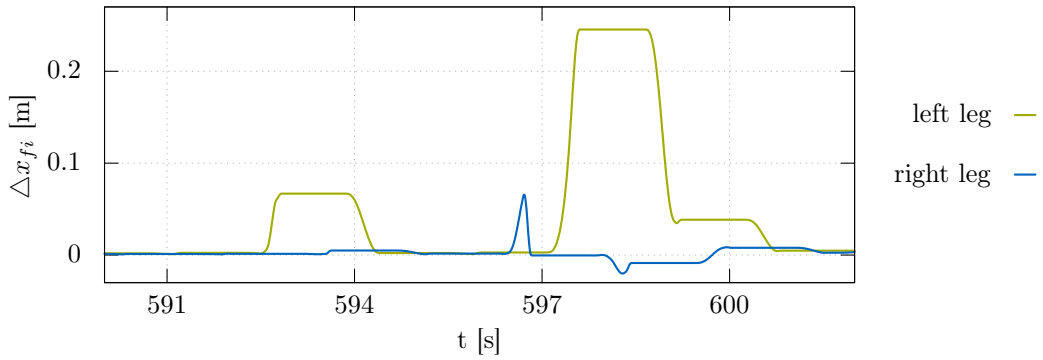


Figure 7.17: Experiment 4a: foot modification trajectories (adopted from Hildebrandt et al. (2017)).

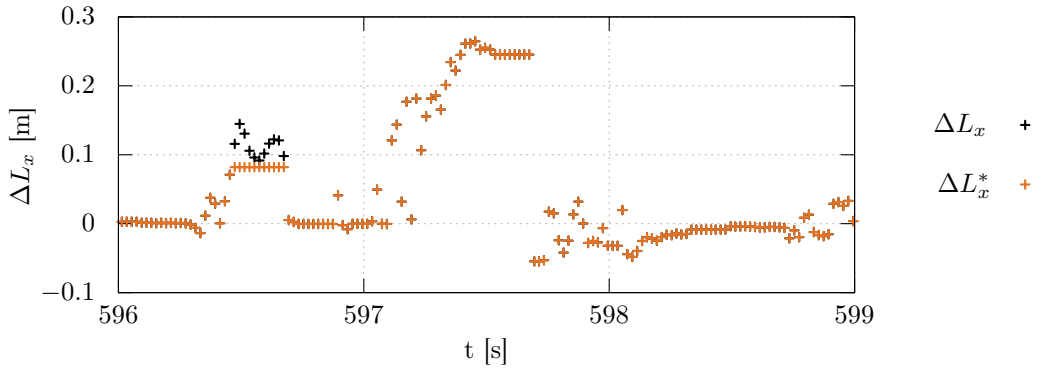


Figure 7.18: Experiment 4a: optimization result before and after evaluation of the constraints for the x direction (adopted from Hildebrandt et al. (2017)).

7.4.2 Forward Walking with Vision System (Experiment 4b)

This experiment also includes the vision system. The setup is presented in Figure 7.21. The robot is commanded to walk forward with 30 cm steps. While walking the robot's walking control receives the online detected obstacles. The module *A*-based step planner & parameter optimization* calculates in real-time an ideal step sequence and parameter set that ensure collision-free movements. The ideal motion is modified based on the robot's state. The robot is pushed several times during the experiment and recovers from the disturbances. The overall ideal step sequence and modified footholds calculated are shown in Figure 7.20. The resulting inclination errors are shown in Figure 7.19.

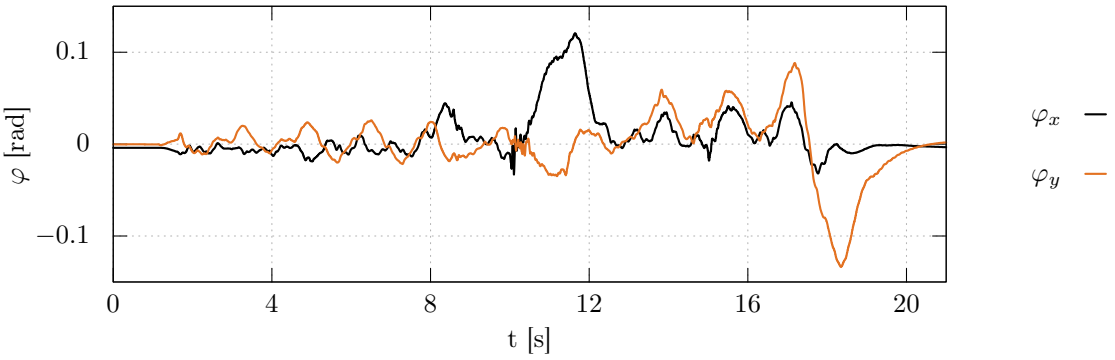


Figure 7.19: Experiment 4b: Inclination errors of the upper body.

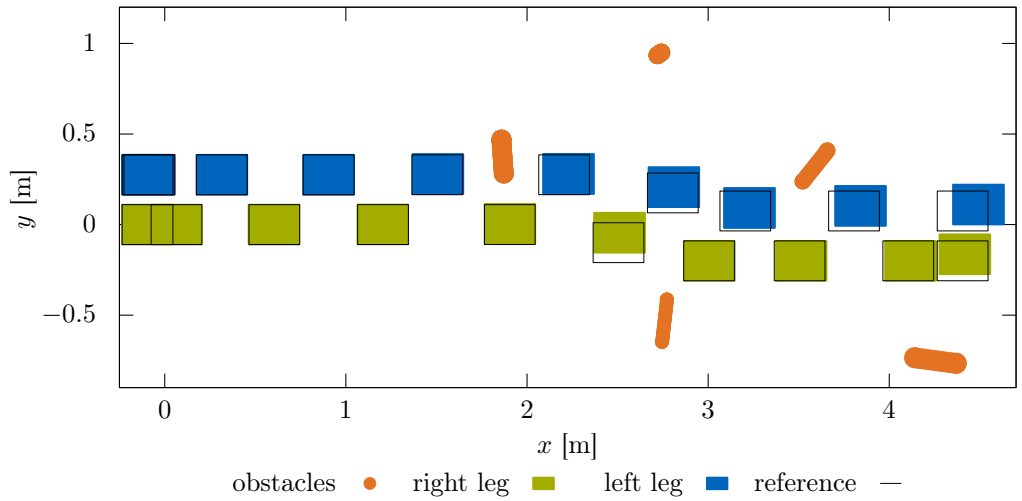


Figure 7.20: Experiment 4b: Ideal and modified foot steps of the robot walking while it is pushed several times.

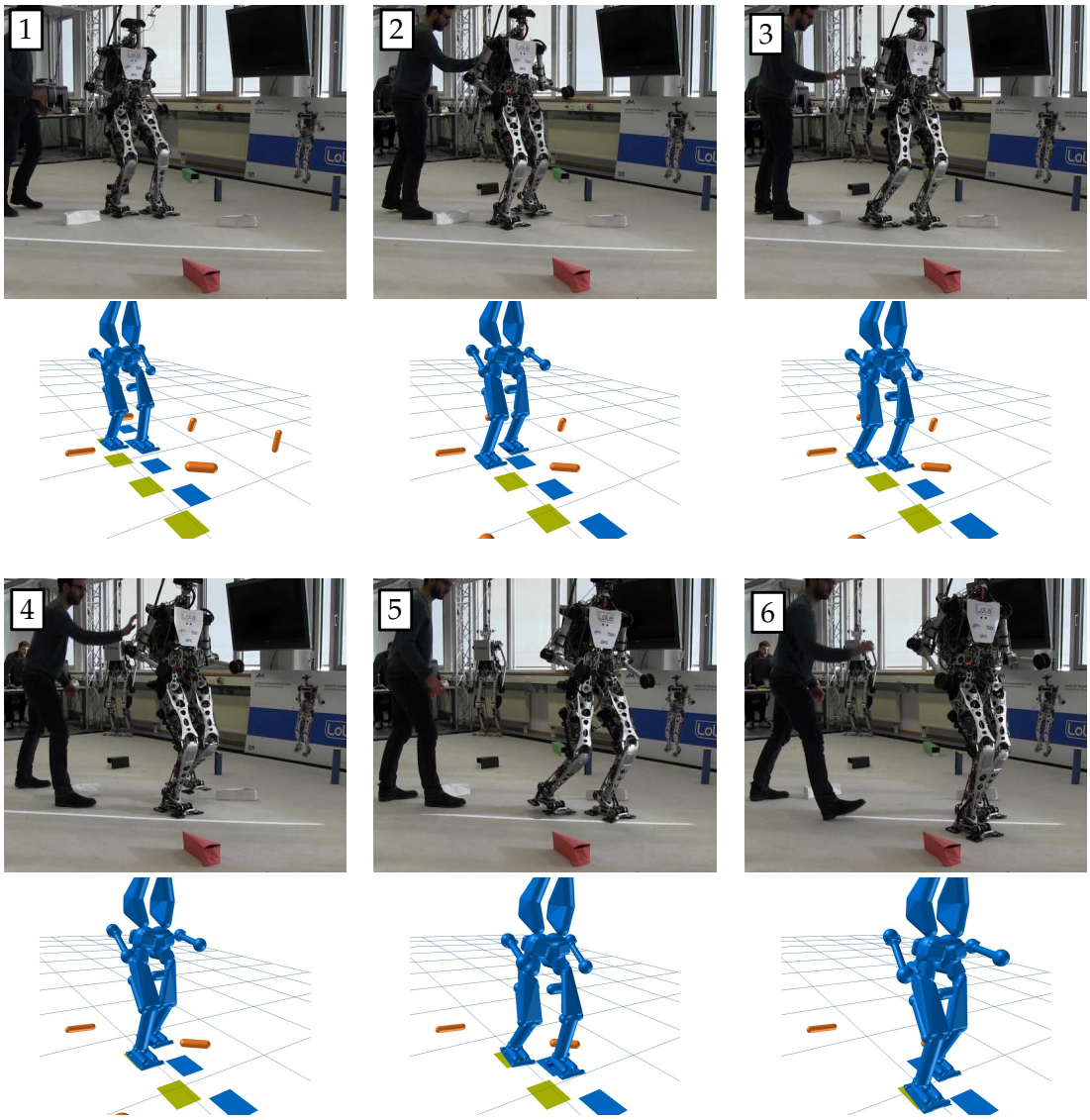


Figure 7.21: Experiment 4b: The robot is pushed several times and stabilizes itself.

Chapter 8

Conclusions

This thesis described methods to increase the robustness of bipedal robots in unknown environments. The following chapter summarizes the methods, discusses the results and finally describes directions for future work.

8.1 Summary and Discussion

Creating stabilization methods for humanoid robots requires to understand the underlying dynamics. Chapter 2 gives an overview of the mechanics of bipedal locomotion. Unilateral contacts constrain the robot's possible motions while the underactuated dynamics cause that the robot's full state can not be controlled directly. Existing motion planning and stabilization methods of bipedal robots are reviewed. The methods are discussed considering the choice of input and output. Special effort is put on the design of overall feedback frameworks. Many walking control systems in literature consist of several feedback mechanisms. Outer feedback loops have to consider the effect of the inner ones and should show a lower dynamics. A consistent consideration of these design concepts is essential for a powerful walking controller.

The experimental platform LOLA is described in Chapter 3. Its hardware and overall walking control system is summarized. The control system has a hierarchical architecture. Walking commands that were sent from a human operator are used to generate online a walking pattern in the global control. The local control adapts this ideal motion locally and generates joint target data that is finally sent to the decentralized joint controls. Based on the existing system architecture a sensor based trajectory adaptation is introduced. It is located in the global control and adapts the ideal planned trajectories according to sensor data. It consists of a nonlinear prediction model, a state estimator and a trajectory adaptation, see Figure 8.1. The input is the upper body inclination state obtained from the IMU. Adapted foot and CoG trajectories are the method's output. Additionally, a learning based feedforward adaptation of the joint target data is presented that improves the tracking performance of all joints by more than 90 %.

The prediction models proposed in Chapter 4 are motivated by the observation that the popular LIPM produces poor prediction results for disturbed states. The planar models are designed to produce a better prediction result of the absolute inclination of the robot. This corresponds directly to its absolute CoG state. It is achieved by adding two passive DoFs which make the model underactuated and by adding unilateral contacts. This way the model maintains two important properties of bipedal robots. The robot is approximated by three point masses that are assumed to perfectly follow planned trajectories in a FoR that rotates with the robot. It is shown that the model can predict fast and accurately the robot's behavior. Trajectory modifications for the feet and the CoG are added to the model to enable the determination and evaluation of stabilizing motions.

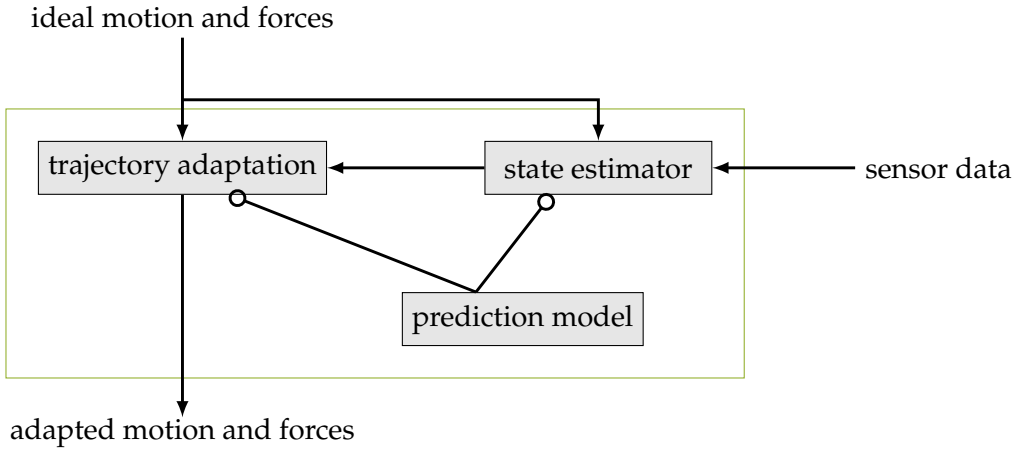


Figure 8.1: Summary of the model predictive trajectory adaptation.

A state estimator is developed in Chapter 5 that extracts the main trend of the robot's motion from the IMU data. Undesired oscillations are removed. This is realized by using the previously introduced prediction model. A state estimator is introduced that is based on an extended Kalman filter. It is extended by an external force state to compensate for unknown disturbances. Additionally model errors are taken into account. It is shown that the estimator produces smoothed inclination rates with less time delay compared to conventional low pass filters. The estimator provides initial values that are used in the trajectory modifications.

Chapter 6 presents methods to modify the robot's ideal planned motion by using the prediction model and the estimated current state. The first part presents an algorithm to adapt the overall foot trajectories. It uses a parameter optimization to determine the next horizontal footstep position. The optimization mainly penalizes the predicted upper body inclination and is solved by a direct shooting method. Final swing foot height and orientations are determined by geometric considerations. The aim is the prevention of early or late touch down of the swing foot which is caused by inclination errors. These parameters are updated continuously and are used to calculate a set of modification trajectories. The second part extends the motion that is adapted by the horizontal CoG trajectory. The resulting problem is stated as optimization problem. A conjugate gradient method is used to determine the unknown trajectory. The algorithm is extended to include additional variables in the optimization. The third part discusses the integration of trajectory adaptation techniques with collision avoidance methods. The main idea preserves the hierarchical structure of the walking control system to first generate a collision free ideal motion that is then adapted when unknown disturbances occur. Consequently obstacles constrain the trajectory adaption. A method is developed that generates a set of geometric constraints which is used to compute safe convex regions for the feet. These regions are used to find a safe foot position as close as possible to the adapted one during a disturbance. The overall framework is designed such that is runs in parallel to collision avoidance methods on the real-time system of the robot.

Experimental results from several walking experiments are presented in Chapter 7. They show how presented methods perform under real world conditions with the bipedal robot LOLA. The following situations are investigated: stepping in place and walking forward with external pushes from arbitrary directions, walking over unknown rough terrain and walking with detected obstacles while the robot is pushed. Especially the last experiment generates a challenging situation since environment recognition, collision avoidance and stabilization methods have to run and interact at the same time online with real sensor data.

The capability to adapt the robot's future motion according to its disturbed state is a very powerful approach. The results show that the walking control system can stabilize the robot for moderate disturbances such as pushes. However by increasing the impulse of the pushes the robot can still be destabilized. The formulated optimization problems use hand-tuned cost functions which may not be the best choice for every situation. Another limiting factor is that up to now only the motion of the feet and the CoG is modified. Since the methods use a dynamic model of the biped further investigations should consider to adapt the model parameters if necessary. Failures in the real-time system, its communication system or the hardware are also related to the robustness of the robot. Examples are rare cabling issues, sensor errors or crashing processes. However, the new ETHERCAT-based communication system and appropriate safety checks reduced such crashes of the overall system.

8.2 Recommendations for Future Work

From the experience gained through the work of the last years and the discussion of the results above, several suggestions for future research can be made:

- **Model based approach combined with learning**:
  The presented approach uses the knowledge of robot models to predict the behavior. However for control approaches there often remain few tuning parameters that are chosen only with experience. Those parameters could be determined similar to the reinforcement learning of the joint feedforward gains. This would combine the knowledge of the robot's mechanics with learning of very few parameters. Therefore investigations are suggested to identify such learning parameters and identify criteria to evaluate the robot's overall performance.

- **Adaptive model parameters**:
  Model prediction accuracy can be improved in situations where the model parameters vary. An example is walking on terrain that is compliant. This changes the contact model between foot and ground. Adding an online parameter estimator for critical values could improve the system's behavior.

- **Using the arms for stabilization**:
  While the proposed model predictive trajectory adaptation for the feet and the CoG works well, very large disturbances can still cause the robot to fall. One possibility beside tuning the method's parameters is to extend the adaptation strategy to the arms. Using the arms for stabilization could improve the overall robustness. For the prediction model it was shown how the additional contact points from the arms could be integrated into the existing framework. With a faster onboard computer and efficient optimization methods the adaptation of the overall motion can be solved in real-time.

- **Adapt step time**:
  A parameter that has large influence to the bipeds motion is the step time. Increasing robustness could be achieved by adding this parameter to the optimization. The reduction of time until the next foot touches the ground shortens the time until a modified step position becomes active. However the coupling with the biped's dynamics is nonlinear.

- **State estimation and error handling**:
  This suggestion is not related to the development of new stabilization methods but has to be considered when the overall walking performance and robustness has

to be increased. The developed ETHERCAT-based communication system together with the motor controls from ELMO increased the robustness of the hardware and communication system and reduced errors. Future work should improve the error handling for the remaining failures. A possible solution is a comprehensive state estimator that monitors the robot's overall "health state" and a state machine that triggers an appropriate recovery or if necessary safe shutdown behavior.

Appendix A

Joint Tracking Performance

The following part contains results from the reinforcement learning process. Additionally measurements of the joint tracking performance and the commanded joint velocities are shown for an experiment with a maximal mean walking speed 0.75 m/s. Only measurements of the pelvis and the right leg joints are shown.

The measurement data is computed from the incremental encoders (motor side) and is converted to link side quantities using the gear ratio of each joint.

Table A.1: Result of the reinforcement learning process.

joint	value (k_{rl})
pelvis rotation	1.0
pelvis adduction	1.0
hip rotation left/right	1.0
hip adduction left/right	1.0
hip flexion left/right	1.3
knee flexion left/right	1.0
ankle flexion left/right	0.8
ankle adduction left/right	0.8
toe flexion left/right	0.6
shoulder flexion left/right	0.8
shoulder adduction left/right	0.8
elbow flexion left/right	0.8

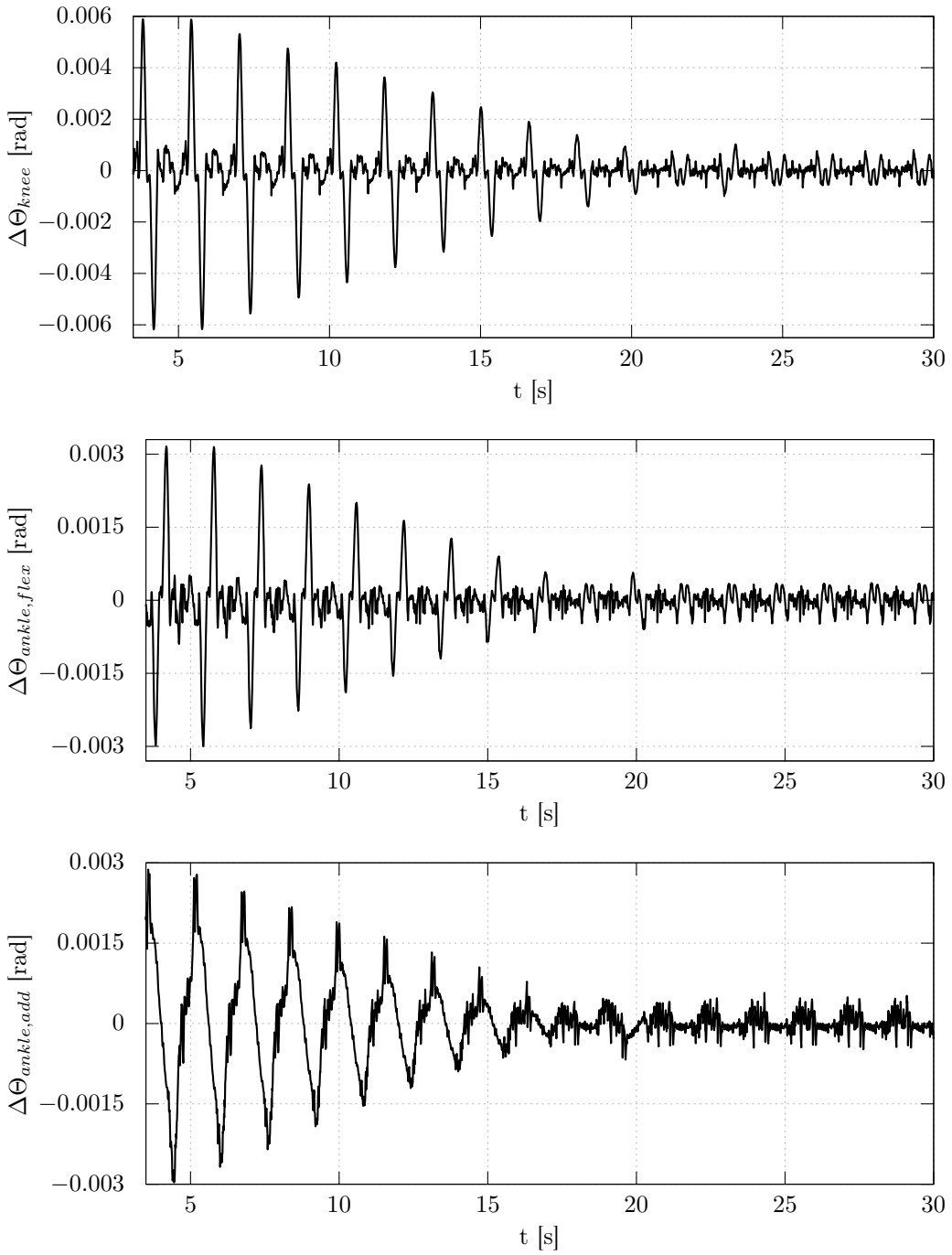


Figure A.1: Reinforcement learning results for knee and ankle joints.

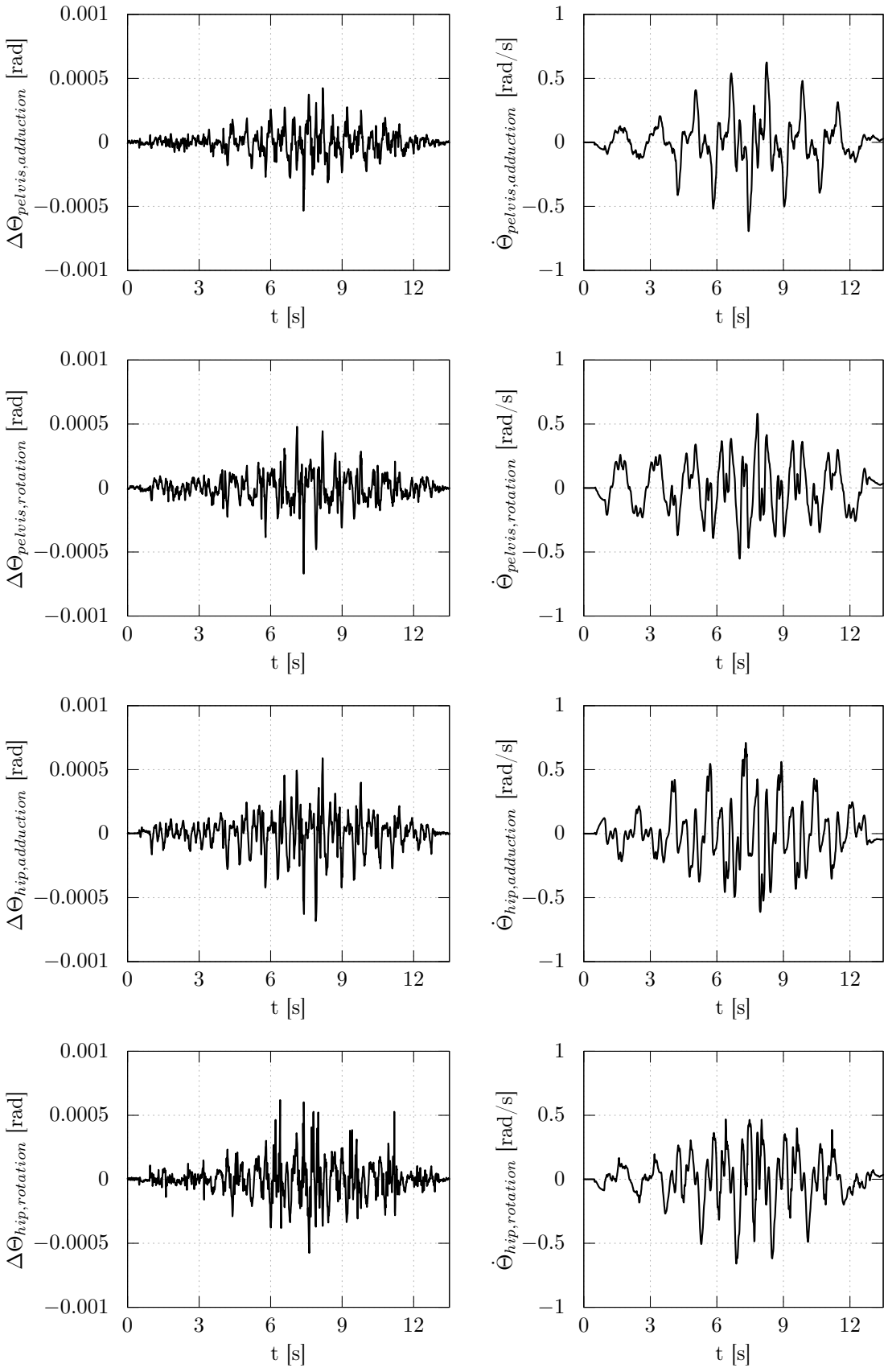


Figure A.2: Joint tracking error and desired velocity while walking with 0.75 m/s.

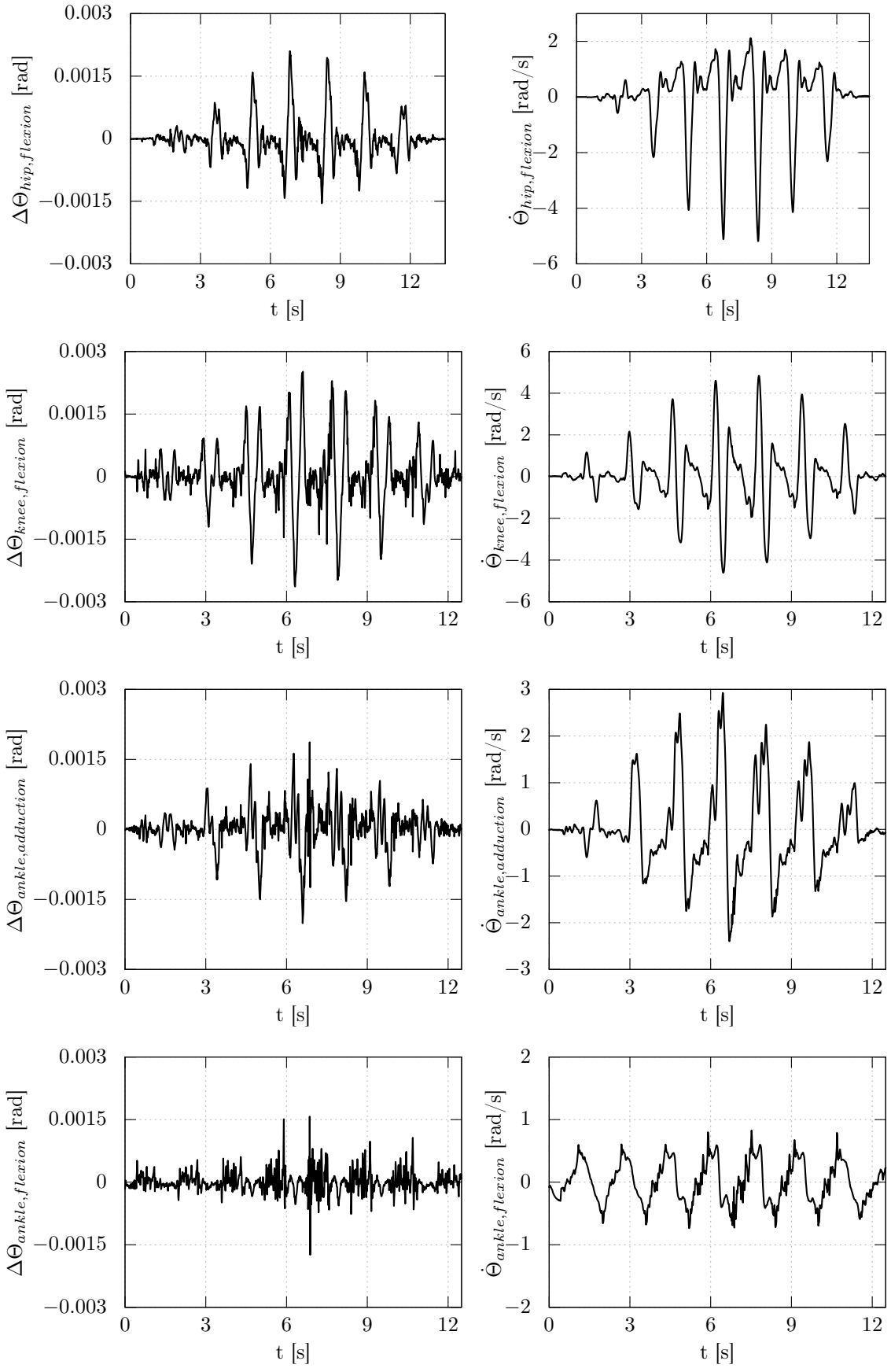


Figure A.3: Joint tracking error and desired velocity while walking with 0.75 m/s.

Appendix B

Prediction Model Gradient Computation

State gradients for the reduced prediction model:

1. Inverse mass matrix $\boldsymbol{M}^{-1}$:

$$\frac{\partial \boldsymbol{M}^{-1}}{\partial z} = \boldsymbol{0}$$
$$\frac{\partial \boldsymbol{M}^{-1}}{\partial \varphi} = -\frac{\partial \det(\boldsymbol{M})}{\partial \varphi}\frac{1}{\det(\boldsymbol{M})^2}\hat{\boldsymbol{M}}^{-1} + \frac{1}{\det(\boldsymbol{M})}\frac{\partial \hat{\boldsymbol{M}}^{-1}}{\partial \varphi}$$

with

$$\boldsymbol{M}^{-1} = \frac{1}{\det(\boldsymbol{M})}\hat{\boldsymbol{M}}^{-1} = \frac{1}{\det(\boldsymbol{M})}\begin{bmatrix} \overline{mx^2 + z^2} + \Theta_{yy} & \overline{mx}c_\varphi + \overline{mz}s_\varphi \\ \overline{mx}c_\varphi + \overline{mz}s_\varphi & m \end{bmatrix}$$
$$\det(\boldsymbol{M}) = m(\overline{mx^2 + z^2} + \Theta_{yy}) + (\overline{mx}c_\varphi + \overline{mz}s_\varphi)^2$$
$$\frac{\partial \det(\boldsymbol{M})}{\partial \varphi} = 2(\overline{mx}c_\varphi + \overline{mz}s_\varphi)(-\overline{mx}s_\varphi + \overline{mz}c_\varphi)$$
$$\frac{\partial \hat{\boldsymbol{M}}^{-1}}{\partial \varphi} = \begin{bmatrix} 0 & -\overline{mx}s_\varphi + \overline{mz}c_\varphi \\ -\overline{mx}s_\varphi + \overline{mz}c_\varphi & 0 \end{bmatrix}$$

2. Vector $\boldsymbol{h}$:

$$\frac{\partial \boldsymbol{h}}{\partial \boldsymbol{q}} = \begin{bmatrix} 0 & -\overline{mz}\ddot{s}_\varphi - \overline{mx}\ddot{c}_\varphi + \overline{mz}\dot{\varphi}_x^2 s_\varphi + \overline{mx}\dot{\varphi}_x^2 c_\varphi + 2\overline{mx}\dot{\varphi}_x s_\varphi - 2\overline{mz}\dot{\varphi}_x c_\varphi \\ 0 & \overline{mx}gs_\varphi - \overline{mz}gc_\varphi \end{bmatrix}$$
$$\frac{\partial \boldsymbol{h}}{\partial \dot{\boldsymbol{q}}} = \begin{bmatrix} 0 & -2\overline{mz}\dot{\varphi}_x c_\varphi + 2\overline{mx}\dot{\varphi}_x s_\varphi - 2\overline{mx}\dot{c}_\varphi - 2\overline{mz}\dot{s}_\varphi \\ 0 & 2\overline{mx}\dot{x} + 2\overline{mz}\dot{z} \end{bmatrix}$$

3. Contact deformation $\Delta z_{r,i}$:

$$\frac{\partial \Delta z_{r,i}}{\partial \boldsymbol{q}} = \begin{bmatrix} 1 & -x_{fi}(t)\cos\varphi_x - z_{fi}(t)\sin\varphi_x \end{bmatrix}$$
$$\frac{\partial \Delta \dot{z}_{r,i}}{\partial \boldsymbol{q}} = \begin{bmatrix} 0 & -\dot{x}_{fi}(t)\cos\varphi_x + x_{fi}(t)\dot{\varphi}_x\sin\varphi_x - \dot{z}_{fi}(t)\sin\varphi_x - z_{fi}(t)\dot{\varphi}_x\cos\varphi_x \end{bmatrix}$$
$$\frac{\partial^2 \Delta z_{r,i}}{\partial \boldsymbol{q}^2} = \begin{bmatrix} 0 & 0 \\ 0 & x_{fi}(t)\sin\varphi_x - z_{fi}(t)\cos\varphi_x \end{bmatrix}$$
$$\frac{\partial^2 \Delta z_{r,i}}{\partial \boldsymbol{q}\partial \dot{\boldsymbol{q}}} = \frac{\partial \Delta z_{r,i}}{\partial \dot{\boldsymbol{q}}} = \boldsymbol{0}$$

Modification gradients for $\Delta L_x = p$:

$$\begin{aligned}
\frac{\partial \Delta z_{r,i}}{\partial p} &= -\sin\varphi_x \\
\frac{\partial \Delta \dot{z}_{r,i}}{\partial p} &= -\dot{\varphi}_x \cos\varphi_x \\
\frac{\partial^2 \Delta z_{r,i}}{\partial \boldsymbol{q} \partial p} &= \begin{bmatrix} 0 & -\cos\varphi_x \end{bmatrix}
\end{aligned}$$

Modification gradients for $x_b(t)$ and its derivatives:

$$\begin{aligned}
\frac{\partial \boldsymbol{M}^{-1}}{\partial x_b} &= -\frac{\partial \det(\boldsymbol{M})}{\partial x_b} \frac{1}{\det(\boldsymbol{M})^2} \hat{\boldsymbol{M}}^{-1} + \frac{1}{\det(\boldsymbol{M})} \frac{\partial \hat{\boldsymbol{M}}^{-1}}{\partial x_b} \\
\frac{\partial \det(\boldsymbol{M})}{\partial x_b} &= 2m_b^2 x_b + 2(\overline{mx} c_\varphi + \overline{mz} s_\varphi) m_b c_\varphi \\
\frac{\partial \hat{\boldsymbol{M}}}{\partial x_b} &= \begin{bmatrix} 2m_b x_b & m_b c_\varphi \\ m_b c_\varphi & 0 \end{bmatrix} \\
\frac{\partial \boldsymbol{h}}{\partial x_b} &= \begin{bmatrix} m_b \dot{\varphi}_x^2 s_\varphi \\ m_b \ddot{z}_b + 2m_b \dot{x}_b \dot{\varphi}_x - m_b g c_\varphi \end{bmatrix} \\
\frac{\partial \boldsymbol{h}}{\partial \dot{x}_b} &= \begin{bmatrix} -2m_b \dot{\varphi}_x c_\varphi \\ 2m_b x_b \dot{\varphi}_x \end{bmatrix} \\
\frac{\partial \boldsymbol{h}}{\partial \ddot{x}_b} &= \begin{bmatrix} -m_b s_\varphi \\ m_b z_b \end{bmatrix}
\end{aligned}$$

Appendix C

Alternative Derivation of Pontryagin's Minimum Principle for Problem B

This part gives an alternative derivation of Pontryagin's Minimum Principle for problem B. It can be derived by stating the augmented cost function for the original problem (6.5) and (6.9)

$$\begin{aligned}\bar{J} =& s(\boldsymbol{z}_{r,ext}(t_e), p) + \int_{t_a}^{t_e} h(\boldsymbol{z}_{r,ext}, p, \boldsymbol{u}, t) + \boldsymbol{\psi}^T(\boldsymbol{f}(\boldsymbol{z}_{r,ext}, p, \boldsymbol{u}, t) - \dot{\boldsymbol{z}}_{r,ext})\mathrm{d}t \\ &+ \boldsymbol{\psi}_a^T\left(\boldsymbol{z}_{r,ext,a} - \boldsymbol{z}_{r,ext}(t_a)\right)\end{aligned}$$

with the Lagrange multipliers $\boldsymbol{\psi}(t)$ and $\boldsymbol{\psi}_a$. Using the Hamiltonian $H(\boldsymbol{z}_{r,ext}, p, \boldsymbol{u}, t, \boldsymbol{\psi}) = h(\boldsymbol{z}_{r,ext}, p, \boldsymbol{u}, t) + \boldsymbol{\psi}^T\boldsymbol{f}(\boldsymbol{z}_{r,ext}, p, \boldsymbol{u}, t)$ the first variation is

$$\begin{aligned}\delta\bar{J} = &\left.\frac{\partial s}{\partial \boldsymbol{z}_{r,ext}}\right|_{t_e}\delta\boldsymbol{z}_{r,ext}(t_e) + \left.\frac{\partial s}{\partial p}\right|_{t_e}\delta p + \delta\boldsymbol{\psi}_a^T\left(\boldsymbol{z}_{r,ext,a} - \boldsymbol{z}_{r,ext}(t_a)\right) + \boldsymbol{\psi}_a\cdot\boldsymbol{0} \\ &+ \int_{t_a}^{t_e}\frac{\partial H}{\partial \boldsymbol{z}_{r,ext}}\delta z + \frac{\partial H}{\partial p}\delta p + \frac{\partial H}{\partial \boldsymbol{u}}\delta\boldsymbol{u} + \frac{\partial H}{\partial \boldsymbol{\psi}}\delta\boldsymbol{\psi} - \delta\boldsymbol{\psi}^T\dot{\boldsymbol{y}} - \boldsymbol{\psi}^T\delta\dot{\boldsymbol{y}}\ \mathrm{d}t.\end{aligned}$$

Integrating $\int \boldsymbol{\psi}^T\delta\dot{\boldsymbol{y}}$ by parts and rearranging leads to

$$\begin{aligned}\delta\bar{J} = &\left(\left.\frac{\partial s}{\partial \boldsymbol{z}_{r,ext}}\right|_{t_e} - \boldsymbol{\psi}(t_e)^T\right)\delta\boldsymbol{z}_{r,ext}(t_e) + \delta\boldsymbol{\psi}_0^T\left(\boldsymbol{z}_{r,ext,a} - \boldsymbol{z}_{r,ext}(t_a)\right) \\ &+ \int_{t_a}^{t_e}\left(\frac{\partial H}{\partial \boldsymbol{z}_{r,ext}} + \dot{\boldsymbol{\psi}}^T\right)\delta\boldsymbol{z}_{r,ext} + \left(\frac{\partial H}{\partial \boldsymbol{\psi}} - \dot{\boldsymbol{y}}^T\right)\delta\boldsymbol{\psi} + \frac{\partial H}{\partial \boldsymbol{u}}\delta\boldsymbol{u}\ \mathrm{d}t \\ &+ \left.\frac{\partial s}{\partial p}\right|_{t_e}\delta p + \int_{t_a}^{t_e}\frac{\partial H}{\partial p}\delta p\ \mathrm{d}t\end{aligned}$$

The variation of the parameter δp in the last term can be extracted from the integrand because it is constant over time. The conditions for the optimal solution are finally

$$\left.\frac{\partial s}{\partial \boldsymbol{z}_{r,ext}}\right|_{t_e} - \boldsymbol{\psi}(t_e)^T = 0 \qquad \boldsymbol{z}_{r,ext,a} - \boldsymbol{z}_{r,ext}(t_a) = 0 \tag{C.1}$$

$$\frac{\partial H}{\partial \boldsymbol{z}_{r,ext}} + \dot{\boldsymbol{\psi}}^T = 0 \qquad \frac{\partial H}{\partial \boldsymbol{\psi}} - \dot{\boldsymbol{z}}_{r,ext}^T = 0 \tag{C.2}$$

$$\frac{\partial H}{\partial \boldsymbol{u}}\delta\boldsymbol{u} = 0 \tag{C.3}$$

$$\left(\left. \frac{\partial s}{\partial p} \right|_{t_e} + \int_{t_a}^{t_e} \frac{\partial H}{\partial p} \, \mathrm{d}t \right) \delta p = 0. \tag{C.4}$$

These are the same conditions as for an optimal control problem with free final state and fixed final time with the additional condition (C.4) for the free parameters p.

Appendix D

Supervised Student Theses

Within the scope of this thesis a number of student theses were supervised by the author at the chair of applied mechanics. They result with the scientific, technical and content guidance of the author of the present work. Various issues were investigated concerning robust bipedal walking. Some parts of those supervised theses may have been incorporated into the present thesis. The author would like to express his sincere gratitude to all formerly supervised students for their commitment supporting this research project. The following list chronologically summarizes student theses related to the topic:

- TOBIAS BERNINGER:
  Dezentrale Gelenkregelung für humanoide Roboter. Master thesis 2016.
  Parts of the thesis influenced the presented method in Subsection 3.3.3.

- LISA JESCHEK:
  Robustes Gehen unter geometrischen Beschränkungen. Master thesis 2016.
  Methods and results of the thesis are the base for Section 6.6.

- STEFANIE ZIMMERMANN:
  Modellbasierte Optimierung für robustes zweibeiniges Laufen. Bachelor thesis 2016.
  Initial implementation of the spatial model of Section 4.5 and its application in Subsection 6.4.2 was done in this work.

- MARC ANDREAS KLESER:
  Modellprädiktive Trajektorienoptimierung für robustes zweibeiniges Laufen. Diploma thesis 2016.
  The method presented in Section 6.5 is based on the algorithm of the thesis.

- FELIX ELLENSOHN:
  Entwicklung einer Kollokationsmethode zur Online-Trajektorienoptimierung für Humanoide Roboter. Master thesis 2016.

- PHILIPP SEIWALD:
  Entwicklung eines Zustandsschätzers für zweibeinige Laufroboter. Semester thesis 2015.
  The experience with this thesis helped to develop the state estimator in Section 5.2.

- ANNA-KAARINA SEPPÄLÄ:
  Integration of kinematic information into the state estimator of a humanoid robot. Master thesis 2015.

Bibliography

Adamy, J. (2009). *Nichtlineare Regelungen*. Berlin, Heidelberg: Springer Berlin Heidelberg.

Aftab, Z., T. Robert, and P.-B. Wieber (2012). "Predicting multiple step placements for human balance recovery tasks." In: *Journal of biomechanics* 45.16.

Antoulas, A. C. (2009). *Approximation of Large-Scale Dynamical Systems*. Advances in Design and Control. Society for Industrial and Applied Mathematics.

Aubin, J.-P. (1991). *Viability theory*. Springer Science & Business Media.

Bessonnet, G. (2004). "Optimal Gait Synthesis of a Seven-Link Planar Biped". In: *The International Journal of Robotics Research* 23.10-11, pp. 1059–1073.

Betts, J. T. (1998). "Survey of Numerical Methods for Trajectory Optimization". In: *Survey of numerical methods for trajectory optimization* 21.2, pp. 193–207.

Bloesch, M., M. Hutter, M. A. Hoepflinger, S. Leutenegger, C. Gehring, C. D. Remy, and R. Siegwart (2012). "State Estimation for Legged Robots - Consistent Fusion of Leg Kinematics and IMU". In: *Proceedings of Robotics: Science and Systems*.

Boček, M. (1980). "Conjugate gradient algorithm for optimal control problems with parameters". In: *Kybernetika* 16.5, pp. 454–461.

Bogacki, P. and L. Shampine (1989). "A 3(2) Pair of Runge-Kutta Formulas". In: *Appl Math Lett* 2.4, pp. 321–325.

Bryson, A. E. (1969). *Applied optimal control: optimization, estimation and control*. CRC Press.

Buschmann, T., S. Lohmeier, H. Ulbrich, and F. Pfeiffer (2005). "Optimization based gait pattern generation for a biped robot". In: *IEEE/RAS International Conference on Humanoid Robots*. IEEE, pp. 98–103.

Buschmann, T. (2010). "Simulation and Control of Biped Walking Robots". PhD thesis.

Buschmann, T., A. Ewald, H. Ulbrich, and A. Buschges (2012). "Event-based walking control - From neurobiology to biped robots". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, pp. 1793–1800.

Buschmann, T., V. Favot, S. Lohmeier, M. Schwienbacher, and H. Ulbrich (2011). "Experiments in fast biped walking". In: *IEEE International Conference on Mechatronics*. IEEE, pp. 863–868.

Buschmann, T., S. Lohmeier, M. Bachmayer, H. Ulbrich, and F. Pfeiffer (2007). "A collocation method for real-time walking pattern generation". In: *IEEE/RAS International Conference on Humanoid Robots*. IEEE, pp. 1–6.

Buschmann, T., S. Lohmeier, and H. Ulbrich (2009). "Biped walking control based on hybrid position/force control". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, pp. 3019–3024.

Chestnutt, J. and Y. Takaoka (2010). "Safe adjustment regions for legged locomotion paths". In: *IEEE International Conference on Humanoid Robotics*. IEEE, pp. 224–229.

Chevallereau, C., D. Djoudi, and J. Grizzle (2008). "Stable Bipedal Walking With Foot Rotation Through Direct Regulation of the Zero Moment Point". In: *IEEE Transactions on Robotics* 24.2, pp. 390–401.

Deits, R. and R. Tedrake (2014). "Footstep planning on uneven terrain with mixed-integer convex optimization". In: *2014 IEEE-RAS International Conference on Humanoid Robots*. Vol. 2015-Febru. IEEE, pp. 279–286.

— (2015). "Computing large convex regions of obstacle-free space through semidefinite programming". In: *Springer Tracts in Advanced Robotics* 107, pp. 109–124.

Diedam, H., D. Dimitrov, P.-B. Wieber, K. Mombaur, and M. Diehl (2008). "Online walking gait generation with adaptive foot positioning through Linear Model Predictive control". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, pp. 1121–1126.

Diehl, M., H. Ferreau, and N. Haverbeke (2009). "Efficient numerical methods for nonlinear MPC and moving horizon estimation". In: *Nonlinear Model Predictive Control*, pp. 391–417.

Dimitrov, D., P.-B. Wieber, H. J. Ferreau, and M. Diehl (2008). "On the implementation of model predictive control for on-line walking pattern generation". In: *IEEE International Conference on Robotics and Automation*. IEEE, pp. 2685–2690.

Englsberger, J., C. Ott, M. a. Roa, A. Albu-Schaffer, and G. Hirzinger (2011). "Bipedal walking control based on Capture Point dynamics". In: *IEEE/RSJ International Conference on Intelligent Robots and System*. IEEE, pp. 4420–4427.

Englsberger, J. and C. Ott (2012). "Integration of vertical COM motion and angular momentum in an extended Capture Point tracking controller for bipedal walking". In: *IEEE-RAS International Conference on Humanoid Robots*, pp. 183–189.

Feng, S., E. Whitman, X. Xinjilefu, and C. G. Atkeson (2014). "Optimization based full body control for the atlas robot". In: *14th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*. IEEE, pp. 120–127.

Fujimoto, Y., S. Obata, and A. Kawamura (1998). "Robust biped walking with active interaction control between foot and ground". In: *IEEE International Conference on Robotics and Automation*. Vol. 3. May. IEEE, pp. 2030–2035.

Furusho, J. and M. Masubuchi (1987). "A theoretically motivated reduced order model for the control of dynamic biped locomotion". In: *Journal of Dynamic Systems, Measurement, and Control* 109.2, pp. 155–163.

Geering, H. (2007). *Optimal control with engineering applications*.

Goswami, A. (1999). "Foot rotation indicator (FRI) point: a new gait planning tool to evaluate postural stability of biped robots". In: *Proceedings 1999 IEEE International Conference on Robotics and Automation*. Vol. 1. May, pp. 47–52.

Goswami, A. and V. Kallem (2004). "Rate of change of angular momentum and balance maintenance of biped robots". In: *IEEE International Conference on Robotics and Automation*. IEEE, 3785–3790 Vol.4.

Graichen, K. (2015). *Methoden der optimierung und optimalen steuerung*.

Graichen, K. and B. Käpernick (2011). "A Real-Time Gradient Method for Nonlinear Model Predictive Control". In: *Frontiers of Model Predictive Control* 2010, pp. 9–28.

Griewank, A. (2000). *Evaluating derivatives : principles and techniques of algorithmic differentiation*. 19. Siam.

Grizzle, J., E. Westervelt, and C. Canudas-de-Wit (2003). "Event-based PI control of an underactuated biped walker". In: *42nd IEEE International Conference on Decision and Control*. Vol. 3. December. IEEE, pp. 3091–3096.

Harada, K., S. Kajita, K. Kaneko, and H. Hirukawa (2004). "An analytical method on real-time gait planning for a humanoid robot". In: *IEEE/RAS International Conference on Humanoid Robots*. Vol. 2. IEEE, pp. 640–655.

Hashimoto, K., H.-j. Kang, M. Nakamura, E. Falotico, H.-o. Lim, A. Takanishi, C. Laschi, P. Dario, and A. Berthoz (2012). "Realization of biped walking on soft ground with stabilization control based on gait analysis". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2. IEEE, pp. 2064–2069.

Hermann, R. and A. Krener (1977). "Nonlinear controllability and observability". In: *IEEE Transactions on Automatic Control* 22.5, pp. 728–740.

Hildebrandt, A.-C., R. Wittmann, D. Wahrmann, A. Ewald, and T. Buschmann (2014). "Real-time 3D collision avoidance for biped robots". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, pp. 4184–4190.

Hildebrandt, A.-C., M. Demmeler, R. Wittmann, D. Wahrmann, F. Sygulla, D. Rixen, and T. Buschmann (2016). "Real-Time Predictive Kinematic Evaluation and Optimization for Biped Robots". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. Ed. by IEEE.

Hildebrandt, A.-C., D. Wahrmann, R. Wittmann, D. Rixen, and T. Buschmann (2015). "Real-Time Pattern Generation Among Obstacles for Biped Robots". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, pp. 2780–2786.

Hildebrandt, A.-C., R. Wittmann, D. Wahrmann, F. Sygulla, D. Rixen, and T. Buschmann (2017). "Versatile and Robust Bipedal Walking in Unknown Environments". In: *IEEE Transactions on Robotics (submitted)*.

Hirai, K., M. Hirose, Y. Haikawa, and T. Takenaka (1998). "The development of Honda humanoid robot". In: *IEEE International Conference on Robotics and Automation*. Vol. 2. IEEE, pp. 1321–1326.

Hirose, M. and K. Ogawa (2007). "Honda humanoid robots development." In: *Philosophical transactions. Series A, Mathematical, physical, and engineering sciences* 365.1850, pp. 11–19.

Hobbelen, D. G. E. and M. Wisse (2009). "Active Lateral Foot Placement for 3D Stabilization of a Limit Cycle Walker Prototype". In: *International Journal of Humanoid Robotics* 06.01, pp. 93–116.

Hodgins, J. and M. Raibert (1991). "Adjusting step length for rough terrain locomotion". In: *IEEE Transactions on Robotics and Automation* 7.3, pp. 289–298.

Huang, A.-C., Y.-F. Chen, and C.-Y. Kai (2015). *Adaptive Control of Underactuated Mechanical Systems*. River Edge, NJ, USA: World Scientific Publishing Co., Inc.

Huang, Q. H. Q., S. Kajita, N. Koyachi, K. Kaneko, K. Yokoi, H. Arai, K. Komoriya, and K. Tanie (1999). "A high stability, smooth walking pattern for a biped robot". In: *Proceedings 1999 IEEE International Conference on Robotics and Automation*. Vol. 1. May, pp. 65–71.

Isermann, R. (2011). *Identification of Dynamic Systems*. Springer Berlin.

Kajita, S., H. Hirukawa, K. Harada, and K. Yokoi (2014). *Introduction to Humanoid Robotics*. Springer Tracts in Advanced Robotics. Springer Berlin Heidelberg.

Kajita, S., F. Kanehiro, K. Kaneko, K. Fujiwara, K. Harada, K. Yokoi, and H. Hirukawa (2003). "Biped walking pattern generation by using preview control of zero-moment point". In: *IEEE International Conference on Robotics and Automation*. IEEE, pp. 1620–1626.

Kajita, S., M. Morisawa, K. Miura, S. Nakaoka, K. Harada, K. Kaneko, F. Kanehiro, and K. Yokoi (2010). "Biped walking stabilization based on linear inverted pendulum tracking". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, pp. 4489–4496.

Kajita, S. and K. Tani (1995). "Experimental study of biped dynamic walking in the linear inverted pendulum mode". In: *Proceedings of 1995 IEEE International Conference on Robotics and Automation*. Vol. 3, pp. 2885–2891.

Kajita, S., F. Kanehiro, K. Kando, K. Yokoi, and H. Hirukawa (2001). "The 3D Linear Inverted Pendulum Mode: A simple modeling for a biped walking pattern generation". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, pp. 239–246.

Kajita, S., A. Kobayashit, and K. Tani (1990). "Dynamic walking control of a biped robot along a potential energy conserving orbit". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, pp. 789–794.

Kalman, R. (1960). "A new approach to linear filtering and prediction problems". In: *Journal of basic Engineering* 82.Series D, pp. 35–45.

Kaneko, K., F. Kanehiro, S. Kajita, H. Hirukawa, T. Kawasaki, M. Hirata, K. Akachi, and T. Isozumi (2004). "Humanoid robot HRP-2". In: *IEEE International Conference on Robotics and Automation*. April. IEEE, 1083–1090 Vol.2.

Khalil, H. K. and J. W. Grizzle (2002). *Nonlinear systems*. Vol. 3. Prentice hall Upper Saddle River.

Koch, K. H., D. Clever, K. Mombaur, and D. M. Endres (2015). "Learning Movement Primitives from Optimal and Dynamically Feasible Trajectories for Humanoid Walking". In: *IEEE/RAS International Conference on Humanoid Robots (Humanoids 2015)*, pp. 866–873.

Koolen, T., T. de Boer, J. Rebula, A. Goswami, and J. Pratt (2012). "Capturability-based analysis and control of legged locomotion, Part 1: Theory and application to three simple gait models". In: *The International Journal of Robotics Research* 31.9, pp. 1094–1113.

Kröger, T. (2010). *On-Line Trajectory Generation in Robotic Systems: Basic Concepts for Instantaneous Reactions to Unforeseen (Sensor) Events*. Vol. 58. Springer.

Kuindersma, S., R. Deits, M. F. Andr, H. Dai, F. Permenter, K. Pat, and M. Russ (2015). "Optimization-based Locomotion Planning , Estimation , and Control Design for the Atlas Humanoid Robot". In: *Autonomous Robots* 40.3, pp. 1–27.

Kuindersma, S., F. Permenter, and R. Tedrake (2014). "An efficiently solvable quadratic program for stabilizing dynamic locomotion". In: *IEEE International Conference on Robotics and Automation*. IEEE, pp. 2589–2594.

Kwon, S. (2007). "Estimation of the center of mass of humanoid robot". In: *International Conference on Control, Automation and Systems*. IEEE, pp. 2705–2709.

Lanari, L. and S. Hutchinson (2015). "Inversion-based gait generation for humanoid robots". In: *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 1. IEEE, pp. 1592–1598.

Lasdon, L., S. Mitter, and A. Waren (1967). "The conjugate gradient method for optimal control problems". In: *IEEE Transactions on Automatic Control* 12.2, pp. 132–138.

Leine, R. I. and H. Nijmeijer (2004). *Dynamics and Bifurcations of Non-Smooth Mechanical Systems*. Vol. 18. Lecture Notes in Applied and Computational Mechanics. Berlin, Heidelberg: Springer Berlin Heidelberg.

Li, W., L. Jolla, and E. Todorov (2004). "Iterative Linear Quadratic Regulator Design for Nonlinear Biological Movement Systems". In: *International Conference on Informatics in Control, Automation and Robotics*, pp. 222–229.

Liégeois, A. (1977). "Automatic Supervisory Control of the Configuration and Behavior of Multibody Mechanisms". In: *IEEE Transactions on Systems, Man, and Cybernetics* 7.12, pp. 868–871.

Liu, H., W. Liu, and L. J. Latecki (2010). "Convex shape decomposition". In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 97–104.

Löffler, K., M. Gienger, and F. Pfeiffer (2002). "Model based control of a biped robot". In: *7th International Workshop on Advanced Motion Control*. IEEE, pp. 443–448.

Löffler, K. (2006). "Dynamik und Regelung einer zweibeinigen Laufmaschine". PhD thesis.

Lohmeier, S. (2009). "System design and control of anthropomorphic walking robot LOLA". In: *IEEE/ASME Transactions on Mechatronics* 14.6, pp. 658–666.

— (2010). "Design and Realization of a Humanoid Robot for Fast and Autonomous Bipedal Locomotion". PhD thesis.

Masuya, K. and T. Sugihara (2013). "A Dual-stage Complementary Filter for Dead Reckoning of a Biped Robot via Estimated Contact Point". In: *IEEE/RAS International Conference on Humanoid Robots*. IEEE.

Matsumoto, T., T. Takenaka, and T. Yoshiike (2004). *Gait generation device for legged mobile robot*.

Maybeck, P. S. (1979). *Stochastic models, estimation, and control*. Vol. 141. Mathematics in Science and Engineering.

Mayr, J., H. Gattringer, and H. Bremer (2012). "A Bipedal Walking Pattern Generator that Considers Multi-Body Dynamics by Angular Momentum Estimation". In: pp. 177–182.

M'Closkey, R. T. and J. W. Burdick (1993). "Periodic Motions of a Hopping Robot With Vertical and Forward Motion". In: *The International Journal of Robotics Research* 12.3, pp. 197–218.

Missura, M. and S. Behnke (2015). "Gradient-Driven Online Learning of Bipedal Push Recovery". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 387–392.

Morisawa, M., K. Harada, and S. Kajita (2007). "Experimentation of humanoid walking allowing immediate modification of foot place based on analytical solution". In: *IEEE International Conference on Robotics and Automation*. April. IEEE, pp. 10–14.

Morisawa, M., S. Kajita, F. Kanehiro, K. Kaneko, K. Miura, and K. Yokoi (2012). "Balance control based on Capture Point error compensation for biped walking on uneven terrain". In: *IEEE/RAS International Conference on Humanoid Robots*. IEEE, pp. 734–740.

Naveau, M., M. Kudruss, O. Stasse, C. Kirches, K. Mombaur, and P. Souères (2017). "A ReactiveWalking Pattern Generator Based on Nonlinear Model Predictive Control". In: *IEEE Robotics and Automation Letters* 2.1, pp. 10–17.

Nelson, G., A. Saunders, N. Neville, B. Swilling, J. Bondaryk, D. Billings, C. Lee, R. Playter, and M. Raibert (2012). "PETMAN: A Humanoid Robot for Testing Chemical Protective Clothing". In: *Journal of the Robotics Society of Japan* 30.4, pp. 372–377.

Nishiwaki, K., J. Chestnutt, and S. Kagami (2012). "Autonomous Navigation of a Humanoid Robot over Unknown Rough Terrain using a Laser Range Sensor". In: *The International Journal of Robotics Research*.

Nishiwaki, K. and S. Kagami (2009a). "Online Walking Control System for Humanoids with Short Cycle Pattern Generation". In: *The International Journal of Robotics Research* 28.6, pp. 729–742.

Nishiwaki, K. and S. Kagami (2006). "High frequency walking pattern generation based on preview control of ZMP". In: *IEEE/RAS International Conference on Humanoid Robots*. May. IEEE, pp. 2667–2672.

— (2007a). "Sensor feedback modification methods that are suitable for the short cycle pattern generation of humanoid walking". In: *IEEE/RSJ International Conference on Intelligent Robots and System*. IEEE, pp. 4214–4220.

— (2007b). "Walking control on uneven terrain with short cycle pattern generation". In: *IEEE/RAS International Conference on Humanoid Robots*. IEEE, pp. 447–453.

— (2009b). "Frequent walking pattern generation that uses estimated actual posture for robust walking control". In: *IEEE/RAS International Conference on Humanoid Robots*. IEEE, pp. 535–541.

Nocedal, J. and S. J. Wright (2004). "Numerical Optimization". In: pp. 1–651.

Ott, C., R. Mukherjee, and Y. Nakamura (2010). "Unified Impedance and Admittance Control". In: *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pp. 554–561.

Ott, C., M. a. Roa, and G. Hirzinger (2011). "Posture and balance control for biped robots based on contact force optimization". In: *IEEE/RAS International Conference on Humanoid Robots*. IEEE, pp. 26–33.

Park, I.-w., J.-y. Kim, J. Lee, and J.-h. Oh (2005). "Mechanical design of humanoid robot platform KHR-3 (KAIST humanoid robot - 3: HUBO)". In: *IEEE/RAS International Conference on Humanoid Robots*. Vol. 3. IEEE, pp. 321–326.

Pongsak, L., O. Masafumi, and Y. Nakamura (2002). "Optimal filtering for humanoid robot state estimators". In: *Proceedings of SICE System Integration Division Annual Conference*. 4, pp. 5–6.

Posa, M., C. Cantu, and R. Tedrake (2013). "A direct method for trajectory optimization of rigid bodies through contact". In: *The International Journal of Robotics Research* 33.1, pp. 69–81.

Pratt, J. (2001). "Virtual Model Control: An Intuitive Approach for Bipedal Locomotion". In: *The International Journal of Robotics Research* 20.2, pp. 129–143.

Pratt, J., T. Koolen, T. de Boer, J. Rebula, S. Cotton, J. Carff, M. Johnson, and P. Neuhaus (2012). "Capturability-based analysis and control of legged locomotion, Part 2: Application to M2V2, a lower-body humanoid". In: *The International Journal of Robotics Research* 31.10, pp. 1117–1133.

Pratt, J. and R. Tedrake (2006). "Velocity-based stability margins for fast bipedal walking". In: *Fast Motions in Biomechanics and Robotics*, pp. 1–27.

Pratt, J., J. Carff, S. Drakunov, and A. Goswami (2006). "Capture Point: A Step toward Humanoid Push Recovery". In: *IEEE/RAS International Conference on Humanoid Robots*. IEEE, pp. 200–207.

Raibert, M. (1986). *Legged Robots that Balance*. Cambridge, Massachusetts: The MIT Press series in artificial intelligence.

Ramos, O. E., M. García, N. Mansard, O. Stasse, J.-B. Hayet, and P. Souères (2014). "Toward Reactive Vision-Guided Walking on Rough Terrain: An Inverse-Dynamics Based Approach". In: *International Journal of Humanoid Robotics* 11.02, p. 1441004.

Rebula, J., F. Canas, J. Pratt, and A. Goswami (2007). "Learning Capture Points for humanoid push recovery". In: *IEEE/RAS International Conference on Humanoid Robots*. IEEE, pp. 65–72.

Renjewski, D., A. Sprowitz, A. Peekema, M. Jones, and J. Hurst (2015). "Exciting Engineered Passive Dynamics in a Bipedal Robot". In: *IEEE Transactions on Robotics* 31.5, pp. 1244–1251.

Romano, F., D. Pucci, and F. Nori (2014). "Collocated Adaptive Control of Underactuated Mechanical Systems". In: 31.6, pp. 1527–1536.

Sardain, P. and G. Bessonnet (2004). "Forces Acting on a Biped Robot. Center of Pressure - Zero Moment Point". In: *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans* 34.5, pp. 630–637.

Sarmiento, A., R. Murrieta-Cid, and S. Hutchinson (2005). "A sample-based convex cover for rapidly finding an object in a 3-D environment". In: *Proceedings - IEEE International Conference on Robotics and Automation*. Vol. 2005. April, pp. 3486–3491.

Schuetz, C., T. Buschmann, J. Baur, J. Pfaff, and H. Ulbrich (2014). "Predictive Online Inverse Kinematics for Redundant Manipulators". In: *IEEE International Conference on Robotics and Automation*. IEEE, pp. 5056–5061.

Schwienbacher, M. (2014). "Efficient Algorithms for Biped Robots". PhD thesis.

Schwienbacher, M., T. Buschmann, S. Lohmeier, V. Favot, and H. Ulbrich (2011). "Self-collision avoidance and angular momentum compensation for a biped humanoid robot". In: *IEEE International Conference on Robotics and Automation*. IEEE, pp. 581–586.

Sherikov, A., D. Dimitrov, and P.-b. Wieber (2014). "Whole body motion controller with long-term balance constraints". In: *IEEE/RAS International Conference on Humanoid Robots*. IEEE, pp. 444–450.

Siciliano, B., L. Sciavicco, and L. Villani (2009). *Robotics: modelling, planning and control*. Ed. by And. Springer.

Stephens, B. J. (2011). "State estimation for force-controlled humanoid balance using simple models in the presence of modeling error". In: *IEEE International Conference on Robotics and Automation*. IEEE, pp. 3994–3999.

Stephens, B. J. and C. G. Atkeson (2010). "Push Recovery by stepping for humanoid robots with force controlled joints". In: *IEEE/RAS International Conference on Humanoid Robots*. IEEE, pp. 52–59.

Sugihara, T. and Y. Nakamura (2005). "A Fast Online Gait Planning with Boundary Condition Relaxation for Humanoid Robots". In: *IEEE International Conference on Robotics and Automation*. April. IEEE, pp. 305–310.

Sutton, R. S. and A. G. Barto (1998). *Reinforcement learning: An introduction*. Cambridge: MIT press.

Tajima, R., D. Honda, and K. Suga (2009). "Fast running experiments involving a humanoid robot". In: *IEEE International Conference on Robotics and Automation*. IEEE, pp. 1571–1576.

Tajima, R. and K. Suga (2006). "Motion having a Flight Phase: Experiments Involving a One-legged Robot". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, pp. 1726–1731.

Takenaka, T., T. Matsumoto, and T. Yoshiike (2009a). "Real time motion generation and control for biped robot - 1st report: Walking gait pattern generation-". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, pp. 1084–1091.

— (2009b). "Real time motion generation and control for biped robot - 3rd report: Dynamics error compensation-". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, pp. 1594–1600.

Takenaka, T., T. Matsumoto, T. Yoshiike, T. Hasegawa, S. Shirokura, H. Kaneko, and A. Orita (2009c). "Real time motion generation and control for biped robot -4th report: Integrated balance control". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, pp. 1601–1608.

Tassa, Y., T. Erez, and E. Todorov (2012). "Synthesis and stabilization of complex behaviors through online trajectory optimization". In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4906–4913.

Ulbrich, H. (1996). *Maschinendynamik*. Teubner Verlag.

Urata, J., Y. Nakanishi, K. Okada, and M. Inaba (2010). "Design of High Torque and High Speed Leg Module". In: *IEEE International Conference on Intelligent Robots and Systems*. IEEE, pp. 4497–4502.

Urata, J., K. Nishiwaki, Y. Nakanishi, K. Okada, S. Kagami, and M. Inaba (2011). "Online decision of foot placement using singular LQ preview regulation". In: *IEEE-RAS International Conference on Humanoid Robots*. IEEE, pp. 13–18.

Vukobratovic, M. and B. Borovac (2004). "Zero-moment point - thirty five years of its life". In: *International Journal of Humanoid Robotics* 1.1, pp. 157–173.

Vukobratovic, M. and J. Stepanenko (1972). "On The Stability of Anthropomorphic Systems". In: *Mathematical Bioscience* 15, pp. 1–37.

Wahrmann, D., A.-C. Hildebrandt, R. Wittmann, D. Rixen, and T. Buschmann (2016). "Fast Object Approximation for Real-Time 3D Obstacle Avoidance with Biped Robots (submitted)". In: *IEEE International Conference on Advanced Intelligent Mechatronics.*

Westervelt, E., J. Grizzle, and D. Koditschek (2003). "Hybrid zero dynamics of planar biped walkers". In: *IEEE Transactions on Automatic Control* 48.1, pp. 42–56.

Westervelt, E., C. Chevallereau, B. Morris, J. Grizzle, and J. Ho Choi (2007). *Feedback Control of Dynamic Bipedal Robot Locomotion*. Vol. 26. Automation and Control Engineering. CRC Press.

Wieber, P.-B. (2002). "On the stability of walking systems". In: *Proceedings of the Third IARP International Workshop on Humanoid and Human Friendly Robotics*, pp. 1–7.

Wieber, P.-B. (2006). "Trajectory Free Linear Model Predictive Control for Stable Walking in the Presence of Strong Perturbations". In: *IEEE-RAS International Conference on Humanoid Robots*. IEEE, pp. 137–142.

— (2008). "Viability and predictive control for safe locomotion". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, pp. 1103–1108.

Wight, D. L., E. G. Kubica, and D. W. L. Wang (2008). "Introduction of the Foot Placement Estimator: A Dynamic Measure of Balance for Bipedal Robotics". In: *Journal of Computational and Nonlinear Dynamics* 3.1, p. 011009.

Wittmann, R., A.-C. Hildebrandt, A. Ewald, and T. Buschmann (2014). "An Estimation Model for Footstep Modifications of Biped Robots". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, pp. 2572–2578.

Wittmann, R., A.-C. Hildebrandt, D. Wahrmann, T. Buschmann, and D. Rixen (2015a). "State Estimation for Biped Robots Using Multibody Dynamics". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, pp. 2166–2172.

Wittmann, R., A.-C. Hildebrandt, D. Wahrmann, F. Sygulla, D. Rixen, and T. Buschmann (2016). "Model-Based Predictive Bipedal Walking Stabilization". In: *IEEE-RAS International Conference on Humanoid Robots*.

Wittmann, R., A.-C. Hildebrandt, D. Wahrmann, D. Rixen, and T. Buschmann (2015b). "Real-Time Nonlinear Model Predictive Footstep Optimization for Biped Robots". In: *IEEE-RAS International Conference on Humanoid Robots*, pp. 711–717.

Wittmann, R. and D. Rixen (2016). "A Prediction Model for State Observation and Model Predictive Control of Biped Robots". In: *Proc. Appl. Math. Mech.* Vol. 16. 1. Wiley-Blackwell, pp. 65–66.

Xinjilefu, X., S. Feng, and C. G. Atkeson (2014). "Dynamic State Estimation using Quadratic Programming". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. 1. IEEE.

Xinjilefu and C. G. Atkeson (2012). "State estimation of a walking humanoid robot". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, pp. 3693–3699.

Yamaguchi, J. and A. Takanishi (1996). "Multisensor foot mechanism with shock absorbing material for dynamic biped walking adapting to unknown uneven surfaces". In: *IEEE/SICE/RSJ International Conference on Multisensor Fusion and Integration for Intelligent Systems*. IEEE.