University of Duisburg-Essen
Faculty of Engineering
Chair of Mechatronics

Highly-Dynamic Movements of a Humanoid Robot Using Whole-Body Trajectory Optimization

Master Thesis

Maschinenbau (M.Sc.)

Julian Eßer

Student ID: 3015459

First examiner	Prof. Dr. Dr. h.c. Frank Kirchner (DFKI)
Second examiner	Dr.-Ing. Tobias Bruckmann (UDE)
Supervisor	Dr. rer. nat. Shivesh Kumar (DFKI)
Supervisor	Dr. Carlos Mastalli (University of Edinburgh)
Supervisor	Dr. Olivier Stasse (LAAS-CNRS)

September 18, 2020

UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

Deutsches Forschungszentrum für Künstliche Intelligenz GmbH
Robotics Innovation Center

Declaration

This study was carried out at the Robotics Innovation Center of the German Research Center for Artificial Intelligence in the Advanced AI Team on Mechanics & Control.

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

Bremen, September 18, 2020

Julian Eßer

Abstract

In order to further close the gap between robots and their natural counterparts, current research is driving towards exploiting the natural dynamics of robots. This requires a rethinking about the way we control robots for moving in a more dynamic, efficient and natural way.

Dynamic bipedal locomotion is a challenging problem and remains an open area of research. A key characteristic is the decoupling between the center of mass and the multi-body dynamics. Particular difficulties arise from effective underactuation, the mechanism complexity, as well as nonlinear and hybrid dynamics.

A common approach is to decompose this problem into smaller sub-problems that are solved sequentially. Many state-of-the-art frameworks rely on trajectory optimization based on reduced centroidal dynamics, which is transferred to a whole-body trajectory via feedback linearization using Inverse Kinematics (IK) or Inverse Dynamics (ID). Recent research indicates that solving this mapping with a local optimal control solver, namely Differential Dynamic Programming (DDP), instead of IK/ID, produces more efficient motions, with lower forces and impacts.

This master thesis contributes to the research field of dynamic bipedal locomotion by applying, evaluating and extending DDP-based whole-body trajectory optimization, pursuing three objectives: First, we develop an approach to constrain the optimal control problem allowing DDP to produce inherently balanced motions. Second, we evaluate our approach for quasi-static and dynamic motions in a real-time physics simulation and in real-world experiments on the lightweight and biologically inspired RH5 humanoid robot. Third, we examine the limits of the derived whole-body planning approach and the system design via solving highly-dynamic movements.

Keywords: Humanoid Robots, Dynamic Bipedal Walking, Motion Planning, Multi-Contact Optimal Control, Differential Dynamic Programming, Whole-Body Trajectory Optimization

Contents

Acronyms

CoM Center of Mass

CoP Center of Pressure

DDP Differential Dynamic Programming

DoF Degrees of Freedom

DS Double Support

EoM Equations of Motion

FCoM Floor Projection of Center of Mass

FD Forward Dynamics

FDDP Feasibility-driven Differential Dynamic Programming

ID Inverse Dynamics

IK Inverse Kinematics

KKT Karush-Kuhn-Tucker

LF left foot

LIP Linear Inverted Pendulum

MPC Model Predictive Control

OC Optimal Control

RF right foot

SP Support Polygon

SQP Sequential Quadratic Programming

TO Trajectory Optimization

ZMP Zero-Moment Point

CHAPTER 1

Introduction

This introduction guides the reader to the goal of the master's thesis. Beginning with a motivation on humanoid robots a general problem statement is derived. Thereupon, two approaches are presented for solving the motion planning problem as well as the used framework and experimental platform. Building upon this related work, the specific objectives of the thesis are defined and a brief overview of the structure is provided.

1.1 Motivation

Robotics research is highly motivated by the idea of creating machines with the ability to autonomously explore and interact with complex and dynamic environments. These intelligent agents can act in surroundings that are either inaccessible or dangerous to humans or support us in everyday life tasks.

The key promise of using legs for locomotion is the improved mobility over wheeled systems. Significant advantages are gained due to the ability of using isolated footholds and active suspension, which effectively decouples the main body from the roughness of the environment and allows e.g. to step over obstacles [**?**]. These benefits come at the expense of a significant increase in complexity, since there is a need of ongoing, active balancing of the robot in order to avoid falling down [**?**].

Nature often plays a crucial rule and serves as source of inspiration in the design process of such systems. This is especially true for the research field of humanoid robots, which deals with robots that are generally inspired by human capabilities and share similar kinematics, sensing and behavior. Many of the objects that we interact with on a daily basis, are tailored to human form and human behavior, e.g. doors, stairs or tools. This is equally true for the environments that we move in. Humans make use of their legs to climb stairs, lean towards difficult postures

or traverse rough terrain [**?**]. These capabilities of humanoid robots have the potential to benefit mankind. Walking robot nurses would have the ability to freely move around and interact with the elderly. In disaster scenarios, humanoids could be sent to check for an rescue persons. When exploiting foreign planets, humanoid robots have the ability to collaborate with humans in an intuitive and effective way (see Fig. 1.1).

Figure 1.1: Humanoid robots can interact with humans in a very intuitive manner since they share similar kinematics, sensing and behavior. This opens up new opportunities for the cooperation needed, e.g. when assembling and installing infrastructure on foreign planets [**?**].

We perform all these tasks seemingly effortless. But compared to current robots, the dynamic capabilities of humans and animals are still outstanding in terms of versatility, speed, efficiency and robustness [**?**]. In order to further close the gap between robots and their natural counterparts, current research is driving towards exploiting the natural dynamics of robots. This implies mutually dependent changes about how we think about the robots design [**?**] on the one hand and about ways of controlling them on the other hand, in order to move in a more dynamic, efficient and natural way [**?** **?** **?**].

The specific difficulty of creating bipedal walking motions has several causes. The first difficulty is due to the mechanism complexity, i.e. dealing with high-dimensional degrees of freedom, leading to potentially expensive computations. Secondly, legged locomotion is subject to different contact and impact collisions, resulting in multi-phase models and hence hybrid dynamics. Finally, bipeds face the problem of effective underactuation, further restricting the applicable control approaches.

Dynamic bipedal locomotion adds additional complexity to this problem. A central characteristic of walking dynamically is that the center of mass (COM) partially leaves the biped's support polygon. Furthermore there is the need of generating feasible, controllable limit cycles. When considering running motions, difficulties are faced regarding the conservation of angular momentum, which implies restrictions on the controllability during flight phases [**?**].

1.2 Related Work

There are existing numerous ways to generate feasible motion plans for robotic systems. Common approaches are relying on simplified dynamic models, while recent ones make use of numerical optimization for generating efficient motion plans.

1.2.1 Traditional Legged Locomotion Planning

Previously we have already seen, why locomotion synthesis for legged robots is a challenging problem. Solving this global problem typically is approached by splitting it up into successive subproblems that are solved sequentially: (i) Contact planner, (ii) Centroidal pattern generator, (iii) Whole-body motion generator (Fig. 1.2) [**?**]. The first stage of traditional legged locomotion planning is a contact planner

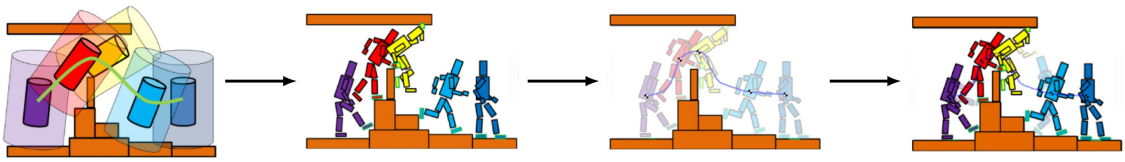


Figure 1.2: Tradional Legged Locomotion Planning [**?**].

that selects appropriate foothold position and step timings. In other words, this component predefines *where* and *when* external forces can act on the system. Once these force locations have been predetermined, the goal is to generate a body motion, i.e. a Center of Mass (CoM) and end-effector trajectories, which can be created with the available forces. A common approach is to model the robot as a Linear Inverted Pendulum (LIP) and find a motion where the Center of Pressure (CoP) remains inside the Support Polygon (SP). The LIP can be solved analytically and efficiently and consequently has been used in a variety of approaches [**?** **?** **?** **?**]. From the contact sequence and the centroidal trajectory, a dynamic-physical whole-body trajectory has to be found. This often is done by solving the Inverse Kinematics (IK) [**?**] or operational-space Inverse Dynamics (ID) [**?**] of the system and produces appropriate joint positions or torques, respectively that can be applied to a physical system.

This approach is beneficial in terms of computation time but comes at the cost of limited motion complexity and energy efficiency. The first limitation comes from the underlying assumptions of the LIP model. More complex motions and terrains might require to place feet at different heights, reorientation of the base or jump vertically [**?**]. Another limitation arises from the fact whole-body planning produces more efficient motions than a simple program such as an IK solver [**?**].

1.2.2 Trajectory Optimization

Generating motion plans for dynamic systems is subject to Trajectory Optimization (TO) algorithms that belong to the broader research field of Optimal Control (OC).

TO is a numerical optimization technique with the goal of finding a state-control sequence, which locally minimizes a predefined cost function given a set of constraints. The approach allows to specify the behavior of a robot directly in the task space, e.g. a desired end-effector trajectory, while reducing the amount of hand-crafted components.

There are existing different methods to formulate a TO, namely *direct* and *indirect* methods. Unlike *direct* methods which explicitly represent the state, *indirect* methods only represent the controls while the state is obtained from forward simulation of the system. In direct methods, the OC problem is transcribed into a Sequential Quadratic Programming (SQP), which easily handles both equality and inequality constraints and is implemented in generic off-the-shelf solvers. Consequently, the direct approaches are forced to search in a constrained optimization space which is slower but finds better optima. The indirect approaches are more sensitive to local minima, but are faster and better suited for warm-starting. For a comprehensive overview and further information on TO methods see [**?** **?** **?**].

TO based on reduced centroidal dynamics [**?**] has become a popular approach in the legged robotics community. In some approaches it is used after planning the contacts [**?** **?** **?**] while other approaches simultaneously optimize the centroidal trajectory and the contacts [**?** **?** **?**]. Either way, the transfer from centroidal to whole-body dynamics is only achieved by instantaneous feedback linearization where typically quadratic programs with task-space dynamics are solved [**?** **?** **?**].

While TO based on reduced dynamics models has shown great experimental results, whole-body TO instead is proven to produce more efficient motions, with lower forces and impacts [**?**]. Hence we will focus on *indirect methods*, namely Differential Dynamic Programming (DDP) to compute the whole-body motion. DDP allows to efficiently solve nonlinear OC problems due to its intrinsic sparse structure and is introduced in Sections 2.3 to 2.4 in more detail.

1.2.3 Crocoddyl Framework

Crocoddyl (Contact RObot COntrol by Differential DY1namic Library) is a recently presented open-source framework for efficient multi-contact optimal control [**?**], which is used within this thesis to plan whole-body motions.

The framework allows to efficiently compute optimal robot trajectories with predefined contact phases. Its solver is based on various efficient DDP-like algorithms. Along with the Crocoddyl, a novel optimal control algorithm called Feasibility-driven Differential Dynamic Programming (FDDP) is introduced. The FDDP algorithm is an improved version of the classical DDP algorithm, which shows a greater globalization strategy due to two modifications. First, the backward pass also allows for infeasible state and control trajectories. Second, forward pass keeps the gaps open within the first iterations. The framework can be used in one of two ways: Either offline, where a stabilizing controller is build around the nominal trajectory [**?**] or in a Model Predictive Control (MPC) sense, where the optimal trajectory is (re-)computed online.

Crocoddyl allows the computation of highly-dynamic movements (e.g. jumping, front-flip) within few milliseconds. However, these exemplary case studies do not guarantee inherent balance of the motions, leading to trajectories that are hard to stabilize on a real system. To this end, we present a generic method for constraining DDP-like solvers in order to generate inherently balanced, dynamic motions that are applicable on real robots.

1.2.4 RH5 Humanoid Robot

The derived motion planning approach has been tested both in simulation and real-world experiments on a full-size humanoid robot. RH5 is a lightweight and biologically inspired humanoid that has recently been developed at DFKI Robotics Innovation Center [**?**].

The RH5 humanoid robot (see Fig. 1.3) is designed to mimic the human anatomy with a total size of 200cm, a weight of 62kg and a total of 32 Degrees of Freedom (DoF). The two legs account for 12 DoF, the torso and neck kinematics each for three and the arms and grippers of the robot for 14 DoF. In order to achieve a high dynamic performance, the robot's design follows a series-parallel hybrid approach. Consequently,linkages and parallel mechanisms are utilized in most of the robots joints, e.g. the hip-flexion-extension, knee, ankle, torso and wrist. A comparison of RH5 with other state of the art humanoid robots revealed several advantages of this design approach, including better maximum velocity and torque of the ankle as well as an advantageous weight of the lower leg [**?**]. The interested reader can find a comprehensive introduction on series-parallel hybrid robots in [**?** , Ch.2].

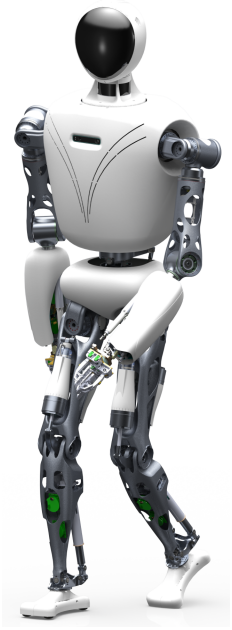


Figure 1.3: The recently presented RH5 is a lightweight and biologically inspired humanoid used as experimental platform within this thesis.

1.3 Contributions

The overall goal of this master's thesis is to contribute to the research field of humanoid robotics by applying, evaluating and extending recently presented whole-body TO approaches on the RH5 humanoid robot.

The proposed motion planning approach (see Fig. 1.4) consists of a DDP-based whole-body TO. Beneath the contact position and timings, it also considers a set of contact stability constraints that allow computing inherently balanced motions. In contrast to many other motion planning concepts followed by the legged robotics community, our approach (i) considers the full robot dynamics instead of using a reduced dynamics model such as the LIP model (ii) simultaneously optimizes for the centroidal motion, contact stability and whole-body trajectory instead of a sequential optimization. The specific contributions of this thesis are summarized below.

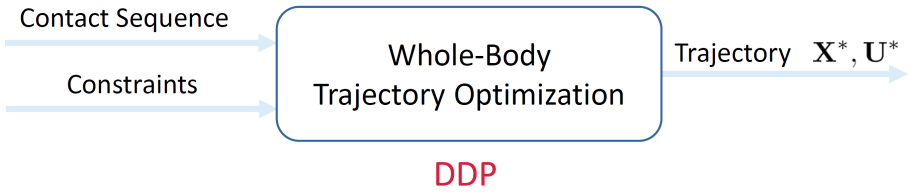


Figure 1.4: The motion planning approach proposed within this thesis. Based on predefined foothold positions, step timings and stability constraints a DDP-based whole-body TO is used to generate inherently balanced motions plans.

C1. A generic method for constraining DDP-like solvers in order to generate inherently balanced dynamic motions. The results are integrated into the open-source framework Crocoddyl.

C2. Evaluation of the stability of the proposed motion planning approach for bipedal walking gaits of increasing complexity in simulation.

C3. Identification of range of motions for the RH5 humanoid by performing highly-dynamic movements as basis for future design iterations.

C4. An experimental pipeline for executing optimization-based whole-body motions on a series-parallel hybrid robot.

1.4 Structure

This thesis is organized in a total of 8 chapters. Fig. XXX shows the overall outline of this thesis. In the following, a summary of each chapter is provided which allows the readers to easily navigate through this document.

Chapter 1 (Introduction) motivates the problem and presents the related work and specific contributions. It also provides details on the structure of the thesis.

Chapter 2 (Mathematical Background) provides the reader with fundamental background in bipedal locomotion and stability analysis. Furthermore, it introduces the class of algorithms and relevant extensions used in this work.

Chapter 3 (Contact Stability Constrained DDP) presents a generic method for integrating stability constraints into DDP-like solvers in order to generate inherently balanced dynamic motions.

Chapter 4 (Bipedal Walking Variants) studies the proposed motion planning approach for bipedal walking gaits of increasing complexity in simulation.

Chapter 5 (Highly-Dynamic Movements) presents an analysis of highly-dynamic movements based on the proposed motion planning approach with the goal of identifying the system limits of the RH5 humanoid robot.

Chapter 6 (Validation in Real-Time Physics Simulation) presents a validation of the generated motions by application of a simple control architecture in an real-time physics simulation environment.

Chapter 7 (Validation in Real-World Experiments) presents an experimental pipeline for validating the motions on a full-size humanoid robot with series-parallel hybrid mechanisms.

Chapter 8 (Conclusion and Outlook) presents the summary of the thesis and identifies future research directions.

CHAPTER 2

Mathematical Background: Optimal Bipedal Locomotion

The second chapter provides the reader with fundamentals regarding terminology, modeling and stability analysis in the context of humanoid robotics, presents the class of used algorithms and relevant extensions and introduces the RH5 humanoid robot serving as experimental platform for this thesis.

2.1 Foundations of Bipedal Locomotion

2.1.1 Terminology

In order to describe the locomotion of a humanoid robot, specific terms are required that are introduced within this section. **?** provide an extensive introduction to the terminology related to bipedal walking [**?**], concisely summarized by **?** [**?**].

Walk
Walk can be defined as: "*Movement by putting forward each foot in turn, not having both feet off the ground at once*".

Run
Run in turn is characterized by a movement where partially both feet leaving the ground at the same time.

Gait
The way each human walks and runs is unique, hence gait can be defined as: "*Manner of walking or running*".

Periodic gait
If a gait is realized by repeating each locomotion phase in an identical [1] way, the gait is referred to as *periodic*.

Symmetric gait
If the left and right leg move in an identical but time-shifted manner, the gait is referred to as *symmetric*.

Double Support
A situation where the humanoid has two isolated contact surfaces with the ground.

Single Support
A situation where the humanoid has only one contact surface with the ground.

Support Polygon
The support polygon is formed by the *convex hull* about the ground contact points.

Swing foot
This term refers to the leg that is performing a step, i.e. moving through the air.

Supporting foot
This term refers to the leg that is in contact with the ground, supporting all the weight of the humanoid.

2.1.2 Dynamic Modeling of Legged Robots

In the following, the dynamic model for floating base systems, such as legged robots, is derived based on a general formulation. A concise introduction to dynamic modeling is presented with [**?**], comprehensive studies can be found in [**?** **?** **?**].

General Formulation

Mathematical models of a robot's dynamics describe the motion as a function of time and control inputs. These models are the basis for both simulation and control of robotic systems. In an abstract form, the Equations of Motion (EoM) can be written as:

$$F(\boldsymbol{q}(t), \dot{\boldsymbol{q}}(t), \ddot{\boldsymbol{q}}(t), \boldsymbol{u}(t), t) = 0, \tag{2.1}$$

where

- **t** is the time variable,

[1] The locomotion phase can be identical w.r.t. the step size or the duration, depending on the index.

- $\mathbf{q}$ is the vector of generalized coordinates,
- $\dot{\boldsymbol{q}}$ is the first time derivative (velocity) of $\mathbf{q}$,
- $\ddot{\boldsymbol{q}}$ is the second time derivative (acceleration) of $\mathbf{q}$ and
- $\mathbf{u}$ is the vector of control inputs.

Consequently, the EoM provide a mapping between the control space on the one hand and the state space of robot on the other hand. Typical methods for computing the closed-form solution of the EoM are e.g. the classical *Newton-Euler* [**?**] method or the *Lagrange method* [**?**], where the former is based on principles for conservation of linear and angular momenta and the latter utilizes energy-based functions expressed in generalized coordinates.

Fixed Base Systems

For applications with fixed-based robots, e.g. a robotic manipulator, the multi-body dynamics can be formulated as

$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \dot{\boldsymbol{q}}^T \boldsymbol{C}(\boldsymbol{q})\dot{\boldsymbol{q}} = \boldsymbol{\tau} + \boldsymbol{\tau}_g(\boldsymbol{q}), \tag{2.2}$$

where

- $\boldsymbol{M}(\boldsymbol{q})$ is the generalized inertia matrix,
- $\boldsymbol{C}(\boldsymbol{q})$ is the coriolis tensor,
- $\boldsymbol{\tau}$ is the vector of actuated joint torques and
- $\boldsymbol{\tau}_g(\boldsymbol{q})$ is the vector of external joint torques caused by gravity.

In contrast to the general formulation in Eq. (2.1), this expression is time-invariant. Hence, Eq. (2.2) can be used for computing the Forward Dynamics (FD), as well as the ID of a robotic system.

Floating Base Systems

A floating base system is characterized by having a base that is free to move, rather than being fixed in space. Consequently, the vector of generalized coordinates $\boldsymbol{q}$ not only contains the joints angles, but also accounts for the position and orientation of the floating base. Legged robots belong to this category of rigid-body systems as they make and break contacts with their environment in order to move. Contrary to manipulators, contacts need to be actively enforced by holonomic constraints for legged robots. There are namely two different types of contact constraints that can be applied: point contacts (3d) or surface contacts (6d).

For the case of point contacts, the dynamics of the floating base system become

$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \dot{\boldsymbol{q}}^T \boldsymbol{C}(\boldsymbol{q})\dot{\boldsymbol{q}} = \boldsymbol{S}^T\boldsymbol{\tau} + \boldsymbol{\tau}_g(\boldsymbol{q}) + \sum_{i=1}^{k} \boldsymbol{J}_{C_i}^T \boldsymbol{f}_i,$$

where

- **S** is the selection matrix of actuated joints,
- $\boldsymbol{J}_{C_i}$ is the Jacobian at the location of a contact point C_i and
- $\boldsymbol{f}_i$ is the contact force acting at the contact point C_i.

For the case of surface contacts, such as a flat foot on a flat floor, modeling a point contact is not sufficient since it only constrains the translation. In order to also account for the rotational constraints enforced by the geometry one could take into account multiple point contacts. A non-redundant alternative is to model more general frame contact constraints as

$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \dot{\boldsymbol{q}}^T \boldsymbol{C}(\boldsymbol{q})\dot{\boldsymbol{q}} = \boldsymbol{S}^T\boldsymbol{\tau} + \boldsymbol{\tau}_g(\boldsymbol{q}) + \sum_{i=1}^{k} \boldsymbol{J}_{C_i}^T \boldsymbol{w}_i, \tag{2.3}$$

where $\boldsymbol{w}_i$ is referred to as the *contact wrench* acting on the contact link i. This wrench stacks the resultant $\boldsymbol{f}_i$ of contact forces and the moment $\boldsymbol{\tau}_i$ exerted by these forces around the contact frame as

$$\boldsymbol{w}_i = (\boldsymbol{f}_i, \boldsymbol{\tau}_i)_{6\times 1}.$$

For more details on contact wrenches and spatial vector algebra in general, the interested reader is referred to e.g. [**?** , Ch.2].

2.2 Stability Analysis: Not Falling Down

Humanoid robots are high-dimensional, constrained and nonlinear dynamical systems. In this section, the most common criteria for analyzing the long-term stability behavior of such complex systems are presented. Exhaustive studies on stability criteria and their relation can be found in [**? ? ?**].

2.2.1 Static Stability Criteria

Floor Projection of the Center of Mass (FCoM)

Consider the case of a robot that is not moving, i.e. a humanoid in static double support. In that case, the only forces acting on the humanoid are the ones caused

by gravity. These forces can be represented by a virtual force acting on the CoM of the robot. The position of the CoM w.r.t. the base frame can be described by

$$\boldsymbol{p}_{CoM} = \frac{\sum_{i=1}^{n} m_i \boldsymbol{p}_i}{\sum_{i=1}^{n} m_i},$$

where the robot has n links and $\boldsymbol{p}_i$ indicate the according link distances of the individual CoMs. The Floor Projection of Center of Mass (FCoM) equals the first two components of the CoM position vector $\boldsymbol{p}_{CoM}$ and the following relation holds:

$$\sum_{i=1}^{n}((\boldsymbol{p}_{FCoM} - \boldsymbol{p}_i) \times m_i \boldsymbol{g}) = \boldsymbol{0}.$$

The FCoM can be used as a static stability margin, ensuring the motionless robot will not tip over or fall, if $\boldsymbol{p}_{FCoM}$ always remains inside the SP. Note that this criteria is also applicable in so called *quasi-static* movements, where static forces are still dominating dynamic forces.

2.2.2 Dynamic Stability Criteria

In case of faster motions, dynamic forces will exceed the static forces and can not be neglected anymore. The acting forces can be divided into contact forces and gravity/inertial forces, where the so called Zero-Moment Point (ZMP) is based on the former, and the CoP on the latter. In the following, both concepts are introduced according to the description in [**?**] with a nomenclature equivalent to [**?**].

Center of Pressure (CoP)

The CoP is defined as the point, where the field of pressure forces acting on the sole is equivalent to a single resultant force where the resultant moment is zero. Hence the CoP is a local quantity that is derived from the interaction forces at the contact surface.

Considering the case of a foot contacting a plane surface, the resultant contact force $\boldsymbol{f}^c$ is exerted by the environment onto the robot. This force consists of the resultant pressure force $\boldsymbol{f}^p = (\boldsymbol{f}^c \cdot \boldsymbol{n})\boldsymbol{n}$, as well as the resultant friction force $\boldsymbol{f}^f = \boldsymbol{f}^c - \boldsymbol{f}^p$. Hence, the following conditions hold:

$$\begin{aligned} \boldsymbol{\tau}_O^p &= \boldsymbol{0} \\ \boldsymbol{p}_{CoP} \times (\boldsymbol{f}^p \cdot \boldsymbol{n})\boldsymbol{n} &= -\boldsymbol{\tau}_O^P \\ (\boldsymbol{f}^p \cdot \boldsymbol{n})\boldsymbol{n} \times \boldsymbol{p}_{CoP} \times \boldsymbol{n} &= -\boldsymbol{n} \times \boldsymbol{\tau}_O^p \end{aligned}$$

Since both the sole point O and $\boldsymbol{p}_{CoP}$ belong to the same plane, we get:

$$\boldsymbol{p}_{CoP} = \frac{\boldsymbol{n} \times \boldsymbol{\tau}_O^p}{\boldsymbol{f}^p \cdot \boldsymbol{n}}.$$

Finally, friction forces are tangent to the contact surface and their moment is aligned with $\boldsymbol{n}$, so we equivalently can write this relationship as:

$$\boldsymbol{p}_{CoP} = \frac{\boldsymbol{n} \times \boldsymbol{\tau}_O^c}{\boldsymbol{f}^c \cdot \boldsymbol{n}}. \tag{2.4}$$

Equation (2.4) can be used to compute the CoP expressed in the local contact frame.

Zero-Moment Point (ZMP)

The ZMP is defined as a point on the ground where the *tipping moment* acting on the biped equals zero. This condition can be interpreted as a constraint on the contact moments, which contains *at least* the roll and pitch direction. Originally, the concept has been introduced in [**?**], it has been reviewed in [**?**] and made popular with [**?**].

The concept is build upon two key assumptions:

- There exists one planar contact surface (i.e. no multiple surfaces like on rough terrain)
- The friction is sufficiently high to prevent sliding of the feet

From the Newton-Euler equations, the motion of the biped can be written as

$$\begin{aligned} m\ddot{\boldsymbol{p}}_{CoM} &= m\boldsymbol{g} + \boldsymbol{f}^c \\ \dot{\boldsymbol{L}}_O &= \boldsymbol{p}_{CoM} \times m\boldsymbol{g} + \boldsymbol{\tau}_{CoM}^c, \end{aligned}$$

where m denotes the total mass of the robot, $\boldsymbol{g}$ is the gravity vector, $\ddot{\boldsymbol{p}}_{CoM}$ the centroidal acceleration, $\dot{\boldsymbol{L}}_O$ the change of the angular momentum. $\boldsymbol{w}_{CoM}^c = (\boldsymbol{\tau}_{CoM}^c, \boldsymbol{f}^c)_{6\times 1}$ denotes the sum of all contact wrenches in the CoM frame. The gravito-inertial wrench of the robot can be defined as

$$\begin{aligned} \boldsymbol{f}^{gi} &= m(\boldsymbol{g} - m\ddot{\boldsymbol{p}}_{CoM}) \\ \boldsymbol{\tau}_O^{gi} &= \boldsymbol{p}_{CoM} \times m\boldsymbol{g} - \dot{\boldsymbol{L}}_O. \end{aligned}$$

Using the wrench form of the Newton-Euler equations

$$\boldsymbol{w}^{gi} + \boldsymbol{w}^c = \boldsymbol{0}, \tag{2.5}$$

one can derive the ZMP, for the case of a planar surface, as

$$\boldsymbol{p}_{ZMP} = \frac{\boldsymbol{n} \times \boldsymbol{\tau}_O^{gi}}{\boldsymbol{f}^{gi} \cdot \boldsymbol{n}}. \tag{2.6}$$

In practice, one can use this formula to compute the ZMP from force sensors or from an inertial measurement unit.

Coincidence of ZMP and CoP

As **?** outline, both the ZMP and the CoP yield the same point for the case of bipedal walking on a single plane surface. Comparing Eq. (2.6) with Eq. (2.4), we recognize the only difference is that the former is applied to the (global) gravito-inertial wrench, while the latter is applied to the (local) contact wrench. If we recall the Netwon-Euler equations from Eq. (2.5), it becomes clear why both points coincide when there is only one contact plane.

2.2.3 Stability Classification

There are existing several classifications on stability, which will be defined in the following according to [**?** , Sec.1.2.1] and [**?**]. See **?** for more details on differentiating the terms dynamic stability and dynamic balance [**?**].

Statically Stable Motion

The gait or movement of a humanoid is classified as *statically stable*, if the FCoM does not leave the SP during the entire motion or gait. Consequently, the humanoid will remain in a stable position, whenever the movement is stopped. Typically, these kind of stability are only obtained with very low walking velocities or quasi-static motions, where the static forces dominate the dynamic forces.

Dynamically Stable Motion

If the FCoM partially leaves the SP at some point during the gait, but the CoP (or ZMP) always remains within the SP, the gait or movement is classified as *dynamically stable*. This stability margin is extremely useful for flat-foot dynamic walking since it prevents the foot from rotating around the boundary of the SP.

2.3 Differential Dynamic Programming (DDP)

This section describes the basics of DDP, which is an OC algorithm that belongs to the TO class. The algorithm was introduced in 1966 by **?** [**?**]. A modern description of the algorithm using the same notations as below can be found in [**?** **?**].

2.3.1 Finite Horizon Optimal Control

We consider a system with discrete-time dynamics, which can be modeled as a generic function $\boldsymbol{f}$

$$\boldsymbol{x}_{i+1} = \boldsymbol{f}(\boldsymbol{x}_i, \boldsymbol{u}_i), \tag{2.7}$$

that describes the evolution of the state $\boldsymbol{x} \in \boldsymbol{R}^n$ from time i to $i+1$, given the control $\boldsymbol{u} \in \boldsymbol{R}^m$. A complete trajectory $\{\boldsymbol{X}, \boldsymbol{U}\}$ is a sequence of states $\boldsymbol{X} = \{\boldsymbol{x}_0, \boldsymbol{x}_1, ..., \boldsymbol{x}_N\}$ and control inputs $\boldsymbol{U} = \{\boldsymbol{u}_0, \boldsymbol{u}_1, ..., \boldsymbol{u}_N\}$ satisfying Eq. (2.7). The *total cost* J of a

trajectory can be written as the sum of running costs l and a final cost l_f starting from the initial state $\boldsymbol{x_0}$ and applying the control sequence $\boldsymbol{U}$ along the finite time-horizon:

$$J(\boldsymbol{x}_0, \boldsymbol{U}) = l_f(\boldsymbol{x}_N) + \sum_{i=0}^{N-1} l(\boldsymbol{x}_i, \boldsymbol{u}_i). \tag{2.8}$$

As dicussed in Chapter 1, *indirect* methods such DDP represent the trajectory implicitly solely via the optimal controls $\boldsymbol{U}$. The states $\boldsymbol{X}$ are obtained from forward simulation of the system dynamics, i.e. integration Eq. (2.7). Consequently, the solution of the optimal control problem is the minimizing control sequence

$$\boldsymbol{U}^* = \underset{U}{\operatorname{argmin}}\, J(\boldsymbol{x}_0, \boldsymbol{U}).$$

2.3.2 Local Dynamic Programming

Let $\boldsymbol{U}_i \equiv \{\boldsymbol{u}_i, \boldsymbol{u}_{i+1}..., \boldsymbol{u}_{N-1}\}$ be the partial control sequence, the *cost-to-go* J_i is the partial sum of costs from i to N:

$$J_i(\boldsymbol{x}, \boldsymbol{U}_i) = l_f(\boldsymbol{x}_N) + \sum_{j=i}^{N-1} l(\boldsymbol{x}_j, \boldsymbol{u}_j). \tag{2.9}$$

The *Value function* at time i is the optimal cost-to-go starting at $\boldsymbol{x}$ given the minimizing control sequence

$$V_i(\boldsymbol{x}) = \min_{\boldsymbol{U}_i} J_i(\boldsymbol{x}, \boldsymbol{U}_i),$$

and the Value at the final time is defined as $V_N(\boldsymbol{x}) \equiv l_f(\boldsymbol{x}_N)$. The Dynamic Programming Principle [**?**] reduces the minimization over an entire sequence of controls to a sequence of minimizations over a single control, proceeding backwards in time:

$$V(\boldsymbol{x}) = \min_{\boldsymbol{u}}[l(\boldsymbol{x}, \boldsymbol{u}) + V'(\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}))]. \tag{2.10}$$

Note that Eq. (2.10) is referred to as the *Bellman equation* for *discrete-time* optimization problems [**?**]. For reasons of readability, the time index i is omitted and V' introduced to denote the Value at the next time step. The interested reader may note that the analogous equation for the case of *continuous-time* is a partial differential equation called the *Hamilton-Jacobi-Bellman equation* [**? ?**].

2.3.3 Quadratic Approximation

DDP locally computes the optimal state and control sequences of the OC problem derived with Eq. (2.10) by iteratively performing a forward and backward pass. The *backward pass* on the trajectory generates a new control sequence and is followed by a *forward pass* to compute and evaluate the new trajectory.

Let $\boldsymbol{Q}(\delta\boldsymbol{x}, \delta\boldsymbol{u})$ be the variation in the argument on the right-hand side of Eq. (2.10) around the $i^{th}(\boldsymbol{x}, \boldsymbol{u})$ pair

$$\boldsymbol{Q}(\delta\boldsymbol{x}, \delta\boldsymbol{u}) = l(\boldsymbol{x} + \delta\boldsymbol{x}, \boldsymbol{u} + \delta\boldsymbol{u}) + V'(\boldsymbol{f}(\boldsymbol{x} + \delta\boldsymbol{x}, \boldsymbol{u} + \delta\boldsymbol{u})). \tag{2.11}$$

The DDP algorithm uses a quadratic approximation of this differential change. The quadratic Taylor expansion of $Q(\delta\boldsymbol{x}, \delta\boldsymbol{u})$ leads to

$$\boldsymbol{Q}(\delta\boldsymbol{x}, \delta\boldsymbol{u}) \approx \frac{1}{2} \begin{bmatrix} 1 \\ \delta\boldsymbol{x} \\ \delta\boldsymbol{u} \end{bmatrix}^T \begin{bmatrix} 0 & \boldsymbol{Q}_{\boldsymbol{x}}^T & \boldsymbol{Q}_{\boldsymbol{u}}^T \\ \boldsymbol{Q}_{\boldsymbol{x}} & \boldsymbol{Q}_{\boldsymbol{xx}} & \boldsymbol{Q}_{\boldsymbol{xu}} \\ \boldsymbol{Q}_{\boldsymbol{u}} & \boldsymbol{Q}_{\boldsymbol{ux}} & \boldsymbol{Q}_{\boldsymbol{uu}} \end{bmatrix} \begin{bmatrix} 1 \\ \delta\boldsymbol{x} \\ \delta\boldsymbol{u} \end{bmatrix}. \tag{2.12}$$

The coefficients can be computed as

$$\boldsymbol{Q}_{\boldsymbol{x}} = l_{\boldsymbol{x}} + \boldsymbol{f}_{\boldsymbol{x}}^T \boldsymbol{V}'_{\boldsymbol{x}}, \tag{2.13a}$$
$$\boldsymbol{Q}_{\boldsymbol{u}} = l_{\boldsymbol{u}} + \boldsymbol{f}_{\boldsymbol{u}}^T \boldsymbol{V}'_{\boldsymbol{x}}, \tag{2.13b}$$
$$\boldsymbol{Q}_{\boldsymbol{xx}} = l_{\boldsymbol{xx}} + \boldsymbol{f}_{\boldsymbol{x}}^T \boldsymbol{V}'_{\boldsymbol{xx}} \boldsymbol{f}_{\boldsymbol{x}} + \boldsymbol{V}'_{\boldsymbol{x}} \cdot \boldsymbol{f}_{\boldsymbol{xx}}, \tag{2.13c}$$
$$\boldsymbol{Q}_{\boldsymbol{ux}} = l_{\boldsymbol{ux}} + \boldsymbol{f}_{\boldsymbol{u}}^T \boldsymbol{V}'_{\boldsymbol{xx}} \boldsymbol{f}_{\boldsymbol{x}} + \boldsymbol{V}'_{\boldsymbol{x}} \cdot \boldsymbol{f}_{\boldsymbol{ux}}, \tag{2.13d}$$
$$\boldsymbol{Q}_{\boldsymbol{uu}} = l_{\boldsymbol{uu}} + \boldsymbol{f}_{\boldsymbol{u}}^T \boldsymbol{V}'_{\boldsymbol{xx}} \boldsymbol{f}_{\boldsymbol{u}} + \boldsymbol{V}'_{\boldsymbol{x}} \cdot \boldsymbol{f}_{\boldsymbol{uu}}. \tag{2.13e}$$

where the primes denote the values at the next time-step.

2.3.4 Backward Pass

The first algorithmic step of DDP, namely the backward pass, involves computing a new control sequence on the given trajectory and consequently determining the search direction of a a step in the numerical optimization. To this end, the quadratic approximation obtained from Eq. (2.12), minimized with respect to $\delta\boldsymbol{u}$ for some state perturbation $\delta\boldsymbol{x}$, results in

$$\delta\boldsymbol{u}^*(\delta\boldsymbol{x}) = \underset{\delta\boldsymbol{u}}{\operatorname{argmin}}\, \boldsymbol{Q}(\delta\boldsymbol{x}, \delta\boldsymbol{u}) = -\boldsymbol{Q}_{\boldsymbol{uu}}^{-1}(\boldsymbol{Q}_{\boldsymbol{u}} + \boldsymbol{Q}_{\boldsymbol{ux}}\delta\boldsymbol{x}),$$

giving us an open-loop term $\boldsymbol{k}$ and a feedback gain term $\boldsymbol{K}$:

$$\boldsymbol{k} = -\boldsymbol{Q}_{\boldsymbol{uu}}^{-1}\boldsymbol{Q}_{\boldsymbol{u}} \quad \textit{and} \quad \boldsymbol{K} = -\boldsymbol{Q}_{\boldsymbol{uu}}^{-1}\boldsymbol{Q}_{\boldsymbol{ux}}.$$

The resulting locally-linear feedback policy can be again inserted into Eq. (2.12) leading to a quadratic model of the Value at time i:

$$\begin{aligned} \Delta\boldsymbol{V} &= -\frac{1}{2}\boldsymbol{k}^T\boldsymbol{Q}_{\boldsymbol{uu}}\boldsymbol{k} \\ \boldsymbol{V}_{\boldsymbol{x}} &= \boldsymbol{Q}_{\boldsymbol{x}} - \boldsymbol{K}^T\boldsymbol{Q}_{\boldsymbol{uu}}\boldsymbol{k} \\ \boldsymbol{V}_{\boldsymbol{xx}} &= \boldsymbol{Q}_{\boldsymbol{xx}} - \boldsymbol{K}^T\boldsymbol{Q}_{\boldsymbol{uu}}\boldsymbol{K}. \end{aligned}$$

2.3.5 Forward Pass

After computing the feedback policy in the backward pass, the forward pass computes a corresponding trajectory by integrating the dynamics via

$$\begin{aligned}\hat{\boldsymbol{x}}_0 &= \boldsymbol{x}_0 \\ \hat{\boldsymbol{u}}_i &= \boldsymbol{u}_i + \alpha \boldsymbol{k}_i + \boldsymbol{K}_i(\hat{\boldsymbol{x}}_i - \boldsymbol{x}_i) \\ \hat{\boldsymbol{x}}_{i+1} &= \boldsymbol{f}(\hat{\boldsymbol{x}}_i, \hat{\boldsymbol{u}}_i),\end{aligned}$$

where $\hat{\boldsymbol{x}}_i, \hat{\boldsymbol{u}}_i$ are the new state-control sequences. The step size of the numerical optimization is described by the backtracking line search parameter α, which iteratively is reduced starting from 1. The backward and forward passes of the DDP algorithm are iterated until convergence to the (locally) optimal trajectory.

2.4 Handling Constraints With DDP

By nature, the DDP algorithm presented in Section 2.3 does not take into account constraints. **?** developed a control-limited DDP [**?**] that takes into account box inequality constraints on the controls allowing the consideration of torque limits on real robotic systems. **?** proposed a DDP version for the problem of multi-phase rigid contact dynamics by exploiting the Karush-Kuhn-Tucker constraint of the rigid contact model [**?**]. Since physically consistent bipedal locomotion is highly dependent on making contacts with the ground, this section provides details on the above mentioned approach.

2.4.1 DDP With Constrained Robot Dynamics

Contact Dynamics

In the case of rigid contact dynamics, DDP assumes a set of given contacts of the system with the environment. Then, an equality constrained dynamics can be incorporated by formulating rigid contacts as holonomic constraints to the robot dynamics. In other words, the contact points are assumed to have a fixed position on the ground.

The unconstrained robot dynamics can be represented as

$$\boldsymbol{M}\dot{\boldsymbol{v}}_{free} = \boldsymbol{S}\boldsymbol{\tau} - \boldsymbol{b} = \boldsymbol{\tau}_b, \tag{2.14}$$

with the joint-space intertia matrix $\boldsymbol{M} \in \boldsymbol{R}^{n\times n}$ and the unconstrained acceleration vector $\dot{\boldsymbol{v}}_{free}$. The right-hand side of Eq. (2.14) represents the n-dimensional force-bias vector accounting for the control $\boldsymbol{\tau}$, the Coriolis and gravitational effects $\boldsymbol{b}$ and the selection matrix $\boldsymbol{S}$ of actuated joints.

In order to incorporate the rigid contact constraints to the robot dynamics, one can apply the Gauss principle of least constraint [**?**]. The idea is to minimize the deviation in acceleration between the constrained and unconstrained motion:

$$\begin{aligned} \dot{\boldsymbol{v}} = \arg\min_{\boldsymbol{a}} \quad & \frac{1}{2} \|\dot{\boldsymbol{v}} - \dot{\boldsymbol{v}}_{free}\|_{\boldsymbol{M}} \\ \text{subject to} \quad & \boldsymbol{J}_c \dot{\boldsymbol{v}} + \dot{\boldsymbol{J}}_c \boldsymbol{v} = \boldsymbol{0}, \end{aligned} \tag{2.15}$$

where $\boldsymbol{M}$ formally represents the inertia tensor over the configuration manifold $\boldsymbol{q}$. In order to express the holonomic contact constraint $\phi(\boldsymbol{q})$ in the acceleration space, it needs to be differentiated twice. Consequently, the contact condition can be seen as a second-order kinematic constraints on the contact surface position where $\boldsymbol{J}_c = \begin{bmatrix} \boldsymbol{J}_{c_1} & \cdots & \boldsymbol{J}_c \end{bmatrix}$ is a stack of f contact Jacobians.

Karush-Kuhn-Tucker (KKT) Conditions

The Gauss minimization in Eq. (2.15) corresponds to an equality-constrained quadratic optimization problem. The optimal solutions $(\dot{\boldsymbol{v}}, \boldsymbol{\lambda})$ must satisfy the so-called Karush-Kuhn-Tucker (KKT) conditions given by

$$\begin{bmatrix} \boldsymbol{M} & \boldsymbol{J}_c^\top \\ \boldsymbol{J}_c & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \dot{\boldsymbol{v}} \\ -\boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\tau}_b \\ -\dot{\boldsymbol{J}}_c \boldsymbol{v} \end{bmatrix}. \tag{2.16}$$

These dual variables $\boldsymbol{\lambda}^k$ represent external wrenches at the contact level. For a given robot state and applied torques, Eq. (2.16) allows a direct computation of the contact forces. To this end, the contact constraints can be solved analytically at the level of dynamics instead of introducing additional constraints in the whole-body optimization [**?**].

2.4.2 KKT-Based DDP Algorithm

The KKT dynamics from Eq. (2.16) can be expressed as a function of the state $\boldsymbol{x}_i$ and the control $\boldsymbol{u}_i$:

$$\begin{aligned} \boldsymbol{x}_{i+1} &= \boldsymbol{f}(\boldsymbol{x}_i, \boldsymbol{u}_i), \\ \boldsymbol{\lambda}_i &= \boldsymbol{g}(\boldsymbol{x}_i, \boldsymbol{u}_i), \end{aligned} \tag{2.17}$$

where the concatenation of the configuration vector and its tangent velocity forms the state $\boldsymbol{x} = (\boldsymbol{q}, \boldsymbol{v})$, $\boldsymbol{u}$ is the input torque vector and $\boldsymbol{g}(\cdot)$ is the optimal solution of Eq. (2.16).

Supposing a sequence of predefined contacts, the cost-to-go of the DDP backward-pass and its respective Hessians (compare Eq. (2.9) and 2.13) turn into:

$$J_i(\boldsymbol{x}, \boldsymbol{U}_i) = l_f(\boldsymbol{x}_N) + \sum_{j=i}^{N-1} l(\boldsymbol{x}_j, \boldsymbol{u}_j, \boldsymbol{\lambda}_j)$$

with the control inputs $\boldsymbol{U}_i$ acting on the system dynamics at time i, and first-order approximation of $\boldsymbol{g}(\cdot)$ and $\boldsymbol{f}(\cdot)$ as

$$\begin{aligned}
\boldsymbol{Q}_{\boldsymbol{x}} &= \boldsymbol{l}_{\boldsymbol{x}} + \boldsymbol{g}_{\boldsymbol{x}}^T \boldsymbol{l}_{\boldsymbol{\lambda}} + \boldsymbol{f}_{\boldsymbol{x}}^T \boldsymbol{V}'_{\boldsymbol{x}}, \\
\boldsymbol{Q}_{\boldsymbol{u}} &= \boldsymbol{l}_{\boldsymbol{u}} + \boldsymbol{g}_{\boldsymbol{u}}^T \boldsymbol{l}_{\boldsymbol{\lambda}} + \boldsymbol{f}_{\boldsymbol{u}}^T \boldsymbol{V}'_{\boldsymbol{x}}, \\
\boldsymbol{Q}_{\boldsymbol{xx}} &\approx \boldsymbol{l}_{\boldsymbol{xx}} + \boldsymbol{g}_{\boldsymbol{x}}^T \boldsymbol{l}_{\boldsymbol{\lambda\lambda}} \boldsymbol{g}_{\boldsymbol{x}} + \boldsymbol{f}_{\boldsymbol{x}}^T \boldsymbol{V}'_{\boldsymbol{xx}} \boldsymbol{f}_{\boldsymbol{x}}, \\
\boldsymbol{Q}_{\boldsymbol{ux}} &\approx \boldsymbol{l}_{\boldsymbol{ux}} + \boldsymbol{g}_{\boldsymbol{u}}^T \boldsymbol{l}_{\boldsymbol{\lambda\lambda}} \boldsymbol{g}_{\boldsymbol{x}} + \boldsymbol{f}_{\boldsymbol{u}}^T \boldsymbol{V}'_{\boldsymbol{xx}} \boldsymbol{f}_{\boldsymbol{x}}, \\
\boldsymbol{Q}_{\boldsymbol{uu}} &\approx \boldsymbol{l}_{\boldsymbol{uu}} + \boldsymbol{g}_{\boldsymbol{u}}^T \boldsymbol{l}_{\boldsymbol{\lambda\lambda}} \boldsymbol{g}_{\boldsymbol{u}} + \boldsymbol{f}_{\boldsymbol{u}}^T \boldsymbol{V}'_{\boldsymbol{xx}} \boldsymbol{f}_{\boldsymbol{u}}.
\end{aligned} \tag{2.18}$$

Consequently, the KKT-based DDP algorithm utilizes the set of Eq. (2.18) inside the backward-pass to incorporate the rigid contacts forces, while the updated system dynamics from Eq. (2.17) is utilized during the forward-pass of the algorithm.

2.4.3 Task-Related Constraints

An important part of the motion generation is the execution of desired actions, e.g. grasping an object, moving the CoM or performing a robot step. For formulating these task-related constraints, we follow the notation used in [**?**].

An arbitrary task can be formulated as a regulator:

$$\boldsymbol{h}_{task_k}(\boldsymbol{x}_k, \boldsymbol{u}_k) = \boldsymbol{s}_{task}^d - \boldsymbol{s}_{task}(\boldsymbol{x}_k, \boldsymbol{u}_k),$$

where the task is defined as the difference between the desired and current feature vectors $\boldsymbol{s}_{task}^d$ and $\boldsymbol{s}_{task}(\boldsymbol{x}_k, \boldsymbol{u}_k)$, respectively. The task at each node can be added to the cost function via penalization as:

$$l_k(\boldsymbol{x}_k, \boldsymbol{u}_k) = \sum_{j \in tasks} \boldsymbol{w}_{j_k} \mid\mid \boldsymbol{h}_{j_k}(\boldsymbol{x}_k, \boldsymbol{u}_k) \mid\mid^2,$$

where $\boldsymbol{w}_{j_k}$ assigned to task j at corresponding time k. The DDP algorithm utilized the derivatives of the regulators functions, namely computing the Jacobians and Hessians of the cost functions. In the scope of this thesis, the following tasks are handled

$$tasks \subseteq \{CoM, LF_{SE(3)}, RF_{SE(3)}\} : \tag{2.19}$$

1) the CoM tracking (CoM) and 2) the tracking of the left- and right-feet pose ($LF_{SE(3)}, RF_{SE(3)}$).

CHAPTER 3

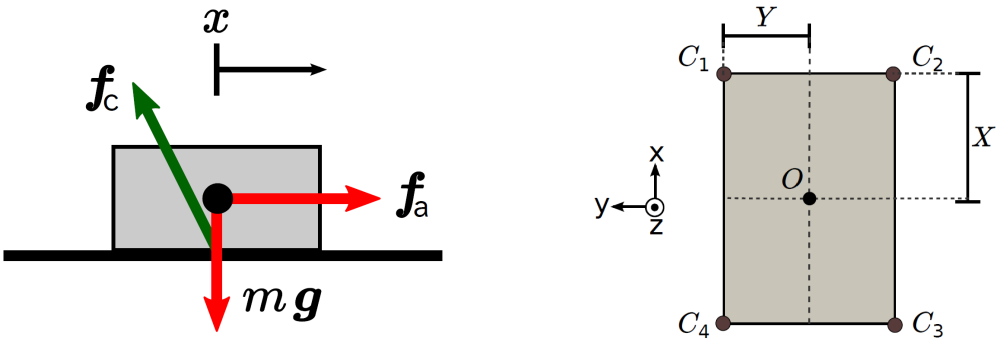
Contact Stability Constrained DDP

This chapter presents a generic method for integrating contact stability constraints into DDP-like solvers. The key idea is to define inequality constraints for unilaterality, friction and the CoP of each contact surface with the goal of generating inherently balanced motions.

3.1 The Idea

Stability of the contacts is an essential objective of motion planning since prevents the robot from sliding and falling down. In Section 2.2 we have explored two different criteria for ensuring contact stability for dynamic systems, namely the ZMP and CoP. As outlined, the application of the ZMP is limited due to the assumptions of sufficiently high friction and the existence of one planar contact surface. Since we want to provide a *generic* method that can also be used for e.g. walking up stairs, these simplifying assumptions do not hold anymore.

Consequently, we decide to model a 6d surface contact, as introduced in 2.3 with dedicated constraints for (i) unilaterality of the contact forces (ii) Coloumb friction on the resultant force, and (iii) CoP inside the support area. This approach can be compared to the concept of contact wrench cone [**?**], without additionally enforcing



a Body on a horizontal surface.

b Notations used for CoP definition.

Figure 3.1: Contact in the surface plane [**?**].

the yaw torque constraint. These inequality constraints for surface contacts can compactly be summarized as

$$f_i^z > 0, \tag{3.1a}$$
$$| f_i^x | \leq \mu f_i^z, \tag{3.1b}$$
$$| f_i^y | \leq \mu f_i^z, \tag{3.1c}$$
$$| X | \geq C_x, \tag{3.1d}$$
$$| Y | \geq C_y. \tag{3.1e}$$

Let us now detail each line of the approach. The first inequality Eq. (3.1a) accounts for the unilaterality of the contact force. By nature, contact forces always have to be positive since the robot can only *push* from the ground, not *pull* to the ground (Fig. 3.1a). Inequality Eqs. (3.1b) to (3.1c) corresponds to the Coloumb friction, where μ denotes the static coefficient of friction. From a modeling perspective, this can be interpreted via the concept of spatial friction cones [**?**]. If, and only if the distributed contact forces lie inside their respective friction cones, these constraints are satisfied. Finally, inequalitiy Eqs. (3.1d) to (3.1e) constrain the CoP to lie inside the rectangular contact area of each foot (see Fig. 3.1b). C_x and C_y denote the x and y position of $\boldsymbol{p}_{CoP}$, respectively. These CoP constraints prevent the robot from tilting about the edges of the rectangular surface contact. In particular, Eq. (3.1d) corresponds to a constraint of tilting around the pitch axis and Eq. (3.1e) prevents tilting around the roll axis.

Both, the unilaterality of the contact forces and the friction cone constraints, are already implemented inside Crocoddyl. However, the central component, bounding the CoP to lie inside the support area of each contact foot, is missing. Therefore, the rest of this chapter deals with the derivation of a set of implementable CoP constraints and describes the integration of these constraints as a cost function into the Crocoddyl framework.

3.2 Center of Pressure (CoP) Constraints

In this section we will derive a universal set of implementable constraints that bound the CoP to lie inside the rectangular contact area of each foot.

3.2.1 CoP Stability Conditions

Recapitulate the constraints from inequality Eqs. (3.1d) to (3.1e). Instead of using the absolute value of X and Y, one can also formulate the constraints as

$$\begin{aligned} \textbf{Pitch}: \quad & -X \leq C_x \leq X, \\ \textbf{Roll}: \quad & -Y \leq C_y \leq Y. \end{aligned} \tag{3.2}$$

Based on this formulation it becomes evident that the CoP is constraint to lie inside the foot geometry visualized in Fig. 3.1b. In fact, these conditions can be represented via four single inequality equations as

$$\begin{aligned} X + C_x &\geq 0, \\ X - C_x &\geq 0, \\ Y + C_y &\geq 0, \\ Y - C_y &\geq 0. \end{aligned} \tag{3.3}$$

These four inequality equations will be used in the following to formulate the CoP constraints.

3.2.2 CoP Computation

Our goal is to determine explicit expressions for C_x and C_y for arbitrary floor orientations, including inclined ground. To this end, consider the computation routine for the CoP from Eq. (2.4)

$$\boldsymbol{p}_{CoP} = \frac{\boldsymbol{n} \times \boldsymbol{\tau}_{\boldsymbol{O}}^{\boldsymbol{c}}}{\boldsymbol{f}^c \cdot \boldsymbol{n}}.$$

For arbitrary orientations of the contact normal vector $\boldsymbol{n}$, we obtain

$$\boldsymbol{p}_{CoP} = \frac{\boldsymbol{n} \times \boldsymbol{\tau}_{\boldsymbol{O}}^{\boldsymbol{c}}}{\boldsymbol{f}^c \cdot \boldsymbol{n}} = \frac{\begin{bmatrix} n_x \\ n_y \\ n_z \end{bmatrix} \times \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix}}{\begin{bmatrix} f_x \\ f_y \\ f_z \end{bmatrix} \cdot \begin{bmatrix} n_x \\ n_y \\ n_z \end{bmatrix}} = \begin{bmatrix} n_y t_z - n_z t_y \\ n_z t_x - n_x t_z \\ n_x t_y - n_y t_x \end{bmatrix} \cdot \frac{1}{f_x n_x + f_y n_y + f_z n_y}, \tag{3.4}$$

and solve for the desired position C_x and C_y of the CoP as

$$C_x = \frac{n_y t_z - n_z t_y}{f_x n_x + f_y n_y + f_z n_y}, \tag{3.5a}$$

$$C_y = \frac{n_z t_x - n_x t_z}{f_x n_x + f_y n_y + f_z n_y}. \tag{3.5b}$$

3.2.3 CoP Inequality Constraints

Now that we have found explicit expressions for computing the CoP (Eqs. (3.5a) to (3.5b)), we can insert them into Eq. (3.3), which gives a set of four CoP constraints as

$$\begin{aligned} X + \frac{n_y t_z - n_z t_y}{f_x n_x + f_y n_y + f_z n_y} &\geq 0, \\ X - \frac{n_y t_z - n_z t_y}{f_x n_x + f_y n_y + f_z n_y} &\geq 0, \\ Y + \frac{n_z t_x - n_x t_z}{f_x n_x + f_y n_y + f_z n_y} &\geq 0, \\ Y - \frac{n_z t_x - n_x t_z}{f_x n_x + f_y n_y + f_z n_y} &\geq 0. \end{aligned} \tag{3.6}$$

These conditions can be written in matrix form as:

$$\begin{bmatrix} Xn_0 & Xn_1 & Xn_2 & 0 & -n_2 & n_1 \\ Xn_0 & Xn_1 & Xn_2 & 0 & n_2 & -n_1 \\ Yn_0 & Yn_1 & Yn_2 & n_2 & 0 & -n_0 \\ Yn_0 & Yn_1 & Yn_2 & -n_2 & 0 & n_0 \end{bmatrix} \cdot \begin{bmatrix} f^x \\ f^y \\ f^z \\ \tau^x \\ \tau^y \\ \tau^z \end{bmatrix} \geq \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \tag{3.7}$$

and finally yield an implementable set of inquality equations for constraining the CoP to lie inside the rectangular contact area of each foot.

3.3 Integration Into the Crocoddyl Framework

This section presents the integration of the derived CoP inequality constraints from Eq. (3.7) into the Crocoddyl framework.

3.3.1 Inequality Constraints by Penalization

In Section 2.4 we have discussed possible ways of incorporating inequality constraints into DDP-like solvers. Crocoddyl handles inequality constraints, such as joint limits or friction cone, via penalization. In numerical optimization, the goal is to minimize

a given cost function. In Crocoddyl, an *action model* combines dynamics and cost model for each knot of the discretized OC problem from Eq. (2.8). The cost function for an action model at knot n can be written as:

$$l_n = \sum_{c=1}^{C} \alpha_c \Phi_c(\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{\tau}), \tag{3.8}$$

where C different costs Φ_c are weighted by a respective coefficient $\alpha_c \in R$. The goal of the following two parts is to demonstrate how the inequality Eq. (3.7) is implemented inside a novel cost function into the framework.

3.3.2 Computation of the Residual

In numerical analysis, the term *residual* corresponds to the error of an result [**?**]. For the sake of compactness, we abbreviate Eq. (3.7) as

$$\boldsymbol{A} \cdot \boldsymbol{w} \geq \boldsymbol{0}, \tag{3.9}$$

where $\boldsymbol{A}$ corresponds to a matrix of CoP inequality constraints and $\boldsymbol{w}$ is the contact wrench acting on the according foot. The residual $\boldsymbol{r} \in \boldsymbol{R}_{4\times1}$ of the cost is retrieved by a simple matrix-vector multiplication:

$$\boldsymbol{r} = \boldsymbol{A} \cdot \boldsymbol{w}. \tag{3.10}$$

3.3.3 Computation of the Cost

The residual vector depicted in Eq. (3.10) typically contains non-zero numbers. The resulting scalar CoP cost value Φ_{CoP} is computed via a bounded quadratic activation as

$$\begin{aligned} \Phi_{CoP} = 0.5 \cdot ||\boldsymbol{r}||^2 \quad &| \; lb >= \boldsymbol{r} >= ub \\ \Phi_{CoP} = 0 \quad &| \; lb < \boldsymbol{r} < ub. \end{aligned} \tag{3.11}$$

In order to account for the positiveness of $\boldsymbol{r}$ (see Eq. (3.9)), the bounds are set to $lb = \boldsymbol{0}$ and $ub = \infty$, respectively. Finally, this bounded quadratic activation of the residual vector has the following implications:

- The CoP cost is zero, whenever $\boldsymbol{p}_{CoP}$ lies inside or on the border of the foot area spanned by X and Y,
- The CoP cost increases in a quadratic manner, when $\boldsymbol{p}_{CoP}$ exceeds the foot area spanned by X and Y.

3.3.4 Basic Usage of the CoP Cost

The cost function is implemented in C++, but can be accessed via Python bindings for versatile and fast prototyping. In the following, a basic example is provided to demonstrate the interface of the CoP cost function to the interested reader.

```
# 1. Creating the cost model container
costModel = crocoddyl.CostModelSum(state, actuation.nu)
# 2. Defining the CoP cost
footGeometry = np.array([0.2, 0.08]) # dim [m] of the foot area
CoPCost = crocoddyl.CostModelContactCoPPosition(state,
crocoddyl.FrameCoPSupport(footId, footGeometry), actuation.nu)
# 3. Adding the CoP cost term with assigned weight to the cost model
costModel.addCost("LF_CoPCost", CoPCost, 1e3)
```

3.3.5 List of Contributions

The contributions of this thesis to the open-source framework Crocoddyl are summarized in two main pull requests. The first one, #792 contains the basic formulation of the CoP cost function for contact dynamics action models. With #830, an additional version is added for the special case of impulse dynamics. A functional unittest that checks the cost against numerical differentiation can be found in the according directory of [**?**].

CHAPTER 4

Bipedal Walking Variants

This chapter studies the proposed motion planning approach for bipedal walking gaits of the full-size humanoid RH5. It starts by describing the individual building blocks of the optimization problem, then discusses the simulation results obtained for increasing gait dynamics and finally provides an evaluation of the contact stability of the dynamic motions.

4.1 Formulation of the Optimization Problem

This section gives information about the adopted contact and impact modeling techniques and introduces the constraints used for generating physically compliant walking trajectories. The core formulation is based on the legged gaits described in [**?**], but contains various improvements necessary for application on real robots. All motions presented in this chapter are solved given a predefined sequence of contacts and step timings.

Recapitulating Section 2.3, we formulate the optimization problem as

$$\boldsymbol{X}^*, \boldsymbol{U}^* = \arg\min_{\mathbf{X},\mathbf{U}} l_N(x_N) + \sum_{k=0}^{N-1} \int_{t_k}^{t_k+\Delta t} l(\mathbf{x}, \mathbf{u}) dt. \tag{4.1}$$

4.1.1 Contact and Impact Modeling

During a walking motion, the body is always in contact with the ground either in single support, or in double support. In Section 2.4 we have discovered, how rigid

contacts can be expressed as a kinematic constraint on the EoM (see Eq. (2.14)). Analogously, one can describe the impulse dynamics [1] of a multibody system as

$$\begin{bmatrix} \boldsymbol{M} & \boldsymbol{J}_c^\top \\ \boldsymbol{J}_c & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{v}^+ \\ -\boldsymbol{\Lambda} \end{bmatrix} = \begin{bmatrix} \boldsymbol{M}\boldsymbol{v}^- \\ -e\boldsymbol{J}_c\boldsymbol{v}^- \end{bmatrix}, \tag{4.2}$$

where $\boldsymbol{\Lambda}$ is the contact impulse, $\boldsymbol{v}^-$ and $\boldsymbol{v}^+$ are the generalized velocities before and after the impact and $e \in [0, 1]$ is the restitution coefficient that accounts for the elasticity of the collision. For all motions, we use this impulse model to account for the infinitesimal short change in the contact situation. To improve the numerical integration stability, terms defined by Baumgarte Stabilization [**?**] are used along with the rigid contact constraint described with Eq. (2.15).

4.1.2 Robot Tasks

Robot tasks, such as grasping and object or performing a step, are an essential goal of motion planning. As outlined in Section 2.4, these task-related constraints are considered in the optimization process as regulator functions. In the context of this thesis two tasks are of specific interest, namely the foot tracking and CoM tracking.

Foot Tracking Cost In order to perform a symmetric gait (see Section 2.1), the design of dedicated foot trajectories is crucial. We used piecewise-linear functions to describe the swing foot reference trajectory. Deviation from this time-depended reference foot trajectory is highly penalized. The foot tracking cost can be formulated via the squared euclidean norm (L2 norm) as

$$\textbf{Foot} : \Phi_1 = || \; c(t) - c^{ref}(t) \; ||_2^2,$$

where $c(t)$ is the actual CoM position at time-point t and $c^{ref}(t)$ is the according reference. Start and end position as well as timings are predefined and combined with a desired step height to form the desired foot trajectory.

CoM Tracking Cost In bipedal locomotion, the three dimensional position of the CoM of the whole-body turned out to be crucial [**?**]. This is especially true for quasi-static motions, where the FCoM is used as static stability margin as explained in Section 2.1. Analogously to the foot cost, the CoM tracking cost is formulated as

$$\textbf{CoM} : \Phi_2 = || \; f(t) - f^{ref}(t) \; ||_2^2 \; .$$

In order to account for the static stability in these motions, we perform a dedicated shifting of the FCoM to the foot center, before performing the swing-foot task.

[1] Impulse dynamics account for the physical effects that occur at a switch from non-contact to contact condition. Detailed information can be found e.g. in [**?**].

Additionally, the CoM is kept at a constant height to prevent unnecessary forces acting on the base of the body.

4.1.3 Inequality Constraints for Physical Compliance

Essential demands on physically compliant motion planning are that (i) the robot limits (torque, joints) are considered and (ii) the generated trajectories are inherently balanced. To this end, we consider joint limits, friction cone and the novel CoP bound as inequality constraints in our formulation, while torque constraints are covered in the algorithm itself.

Contact Stability Constraints As detailed in Chapter 3 with the concept of contact stability constrained DDP, we constrain unilaterality, friction and CoP for each foot in contact. For the sake of clarity, Eq. (3.11) is again capitulated as

$$\begin{aligned} \textbf{CoP} : \Phi_3 = 0.5 \cdot ||\boldsymbol{r}||^2 \quad & | \; lb >= \boldsymbol{r} >= ub, \\ \Phi_3 = 0 \quad & | \; lb < \boldsymbol{r} < ub, \end{aligned}$$

where the CoP position is bound to lie inside the foot contact area by the lower and upper bounds lb and ub, respectively. In the same manner friction cone constraints along with the unilaterality are considered as

$$\begin{aligned} \textbf{Friction} : \Phi_4 = 0.5 \cdot ||\boldsymbol{r}||^2 \quad & | \; lb >= \boldsymbol{r} >= ub, \\ \Phi_4 = 0 \quad & | \; lb < \boldsymbol{r} < ub, \end{aligned}$$

where lb and ub bound the resulting contact force to lie inside a 4-sided polygonal approximation of the spatial friction cone [**?**].

Joint Limits Physical boundaries of the joints must not be exceeded to avoid damage to the system. They are covered via a bounded quadratic activation as

$$\begin{aligned} \textbf{Joints} : \Phi_5 = 0.5 \cdot ||\boldsymbol{r}||^2 \quad & | \; lb >= \boldsymbol{r} >= ub \\ \Phi_5 = 0 \quad & | \; lb < \boldsymbol{r} < ub. \end{aligned}$$

where lb and ub correspond to the lower and upper bounds for joint position and velocities, respectively.

4.1.4 Further Regularization Terms

Additional to the described constraints for tasks and physical compliance, we optimize for minimization of the torques and regularize the robot posture.

Torque Minimization In order to improve the energy efficiency of the motions and maintain a human-like torque at the joints [**?**], we minimize the joint torques for realistic dynamic movements via

$$\textbf{Torque} : \Phi_6 = || \boldsymbol{\tau}(t) ||_2^2 .$$

Posture Regularization Finally, we deal with the redundancy of multi-body dynamics by applying a weighted least-squares cost function to regularize the state with respect to the nominal robot posture:

$$\textbf{Posture} : \Phi_7 = || q(t) - q^{ref} ||_2^2 .$$

4.2 Simulation Results for Increasing Gait Dynamics

This section presents the simulation results for bipedal walking motions obtained by solving an optimization problem based on the described building blocks from the previous section. It studies both static and dynamic walking gaits, respectively.

4.2.1 Static Walking

The analysis of static walking gaits provide detailed insights on the optimization structure and allows a thorough experimental validation (see **??**).

Focus of investigation is a slow two step walking motion. We assume a quasi-static motion and hence deploy a static stability criterion as introduced in Section 2.2 by following a dedicated FCoM trajectory. To this end, the optimization problem is composed of a total of five locomotion phases:

1. FCoM shift from initial position to the center of the left foot (LF).
2. Perform a right step, while the FCoM remains at the LF center.
3. FCoM shift from the LF center to the right foot (RF) center.
4. Perform a left step, while the FCoM remains at the RF center.
5. FCoM shift from the RF center to the half length of the line intersecting both feet center while returning to the initial pose.

Additionally, the CoM is optimized for a constant height over the whole gait. Table 4.1 gives a compact overview of the desired gait characteristics and the applied constraints of the optimization. In order to comply with the quasi-static assumption, the motion is performed about a time horizon of 15s with a total stride length of 20cm a robot model with fixed arms.

The results of the optimization problem Eq. (4.1) are shown in Fig. 4.1 and Fig. 4.2.

Table 4.1: Static walking gait characteristics and applied optimization constraints.

Gait Characteristics		**Optimization Constraints**	
Step length:	10 cm	Tasks:	Foot (Φ_1), **CoM** (Φ_2)
Step height:	5 cm	Stability:	Friction Cone (Φ_4)
Time:	3 s/phase	Limits:	Joint (Φ_5), Torque (Φ_6)
Step size:	0.03 s	Regularization:	Posture (Φ_7), Torque (Φ_8)

Fig. 4.1 presents the resulting base and end-effector trajectories. As becomes clear, the FCoM tracking is sufficiently good and the CoM height remains in a reasonable range of ± 1cm. Equally, the desired foot trajectory is tracked with adequate accuracy. Also the foot velocities are reasonable with a maximum velocity of 0.1m/s at the impact. The pursued step height is not reached exactly, but is about one centimeter less. This effect can be explained by the immediately reverse direction at the vertex of the piecewise-linear trajectory, which is smoothed by the solver.

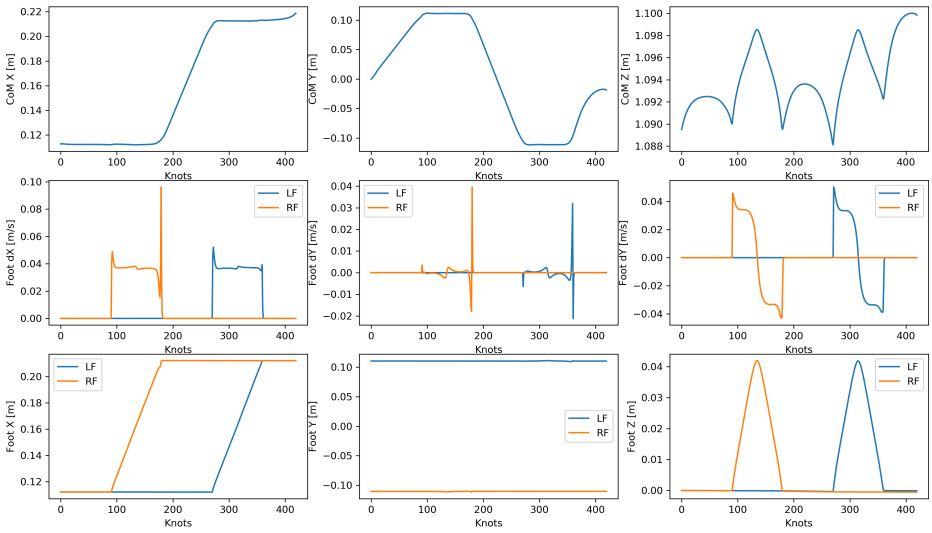


Figure 4.1: Static walking gait solution in task space. Both the FCoM task and the swing foot task is satisfied with acceptable accuracy.

Fig. 4.2 shows the resulting joint trajectories for the torso, LF and RF. Both the joint position limits and the maximum permissible joint speeds remain far below the limits due to the slow nature of the motion. Interestingly, the pursued FCoM shifting turns out to be realized mostly based on a shift in the body roll activation rather than on a shift in the hips. Furthermore, it becomes evident that the posture regularization is effective since all joints end closely to the initial position.

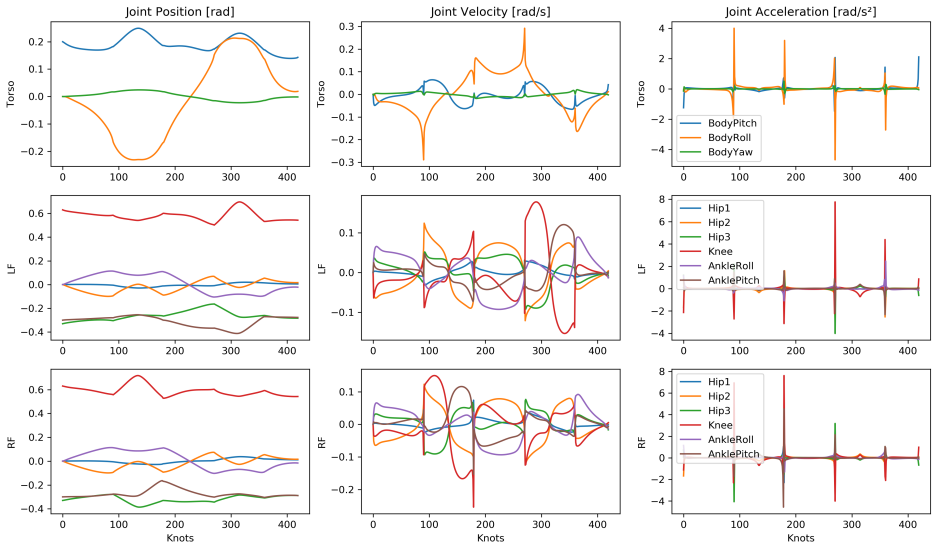


Figure 4.2: Static walking gait solution of the joint states. Due to the slow nature of the motion, joint limits are far from being reached.

4.2.2 Dynamic Walking

The analysis of a dynamic walking gait is concerned about generating efficient motions with higher velocities. This part forms the basis of the stability evaluation in the next section.

Focus of investigation is a dynamic walking motion. Compared to the previously studied static walking gait these motions are characterized by higher velocities and dynamic forces exceed the static ones. These characteristics imply that dynamic stability criteria become necessary. To this end, we apply the proposed approach of contact stability constrained DDP described in Chapter 3. Consequently, the CoP of each foot is constrained instead of following a reference CoM trajectory. By this, the solver is enabled to find an optimal, dynamic CoM shifting along with the requested contact stability constraints.

The optimization problem is composed of a total of five locomotion phases, namely 1. Double Support (DS), 2. right step, 3. DS, 4. left step and 5. recovery to the initial pose. In accordance with biomechanical findings [**?**], we choose a desired step length of 40cm with 1.5s per step and deliberately define a stance/swing ratio of 1/3. Table 4.2 compactly summarizes the dynamic gait characteristics and applied constraints of the optimization.

The solution of the dynamic walking OC problem is visualized in Fig. 4.3. As becomes clear from Fig. 4.4, a natural CoM shifting to the sides emerges resulting from the inequality constraints for the CoP, which is about half of the amount as for the static walking case. Although the CoM height is not explicitly constrained, it

Table 4.2: Dynamic walking gait characteristics and applied optimization constraints.

Gait Characteristics		**Optimization Constraints**	
Step length:	40 cm	Tasks:	Foot (Φ_1)
Step height:	5 cm	Stability:	**CoP** (Φ_3), Friction Cone (Φ_4)
Time:	1.5 s/step	Limits:	Joint (Φ_5), Torque (Φ_6)
Step size:	0.03 s	Regularization:	Posture (Φ_7), Torque (Φ_8)
Stance/Swing Ratio:	1/3		

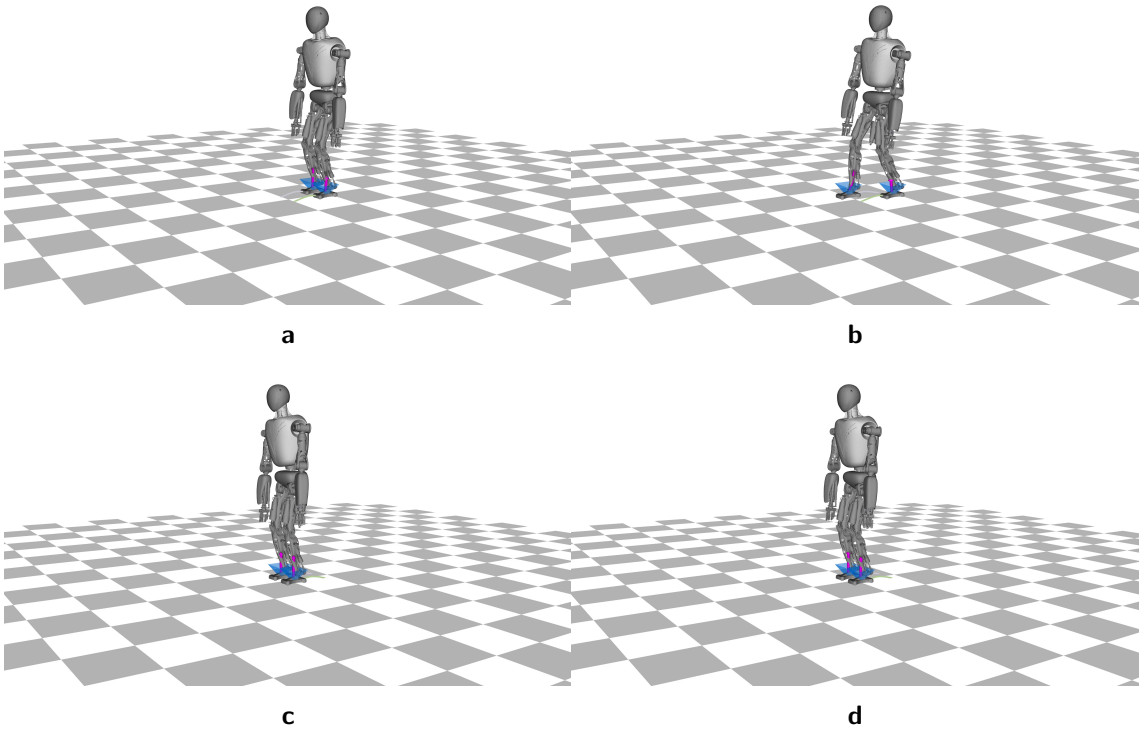


Figure 4.3: Bipedal dynamic walking. Image order: column-wise, from top to bottom and left to right.

stays in a reasonable range of about +3cm, which might be caused from to the final posture regularization. The end-effector velocities in z-direction are about twice as high as for the case of static walking, which is explained by the higher walking speed. Fig. 4.5 shows the resulting joint states for the dynamic walking gait. The velocity and acceleration contain higher peaks and also the joint deflections are stronger, which is reasonable due to the higher walking speed.

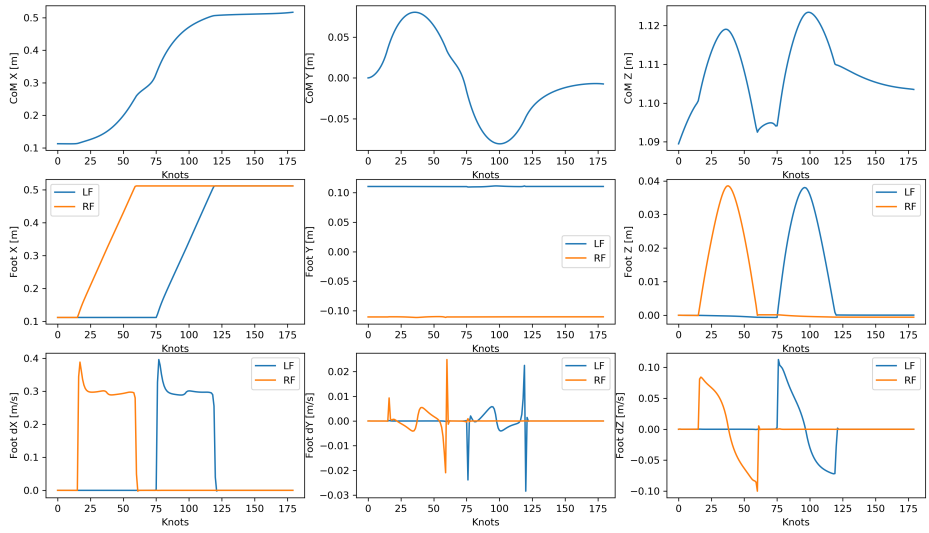


Figure 4.4: Dynamic walking gait solution in task space.

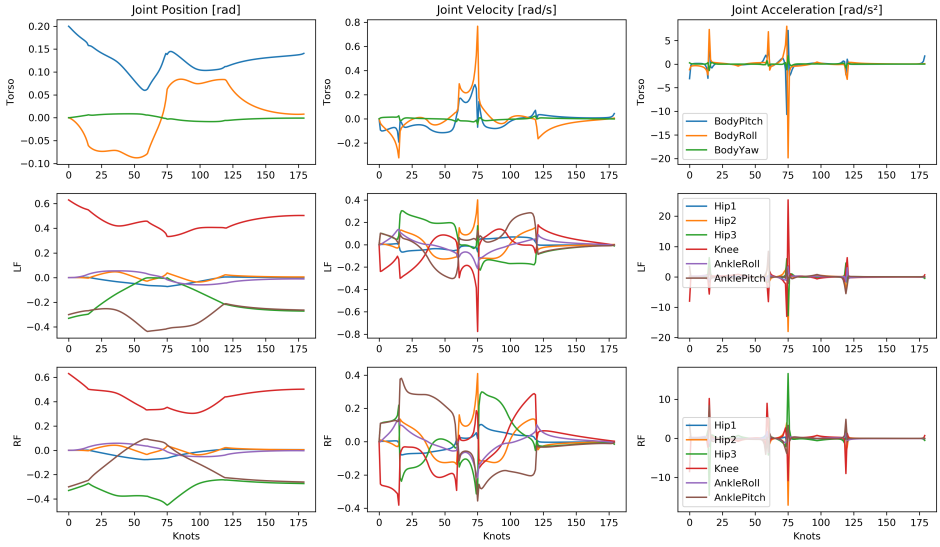


Figure 4.5: Dynamic walking gait solution of the joint states.

4.3 Evaluation of Contact Stability

This section evaluates, based on the presented dynamic walking gait from the previous section, the proposed approach of contact stability constrained DDP (Chapter 3).

4.3.1 Dynamically Balanced Walking Motion

As detailed in Section 2.2, the central characteristic for dynamically balanced motions is that the the CoP or ZMP remains within the SP. The generic approach presented in Chapter 3 has been applied to the dynamic walking gait presented in the previous section. It utilizes the CoP criterion for each contact surface and hence the motion can be called dynamically balanced if, and only if, the CoP of each foot in contact stays within the according SP along the whole motion.

Fig. 4.6 shows the top view of the presented dynamic walking gait. The rectangles correspond to the true-to-scale dimensions of the robot feet. For clarity, only the first and last DS phase are visualized. The blue curve shows the time course of the resulting CoM trajectory, with relevant points in time marked separately. Since the CoM trajectory between lift-off and touch-down is largely outside the respective foot area, no static stability can be present, as expected. The orange and green crosses mark the time-dependent position of the CoPs for both feet. It is evident that both CoPs remain within the corresponding SP over the entire time course of the movement, which is why the movement can be classified as dynamically balanced.

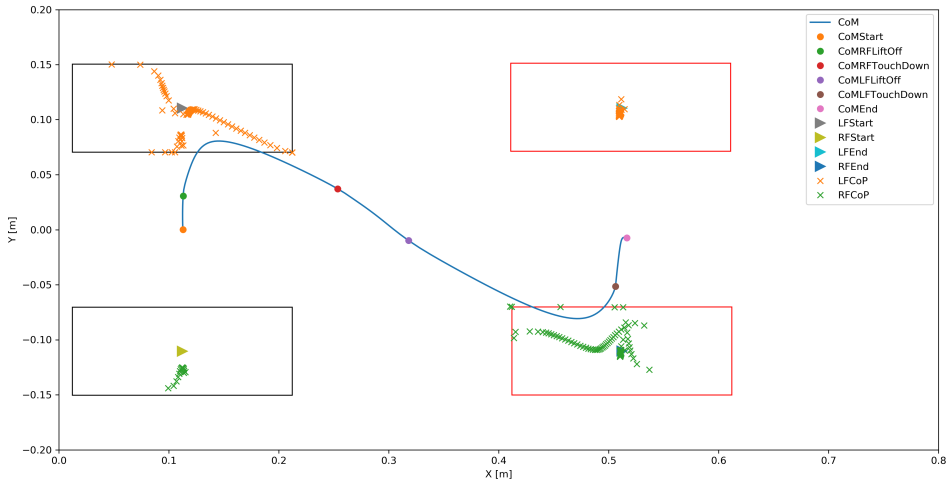


Figure 4.6: Stability Analysis of the dynamic walking gait.

4.3.2 Different Levels of CoP Restriction

Although the dynamic walking motion is inherently balanced, it becomes clear from Fig. 4.6 that the CoP partly lies *on* or *near* the border of the respective foot area. This effect can be attributed to the formulation of the CoP cost function (Eq. (3.11)), which is defined to be zero whenever the CoP lies within the given foot area and a

quadratic penalization prevents the CoP to leave the SP. Theoretically, this formulation is sufficient to generate balanced motions. In practice, however, it might be convenient for real-world experiments to consider a particular safety factor so that the CoPs maintain a certain distance from the edge. With the presented CoP cost function, this objective can be easily achieved by reducing the desired foot geometry. Fig. 4.7 shows the dynamic walking gait, where the CoP inequality constraints are active for a SP reduced to 50 percent of the original foot geometry. It becomes evident that also these more conservative contact stability constraints can be solved, which might be useful for the purpose of experimental validation.

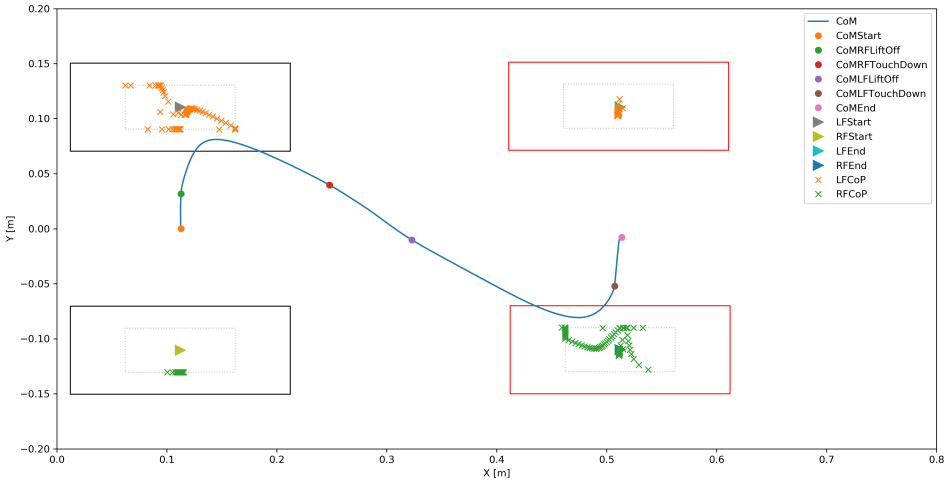


Figure 4.7: Stability Analysis of the dynamic walking gait.

In this section we have seen that the proposed contact stability constrained DDP produces motions that are dynamically balanced. In **??** we will investigate if the generated motions can be tracked by a simple online stabilizer based on position control in joint space. Beforehand, in Chapter 5, we will explore the effect of the motion planning approach and physical system limits on highly dynamic movements.

CHAPTER 5

Highly-Dynamic Movements

This chapter presents an analysis of the proposed motion planning approach for highly-dynamic movements of the full-size humanoid RH5. To begin with, the central building blocks of the optimization problem are again briefly addressed. Then the simulation results for jumping tasks of increasing complexity are presented. Finally, an analysis of the maximal system performance based on the allowed joint and torque limits is presented.

5.1 Formulation of the Optimization Problem

This section concisely summarizes the constraints of the OC problem used to generate highly-dynamic movements as demonstrated in the following up section.

The vertical jump formulation follows an optimization problem of the form described in Eq. (4.1). For performing multiple forward jumps over obstacles instead, we successively solve P individual optimization problems of range N for each jump. To this end, we formulate a multi-phase OC problem as follows:

$$\boldsymbol{X}^*, \boldsymbol{U}^* = \arg\min_{\mathbf{X},\mathbf{U}} \sum_{p=0}^{P} \sum_{k=0}^{N-1} \int_{t_k}^{t_k+\Delta t} l_p(\mathbf{x}, \mathbf{u}) dt. \tag{5.1}$$

We constrain the optimization based on the building blocks introduced in Section 4.1. Precisely, we define a foot tracking cost Φ_1 based on piecewise-linear functions to incorporate the basic jumping height and length. We apply the contact stability constrained DDP (Chapter 3) with dedicated cost functions for CoP (Φ_3) and friction cone (Φ_4). Physical compliance is ensured via the torque bounds of the solver and joint limits costs Φ_5. We use a CoM tracking cost Φ_2 only for the final

Table 5.1: Vertical jump characteristics and applied optimization constraints.

Jump Characteristics		**Optimization Constraints**	
Jump height:	10 cm	Tasks:	Foot (Φ_1)
Total time:	0.9 s	Stability:	CoP (Φ_3), Friction Cone (Φ_4)
Step size:	0.01 s	Limits:	Torque (Φ_6)
		Regularization:	Posture (Φ_7), Torque (Φ_8)

stabilization of the motion. Additionally, torque minimization (Φ_6) and posture regularization (Φ_7) are optimized.

Biomechanical studies have shown that the effect of arm swinging is elementary for the performance in human jumping [1]. To this end, we also include the arms in the optimization for our jumping tasks. In order to account for the dynamic nature of the movements, it turned out to be useful to reduce the integration step size. We found a higher feasibility of the jumping tasks with an integration step size of 10ms compared to 30ms for the bipedal walking gaits (see Section 4.2). Since we do not use a contact planner along with this work, the contact timings had to be chosen to approximately match the physic of flight phases. Highly dynamic movements are subject to higher impulse forces than is the case with walking movements. As described in 2.3, DDP relies on integration of the system dynamics in each step to obtain the states. The higher impulse forces then in turn cause numerical drifts of higher order in the contact constraints. This effect is visible as drifting of the support feet after touchdown. Through an extensive grid search, we found a set of valid Baumgarte gains that reduce these numerical drifts for highly-dynamic movements.

5.2 Simulation Results for Increasing Task Complexity

This section presents the simulation results for three case studies of highly-dynamic movements obtained by solving an optimization problem based on the description in the previous section. First, we study the task of vertically jumping upwards, then we investigate forward jumping and finally we explore a challenging sequence of multiple forward jumps over obstacles.

5.2.1 A Simple Vertical Jump

The analysis of a simple vertical jump is a good example to understand the underlying challenges highly-dynamic movements bring along in the context of numerical optimization.

The vertical jumping task (see Table 5.1) consists of five phases as depicted in Fig. 5.1. From the initial position (a), a descending into the jump (b) takes place. Then, an upward motions accelerates the base until the take off (c). The symmetrical flight phase (d) is ended with a touch down (e) followed by a pose recovery (f).

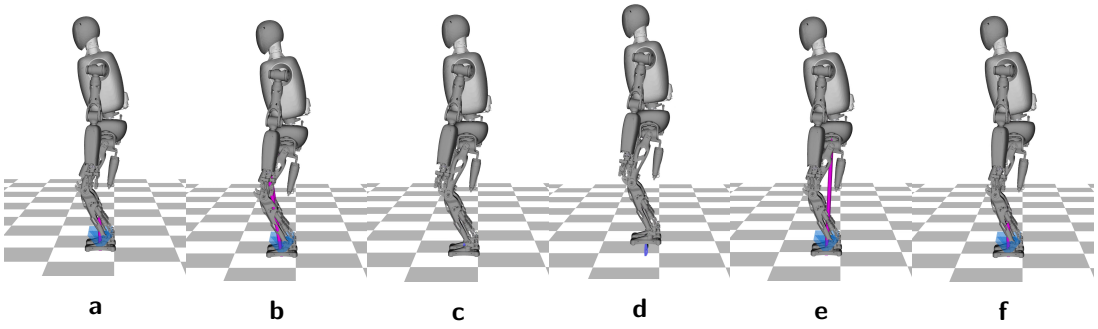


Figure 5.1: A simple vertical jump.

Fig. 5.2 provides insights in the dynamic nature of the motion. As can be seen, the joint positions stay within reasonable ranges. However, velocity peaks at the take off exceed the maximum joint velocities of the body pitch and knee joints by a factor of two and four, respectively. This effect is plausible, since both the knee deflection as well as the torso swing are essential for a jump. Consequently, the short time horizon of the jump requires high peak velocities in these task-relevant joints.

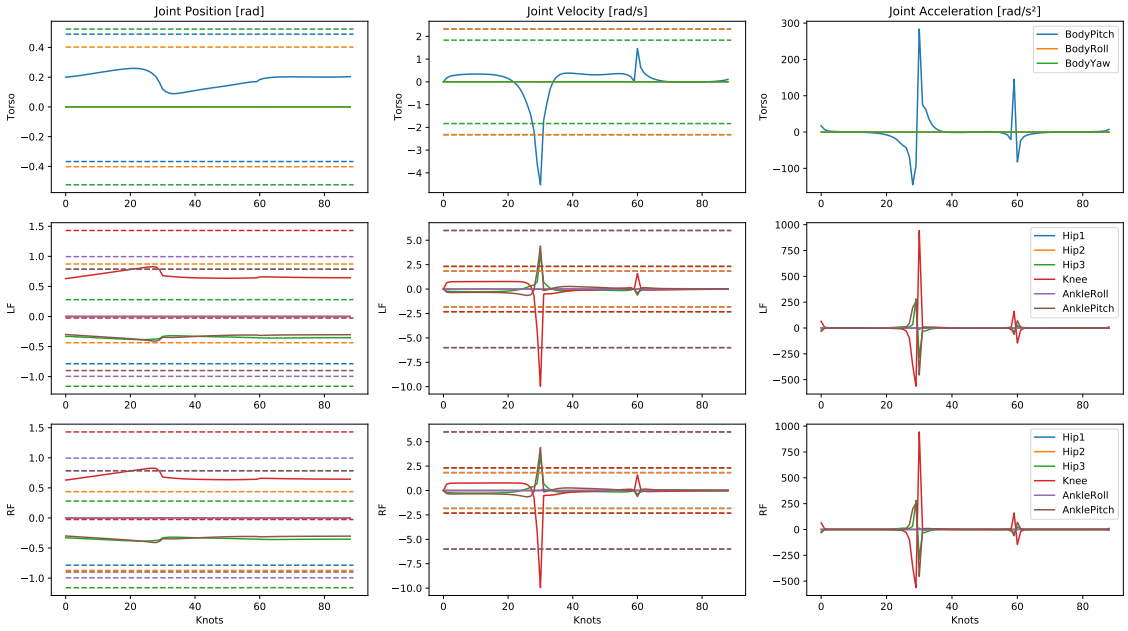


Figure 5.2: Vertical jump solution of the joint states with according joint limits visualized as dashed lines. The maximum velocities of the body pitch and knee joints turn out to be insufficient for the highly-dynamic take off.

A similar observation applies to the joint torques as shown in Fig. 5.3. Although the solver prevents the body pitch and knee torques to exceed the limits in this case, it becomes evident that maximum torques for these joints are necessary over longer

Table 5.2: Forward jump characteristics and applied optimization constraints.

Jump Characteristics		**Optimization Constraints**	
Jump length:	30 cm	Tasks:	Foot (Φ_1)
Jump height:	10 cm	Stability:	CoP (Φ_3), Friction Cone (Φ_4)
Total time:	0.9 s	Limits:	Torque (Φ_6)
Step size:	0.01 s	Regularization:	Posture (Φ_7), Torque (Φ_8)

time horizons. Such continuous loads should be avoided as far as possible in order to guarantee the durability of the robotic system.

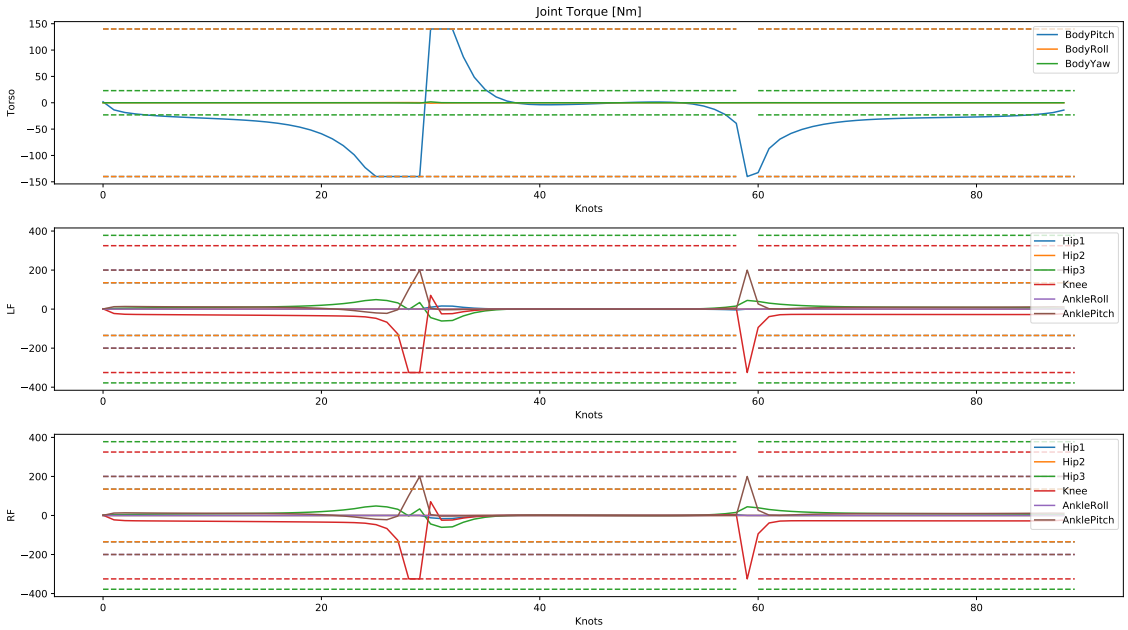


Figure 5.3: Vertical jump solution of the joint torques with according motor limits visualized as dashed lines. The solver prevents the body pitch and knee torques to exceed the limits, but still maximum torques are required for these joints over longer time horizons.

5.2.2 A Simple Forward Jump

Focus of investigation is a more dynamic forward jump to study the effect of increasing task complexity on the contact forces and stability.

Table 5.2 summarizes the characteristics and optimization constraints. The forward jumping problem consists of the same five locomotion phases as the vertical jump described previously (see Fig. 5.4). We can draw similar conclusions regarding the physical compliance of the system, namely appropriate joint position ranges and torque limits, but insufficient maximal velocities for the body pitch and knee joints.

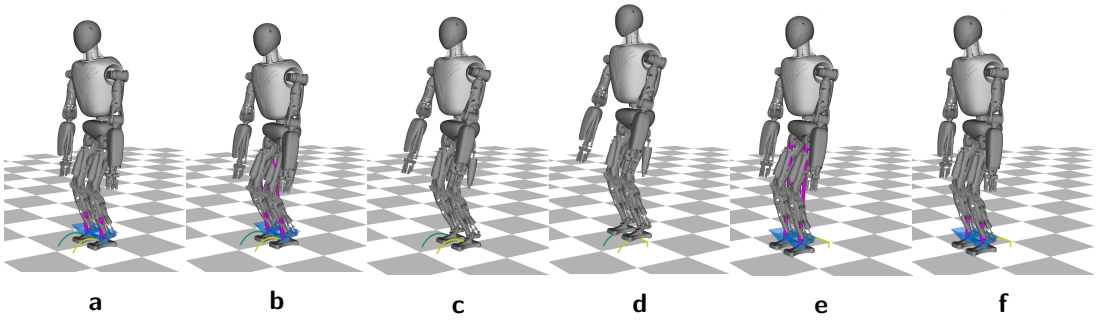


Figure 5.4: A simple forward jump.

Fig. 5.5 presents an overview about the solved optimal contact wrenches. When the robot does not move, the sum of vertical forces F_z for left and right foot equals the body's weight, which is about 600 N. It becomes evident that about 4000 N (7 x robot weight) are exerted to the ground at lift off and about 8000 N (14 x robot weight) at touchdown, which is reasonable according to biomechanical findings [2].

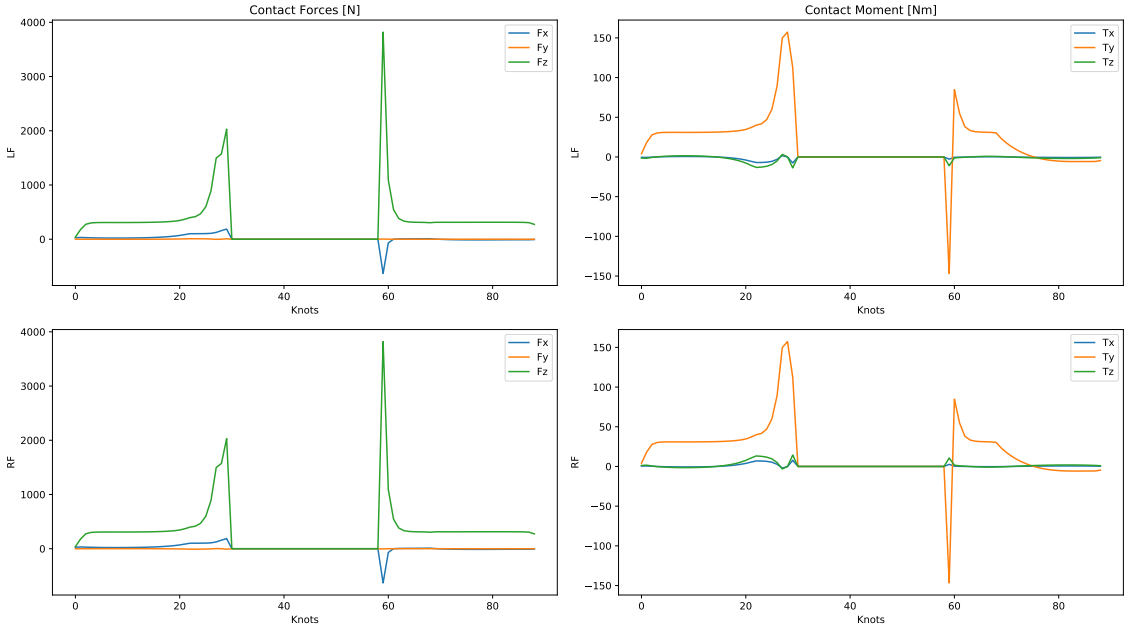


Figure 5.5: Forward jump solution of the contact wrenches.

We have shown in Section 4.3 that the proposed motion planning approach allows computation of dynamically balanced walking gaits. In the following, we want to investigate the feasibility of contact stability for highly-dynamic movements. Fig. 5.6 shows the stability analysis for the forward jumping task. It becomes clear that the CoPs lie within the desired contact area of the left and right foot, respectively. Moreover, it can be observed that CoPs are located more in the rear area of the sole of the foot before the jump or in the front area after landing. The first effect

Table 5.3: Multiple obstacles jumping characteristics and applied optimization constraints.

Jump Characteristics		**Optimization Constraints**	
Number of jumps:	3	Tasks:	Foot (Φ_1)
Jump length:	60 cm	Stability:	CoP (Φ_3), Friction Cone (Φ_4)
Jump height:	25 cm	Limits:	/
Total time:	1.2 s / jump	Regularization:	Posture (Φ_7), Torque (Φ_8)
Step size:	0.01 s		

can be explained due to the angular momentum gained during descending of the base. The latter observation accounts for the rapid deceleration of the base motion in x-direction resulting from the instantaneous impact after the flight phase.

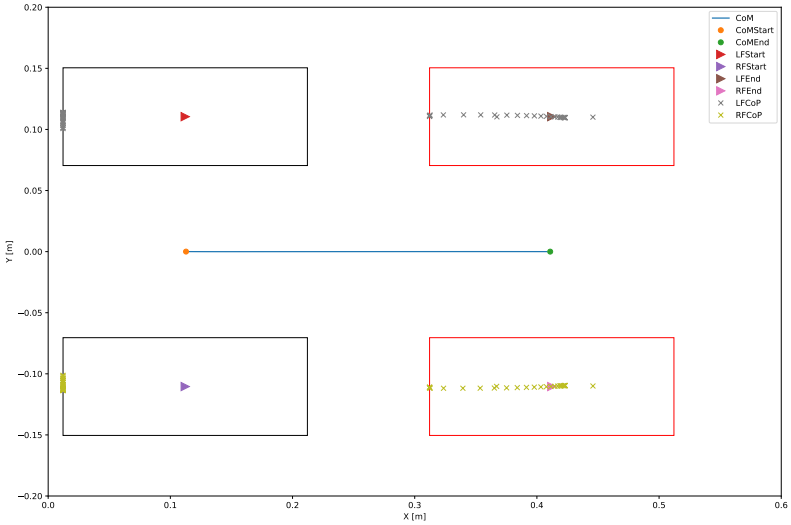


Figure 5.6: Stability analysis of the simple forward jump.

5.2.3 Forward Jumping Over Multiple Obstacles

Finally, we want to further increase the task dynamics and investigate the proposed motion planning approach for a challenging sequence of multiple forward jumps over obstacles.

We build a sequence of OC problems, as introduced in Eq. (5.1), consisting of multiple forward jumps with increased jumping height and length (see Table 5.3, accordingly. Since the RH5 humanoid was not designed for tasks of this dynamics, neither joint nor torque limits can be satisfied. In contrast to the simple vertical and forward jumps, this case study is supposed to proof the flexibility of the proposed motion planning approach rather than analyzing the system design.

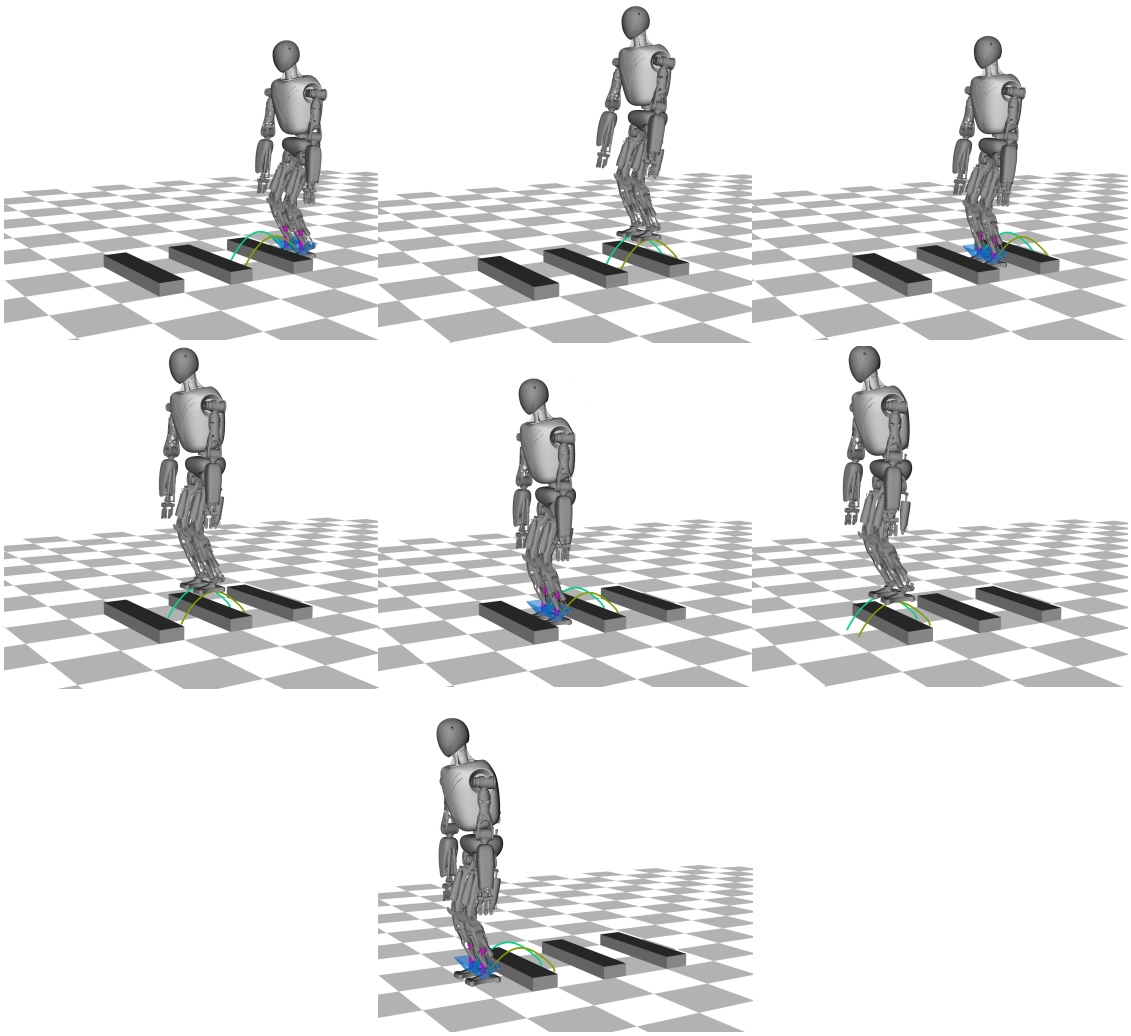


Figure 5.7: Multi-phase OC problem of forward jumping over obstacles. Image order: column-wise, from top to bottom and left to right.

Previously, we have demonstrated that the contact stability constrained DDP allows computation of a simple forward jump with dynamical balanced motions. Following up, we want to study if the concept also holds for multiple forward jumps.

Fig. 5.8 shows the obtained results for stability analysis for sequentially solving the sequence of optimization problems. The robot starts in an initial pose in DS, performs a jump and recovers to the intial pose again. This OC problem is repeated for three times with identical constraints and timings until the robot reaches the final DS pose marked by red rectangles. As becomes evident, the contact stability constrained DDP forces the CoPs of both feet to lie within the desired contact area proofing for dynamically balanced motions.

Recapitulating the results for the simple forward jump, we identified that the CoPs are located in the rear area of the foot sole during takeoff. We can draw similar conclusions for the first jump of the multi-phase OC problem. However, the

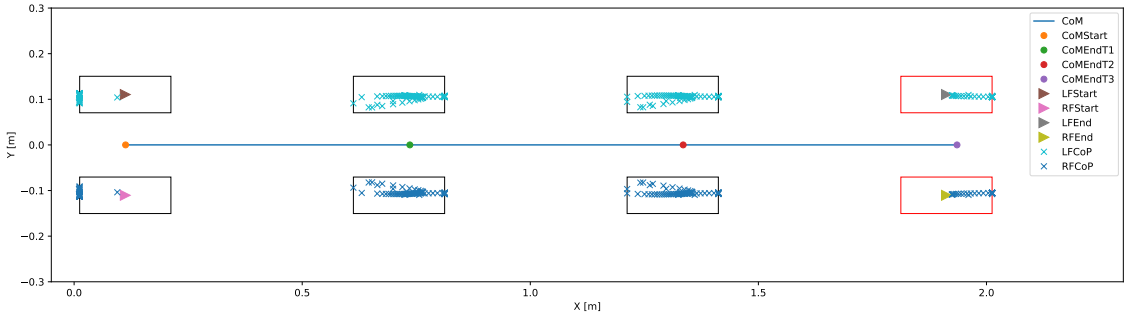


Figure 5.8: Stability analysis of forward jumping over multiple obstacle.

CoPs for the second and third jump do not share this pattern. This effect can be attributed to the dynamic forces already acting on the robot while the second and second impact phase, respectively.

CHAPTER 6

Online Stabilization of the Planned Motions

CHAPTER 7

Conclusion and Outlook

7.1 Thesis Summary

7.2 Future Directions

Bibliography

[1] Everett A Harman, Michael T Rosenstein, Peter N Frykman, Richard M ROSenStein, et al. The effects of arms and countermovement on vertical jumping. *Med Sci Sports Exerc*, 22(6):825–833, 1990.

[2] Ali Meghdari and M Aryanpour. Dynamical modeling and analysis of the human jumping process. In *ASME International Mechanical Engineering Congress and Exposition*, volume 36290, pages 453–461, 2002.