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

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

LIP Linear Inverted Pendulum

MPC Model Predictive Control

OC Optimal Control

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 [1]. 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 [2].

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 [3]. 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 [4].

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 [5]. 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 [6] 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 [7, 8, 9].

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 [10].

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) [11]. The first stage of traditional legged locomotion planning is a contact planner that

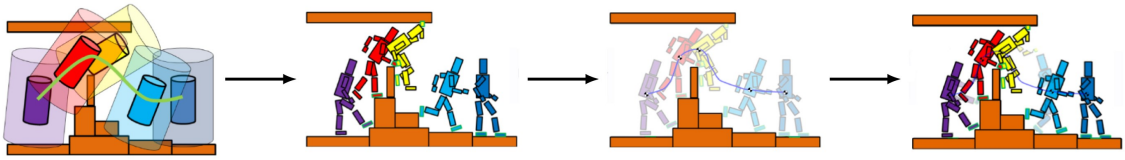


Figure 1.2.: Tradional Legged Locomotion Planning [12].

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 [13, 14, 15, 16]. 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) [17] or operational-space Inverse Dynamics (ID) [18] 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 [19]. Another limitation arises from the fact whole-body planning produces more efficient motions than a simple program such as an IK solver [20].

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 [21, 22, 23].

TO based on reduced centroidal dynamics [24] has become a popular approach in the legged robotics community. In some approaches it is used after planning the contacts [25, 26, 27] while other approaches simultaneously optimize the centroidal trajectory and the contacts [28, 29, 30]. 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 [31, 32, 33].

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 [20]. 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 [34], 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 [12] 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 [35].

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 [36]. The interested reader can find a comprehensive introduction on series-parallel hybrid robots in [37, Ch.2].

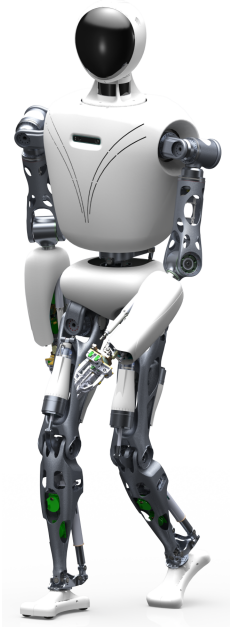


Figure 1.3.: The recently presented RH5 is a lightweight and biologically inspired humanoid used as experimental platform within this thesis.

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.

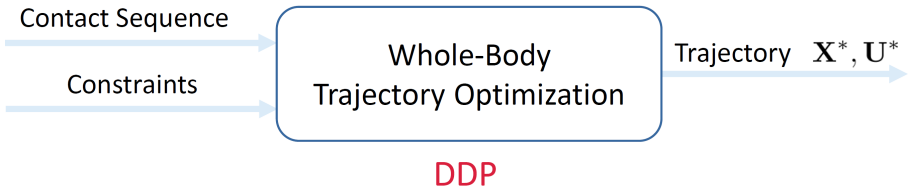


Figure 1.4.: The motion planning approach proposed within this thesis. Based on predefined foothold positions, step timings and stability constraints a DDP-based whole-body TO is used to generate inherently balanced motions plans.

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. Vukobratović et al. provide an extensive introduction to the terminology related to bipedal walking [38], concisely summarized by Dekker [39].

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: "*Man-*

ner 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 [40], comprehensive studies can be found in [41, 42, 43].

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* [44] method or the *Lagrange method* [45], 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

- $\mathbf{S}$ 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. [43, 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 [39, 46, 47].

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 [48] with a nomenclature equivalent to [40].

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 [49], it has been reviewed in [50] and made popular with [13].

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. $\boldsymbol{w}_{CoM}^c = (\boldsymbol{\tau}_{CoM}^c, \boldsymbol{f}^c)_{6\times1}$ 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 Sardain and Bessonnet 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 [10, Sec.1.2.1] and [46]. See Vukobratović et al. for more details on differentiating the terms dynamic stability and dynamic balance [38].

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 Mayne [51]. A modern description of the algorithm using the same notations as below can be found in [23, 52].

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 [53] 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 [54]. 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* [55, 56].

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. Tassa et al. developed a control-limited DDP [23] that takes into account box inequality constraints on the controls allowing the consideration of torque limits on real robotic systems. Budhiraja et al. 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 [20]. 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 [57]. The idea is to minimize the deviation in acceleration between the constrained and unconstrained motion:

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

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 [31].

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 [12].

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

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

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

2.4.4. Inequality Constraints

DDP has a hard time with these... (motivate penalization, mention active-set and augmented lagrangian)

CHAPTER 3

Contact Stability Constrained DDP

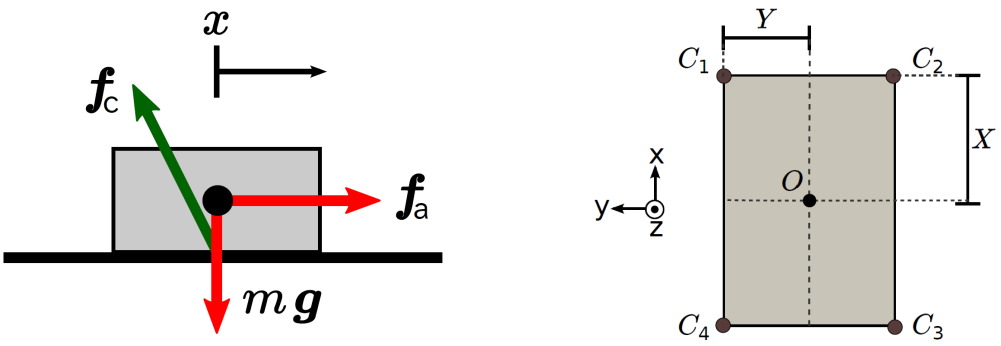
This chapter presents a generic method for integrating contact stability constraints into DDP-like solvers. The key idea is to define inequality constraints for unilaterality, friction and the CoP of each contact surface with the goal of generating inherently balanced motions.

3.1. The Idea

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 [58], without additionally enforcing the yaw torque constraint. These inequality constraints for surface contacts can compactly be summarized as

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



a Body on a horizontal surface.

b Notations used for CoP definition.

Figure 3.1.: Contact in the surface plane [58].

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 [59]. 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)

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

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

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

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 [60]. 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 the special case of impulse dynamics. A functional unittest that checks the cost against numerical differentiation can be found in the according directory of [61].

CHAPTER 4

Bipedal Walking Variants

CHAPTER 5

Highly-Dynamic Movements

CHAPTER 6

Validation in Real-Time Physics Simulation

6.1. Simulation Setup

6.1.1. PyBullet Simulation Environment

6.1.2. Spline Interpolation of the Trajectories

6.1.3. Control Approach

6.2. Motion Stabilization with Joint Space Control

6.2.1. Quasi-Static Movements

6.2.2. Bipedal Walking Variants

6.2.3. Highly-Dynamic Movements

CHAPTER 7

Validation in Real-World Experiments

CHAPTER 8

Conclusion and Outlook

8.1. Thesis Summary

8.2. Future Directions

APPENDIX A

Appendix

A.0.1. Carlos Talk: Essentials

In the end we want to solve a bilevel (nested) optimization

$$\begin{aligned} \boldsymbol{X}^*, \boldsymbol{U}^* &= \arg\min_{\boldsymbol{X},\boldsymbol{U}} \sum_{k=0}^{N-1} task(x_k, u_k) \\ x_k &= \arg\min physics(x_k, u_k), \\ s.t. \quad & constraints(x_k, u_k) \end{aligned} \tag{A.1}$$

Which more formally looks like

$$\begin{aligned} \boldsymbol{X}^*, \boldsymbol{U}^* = \begin{Bmatrix} \boldsymbol{x}_0^*, \cdots, \boldsymbol{x}_N^* \\ \boldsymbol{u}_0^*, \cdots, \boldsymbol{u}_N^* \end{Bmatrix} &= \arg\min_{\boldsymbol{X},\boldsymbol{U}} l_N(x_N) + \sum_{k=0}^{N-1} \int_{t_k}^{t_k+\Delta t} l(\boldsymbol{x}, \boldsymbol{u}) dt \\ s.t. \quad \dot{\boldsymbol{v}}, \boldsymbol{\lambda} &= \arg\min_{\dot{\boldsymbol{v}},\boldsymbol{\lambda}} ||\dot{\boldsymbol{v}} - \dot{\boldsymbol{v}}_{free}||_M, \\ \boldsymbol{x} \in \mathcal{X}, \boldsymbol{u} &\in \mathcal{U} \end{aligned}$$

KKT Matrix:

$$\begin{aligned} \boldsymbol{X}^*, \boldsymbol{U}^* &= \arg\min_{\boldsymbol{X},\boldsymbol{U}} \sum_{k=0}^{N-1} task(x_k, u_k) \\ & KKT - Dynamics(x_k, u_k) \end{aligned}$$

Multi-contact dynamics as holonomic constraints:

$$\begin{bmatrix} \dot{\boldsymbol{v}} \\ -\boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \boldsymbol{M} & \boldsymbol{J}_c^\top \\ \boldsymbol{J}_c & \boldsymbol{0} \end{bmatrix}^{-1} \begin{bmatrix} \boldsymbol{\tau}_b \\ -\boldsymbol{a}_0 \end{bmatrix}$$

A.1. Crocoddyl: Contact RObot COntrol by Differential DYnamic programming Library (Wiki Home)

A.1.1. Welcome to Crocoddyl

Crocoddyl is an optimal control library for robot control under contact sequence. Its solver is based on an efficient Differential Dynamic Programming (DDP) algorithm. Crocoddyl computes optimal trajectories along to optimal feedback gains. It uses Pinocchio for fast computation of robot dynamics and its analytical derivatives. Crocoddyl is focused on multi-contact optimal control problem (MCOP) which as the form:

$$\boldsymbol{X}^*, \boldsymbol{U}^* = \begin{Bmatrix} \boldsymbol{x}_0^*, \cdots, \boldsymbol{x}_N^* \\ \boldsymbol{u}_0^*, \cdots, \boldsymbol{u}_N^* \end{Bmatrix} = \arg\min_{\boldsymbol{X},\boldsymbol{U}} \sum_{k=1}^{N} \int_{t_k}^{t_k+\Delta t} l(\boldsymbol{x}, \boldsymbol{u}) dt$$

subject to

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}),$$
$$\boldsymbol{x} \in \mathcal{X}, \boldsymbol{u} \in \mathcal{U}, \boldsymbol{\lambda} \in \mathcal{K}.$$

where

- the state $\boldsymbol{x} = (\boldsymbol{q}, \boldsymbol{v})$ lies in a manifold, e.g. Lie manifold $\boldsymbol{q} \in SE(3) \times \boldsymbol{R}^{n_j}$,
- the system has underactuacted dynamics, i.e. $\boldsymbol{u} = (\boldsymbol{0}, \boldsymbol{\tau})$,
- $\mathcal{X}, \mathcal{U}$ are the state and control admissible sets, and
- $\mathcal{K}$ represents the contact constraints.

Note that $\boldsymbol{\lambda} = \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{u})$ denotes the contact force, and is dependent on the state and control.

Let's start by understanding the concept behind crocoddyl design.

A.1.2. Action Models

In crocoddyl, an action model combines dynamics and cost models. Each node, in our optimal control problem, is described through an action model. Every time that we want describe a problem, we need to provide ways of computing the dynamics, cost functions and their derivatives. All these is described inside the action model.

To understand the mathematical aspects behind an action model, let's first get a locally linearize version of our optimal control problem as:

$$\boldsymbol{X}^*(\boldsymbol{x}_0),\, \boldsymbol{U}^*(\boldsymbol{x}_0) = \arg\min_{\boldsymbol{X},\boldsymbol{U}} = cost_T(\delta\boldsymbol{x}_N) + \sum_{k=1}^{N} cost_t(\delta\boldsymbol{x}_k, \delta\boldsymbol{u}_k)$$

subject to

$$dynamics(\delta\boldsymbol{x}_{k+1}, \delta\boldsymbol{x}_k, \delta\boldsymbol{u}_k) = \boldsymbol{0},$$

where

$$cost_T(\delta\boldsymbol{x}_k) = \frac{1}{2}\begin{bmatrix}1\\ \delta\boldsymbol{x}_k\end{bmatrix}^\top \begin{bmatrix}0 & \boldsymbol{l}_{\boldsymbol{x}k}^\top\\ \boldsymbol{l}_{\boldsymbol{x}k} & \boldsymbol{l}_{\boldsymbol{x}\boldsymbol{x}k}\end{bmatrix}\begin{bmatrix}1\\ \delta\boldsymbol{x}_k\end{bmatrix},$$

$$cost_t(\delta\boldsymbol{x}_k, \delta\boldsymbol{u}_k) = \frac{1}{2}\begin{bmatrix}1\\ \delta\boldsymbol{x}_k\\ \delta\boldsymbol{u}_k\end{bmatrix}^\top \begin{bmatrix}0 & \boldsymbol{l}_{\boldsymbol{x}k}^\top & \boldsymbol{l}_{\boldsymbol{u}k}^\top\\ \boldsymbol{l}_{\boldsymbol{x}k} & \boldsymbol{l}_{\boldsymbol{x}\boldsymbol{x}k} & \boldsymbol{l}_{\boldsymbol{u}\boldsymbol{x}k}^\top\\ \boldsymbol{l}_{\boldsymbol{u}k} & \boldsymbol{l}_{\boldsymbol{u}\boldsymbol{x}k} & \boldsymbol{l}_{\boldsymbol{u}\boldsymbol{u}k}\end{bmatrix}\begin{bmatrix}1\\ \delta\boldsymbol{x}_k\\ \delta\boldsymbol{u}_k\end{bmatrix}$$

$$dynamics(\delta\boldsymbol{x}_{k+1}, \delta\boldsymbol{x}_k, \delta\boldsymbol{u}_k) = \delta\boldsymbol{x}_{k+1} - (\boldsymbol{f}_{\boldsymbol{x}k}\delta\boldsymbol{x}_k + \boldsymbol{f}_{\boldsymbol{u}k}\delta\boldsymbol{u}_k)$$

Notes

- An action model describes the dynamics and cost functions for a node in our optimal control problem.
- Action models lie in the discrete time space.
- For debugging and prototyping, we have also implemented NumDiff abstractions. These computations depend only in the defining of the dynamics equation and cost functions. However to asses efficiency, crocoddyl uses analytical derivatives computed from Pinocchio.

Differential and Integrated Action Models

It's often convenient to implement action models in continuous time. In crocoddyl, this continuous-time action models are called Differential Action Model (DAM). And together with predefined Integrated Action Models (IAM), it possible to retrieve the time-discrete action model needed by the solver. At the moment, we have the following integration rules:

- simpletic Euler and
- Runge-Kutta 4.

Add On from Introduction.jpnb

Optimal control solvers often need to compute a quadratic approximation of the action model (as previously described); this provides a search direction (computeDirection). Then it's needed to try the step along this direction (tryStep).

Typically calc and calcDiff do the precomputations that are required before computeDirection and tryStep respectively (inside the solver). These functions update the information of:

- **calc**: update the next state and its cost value

$$\delta\dot{\boldsymbol{x}}_{k+1} = \boldsymbol{f}(\delta\boldsymbol{x}_k, \boldsymbol{u}_k)$$

- **calcDiff**: update the derivatives of the dynamics and cost (quadratic approximation)

$$\boldsymbol{f_x}, \boldsymbol{f_u} \quad (dynamics)$$
$$\boldsymbol{l_x}, \boldsymbol{l_u}, \boldsymbol{l_{xx}}, \boldsymbol{l_{ux}}, \boldsymbol{l_{uu}} \quad (cost)$$

A.1.3. State and its Integrate and Difference Rules

General speaking, the system's state can lie in a manifold M where the state rate of change lies in its tangent space $T_{\mathbf{x}}M$. There are few operators that needs to be defined for different routines inside our solvers:

$$\boldsymbol{x}_{k+1} = integrate(\boldsymbol{x}_k, \delta\boldsymbol{x}_k) = \boldsymbol{x}_k \oplus \delta\boldsymbol{x}_k$$

$$\delta\boldsymbol{x}_k = difference(\boldsymbol{x}_{k+1}, \boldsymbol{x}_k) = \boldsymbol{x}_{k+1} \ominus \boldsymbol{x}_k$$

where $\mathbf{x} \in M$ and $\delta\mathbf{x} \in T_{\mathbf{x}}M$. And we also need to defined the Jacobians of these operators with respect to the first and second arguments:

$$\frac{\partial \boldsymbol{x} \oplus \delta\boldsymbol{x}}{\partial \boldsymbol{x}}, \frac{\partial \boldsymbol{x} \oplus \delta\boldsymbol{x}}{\partial \delta\boldsymbol{x}} = Jintegrante(\boldsymbol{x}, \delta\boldsymbol{x})$$

$$\frac{\partial \boldsymbol{x}_2 \ominus \boldsymbol{x}_2}{\partial \boldsymbol{x}_1}, \frac{\partial \boldsymbol{x}_2 \ominus \boldsymbol{x}_1}{\partial \boldsymbol{x}_1} = Jdifference(\boldsymbol{x}_2, \boldsymbol{x}_1)$$

For instance, a state that lies in the Euclidean space will the typical operators:

$$integrate(\boldsymbol{x}, \delta\boldsymbol{x}) = \boldsymbol{x} + \delta\boldsymbol{x}$$

$$difference(\boldsymbol{x}_2, \boldsymbol{x}_1) = \boldsymbol{x}_2 - \boldsymbol{x}_1$$
$$Jintegrate(\cdot,\cdot) = Jdifference(\cdot,\cdot) = \boldsymbol{I}$$

All these functions are encapsulate inside the State class. For Pinocchio models, we have implemented the StateMultibody class which can be used for any robot model.

A.2. Crocoddyl Wiki: Differential Action Model for Floating in Contact Systems (DAMFIC)

A.2.1. System Dynamics

As you might know, a differential action model describes the systems dynamics and cost function in continuous-time. For multi-contact locomotion, we account for the rigid contact by applying the Gauss principle over holonomic constraints in a set of predefined contact placements, i.e.:

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

This is equality-constrained quadratic problem with an analytical solution of the form:

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

in which

$$(\dot{\boldsymbol{v}}, \boldsymbol{\lambda}) \in (\boldsymbol{R}^{nv}, \boldsymbol{R}^{nf})$$

are the primal and dual solutions,

$$\boldsymbol{M} \in \boldsymbol{R}^{nv \times nv}$$

is formally the metric tensor over the configuration manifold $\boldsymbol{q} \in \boldsymbol{R}^{nq}$,

$$\boldsymbol{J}_c = \begin{bmatrix} \boldsymbol{J}_{c_1} & \cdots & \boldsymbol{J}_{c_f} \end{bmatrix} \in \boldsymbol{R}^{nf \times nv}$$

is a stack of f contact Jacobians, $\boldsymbol{\tau}_b = \boldsymbol{S}\boldsymbol{\tau} - \boldsymbol{b} \in \boldsymbol{R}^{nv}$ is the force-bias vector that accounts for the control $\boldsymbol{\tau} \in \boldsymbol{R}^{nu}$, the Coriolis and gravitational effects $\boldsymbol{b}$, and $\boldsymbol{S}$ is the selection matrix of the actuated joint coordinates, and nq, nv, nu and nf are the number of coordinates used to describe the configuration manifold, its tangent-space dimension, control commands and contact forces, respectively.

And this equality-constrained forward dynamics can be formulated using state space representation, i.e.:

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u})$$

where $\boldsymbol{x} = (\boldsymbol{q}, \boldsymbol{v}) \in \boldsymbol{R}^{nq+nv}$ and $\boldsymbol{u} = \boldsymbol{\tau} \in \boldsymbol{R}^{nu}$ are the state and control vectors, respectively. Note that $\dot{\boldsymbol{x}}$ lies in the tangent-space of $\boldsymbol{x}$, and their dimension are not the same.

A.2.2. Add On from Introduction.jpnb

A.2.3. Solving the Optimal Control Problem

Our optimal control solver interacts with a defined ShootingProblem. A **shooting problem** represents a **stack of action models** in which an action model defines a specific node along the OC problem.

First we need to create an action model from DifferentialFwdDynamics. We use it for building terminal and running action models. In this example, we employ an simpletic Euler integration rule.

Next we define the set of cost functions for this problem. One could formulate

- Running costs (related to individual states)
- Terminal costs (related to the final state)

in order to penalize, for example, the state error, control error, or end-effector pose error.

Onces we have defined our shooting problem, we create a DDP solver object and pass some callback functions for analysing its performance.

Application to Bipedal Walking

In crocoddyl, we can describe the multi-contact dynamics through holonomic constraints for the support legs. From the Gauss principle, we have derived the model as:

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

.

This DAM is defined in "DifferentialActionModelFloatingInContact" class. Given a predefined contact sequence and timings, we build per each phase a specific multi-contact dynamics. Indeed we need to describe **multi-phase optimal control problem**. One can formulate the multi-contact optimal control problem (MCOP) as follows:

$$\boldsymbol{X}^*, \boldsymbol{U}^* = \begin{Bmatrix} \boldsymbol{x}_0^*, \cdots, \boldsymbol{x}_N^* \\ \boldsymbol{u}_0^*, \cdots, \boldsymbol{u}_N^* \end{Bmatrix} = \arg\min_{\boldsymbol{X},\boldsymbol{U}} \sum_{p=0}^{P} \sum_{k=1}^{N(p)} \int_{t_k}^{t_k+\Delta t} l_p(\boldsymbol{x}, \boldsymbol{u}) dt$$

subject to

$$\dot{\boldsymbol{x}} = \boldsymbol{f}_p(\boldsymbol{x}, \boldsymbol{u}), \text{for } t \in [\tau_p, \tau_{p+1}]$$

$$\boldsymbol{g}(\boldsymbol{v}^{p+1}, \boldsymbol{v}^p) = \boldsymbol{0}$$

$$\boldsymbol{x} \in \mathcal{X}_p, \boldsymbol{u} \in \mathcal{U}_p, \boldsymbol{\lambda} \in \mathcal{K}_p.$$

where $\boldsymbol{g}(\cdot,\cdot,\cdot)$ describes the contact dynamics, and they represents terminal constraints in each walking phase. In this example we use the following **impact model**:

$$\boldsymbol{M}(\boldsymbol{v}_{next} - \boldsymbol{v}) = \boldsymbol{J}_{impulse}^{T}$$

$$\boldsymbol{J}_{impulse}\boldsymbol{v}_{next} = \boldsymbol{0}$$

$$\boldsymbol{J}_{c}\boldsymbol{v}_{next} = \boldsymbol{J}_{c}\boldsymbol{v}$$

Bibliography

[1] Marc H Raibert. *Legged robots that balance*. MIT press, 1986.

[2] Pierre-Brice Wieber, Russ Tedrake, and Scott Kuindersma. Modeling and control of legged robots. In *Springer handbook of robotics*, pages 1203–1234. Springer, 2016.

[3] Paul Fitzpatrick, Kensuke Harada, Charles C Kemp, Yoshio Matsumoto, Kazuhito Yokoi, and Eiichi Yoshida. Humanoids. In *Springer handbook of robotics*, pages 1789–1818. Springer, 2016.

[4] Frank Kirchner, Elsa Kirchner, Manuel Meder, and Georg von Wichert. Transfit: Flexible interaction for infrastructures establishment by means of teleoperation and direct collaboration; transfer into industry 4.0. https://robotik.dfki-bremen.de/en/research/projects/transfit.html. Accessed: 2020-09-02.

[5] Marco Hutter, Christian Gehring, Michael Bloesch, Mark A Hoepflinger, C David Remy, and Roland Siegwart. Starleth: A compliant quadrupedal robot for fast, efficient, and versatile locomotion. In *Adaptive Mobile Robotics*, pages 483–490. World Scientific, 2012.

[6] Jerry E Pratt and Benjamin T Krupp. Series elastic actuators for legged robots. In *Unmanned Ground Vehicle Technology Vi*, volume 5422, pages 135–144. International Society for Optics and Photonics, 2004.

[7] Steve Collins, Andy Ruina, Russ Tedrake, and Martijn Wisse. Efficient bipedal robots based on passive-dynamic walkers. *Science*, 307(5712):1082–1085, 2005.

[8] Sami Haddadin, Felix Huber, and Alin Albu-Schäffer. Optimal control for exploiting the natural dynamics of variable stiffness robots. In *2012 IEEE International Conference on Robotics and Automation*, pages 3347–3354. IEEE, 2012.

[9] Jerry E Pratt. Exploiting inherent robustness and natural dynamics in the control of bipedal walking robots. Technical report, Massachusetts Institute of Technology Cambridge Departmend of Electrical Engineering, 2000.

[10] Eric R Westervelt, Jessy W Grizzle, Christine Chevallereau, Jun Ho Choi, and Benjamin Morris. *Feedback control of dynamic bipedal robot locomotion*. CRC press, 2018.

[11] Justin Carpentier, Andrea Del Prete, Steve Tonneau, Thomas Flayols, Florent Forget, Alexis Mifsud, Kevin Giraud, Dinesh Atchuthan, Pierre Fernbach, Rohan Budhiraja, et al. Multi-contact locomotion of legged robots in complex environments–the loco3d project. In *RSS Workshop on Challenges in Dynamic Legged Locomotion*, page 3p, 2017.

[12] Kevin Giraud, Pierre Fernbach, Gabriele Buondonno, Carlos Mastalli, Olivier Stasse, et al. Motion planning with multi-contact and visual servoing on humanoid robots. 2020.

[13] Shuuji Kajita, Fumio Kanehiro, Kenji Kaneko, Kiyoshi Fujiwara, Kensuke Harada, Kazuhito Yokoi, and Hirohisa Hirukawa. Biped walking pattern generation by using preview control of zero-moment point. In *2003 IEEE International Conference on Robotics and Automation (Cat. No. 03CH37422)*, volume 2, pages 1620–1626. IEEE, 2003.

[14] Mrinal Kalakrishnan, Jonas Buchli, Peter Pastor, Michael Mistry, and Stefan Schaal. Fast, robust quadruped locomotion over challenging terrain. In *2010 IEEE International Conference on Robotics and Automation*, pages 2665–2670. IEEE, 2010.

[15] Alexander W Winkler, Carlos Mastalli, Ioannis Havoutis, Michele Focchi, Darwin G Caldwell, and Claudio Semini. Planning and execution of dynamic whole-body locomotion for a hydraulic quadruped on challenging terrain. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5148–5154. IEEE, 2015.

[16] C Dario Bellicoso, Fabian Jenelten, Péter Fankhauser, Christian Gehring, Jemin Hwangbo, and Marco Hutter. Dynamic locomotion and whole-body control for quadrupedal robots. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3359–3365. IEEE, 2017.

[17] Bernard Espiau, François Chaumette, and Patrick Rives. A new approach to visual servoing in robotics. *ieee Transactions on Robotics and Automation*, 8 (3):313–326, 1992.

[18] Oussama Khatib. A unified approach for motion and force control of robot manipulators: The operational space formulation. *IEEE Journal on Robotics and Automation*, 3(1):43–53, 1987.

[19] Alexander W Winkler. *Optimization-based motion planning for legged robots*. PhD thesis, ETH Zurich, 2018.

[20] Rohan Budhiraja, Justin Carpentier, Carlos Mastalli, and Nicolas Mansard. Differential dynamic programming for multi-phase rigid contact dynamics. In *2018 IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids)*, pages 1–9. IEEE, 2018.

[21] John T Betts. Survey of numerical methods for trajectory optimization. *Journal of guidance, control, and dynamics*, 21(2):193–207, 1998.

[22] Matthew P Kelly. Transcription methods for trajectory optimization: a beginners tutorial. *arXiv preprint arXiv:1707.00284*, 2017.

[23] Yuval Tassa, Nicolas Mansard, and Emo Todorov. Control-limited differential dynamic programming. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1168–1175. IEEE, 2014.

[24] David E Orin, Ambarish Goswami, and Sung-Hee Lee. Centroidal dynamics of a humanoid robot. *Autonomous robots*, 35(2-3):161–176, 2013.

[25] Hongkai Dai, Andrés Valenzuela, and Russ Tedrake. Whole-body motion planning with centroidal dynamics and full kinematics. In *2014 IEEE-RAS International Conference on Humanoid Robots*, pages 295–302. IEEE, 2014.

[26] Justin Carpentier, Steve Tonneau, Maximilien Naveau, Olivier Stasse, and Nicolas Mansard. A versatile and efficient pattern generator for generalized legged locomotion. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3555–3561. IEEE, 2016.

[27] Alexander Herzog, Nicholas Rotella, Stefan Schaal, and Ludovic Righetti. Trajectory generation for multi-contact momentum control. In *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, pages 874–880. IEEE, 2015.

[28] Carlos Mastalli, Michele Focchi, Ioannis Havoutis, Andreea Radulescu, Sylvain Calinon, Jonas Buchli, Darwin G Caldwell, and Claudio Semini. Trajectory and foothold optimization using low-dimensional models for rough terrain locomotion. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1096–1103. IEEE, 2017.

[29] Alexander W Winkler, C Dario Bellicoso, Marco Hutter, and Jonas Buchli. Gait and trajectory optimization for legged systems through phase-based end-effector parameterization. *IEEE Robotics and Automation Letters*, 3(3):1560–1567, 2018.

[30] Bernardo Aceituno-Cabezas, Carlos Mastalli, Hongkai Dai, Michele Focchi, Andreea Radulescu, Darwin G Caldwell, José Cappelletto, Juan C Grieco, Gerardo Fernández-López, and Claudio Semini. Simultaneous contact, gait, and motion planning for robust multilegged locomotion via mixed-integer convex optimization. *IEEE Robotics and Automation Letters*, 3(3):2531–2538, 2017.

[31] Layale Saab, Oscar E Ramos, François Keith, Nicolas Mansard, Philippe Soueres, and Jean-Yves Fourquet. Dynamic whole-body motion generation under rigid contacts and other unilateral constraints. *IEEE Transactions on Robotics*, 29(2):346–362, 2013.

[32] Alexander Herzog, Nicholas Rotella, Sean Mason, Felix Grimminger, Stefan Schaal, and Ludovic Righetti. Momentum control with hierarchical inverse dynamics on a torque-controlled humanoid. *Autonomous Robots*, 40(3):473–491, 2016.

[33] Joris Vaillant, Abderrahmane Kheddar, Hervé Audren, François Keith, Stanislas Brossette, Adrien Escande, Karim Bouyarmane, Kenji Kaneko, Mitsuharu Morisawa, Pierre Gergondet, et al. Multi-contact vertical ladder climbing with an hrp-2 humanoid. *Autonomous Robots*, 40(3):561–580, 2016.

[34] Carlos Mastalli, Rohan Budhiraja, Wolfgang Merkt, Guilhem Saurel, Bilal Hammoud, Maximilien Naveau, Justin Carpentier, Ludovic Righetti, Sethu Vijayakumar, and Nicolas Mansard. Crocoddyl: An Efficient and Versatile Framework for Multi-Contact Optimal Control. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

[35] H Peters, P Kampmann, and M Simnofske. Konstruktion eines zweibeinigen humanoiden roboters. *Proceedings of the 2. VDI Fachkonferenz Humanoide Roboter, VDI Fachkonferenz Humanoide Roboter*, 2017.

[36] Shivesh Kumar, Hendrik Wöhrle, José de Gea Fernández, Andreas Müller, and Frank Kirchner. A survey on modularity and distributivity in series-parallel hybrid robots. *Mechatronics*, 68:102367, 2020.

[37] Shivesh Kumar. *Modular and Analytical Methods for Solving Kinematics and Dynamics of Series-Parallel Hybrid Robots*. PhD thesis, Universität Bremen, 2019.

[38] M Vukobratović, Branislav Borovac, and Veljko Potkonjak. Towards a unified understanding of basic notions and terms in humanoid robotics. *Robotica*, 25 (1):87–101, 2007.

[39] MHP Dekker. Zero-moment point method for stable biped walking. *Eindhoven University of Technology*, 2009.

[40] Stephane Caron. Legged locomotion teaching material. https://scaron.info/category/teaching.html. Accessed: 2020-06-22.

[41] Friedrich Pfeiffer and Christoph Glocker. *Multibody dynamics with unilateral contacts*. John Wiley & Sons, 1996.

[42] Abhinandan Jain. *Robot and multibody dynamics: analysis and algorithms*. Springer Science & Business Media, 2010.

[43] Roy Featherstone. *Rigid body dynamics algorithms*. Springer, 2014.

[44] John YS Luh, Michael W Walker, and Richard PC Paul. On-line computational scheme for mechanical manipulators. 1980.

[45] John M Hollerbach. A recursive lagrangian formulation of maniputator dynamics and a comparative study of dynamics formulation complexity. *IEEE Transactions on Systems, Man, and Cybernetics*, 10(11):730–736, 1980.

[46] Elena Garcia, Joaquin Estremera, and Pablo Gonzalez-de Santos. A classification of stability margins for walking robots. *Robotica*, 20(6):595–606, 2002.

[47] Bruno Siciliano and Oussama Khatib. *Springer handbook of robotics*. Springer, 2016.

[48] Philippe Sardain and Guy Bessonnet. Forces acting on a biped robot. center of pressure-zero moment point. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 34(5):630–637, 2004.

[49] Miomir Vukobratović and J Stepanenko. On the stability of anthropomorphic systems. *Mathematical biosciences*, 15(1-2):1–37, 1972.

[50] Miomir Vukobratović and Branislav Borovac. Zero-moment pointâ€"thirty five years of its life. *International journal of humanoid robotics*, 1(01):157–173, 2004.

[51] David Mayne. A second-order gradient method for determining optimal trajectories of non-linear discrete-time systems. *International Journal of Control*, 3 (1):85–95, jan 1966. doi: 10.1080/00207176608921369.

[52] Yuval Tassa, Tom Erez, and Emanuel Todorov. Synthesis and stabilization of complex behaviors through online trajectory optimization. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4906–4913. IEEE, 2012.

[53] Richard Bellman. Dynamic programming. *Science*, 153(3731):34–37, 1966.

[54] Donald E Kirk. *Optimal control theory: an introduction*. Courier Corporation, 2004.

[55] Russ Tedrake. Underactuated robotics: Algorithms for walking, running, swimming, flying, and manipulation (course notes for mit 6.832). http://underactuated.mit.edu/. Accessed: 2020-06-01.

[56] Morton I Kamien and Nancy Lou Schwartz. *Dynamic optimization: the calculus of variations and optimal control in economics and management*. Courier Corporation, 2012.

[57] Firdaus E Udwadia and Robert E Kalaba. A new perspective on constrained motion. *Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences*, 439(1906):407–410, 1992.

[58] Stéphane Caron, Quang-Cuong Pham, and Yoshihiko Nakamura. Stability of surface contacts for humanoid robots: Closed-form formulae of the contact wrench cone for rectangular support areas. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5107–5112. IEEE, 2015.

[59] Imin Kao, Kevin M Lynch, and Joel W Burdick. Contact modeling and manipulation. In *Springer Handbook of Robotics*, pages 931–954. Springer, 2016.

[60] Jonathan Richard Shewchuk et al. An introduction to the conjugate gradient method without the agonizing pain, 1994.

[61] Rohan Budhiraja Carlos Mastalli, Nicolas Mansard, et al. Crocoddyl: a fast and flexible optimal control library for robot control under contact sequence. https://github.com/loco-3d/crocoddyl/wikis/home, 2020.