HAL
open science

Multi-body Locomotion : Problem Structure and Efficient Resolution

Rohan Budhiraja

To cite this version:

Rohan Budhiraja. Multi-body Locomotion : Problem Structure and Efficient Resolution. Automatic. INSA de Toulouse, 2019. English. NNT : 2019ISAT0036 . tel-02880584v2

HAL Id: tel-02880584

https://theses.hal.science/tel-02880584v2

Submitted on 16 Jul 2020

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ FÉDÉRALE TOULOUSE MIDI-PYRÉNÉES

Délivré par :

l'Institut National des Sciences Appliquées de Toulouse (INSA de Toulouse)

Présentée et soutenue le *29/11/2019* **par :**

Rohan BUDHIRAJA

Multi-body Locomotion: Problem Structure and Efficient Resolution

---

JURY

Adrien Escande	Chargé de Recherche	Président du Jury
Nicolas Mansard	Directeur de Recherche	Directeur de thése
Olivier Stasse	Directeur de Recherche	Directeur de thése
Katja Mombaur	Professeur d'université	Membre du Jury
Serena Ivaldi	Chargé de Recherche	Membre du Jury
Jerome Bolte	Professeur d'université	Membre du Jury

École doctorale et spécialité :
EDSYS : Robotique 4200046

Unité de Recherche :
Laboratoire d'analyse et d'architecture des systèmes

Directeur(s) de Thèse :
Nicolas MANSARD et *Olivier STASSE*

Rapporteurs :
Katja Mombaur et *Adrien Escande*

Multi-body Locomotion:
Problem Structure and Efficient Resolution

Rohan Budhiraja

November 29, 2019
Version: 2.0

Phd Thesis

Multi-body Locomotion: Problem Structure and Efficient Resolution

Rohan Budhiraja

Reviewer Katja Mombaur
ZITI - Institute of Computer Engineering
Heidelberg University, Heidelberg, Germany

Reviewer Adrien Escande
Japanese-French Joint Robotic Laboratory (JRL)
CNRS/AIST, Tsukuba, Japan

Examiner Jerome Bolte
Toulouse School of Economics
Université Toulouse 1 Capitole, Toulouse, France

Examiner Serena Ivaldi
Project-Team LARSEN
INRIA Nancy Grand-Est, France

Supervisors Nicolas Mansard and Olivier Stasse
Laboratoire d'analyse et d'architecture des systèmes
LAAS-CNRS, Toulouse, France

November 29, 2019

Rohan Budhiraja
Multi-body Locomotion: Problem Structure and Efficient Resolution
Phd Thesis, November 29, 2019
President of Jury: Adrien Escande
Reviewers: Katja Mombaur and Adrien Escande
Examiners: Jerome Bolte and Serena Ivaldi
Supervisors: Nicolas Mansard and Olivier Stasse

Laboratoire d'analyse et d'architecture des systèmes
Gepetto Team
7 Avenue du Colonel Roche
31400 and Toulouse

Abstract

> *I have passed through fire and deep water, since we parted. I have forgotten much that I thought I knew, and learned again much that I had forgotten. I can see many things far off, but many things that are close at hand I cannot see.*
>
> — **Gandalf, to the Three Hunters in Fangorn**
> (J.R.R. Tolkien, LOTR-The Two Towers)

The locomotion problem in robotics is usually written as a large optimization problem. However, the full problem is marred by many undesirable properties, such as high dimensionality, non-linearity, non convex constraints etc., which make it difficult to find a direct solution. A common approach then is to simplify the numerical optimization by solving for reduced models instead (inverted pendulum, capture points, centroidal). While these reduced models are simpler to solve than the full problem, they lack generality, are suboptimal, and hence are difficult to trust. Nevertheless, a typical approach in robotics relies on a reduced template model optimization, followed by an instantaneous linearization (such as Inverse Kinematics/Inverse Dynamics) for the whole-body, with possible variations in both (Kajita et al., 2003; Herzog et al., 2016a). While some research has been conducted to solve the full problem in a brute-force way, and without utilizing the structure of the problem (Heess et al., 2017; Rajeswaran et al., 2017), it is not mature or applicable on robots.

In this thesis, we explore an approach that tries to solve the general/holistic locomotion problem, while exploiting fully the problem structure to obtain an efficient and realistic solution. We aim to provide the guarantees offered to us by the brute-force approaches to the full problem, but with an efficient resolution provided by proper use of the problem structure. We support our proposition by dynamic proof-of-concept motions, and rely on three core contributions.

The first contribution of this thesis is the introduction of a rigorous mathematical framework, based on the Alternating Direction Method of Multipliers, to enforce

the consensus between the centroidal state dynamics at reduced and whole-body level. We propose an exact splitting of the whole-body optimal control problem between the centroidal dynamics (under-actuation) and the manipulator dynamics (full actuation), corresponding to a re-arrangement of the equations already stated in state-of-the-art. We describe with details how alternating descent is a good solution to implement an effective locomotion solver.

Such a split between the whole-body and centroidal components of locomotion puts the onus of producing efficient angular momentum trajectories on the whole-body solver. The second contribution of this thesis is to introduce a local optimal control framework, based on popularly known method called Differential Dynamic Programming, which deals implicitly with the contact constraints of the locomotion problem. By defining the contact constraints inside the dynamics of the problem, we are able to exploit fully the sparsity resolution capability of DDP, and optimize the solution over a horizon. Our efforts with using DDP for contact-constrained whole-body optimization resulted in the piece of software called "Crocoddyl". It is a python-based solver for shooting problems, and under the aegis of this software we implement our contact-constrained locomotion problem, and the corresponding DDP resolution. We make a distinction between the structure of the optimization problem, and its resolution, and use efficient memory allocations in order to speed up problem resolution in python. Similar effort has been made to unit-test the software, and to use a clear API for problem description.

One way of managing the constraints between the centroidal and whole-body solvers is to explicitly learn and pre-code the constraint when solving one problem. With simple kinematic values such as the "Center of Mass of the system given a foot position", such an information can be easily transferred to the centroidal solver by means of proxy constraints. The third contribution of this thesis is to the centroidal dynamics optimization problem. We learn the feasibility of the center of mass solution of the centroidal optimization, with respect to the whole-body kinematics, by means of an occupancy measure. Maximization of this occupancy measure ensures that the whole-body solver has a large solution space from which it can track the centroidal references.

Résumé

> *Say not, "I have found the truth," but rather, "I have found a truth.". Say not, "I have found the path of the soul." Say rather, "I have met the soul walking upon my path.". For the soul walks upon all paths. The soul walks not upon a line, neither does it grow like a reed. The soul unfolds itself, like a lotus of countless petals.*
>
> — **Kahlil Gibran**
> (The Prophet)

En robotique, un problème de locomotion est généralement transcrit comme un grand problème d'optimisation. Cependant, le problème complet est caractérisé par de nombreuses propriétés indésirables, telles qu'une grande dimensionnalité, de la non-linéarité, des contraintes non convexes, etc., qui rendent difficile la recherche de solution par une méthode directe. Une approche courante consiste alors à simplifier l'optimisation numérique en résolvant plutôt des modèles réduits (pendule inversé, points de capture, centroïdale). Bien que ces modèles réduits soient plus simples à résoudre comparé au problème complet, ils manquent de généralité, sont sous-optimaux et sont donc peu fiables. Néanmoins, une approche typique en robotique repose sur un modèle réduit d'optimisation, suivie d'une linéarisation instantanée du corps-complet (comme l'Inverse Cinématique / l'Inverse Dynamique), avec de possibles variations dans les deux étapes (Kajita et al., 2003; Herzog et al., 2016a). Bien que certaines recherches aient été menées pour résoudre le problème complet en se basant sur la méthode brute-force, sans utiliser la structure du problème (Heess et al., 2017; Rajeswaran et al., 2017), ces méthodes ne sont pas matures ou applicables aux robots.

Dans cette thèse, nous explorons une approche qui tente de résoudre le problème de locomotion général / holistique, tout en exploitant pleinement la structure du problème pour obtenir une solution efficace et réaliste. Nous visons à offrir les garanties que nous offrent les approches par brute-force du problème complet, avec une résolution efficace basé sur une utilisation appropriée de la structure du prob-

lème. Nous appuyons notre proposition par des preuves de concept de mouvements dynamiques et proposons trois contributions essentielles.

La première contribution de cette thèse est l'introduction d'une formulation mathématique rigoureuse, basé sur la méthode de direction alternée des multiplicateurs, afin d'imposer un consensus entre la dynamique de l'état centroïdale au niveau réduit et au niveau du corps-complet. Nous proposons une division exacte du problème de contrôle optimal corps-complet entre la dynamique centroïdale (sous-actionnée) et la dynamique du manipulateur (actionnement complet), correspondant à un réarrangement des équations déjà énoncées dans l'état de l'art. Nous décrivons en détail comment la descente alternée est une solution appropriée pour l'implémentation d'un solveur de locomotion efficace.

Une telle division de la locomotion entre les composants: "corps-complet et centroïdale" oblige le solveur corps-complet à produire des trajectoires de moment angulaire efficaces. La deuxième contribution de cette thèse est l'introduction d'un cadre de contrôle optimal local, basé sur une méthode connue sous le nom de Programmation Dynamique Différentielle (DDP), qui traite implicitement les contraintes de contact du problème de locomotion. En définissant les contraintes de contact à l'intérieur de la dynamique du problème, nous sommes en mesure d'exploiter pleinement la capacité de résolution clairsemée du DDP et d'optimiser la solution sur un horizon. Nos efforts afin d'utiliser le DDP pour l'optimisation corps-complet contraint par contact ont abouti à un logiciel nommé "Crocoddyl". C'est un solveur basé sur python pour des problèmes à tirs multiples, sous l'égide de ce logiciel, nous implémentons notre problème de locomotion contraint aux contacts, ainsi que la résolution DDP correspondante. Une distinction entre la structure du problème d'optimisation et sa résolution est faite, et des allocations de mémoire efficaces sont utilisés afin d'accélérer la résolution du problème en python. Des efforts similaires ont été faits pour les tests unitaires du logiciel, ainsi que l'utilisation d'une API claire pour la description du problème.

Une façon de gérer les contraintes entre les solveurs "centroïdale et corps-complet" est d'apprendre explicitement et de pré-coder la contrainte lors de la résolution du problème. Avec des valeurs cinématiques simples telles que le "Centre de masse du système avec une position du pied", une telle information peut être facilement transférée au solveur centroïdale au moyen de contraintes proxy. La troisième contribution de cette thèse s'intègre dans la problématique d'optimisation de la dynamique centroïdale. Nous apprenons la faisabilité de la solution du centre de masse de l'optimisation centroïdale, par rapport à la cinématique du corps-complet, au moyen d'une mesure d'occupation. La maximisation de cette mesure d'occupation garantit que le solveur du corps-complet dispose d'un grand espace de solutions à partir duquel il peut suivre les références centroïdales.

Preface

> *Whenever we propose a solution to a problem, we ought to try as hard as we can to overthrow our solution, rather than defend it. Few of us, unfortunately, practice this precept; but other people, fortunately, will supply the criticism for us if we fail to supply it ourselves.*
>
> — **Karl Popper**
> (The Logic of Scientific Discovery)

Robotics has evolved at a much faster pace and orderly manner than biological evolution. Unlike the slow process of natural selection and genetic mutation, roboticists have relied on explicitly defined and continuously improving dynamic models to make robots walk within a few years of their conception. Moreover, roboticists require performance guarantees from their models, in order to make sure that the robots are not damaged. While humans have a general intelligence and evolutionary reflexes, robots have relied on control systems that could capture the essence of the problem at hand and provide the necessary stabilizing actuation.

Thus, there has been a great deal of reliance put on the ability of these dynamic models to predict and generate the controls required to move the robot. These models are often guided by the description of the problem at hand, and the resources available for finding a fast solution. For example, the necessary features of the coplanar locomotion problem are captured inside an inverted pendulum, and as a result, researchers were able to make their robots walk with such a model. Even though such an approach was not generic enough to make the robot walk on the stairs at that time, it achieved the desired objective and was a major scientific milestone.

Progressively, over the years, researchers have improved their dynamic models, and tried to capture more and more of the information from the locomotion problem

inside them. As a result, we have moved from inverted pendulum (Kajita et al., 2001) to centroidal dynamics (Orin et al., 2013), from Inverse Kinematics (e.g. in Tevatia and Schaal, 2000) to Inverse Dynamics (e.g. in Khatib et al., 2004) and then to Trajectory Optimization (e.g. in Kalakrishnan et al., 2011; Koenemann et al., 2015). Moreover, there are parallel efforts to solve the discrete contact problem by using either Lagrangian (e.g. in Mombaur et al., 2005a; Posa et al., 2014) or Momentum dynamics (e.g. in Stewart and Trinkle, 2000; Koolen et al., 2016). All these efforts have been made for one reason only, in order to provide as much information as we can about the locomotion problem inside our models, and still be able to get a solution without over-burdening the solvers. As time has progressed, our processors have become faster, and the solutions have become easier. However, the locomotion problem of going from A to B remains an infinite dimensional problem and approximations on some level are still necessary when attempting a solution. The next obvious question would be to decide on which level we should approximate. The fidelity of our models to the locomotion problem remains paramount in the attempt to find a solution.

In this thesis, we will argue that while our solvers might not be equipped to handle the full state yet, we should still be able to approximate at the level of the contact-constrained dynamics. We explain how a simple scheme can help us capture the essentials of the problem inside one single model, without losing on the solver efficiency. This way, our method uses the maximum available information on the contact-constrained locomotion problem when searching for a solution. In the following chapters, you would read our work in this direction, work I did along with my supervisors and colleagues. Additional contributions in the topic of contact-constrained locomotion (in the subspace of the original problem) help to further address the issue of efficiency.

Someone once told me that doing research is like planting a tree in the Amazon rain forest. Your contribution may be really small, but the forest still grows. I hope the tree we plant in this thesis would provide shelter to those who come searching later in this direction.

Rohan Budhiraja

Acknowledgement

> *Why couldnt he stop talking and let them drink his health?*
>
> — **Shirefolk, about Bilbo**
> (Lord of the Rings, The Fellowship of the Ring)

```
from your_abilities import internet_searches

def oscar_speech(category, year, winner):
    return internet_searches(keywords=["oscar",
                                       category,
                                       str(year),
                                       winner,
                                       "transcript"])


print oscar_speech("Best Supporting Actor",
                   1991,
                   "Joe Pesci")
```

Contents

1 Multi-body Locomotion **1**

1.1 Contact and Whole-body Optimization 2

1.2 Decoupled Approach to Motion Generation 5

1.2.1 Contact Planning 6

1.2.2 Reduced Problems 8

1.2.3 Instantaneous tracking of Reduced Patterns 10

1.2.4 Whole-body Trajectory Optimization 12

1.3 The Locomotion Problem Structure 15

1.3.1 Natural splitting of the robot dynamics 15

1.3.2 Coupling Constraints 16

1.3.3 Managing the Coupling Constraints 17

1.3.4 The Global Optimal Control Problem for locomotion 18

1.4 The Loco3D Project 20

1.5 Thesis Structure 22

1.5.1 Chapter Organization 22

1.5.2 Associated Publications 23

2 Learning Feasibility Constraints for Multicontact Locomotion of Legged Robots 25

2.1 Introduction 25

2.1.1 Kinematic Feasibility as a Proxy Constraint 27

2.1.2 Contact model 28

2.2 Feasibility of the centroidal problem 29

2.2.1 Mathematical representation of feasibility constraints 30

2.2.2 Proxy as occupancy measure 30

2.2.3 Maximizing the occupancy measure 31

2.3 Learning the CoM reachability proxy 32

2.3.1 Probabilistic model 32

2.3.2 Kernel density estimation by CoM sampling 33

2.3.3 Reduction of dimension 34

2.3.4 Summary of the learning procedure 34

2.3.5 Proposed optimal control formulation 34

2.4 Results 35

2.4.1 Illustration of the learning procedure 35

2.4.2 Experiments on HRP-2 and Talos robots 36
2.4.3 Experiments in simulation 39
2.5 Conclusion and Perspective 39

3 Differential Dynamic Programming for Multi-Phase Rigid Contact Dynamics 41
3.1 Introduction 41
3.1.1 On the importance of angular momentum 42
3.1.2 Multi-contact Motion Generation 44
3.2 Differential Dynamic Programming 46
3.2.1 Relaxation of Constraints 46
3.2.2 Understanding Bellman's Principal of Optimality 47
3.2.3 Backward Pass 48
3.2.4 Forward pass 49
3.2.5 Line search and regularization 50
3.2.6 DDP as an evolution of Newton Step 50
3.3 DDP with Constrained Robot Dynamics 51
3.3.1 Contact dynamics 51
3.3.2 Karush-Kuhn-Tucker (KKT) conditions 51
3.3.3 KKT-based Differential Dynamic Programming (DDP) algorithm 53
3.4 Results 56
3.4.1 Large stride on a flat ground 56
3.4.2 Attitude regulation through joint motion 57
3.5 Conclusion and Perspective 57

4 Crocoddyl: Contact Robot Control by Differential Dynamic Programming Library 61
4.1 Features of Crocoddyl 61
4.2 Problem Transcription 63
4.2.1 `Action Models`: Unification of Dynamics, Costs and Constraints 63
4.2.2 Non-Euclidean State Models 66
4.2.3 Shooting Problem 67
4.3 Problem Resolution 68
4.3.1 Multi-threading 68
4.3.2 Analytical Derivatives 68
4.3.3 Solvers 69
4.4 Results 71
4.4.1 Simulation Examples 71
4.4.2 Convergence 73
4.4.3 Computational Benchmarks 73
4.5 Conclusion and Perspective 75

5 Alternating Direction Method of Multipliers for locomotion 77

5.1 Introduction 77
5.1.1 Why should we alternate? 78
5.1.2 Outline of the chapter 79
5.2 Short overview of Alternating Direction Method of Multipliers (ADMM) 80
5.3 Rational for our alternative scheme 81
5.4 Application of ADMM for solving the locomotion problem 82
5.4.1 On the advantages of scaling the dual variables 83
5.4.2 The ADMM solver for locomotion 83
5.4.3 Initializing the dual variables 84
5.4.4 Key observations for whole-body solver 84
5.5 Experimental results 85
5.5.1 Locomotion Pipeline 85
5.5.2 Cost functions 88
5.5.3 Convergence analysis 89
5.6 Conclusion and Future Work 89

6 Generation of dynamic motions 91
6.1 Definition of the Motion Problem 91
6.2 Feasibility constraints for warm-starting centroidal solver 92
6.2.1 Centroidal Optimization without feasibility constraint 92
6.2.2 Learning and optimizing with feasibility constraints 94
6.3 Wholebody optimization using centroidal trajectories 95
6.4 Motion generation using ADMM 98

7 Conclusion 101
7.1 Contributions 101
7.1.1 Proxy constraint for kinematic feasibility 101
7.1.2 Whole-body solver for contact-constrained dynamics 101
7.1.3 Alternating to ensure feasibility and consensus 102
7.2 Perspective on Multi-body Locomotion 102
7.2.1 What about contacts? 102
7.2.2 How should we use the duals? 103
7.2.3 Could we go faster? 104

Bibliography 105

1 Multi-body Locomotion

> सत्य की संपूर्णता देती न दिखलाई किसी को,
> हम जिसे हैं देखते, वह सत्य का, बस, एक पहलू है।
> सत्य का प्रेमी भला तब किस भरोसे पर कहे यह
> मैं सही हूँ और सब जन झूठ हैं ?
> *(None realizes the completeness of truth,*
> *what we see is but one of its aspects.*
> *So how can a lover of truth say,*
> *that I am right and the world is wrong?)*
>
> — **Ramdhari Singh 'Dinkar'**
> (Naye Subhashit, Gandhi)

Locomotion is defined as the act of moving from one place to another. This thesis deals with the fundamentals of locomotion by making and breaking contact with the environment. Some examples of such locomotion are shown in Fig. 1.1.

Multiple approaches have been pursued with that objective, and all approaches make a compromise between optimality and ease of solution. Numerical optimization problems in general love convexity, and smoothness of the objective and constraints. The multi-body locomotion problem, when written in Lagrangian form, introduces a discontinuity in its dynamics every time a new contact is made, or an old contact is broken. This difficulty has spurred researchers to pursue avenues which either 1) replace the discontinuous problem with a continuous but more difficult problem; or, 2) remove the discontinuity from the dynamics and treat the piece-wise continuous problem instead

In this chapter, we review the main methods of locomotion. We will explore a simplified and representative history of the locomotion problem. As we will see, research into locomotion has branched again and again into different areas and different approaches, always with the aim of improving the solution. In this chapter, we will try to note all the branches that are around us, and we will try to justify the branch of research that we have chosen. We will then define mathematically the locomotion problem upon which we shall build our contributions, and we shall see and appreciate the full structure of the problem and how to best exploit the

(a) Micro Frog on a thumb

(b) Penguins on a beach

(c) Cheetah on a hunt

(d) Orangutan on ropes

(e) Michael Jackson dancing

(f) Asimo Robot walking

Fig. 1.1: Multi-body Locomotion. Examples of locomotion by making and breaking contacts. The gaits, kinematics, and the number of contacts differ, but the essential features of the movement remain the same.

structure of the problem. Finally, we will expose the organizations of this document, and explain how we deal with the various components of the locomotion problem in different chapters.

1.1 Contact and Whole-body Optimization

Each contact of a multi-body system with the environment corresponds to a unilateral constraint on the body. Optimizing the locomotion problem along with the contacts requires handling these unilateral contact constraints, and the discontinuities arising from it. As a result, the contact and whole-body optimization problem looks like this:

$$\underset{\underline{\mathbf{x}},\underline{\mathbf{u}},\underline{\lambda}}{\text{minimize}} \quad \text{Cost}(\underline{\mathbf{x}}, \underline{\mathbf{u}}, \underline{\lambda}) \tag{1.1a}$$

$$\text{subject to} \quad \text{Dynamics constraint} \tag{1.1b}$$

$$\text{Force unilaterality constraint} \tag{1.1c}$$

$$\text{Force Model is satisfied} \tag{1.1d}$$

where $\mathbf{x}$ is the state trajectory, $\mathbf{u}$ is the control trajectory, λ is the contact force trajectory.

Cost (1.1a)[1] is typically defined as a sum of objective functions, which are only dependent on the current state ($\mathbf{x}$) and control vectors ($\mathbf{u}$,λ). If we define l as the instantaneous objective function based on some efficiency criteria, this sum could be defined as a continuous integral

$$\int_0^T l(\mathbf{x}, \mathbf{u}, \lambda)\, dt \tag{1.2}$$

A practical, and commonly used variation is a discretized sum of l calculated at specific time steps, which can be written as

$$\sum_{n=1}^{T} l_n(\mathbf{x}, \mathbf{u}, \lambda) \tag{1.3}$$

Dynamics Constraint (1.1b) can again be in continuous or discretized. The continuous dynamics is usually defined as a set of differential equations which show the evolution of the state with time.

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}, \lambda) \tag{1.4}$$

The discretized version of the same can be explained as

$$\mathbf{x}_{t+1} = f_t(\mathbf{x_t}, \mathbf{u_t}, \lambda_\mathbf{t}) \tag{1.5}$$

Force unilaterality constraint (1.1c) ensures no penetration between the contacting bodies. If we define $\lambda_\perp$ as the component of the force vector which is normal to the contact surface, the constraint is given by

$$\lambda_\perp \geq 0 \tag{1.6}$$

Force Model (1.1d) puts constraints on the tangential components of the force. To ensure no-slippage between the contacting surfaces, a commonly used model is the friction cone, i.e., the force vector should lie in a cone defined by the maximum allowed static friction between the surfaces. If we define $\mathcal{K}$ as the set of acceptable force vectors which satisfy the friction cone, the constraint is of the form:

$$\lambda \in \mathcal{K} \tag{1.7}$$

Throughout this thesis, we will alternatively use the above representations ((1.2)-(1.7)) to explain the different constraints of (1.1). Also, note that we have not

[1] What is the exact definition of the cost function is usually not very important in this work. It is smooth and regular. It is possible for it to be as simple as 1.

explicitly written constraints such as collision avoidance etc in our optimization problem. While such constraints are necessary to generate locomotion trajectory, these are not necessary to explain the structure of the locomotion problem. Thus, for the sake of simplicity, the optimization problems in this chapter deal only with the constraints that define the structure of the problem, and other constraints are omitted (but not forgotten).

Dynamics and unilaterality are difficult constraints to deal with in the optimization problem (1.1). Moreau, 1988 used the Lagrangian equations of motion as dynamics constraints (1.1b) on top of the unilateral constraints (1.1c) to get a set of acceptable velocity solutions. Working on top of their work, Trinkle et al., 1997 used discrete *Linear Complementarity Problem* formulation of the unilateral constraint. Anitescu and Potra, 1997 updated their LCP to provide guarantees of solution existence for multiple contact condition. Finally, building on top of the previous work, Stewart and Trinkle, 2000; Stewart, 2000 redefined the locomotion problem (1.1) as an LCP using the impulse-momentum equations as the dynamics motion equations (1.1b), and using the complementarity condition $d(x) \perp \lambda_{\perp}$ to replace the unilaterality constraint. $d(x)$ here denotes the distance between the contacting bodies, and $\lambda_{\perp}$ denotes the normal contact force. This converted the problem into a continuous optimization scheme, since it removed the need to decide contact switching or moment of impulse. As a result, they were able to remove any discontinuous effect of the impulsive forces.

These works have been the basis of many further research that came afterwards in this direction. A review of the complemantarity condition and its use in control of dynamic systems of the time was provided by Brogliato, 2003. Continuous force formulation could be modeled in the form of springs and dampers, however in such cases the optimization problem suffers from high stiffness introduced by the spring model (Betts, 2010). As a result, even though the complementarity formulation increases exponentially the search space for the global optimal, LCP formulation with impulsive collisions remains popular in this domain.

While useful for generalizing, these LCP problems are difficult to solve. As a result, research in this direction has always grappled with the question of scalability to the full robot dimensions, with an extremely large number of contact possibilities, and the efficiency of the solvers to find a solution. Recently, Mordatch et al., 2012b use a relaxation of complementarity formulation to generate human-like motions, Mordatch et al., 2012a do similar work with optimization of manipulation by fingers through contacts, and Posa et al., 2014 use tricks proposed by Anitescu, 2005 and Fletcher et al., 2006 to customize SNOPT (Gill et al., 2005) for solving complementarity problems. SNOPT is a sparse Sequential Quadratic Programming (SQP) solver. More recently, Deits et al., 2019 tried to use learning to limit the solution

set to be within the acceptable range, and thus reduce the complexity introduced by the large solution space. However, scalability remains an issue even after the learning. As a result, effort has been made by Todorov, 2010; Todorov et al., 2012 to relax the complementarity constraint to make it easier to solve.

Alternatively to the impulse-momentum equations and LCP formulation, others have used the Lagrangian formulation of dynamics to simplify the problem for the solver. However, using the lagrangian dynamics necessitates that contact switching be accounted for. Mombaur et al., 2005b; Schultz and Mombaur, 2010 used a switching function to change dynamic models depending on contacts and generate impressive human-like running motions. However, while they don't use complementarity and thus simplify their problem, their approach does not consider the problem of obstacle avoidance when deciding contacts, as the gait is meant to be cyclic. Approaches like those of Schultz and Mombaur, 2010 do not actively plan for contacts. They know the exact contact phases, and then use switching conditions to change contacts explicitly. However, they do that only in sagittal plane.

A different approach which promises the advantages of global optimality without the burden and complexity of model-based optimization is based on stochastic training of function approximators. Heess et al., 2017 recently provided an example of how reinforcement learning can extract walking behaviours from simple task definitions. Rajeswaran et al., 2017 build upon something similar, empirically proving that even simple linear representations are able to produce complex walking behaviours. They extend their analysis with an effort to increase the robustness of the output by increasing the variety of the training initial states, eventually learning stability as well as walking. These methods, however, remain in the simulation domain. Efforts to move this learning to the real world are underway (Mansard et al., 2018), but these results still remain in the future. One recent success in this field has been by Hwangbo et al., 2019, where they control the ANYmal quadruped by training an end-to-end neural network in simulation, and achieve an improvement in the robot performance as well. Even though the paper only considers a flat terrain, it is a promising start.

1.2 Decoupled Approach to Motion Generation

While global optimality is theoretically appealing (Todorov, 2004), it still lacks the practical tools to be transferred completely on robots. Human locomotion, on the other hand, does not have the concept of global optimality. Two people might take completely different routes when crossing a room, based on separate logics. Thus, if such an objective function indeed exists, it is deeply rooted in our general intelli-

gence and not obvious. What can be observed is that human locomotion relies on a sense of direction and environment with feedback provided by the proprioceptive sensors, vestibular sensors and vision (Holmes et al., 2006). Holmes et al., 2006 provide a detailed review of the link between biological walking and robotics, and show that once the physics of passive walking (e.g. as shown by McGeer, 1990) is accounted for, the neural centroidal walking pattern generator could be reduced to a template phase oscillator model. Conversely, Schultz and Mombaur, 2010 showed that human-like trajectories can be generated by a cyclic contact sequence and a simple task definition of energy reduction within the local active set. Indeed, Dominici et al., 2011 studied the locomotion patterns of toddlers and adults, and found that our patterns match those found in rats, cats, macaque monkeys, and guineafowl. This suggests that if optimality is the criterion behind our movements, the human locomotion system is based on a concrete objective which is related to some ancestral and neural network carried to us through evolution.

As a result, global optimality can be reconsidered as a goal of locomotion, and a compromise can be made in favor of efficiency of solution. It is easy to see that if contacts were to be decided beforehand, the optimization problem (1.1) would be simplified a great deal. While this would affect global optimality, the result would be a simpler problem which is tractable by the solvers. For a start, there would not be the complementarity constraint (1.1c), since exact contact switchings would be known. Moreover, the combinatorial nature of the optimization would be removed from the subsequent problem,and the discontinuous effects on the dynamics would be known beforehand. As a result, contact and path planning, i.e. finding a path and an associated sequence of contacts that can be feasibly followed by the robot, provides a significant simplification of the locomotion problem.

Once the contacts are predefined, the locomotion problem loses the complementarity constraint (1.1c), and the dynamics constraint (1.1b) becomes piece-wise continuous. Given a predefined sequence of contacts S and contact switching times t_s, the optimization problem (1.1) can be re-written as:

$$\underset{\mathbf{x},\mathbf{u},\lambda}{\text{minimize}} \quad \sum_{s=1}^{S} \int_{t_s}^{t_s+\Delta t_s} l(\mathbf{x},\mathbf{u},\lambda|S)\, dt \tag{1.8a}$$

$$\text{subject to} \quad \dot{\mathbf{x}} = f(\mathbf{x},\mathbf{u},\lambda) \tag{1.8b}$$

$$\lambda \in \mathcal{K} \tag{1.8c}$$

1.2.1 Contact Planning

There are two main fields of research in contact planning: searching inside a discrete set of precomputed contact positions, or a continuous search on the contact

positions. Among approaches which use a discrete search method, *A** (Hart et al., 1968) and Rapidly-exploring Random Trees(*RRT*) (Kuffner and LaValle, 2000) are classical methods which help exploring trees of possible contact sequences. Similar approaches with discrete search has been used by Michel et al., 2005, "Footstep planning among obstacles for biped robots", Baudouin et al., 2011 etc. Indeed, methods like *A** and *RRT* (LaValle and Kuffner, 2001; Kuffner et al., 2001) have been popular in robotics to discretely explore contact placement trees, and LaValle and Kuffner, 2001 was chosen as the *Most Impactful Paper published between 1997 and 2001* during ICRA 2019. On the other side, Deits and Tedrake, 2014 have used mixed-integer convex optimization to solve the continuous search problem of contact sequence generation among previously computed convex regions.

From a robotics perspective, pre-defining contact sequences, whether as a result of a fixed gait cycle like Schultz and Mombaur, 2010 or as a means of obstacle avoidance (e.g. Fernbach et al., 2018; Tonneau et al., 2018a) makes the problem easier to solve and generates a local minima which can be applied on the robot in real-time. Moreover, input needs to be provided to the solver to ensure that a legged robot's locomotion looks natural, even if the natural gait may not be globally optimal with some other objectives. Heess et al., 2017 proved that without any guidance, and with a simple objective function that does not betray information about the human-like walk, a humanoid avatar is able to find multiple possible gaits to move from one point to another while avoiding obstacles.

As a result, pre-defining the contact sequences (with added objective to make the gait human-like), and following the step plan was the main approach followed by the teams during the DARPA robotics challenge. Pratt et al., 2012; Johnson et al., 2015; Johnson et al., 2017 use capturability to determine the future footstep locations, and an operator to manage the exact foot placement. Similarly, Feng et al., 2015; Atkeson et al., 2015 used operator inputs to manage the contact positions and orientations during rough terrains, and used an *A** planner to get initial footstep locations.

In our team, we dealt with the problem of feasibility of trajectory when deciding contacts in multi-contact scenarios. Tonneau et al., 2018a used point mass based model to generate quasi-static feasible trajectories of the center of mass. Later, Fernbach et al., 2018 extend the approach to more general cases. They introduced the algorithm *CROC*, which provided convex reformulation of centroidal dynamics problem in order to ensure that the transition between two contact phases is feasible even for non quasi-static cases. However, they do not consider the angular momentum variations, which are important for dynamic movements.

This is the contact planner that we use in our algorithms to provide the footstep plans, in an efficient and feasible manner, which can be provided to the optimizer to generate whole-body motions. For solving the locomotion problem (1.1), we divide the problem into a contact planning problem followed by optimization problem (1.8).

1.2.2 Reduced Problems

Contact-constrained locomotion problem (1.8) is still hard and challenging. The main difficulty arises from the non-convex dynamics with numerous Degrees of Freedom (DoF) which must be solved together to create a feasible and optimal solution. Such a problem is hardly tractable by modern computers, and particularly the ones embedded in modern legged robots (Koch et al., 2012; Tassa et al., 2012).

To tackle the above limitations, various strategies have been proposed in the literature. Most of them are based upon using reduced models: instead of working with the full dynamics, only a subpart is considered, covering the essential properties of the whole dynamics. The locomotion is then reduced to the problem of finding a trajectory for the reduced model which will then drive the whole-body system. Such an approach is not limited to robotics. Full and Koditschek, 1999 found that by using simpler models (templates), they could not only simplify the high redundancy of walking, but also capture the features of locomotion covering multiple species and multiple gaits. If we consider $\mathbf{x}_c$ and $\mathbf{u}_c$ as the reduced state and control, these reduced models can be written in the following formulation:

$$\begin{aligned}
\underset{\substack{\mathbf{x}_c(\mathbf{x}) \\ \mathbf{u}_c(\mathbf{u},\lambda)}}{\text{minimize}} \quad & \sum_{s=1}^{S} \int_{t_s}^{t_s+\Delta t_s} \ell_c(\mathbf{x}_c, \mathbf{u}_c|\mathbf{S})\, dt \\
\text{subject to} \quad & \forall t \quad \text{Force model is satisfied} && \text{(1.9a)} \\
& \forall t \quad \dot{\mathbf{x}}_c = f_c(\mathbf{x}_c, \mathbf{u}_c) && \text{(1.9b)} \\
& \forall t \quad \text{Other Constraints} && \text{(1.9c)}
\end{aligned}$$

Note that $\mathbf{x}_c$ and $\mathbf{u}_c$ are reduced representations of the full state and control, and similarly, f_c is the reduced dynamics.

In the context of bipedal locomotion in humanoid robots, the most famous reduced model is the linear inverted pendulum model (LIPM) (Kajita et al., 2001). Multiple variations of the LIPM method have been proposed afterwards. Starting with Kajita et al., 2003, efforts have been made to either tackle the robustness problem (Wieber, 2006a), include viability conditions (Sherikov et al., 2014), allow altitude variations of the center of mass(CoM) (Brasseur et al., 2015), or also include foot

placements as parameters of the problem (Herdt et al., 2010a). Pratt et al., 2006 used an extension of LIPM with a flywheel to model the angular momentum, and used this model to demonstrate computation of the capture points (points on the ground where the humanoid must step in order to stop a fall).

However, LIPM-based methods are restricted to basic environments and cannot deal with more complex scenarios as non-coplanar contact cases, climbing stairs using guardrail, etc. Considering non-coplanar contacts breaks the nice linearization leading to the LIPM model. A first approach to handle the non-linear dynamics was proposed in Hirukawa et al., 2006, however requiring technical and dedicated developments based on a limiting assumption (e.g. prior knowledge of force distribution). Caron et al., 2017 use LIPM to do multi-contact motion, but don't take into account the effect of additional angular momentum generated by movement of limbs. Thus, they have to use additional constraint in order to ensure the final motion is stable.

In quite another vein, it has been proposed to simplify the whole-body optimization problem by, for example, assuming unconstrained torque capabilities (Dai et al., 2014). Such approaches indeed boil down to optimizing the so-called centroidal dynamics Orin et al., 2013 as reduced model. Direct resolution of the underlying optimal control problem based on multiple-shooting approach has been recently proposed (Kudruss et al., 2015; Carpentier et al., 2016), leading to real-time performances. Other contributions have also been suggested that exhibit approximate dynamics (with possibly bounded approximations) leading to convex optimization problems, thus ensuring global optimality (Herzog et al., 2015; Dai and Tedrake, 2016; Brasseur et al., 2015). In most of these cases, the footstep sequence is assumed given, although some solvers are also able to discover it while optimizing the centroidal dynamics (Mordatch et al., 2012b), at the price of heavier computational costs.

In recent years, reduced models based on the centroidal dynamics (Orin et al., 2013) have gained in popularity. Centroidal dynamics is later explained in this chapter in Eq (1.13). It is easy to understand the reason behind this popularity: contrary to other approaches, centroidal dynamics is an exact projection of the full dynamics, which does not rely on any assumptions (like the constant altitude of the center of mass(CoM) for the table-cart model). This makes it possible for the solver to consider non-coplanar surfaces, and multi-contact scenarios without making additional efforts to fix the formulation.

In summary, the centroidal model has several advantages with respect to other formulations for an acceptable cost. Non-convexity in the centroidal dynamics model has been empirically validated to be handled by state of the art solvers (even if

formal proofs are still missing). It is for these reasons that in this thesis, we will use the centroidal dynamics as the reduced model to create a tracking reference for our whole-body solver, even though our approach is conceptually valid for any other reduced model.

1.2.3 Instantaneous tracking of Reduced Patterns

Whole-body motion generation tracks a reduced model locomotion pattern. The trajectory of this reduced model captures in itself the essence of the difficulties in locomotion. This reduced model trajectory is then used to generate a whole-body motion. The reduced trajectory contains important information about the global structure of the solution, but it lacks information about the multi-body structure of the dynamics. For example, kinematic limits, torque limits, collision information, and angular momentum generated by limb motion, all these are constraints that can only be expressed at the whole-body level. The whole-body solver, on the other hand, needs to generate a solution that is consistent with the global structure given by the reduced model. This is referred to as "Dynamic Consistency" between the two solvers. This dynamic consistency is tricky to obtain, because the whole-body solver often has to find the sweet spot between blindly following the references provided by the reduced model, and generating a fully new (and feasible) trajectory while ignoring the references completely.

Regardless of the "Dynamic Consistency", the earliest efforts to find the solution have been through instantaneous linearization of the whole-body problem around the current time. As a result, the whole-body control is found only based on the current reduced values. In terms of an optimization problem, this can be written as:

$$\begin{aligned} \underset{\mathbf{x},\mathbf{u},\lambda}{\text{minimize}} \quad & \ell_l(\mathbf{x}, \mathbf{u}, \lambda) \\ \text{subject to} \quad & \dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}) && \text{(1.10a)} \\ & \text{Force model is satisfied} && \text{(1.10b)} \\ & \text{Other Constraints} && \text{(1.10c)} \end{aligned}$$

Note that here, the optimization is being done only over the current state and control, and not over the full trajectory.

Early works from the literature included tracking of reduced model and task-space trajectories with Inverse Kinematics. Kajita et al., 2003 and Wieber, 2006b demonstrated HRP-2 walking with the table-cart reduced model and inverse kinematics. Tevatia and Schaal, 2000 try to track end-effector trajectories with IK, and later D'Souza et al., 2001 used supervised learning of Inverse Kinematics to do the same.

Asfour and Dillmann, 2003 used redundancy of the kinematics in order to solve for a humanoid arm. Mistry et al., 2008 used constraints to deal with the under-actuation of legged robots, and demonstrated IK for task-space full body control of humanoids.

While IK is useful to generate joint-space trajectories, they still lack information about the model dynamics. As a result, the IK trajectories lack consideration of the inertial effects, the centrifugal and coriolis forces, the contact forces etc. These factors become more and more important as we leave the quasi-static domain and try to make the robots more and more dynamic. As a result, Inverse Dynamics control of the robot becomes important.

However, full-body control of floating based systems in the task-space manner is difficult. The system is highly under-actuated, the contact switchings change the dynamics of the system, and the contact forces may be unknown. Khatib, 1987 proposed an operational-space formulation using inverse dynamics, and Sentis and Khatib, 2005; Jaeheung Park and Khatib, 2006 extended it to control floating based systems like humanoid robots. Later Mistry et al., 2010 used orthogonal decomposition of the rigid-body dynamics in order to separately control the contact forces and deal with the under-actuation of the system. Stephens and Atkeson, 2010 used the CoM dynamics model in order to compute the external contact forces, and used these forces as references inside a full dynamic model of the robot. Such a cascade computing Inverse dynamics with a reduced model has been used in Carpentier et al., 2016; Fernbach et al., 2018; Herzog et al., 2015; Winkler et al., 2015 to generate impressive motions on robots and simulations. Even in the cases where the contact information is optimized along with the centroidal trajectory, as in Mastalli et al., 2017; Winkler et al., 2018; Aceituno-Cabezas et al., 2017, this has been the approach most followed. These solvers usually solve quadratic optimization problems written with task-space dynamics to generate the full-body motion, for example as in Saab et al., 2013; Herzog et al., 2016a; Vaillant et al., 2016; Mastalli et al., 2017.

When dealing with such a separation between a reduced model (e.g. CoM dynamics model used by Stephens and Atkeson, 2010; Carpentier et al., 2016) and the full-body model, it becomes important to ensure along with the CoM, the momentum trajectories are also matched by the full model. It is because the rate of momentum trajectories are directly related to the contact wrench being applied on the robot. Koyanagi et al., 2008; Koolen et al., 2013 have shown success in controlling their humanoid by using momentum control. Dai et al., 2014 used a full-body kinematics model with the centroidal dynamics, but enforce that the momentum of the centroidal dynamics is enforced in the full-body model. Herzog et al., 2014 use a momentum based balance controller (similar to the one proposed by Lee

and Goswami, 2012) with hierarchical inverse dynamics in order to balance torque-controller robots. Later Herzog et al., 2016a use a similar cascade of tasks along with momentum tracking in order to do torque control on a robot. Such works have shown the importance of controlling not only the linear, but also the angular momentum, in order to maintain balance of the robot. Komura et al., 2005 have shown that the angular component of the dynamics plays a huge role in maintaining balance. Moreover, even in humans it has been shown that the angular momentum is tightly regulated during walking (Popovic et al., 2004).

1.2.4 Whole-body Trajectory Optimization

Whether it is Kajita et al., 2003, Herdt et al., 2010a or Sherikov et al., 2014, researchers have realized that instantaneous control like IK/ID is not enough to deal with the balance of the biped walking problem. For example, while Kajita et al., 2003 try to ensure ZMP remains within a support polygon over a horizon, Sherikov et al., 2014 used MPC on a point mass model along with instantaneous Inverse Dynamics in order to make sure that the upcoming trajectory keeps satisfying the friction constraints. Similarly, Herdt et al., 2010a; Herdt et al., 2010b used MPC on the center of mass trajectory up to the level of acceleration, in order to make sure that the upcoming trajectory remains smooth. Dai et al., 2014 formulate a similar problem in order to solve for joint kinematics and centroidal dynamics over a horizon. The common factor in such efforts is that preview control over a horizon has been understood to be necessary for stability. Indeed, for a system with discontinuous events like making/breaking contact, information about the future is essential in order to ensure that the robot does not fall. Brockett, 1983 have shown that instantaneous control is not enough for non-holonomic systems, and free-floating systems demonstrate non-holonomic behaviour because of the non-integrability of angular momentum (Nakamura and Mukherjee, 1990; Papadopoulos, 1993).

While using a preview horizon with a simplified model like point-mass (or even centroidal variables) is faster, information is lost in the simplification. The full model of the robot has to be considered in order to provide the optimal control over the horizon.

Using trajectory optimization to generate motions for legged systems and perform preview control is not a new concept. The simplified optimization problem is written as follows:

$$\begin{aligned}
\underset{\underline{\mathbf{x}},\underline{\mathbf{u}},\underline{\lambda}}{\text{minimize}} \quad & \sum_{s=1}^{S} \int_{t_s}^{t_s+\Delta t_s} \ell_l(\mathbf{x}, \mathbf{u}, \lambda | \mathbf{S})\, dt \\
\text{subject to} \quad & \forall t \quad \text{Force model is satisfied} && (1.11a) \\
& \forall t \quad \dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}) && (1.11b) \\
& \forall t \quad \text{Other Constraints} && (1.11c)
\end{aligned}$$

Both direct and indirect approaches have been used in the literature to generate motions. Among the indirect methods, Pontryagin's Maximum Principle (PMP) has been a popular tool for solving two-point boundary value problems. A famous Russian method, PMP has been especially popular in aerospace applications. It was used for the *Apollo 11* mission to the moon. In robot locomotion, Rostami and Bessonnet, 2001 used PMP in order to generate swing phase trajectories.

The adjoint equations in indirect methods serve as a built-in accuracy metric, that leads to precise solutions. However, setting up these equations and their gradients is difficult to do, and they are difficult to solve. Moreover, constraints are difficult to handle, and the active-set of constraints has to be pre-defined in the solution. As a result, direct methods, while less precise than indirect methods, have been more popular in the field of trajectory optimization for motion generation. One major advantage of these direct methods is that they are well-suited for inequality constraints by exploiting the active set changes (Betts, 2010).

In direct methods, Hardt and William, 1999 used an Sequential Quadratic Programming based optimization package for multi-body systems. Roussel et al., 1998 simplified the dynamics and used direct shooting optimization algorithm. Westervelt and Grizzle, 2002 also used Sequential Quadratic Programming (SQP) in order to minimize energy consumption with kinematic and dynamic constraints. Similarly, Djoudi et al., 2005 produced energy-optimal walking motion using polynomial trajectories, by solving for optimal polynomial coefficients. Lengagne et al., 2013 used a parametrized representation of the trajectories with B-splines. Recently, Posa and Tedrake, 2013; Hereid et al., 2016; Hereid and Ames, 2017; Talele and Byl, 2018 have used direct collocation in order to generate stable walking motions on humanoids.

Whole-body trajectory optimization approaches have typically suffered from two problems that prevented the replacement of IK/ID solvers. Namely, they struggled to discover a valid motion, in particular the gait and its timings; and they were

slow to converge. Thus, the bottle-neck in this approach has been both, the computation speed, and the ability of the solvers. For e.g., direct collocation schemes to solve optimization problems might result in sequence of control and state trajectories that might never converge onto a feasible solution. Works like Zucker et al., 2013, while popular tools for trajecory optimization with functional gradients to get around parametrization of trajectories (similarly for Kalakrishnan et al., 2011 in case of stochastic gradients), are difficult to deploy online since they don't exploit the sparse structure of the MPC problem.

With current computational capacity, it is now slowly becoming possible to solve for a large enough horizon even with the full model. A good resolution scheme for the solver, which allows it to exploit the sparsity of the problem properly, tremendously helps in making the preview control real-time.

Differential Dynamic Programming (DDP) is one such method which is built to exploit the sparsity of the structure of the problem. Instead of inverting the full Hessian of the problem, DDP uses the block diagonal structure of the Hessian to iteratively inverts small matrix blocks and find the full inverse of the Hessian. Indeed, DDP is especially suited to Markovian-type dynamics. Tassa et al., 2014 demonstrated in simulation that DDP, is able to meet the control-loop timings constraint, and Koenemann et al., 2015 provided a proof-of-concept. In this thesis, we will use DDP as the base of our solver, and update the method in order to efficiently take into account the contact constraints that are provided by the contact planner. Locomotion movements computed by DDP were never transferred to a real full-size humanoid before this work. DDP does not need to discover the contact switching instants, therefore we can use rigid contact dynamics which is faster to compute and easier to implement.

Other works have shown that DDP is able to discover locomotion gaits applied to a real quadruped e.g. in Neunert et al., 2018. In Rajamaki et al., 2016, DDP is coupled with Monte Carlo tree search to compute the bipedal locomotion pattern of an avatar. While not yet demonstrated on a real humanoid, we might wonder whether this should be pushed further, instead of relying on a decoupling between contact computation, centroidal and whole-body optimization. We believe that DDP is a mature solution to replace IK/ID and is very complementary to centroidal optimization. Indeed, contact and centroidal problems can be efficiently handled within a global search thanks to the low dimension, while DDP is efficient to accurately handle the whole-body dynamics in a large space (but locally).

1.3 The Locomotion Problem Structure

As seen in the previous section, both the coupled approach, and the decoupled approach have their merits and demerits. The coupled approach, as discussed in Sec 1.1, tries to solve the full locomotion problem (1.1) together. On the other hand, the decoupled approach tries to approximate the problem in a simplified pipeline, as explained in Sec 1.2). The first approach tries to find the global optimal, but suffers from issues of efficiency and scalability. These issues don't occur in the decoupled approach, but we make assumptions when we decouple and this reduces the search space and thus sacrifices global optimality.

In this thesis, we have chosen to use the decoupled approach in order to best control our robots in real time. This section explains the structure of the decoupled approach for multi-body systems.

1.3.1 Natural splitting of the robot dynamics

If we consider a legged robot dotted with n Degrees of Freedom (DoF), its whole-body dynamics is represented by the Lagrangian equations of motion (Wieber, 2006a):

$$\begin{bmatrix}\mathbf{M}_u\\ \mathbf{M}_a\end{bmatrix}\ddot{\mathbf{q}}+\begin{bmatrix}\mathbf{b}_u\\ \mathbf{b}_a\end{bmatrix}=\begin{bmatrix}\boldsymbol{g}_u\\ \boldsymbol{g}_a\end{bmatrix}+\begin{bmatrix}\mathbf{0}_6\\ \boldsymbol{\tau}\end{bmatrix}+\sum_{k=1}^{K}\begin{bmatrix}\mathbf{J}_{k,u}^\top\\ \mathbf{J}_{k,a}^\top\end{bmatrix}\boldsymbol{\lambda}_k \tag{1.12}$$

where $\mathbf{M}$ denotes the joint space inertia matrix, $\mathbf{b}$ encompasses the nonlinear effects, $\boldsymbol{g}$ corresponds to the generalized gravity vector, and $\mathbf{J}_k$ is the geometric Jacobian for contact k and λ_k is the vector of contact forces at contact point k. $\boldsymbol{\tau}$ is the vector of joint actuations. This dynamics can be split into two distinct parts: subscript u denotes the 6 rows that correspond to the under-actuated dynamics; subscript a denotes the n rows that correspond to the actuated dynamics.

The under-actuated dynamics of (1.12) is also known as the centroidal dynamics of the robot. It is governed by the Newton-Euler equations of motion which link the variations of the linear momentum and Angular Momentum (AM) to the contact forces:

$$\begin{aligned} m\,\ddot{\mathbf{c}} &= \textstyle\sum_k \boldsymbol{\lambda}_k + m\,\mathbf{g}\\ \dot{\boldsymbol{L}} &= \textstyle\sum_k (\boldsymbol{p}_k - \mathbf{c})\times\boldsymbol{\lambda}_k \end{aligned} \tag{1.13}$$

where $\boldsymbol{p}_k$ is the position of the kth contact point, the operator $\times$ denotes the cross product, m is the total mass of the system, $\mathbf{c}, \dot{\mathbf{c}}, \ddot{\mathbf{c}}$ are the center of mass position, velocity and acceleration vectors and $\boldsymbol{L}, \dot{\boldsymbol{L}}$ are the AM vector and its time derivative.

The centroidal dynamics is a projection of the Lagrangian dynamics onto the Center of Mass (CoM) of the robot. In doing so, the centroidal dynamics absorbs in itself all the interactions of the robot with the environment (external contact forces λ_k). The actuated dynamics then follows the classic equations of a manipulator actuated by a pure torque input $\boldsymbol{\tau}$ (but constrained by the centroidal dynamics). This separation of the dynamics into its underactuated and actuated components is indeed one of the main features of this approach.

Thus, a natural splitting appears between two sets of state and control variables, namely:

$$\begin{aligned} &\textbf{Centroidal set } \mathbf{d}_c \text{ with state } \mathbf{x}_c = (\mathbf{c}, \dot{\mathbf{c}}, \boldsymbol{L}) \text{ and control } \mathbf{u}_c = (\boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_k) \\ &\textbf{Lagrangian set } \mathbf{d}_l \text{ with state } \mathbf{x}_l = (\mathbf{q}, \dot{\mathbf{q}}) \text{ and control } \mathbf{u}_l = \boldsymbol{\tau} \end{aligned} \tag{1.14}$$

1.3.2 Coupling Constraints

When we divide the robot dynamics into its underactuated and actuated components, as in (1.13) and (1.12), the split is not straightforward. The two problems are connected. Each dynamics is restricted by the need to be feasible by the other. (1.8) defines the optimization problem for the contact-constrained locomotion. As a result of the split, there are additional coupling constraints that we need to add to the contact-constrained optimization problem. These constraints are inherent to the decoupling approach, and they couple the two dynamics (1.13) and (1.12) together. Thus, we call them coupling constraints.

$$\forall t \quad \mathbf{c} = CoM(\mathbf{q}) \tag{1.15a}$$

$$\forall t \quad \begin{bmatrix} m\dot{\mathbf{c}} \\ \boldsymbol{L} \end{bmatrix} = \boldsymbol{A}_g(\mathbf{q})\,\dot{\mathbf{q}} \tag{1.15b}$$

$$\forall t \quad \boldsymbol{\lambda} = g_\lambda(\mathbf{q}, \dot{\mathbf{q}}, \boldsymbol{\tau}) \tag{1.15c}$$

where $\boldsymbol{A}_g$ is the so-called centroidal momentum matrix Orin et al., 2013, and CoM maps the current joint configuration $\mathbf{q}$ to the center of mass position. g_λ is the mapping between the whole-body dynamics and contact forces, and may be dependent Budhiraja et al., 2018 or independent Neunert et al., 2018 from $\boldsymbol{\tau}$, depending on the choice of contact model. If we look at these constraints and the sets (1.14),

we can see that the left hand side only contains quantities from the Centroidal Set, and the right hand side only contains quantities from the Lagrangian Set.

As a result of these additional constraints, we can find works in the literature that explicitly or implicitly handle these constraints while splitting the problem into reduced and whole-body structures.

1.3.3 Managing the Coupling Constraints

Splitting the locomotion problem into reduced and a whole-body subproblems has shown great results in the past (for example, Carpentier et al., 2016; Fernbach et al., 2018; Mastalli et al., 2018). However, the reduced subproblem needs to ensure that its solution does not violate the feasibility constraints implied by the whole body (e.g. kinematic or torque limits, footstep lengths etc..) Such constraints cannot be expressed as solely the function of the reduced model. For example, if we consider the splitting shown in Sec 1.3.1, the CoM trajectory must be achievable by the whole-body kinematics. This constraint is expressed as (1.15a). These constraints have been tackled explicitly in the past, for example by adding the corresponding whole-body variable in the optimization scheme (Mordatch et al., 2012b; Dai et al., 2014). In Dai et al., 2014 for example, they optimize the whole-body kinematics along with the centroidal dynamics, in order to make sure that the centroidal solver is aware of the kinematics constraints((1.15a) and (1.15b)) of the model. However, this direct representation is also the most expensive in terms of computation. One way to handle such constraints has been via proxies, where an equivalent constraint (which is independent of the whole-body parameters) is added inside the reduced subproblem. Carpentier et al., 2017a; Carpentier and Mansard, 2018b were the first to use this, and it was later used in Tonneau et al., 2018c. Herdt et al., 2010a and Deits and Tedrake, 2014 use similar proxy constraints to encode maximal step size in biped walking, and Dai and Tedrake, 2016 use proxy constraints to bound the CoM position inside a geometric shape.

While such schemes are able to handle the CoM variation, handling Angular Momentum produced by the limb motion is more tricky. This is notable for humanoid robots which have important masses in the limbs that are put in motion during (for instance) walking. This effect is neither properly captured by the centroidal model, nor by the instantaneous time-invariant linearization. In Herzog et al., 2016b, an alternative scheme aims to compensate the AM variations. Indeed, it properly compensates the momentum changes produced by the flying limbs, however it is not yet able to trigger additional momentum to enable very dynamic movements. This would be needed for generating long steps, running, jumping or salto motions (The necessity of angular-momentum variations is imposed by the motion of the very

heavy swing leg in humanoids). The approach of Herzog et al., 2016b is further exploited in Ponton et al., 2016. In their method, CoM and AM trajectories in the reduced dynamics problem must track the output CoM and AM trajectories resulting from the whole-body dynamics and vice-versa.

While efforts have been made to handle CoM and AM feasibility in the reduced subproblem, in order to ensure that the two subproblems do not produce divergent and incoherent solutions at the global level, the force constraints need to be handled as well. Englsberger et al., 2015 used a heuristic to modify desired forces to ensure feasibility while tracking for the DCM component of motion. Herzog et al., 2016a optimize contact forces in order to ensure that the dynamic constraints are not violated. In the past, these different feasibility constraints have been known, and starting from the first quasi-static walk, effort has been made to make sure that they are not violated by the whole-body solver. Brasseur et al., 2015 found the limits of the dynamic and kinematic feasibility for the LIPM model. Herzog et al., 2016b and Dai et al., 2014 tried to ensure that the feasibility is maintained for the centroidal dynamics model. However, before this thesis, the constraints that bind the centroidal and whole-body models were not explicitly defined and considered for the decoupled problem. In this thesis, we explicitly express the required constraints and propose how to deal with them.

1.3.4 The Global Optimal Control Problem for locomotion

Consider the global motion planning Optimal Control Problem (OCP) problem defined between two states of the robot ($\boldsymbol{x}_{init}$, $\boldsymbol{x}_{final}$). For the decoupled scenario explained in Sec 1.2, the set of contact phases $\mathbf{S}$ and their corresponding contact timings Δt_s have already been solved. Then, we split the locomotion problem into centroidal and whole-body stages. However, to do that we need to assume that the actuators are capable enough to provide sufficient torque, since the torque limits are not a part of the constraints (1.15). This is normally true for current generation of robots. Thus we successively solve for $\underline{\mathbf{d}}_c$ and $\underline{\mathbf{d}}_l$, while making sure that the solution of the first OCP is feasible for the second OCP.

Thus, we divide the contact-constrained optimization problem (1.8) into two problems defined by the sets (1.14). This new motion planning OCP, governed by the dynamics defined by (1.12) and (1.13) is given by:

$$\underset{\substack{\underline{\mathbf{d}}_c:=[\mathbf{c},\dot{\mathbf{c}},\boldsymbol{L},\boldsymbol{\lambda}]\\ \underline{\mathbf{d}}_l:=[\mathbf{q},\dot{\mathbf{q}},\boldsymbol{\tau}]}}{\text{minimize}} \quad \sum_{s=1}^{S}\int_{t_s}^{t_s+\Delta t_s}\ell_c(\mathbf{d}_c|\mathbf{S})\,dt+\sum_{s=1}^{S}\int_{t_s}^{t_s+\Delta t_s}\ell_l(\mathbf{d}_l|\mathbf{S})\,dt \tag{1.16a}$$

$$\text{subject to} \quad \forall t \quad \boldsymbol{\lambda}\in\mathcal{K} \tag{1.16b}$$

$$\forall t \quad g_\lambda(\mathbf{q},\dot{\mathbf{q}},\boldsymbol{\tau})\in\mathcal{K} \tag{1.16c}$$

$$\forall t \quad \dot{\mathbf{x}}_c = f_c(\mathbf{d}_c) \tag{1.16d}$$

$$\forall t \quad \dot{\mathbf{x}}_l = f_l(\mathbf{d}_l) \tag{1.16e}$$

$$\text{Coupling Constraints (1.15) are satisfied} \tag{1.16f}$$

$$\mathbf{x}_c(0) \text{ is given}, \mathbf{x}_c(T) \text{ is viable} \tag{1.16g}$$

$$\mathbf{x}_l(0) \text{ is given}, \mathbf{x}_l(T) \text{ is viable} \tag{1.16h}$$

where s is the index of the contact phase, t_s is the start time of the contact phase s. The cost function l from (1.8) is divided into its centroidal and whole-body objective functions. Consequently, ℓ_c and ℓ_l are local cost functions related to the phase. $\mathcal{K}$ denotes the admissible set of the friction forces corresponding to zero slippage. Note that for all variables, underlines denote a trajectory of the variable over time. Similarly the dependency to the time variable is kept implicit i.e. $\forall t$ $\mathbf{c}$ is preferred to $\forall t$ $\mathbf{c}(t)$.

Most of constraints and the two cost terms only depend on one of the two groups of variables $\underline{\mathbf{d}}_c, \underline{\mathbf{d}}_l$: $\ell_c(\mathbf{d}_c)$, (1.16b), (1.16d) and (1.16g) define a problem over the centroidal dynamics ; $\ell_c(\mathbf{d}_l)$, (1.16c), (1.16e) and (1.16h) define a problem over the Lagrangian dynamics.

One way to solve the two problems independently is to replace the three coupling constraints by some proxy constraint, i.e. reformulation which enforces the existence of a global consensus solution acceptable by both subproblems. In Carpentier et al., 2017a, we have proposed to learn such a proxy constraints for the centroidal optimization. While preparing to resolve this problem in Chapter 5, we will use again this learned proxy in the initial step of our algorithms.

Constraints (1.16b) and (1.16c) are redundant (i.e. (1.16b) and (1.15c) implies (1.16c)). However, consider that (1.16b) enforces the non-slippage condition on the contact forces in the centroidal problem, while (1.16c) does that in the whole-body problem. It is easy to see that (1.16b) and (1.16c) are the definition of the stability constraint (1.1d) from the global locomotion problem (1.1).

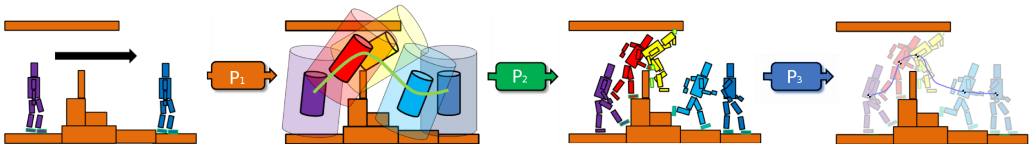


Fig. 1.2: Overview of our multi-stage locomotion framework Carpentier et al., 2017b. Given a requested path request between start and goal positions (left image), $\mathcal{P}_1$ is the problem of computing a guide path in the space of equilibrium feasible root configurations. We achieve this by defining a geometric condition, the reachability condition (abstracted with the transparent cylinders on the middle image). $\mathcal{P}_2$ is then the problem of extending the path into a discrete sequence of contact configurations. Finally, $\mathcal{P}_3$ creates a whole-body trajectory by solving for centroidal dynamics trajectories and using them as a reference.

(1.16b) and (1.16d) enforce consistent centroidal dynamics (1.13), while (1.16c) and (1.16e) enforce consistence of the Lagrangian dynamics (1.12) with respect to the contact model. We explicitly formulate both constraints to make the split evident. Similar remark holds for initial and terminal conditions (1.16g) and (1.16h). As terminal constraints are often difficult to formulate in practice, they should likely be replaced by stopping motion conditions (e.g. capturability)(Wieber, 2008).

The near-perfect split has already been nicely observed (Herzog et al., 2016a). The observation was then mostly used to justify the classical approach of separately solving each subproblem. Here we rather want to insist on the coupling and pave the way to handle this coupling in Chapter 5.

1.4 The Loco3D Project

In our team, we follow the multi-stage decoupling approach defined in Sec 1.3, i.e. we break the global problem into various subproblems of smaller dimensions, and individually solve them in a hierarchical manner. These stages form the individual blocks of a motion generation pipeline that we call the Loco3D project (Carpentier et al., 2017b). The Loco3D pipeline is composed of modules that are summarized below:

- *Contact Sequence Planner*: The first stage consists in an interactive acyclic contact planner (Tonneau et al., 2018c) which is able to compute a sequence of contacts for various scenarios, from a matter of a few hundreds of milliseconds up to few seconds depending on the complexity of the environment. This planner reduces the complexity of the problem by considering only the root of the robot together with the reachability sets of the end-effectors. More precisely, it verifies that the root configuration of a robot is close, but not too close from the obstacles: close to allow contact creation, not too close to avoid col-

lision. With this approximation of the space of admissible root configurations, we decompose the hard contact planning problem into simpler sub-problems: first to plan a guide path for the root without considering the whole-body configuration; then, to generate a discrete sequence of whole-body configurations in static equilibrium along this path. The complete workflow is depicted in Fig. 1.2.

- *Centroidal Pattern Generator*: Carpentier et al., 2016 introduced an optimal control formulation based on centroidal dynamics and using contact forces as control inputs. This formulation takes as input the contact sequence (generated by the previous step) and the initial state of the robot and tries to minimize a tailored cost function to obtain a smooth control while satisfying the friction cone constraints. In addition to that, the formulation seeks a final state that is viable Wieber, 2008. To be effective, this approach is translated into a multiple-shooting formulation, and is fast enough to be implemented in a receding horizon way.

- *Whole-body Motion Generator*: The next step concerns the whole-body motion generation. More precisely, how to plan the trajectories of swing end-effectors. We will explain in detail in Chapter 3 and Chapter 4 our DDP solver which efficiently solves the problem in a receding horizon manner. This approach computes a whole-body motion while following the centroidal trajectories computed by the centroidal pattern generator.

- *Low-level controller*: The above pipeline fuses into a low-level controller on the robot. The interaction of the robot with its environment requires the control of the contact forces. While it is possible to use the DDP solver as a controller on the robot, as shown recently by Grandia et al., 2019, we use inverse dynamics (similar to the one presented by Del Prete et al., 2016) as a control strategy. This allows to control the robot with a much higher bandwidth.

The above pipeline is based on a multi-stage strategy that helps us to exploit the state-of-the-art solutions: interactive computation of contact placements that ensure collision avoidance, real-time computation of centroidal trajectories followed by DDP solver for motion generation and inverse-dynamics based low-level controller.

Our work in this thesis, though self-contained as a research block, lies squarely in the middle of the Loco3D pipeline. We deal with the “Whole-body Motion Generation”, and manage the constraints that bind it to the “Centroidal Pattern Generator”.

1.5 Thesis Structure

The objective of this thesis is to provide an efficient method to solve the complete problem (1.16). In the past, it was either proposed to solve this problem exactly despite the lack of efficiency of the resulting optimization scheme, or to approximate it using some heuristic models in order to get a quick and practical answer. We propose to leverage on this near-perfect split to provide an efficient method which solves the exact problem (1.16), and not an approximation. This split has already been observed, and in particular has been nicely exhibited and exploited in Herzog et al., 2016a. Yet despite their understanding of the problem structure the authors did not come to an exact algorithm that takes the structure into account. Like Herzog et al., 2016a, our work is in continuation of the past works that combines some kind of reduced template dynamics in practical algorithms and a whole body motion generation algorithm. Our main contribution is the proposition to formalize these intuitions with the model split in order to solve the exact problem. We propose improvements in the 1) manner in which the problem is broken, and how the split is handled, 2) manner in which the whole-body solver resolves the contact constrained problem over a horizon, and 3) manner in which the centroidal solver deals with the kinematic feasibility. This thesis defines work done in these three major areas related to the field of multi-contact locomotion.

1.5.1 Chapter Organization

Chapter 2

This chapter details the kinematic constraint (1.15a), and defines the equivalent constraints which simplify the computation of the kinematic feasibility for the whole-body solver. Since the constraint is dependent only on the configuration of the kinematic tree, we propose a way to encode this constraint inside a proxy constraint which is simpler to deal with, and helps to provide a good and kinematically feasible initial solution to our final solver in Chapter 5

Chapter 3

This chapter explains the nitty-gritty of our whole-body solver, which is based on the recently popular method of Differential Dynamic Programming. We explain our choices in using this solver, explain its concepts, an propose changes in the dynamics of the problem in order to deal efficiently with the no-slippage and contact placement constraints. In this chapter, we also see how using DDP effectively improves the performance for aggressive motions when compared to instantaneous solvers like Inverse Dynamics.

Chapter 4

This chapter describes our python-based solver Crocoddyl (**C**ontact **Ro**bot **Co**ntrol by **D**ifferential **Dy**namic Programming **L**ibrary). We explain the API, the structure of the package and its features.

Chapter 5

This chapter provides a resolution of the optimization problem (1.16), and nicely threads the previous chapters together to form one solver scheme. In this chapter we demonstrate how it is possible to consider the full optimization problem (1.16) and use it for motion planning, without losing on the advantages of splitting the problem up.

Chapter 6

This chapter uses an example of a highly dynamic motion to explain how all chapters of this thesis work together to produce a solution to our optimization problem (1.16). We compare the results obtained by each of our contributions separately, and finally combine them together to arrive at a solution that works.

Chapter 7

This chapter provides the final summary of the thesis, its various elements, and proposes a path forward from this work.

1.5.2 Associated Publications

All the chapters in this thesis are in the direction of improving the numerical solutions of the multi-body locomotion optimization problem. Each chapter corresponds to a publication in an internationally renowned and peer-reviewed conference. Our interest lies in the efficient resolution of the locomotion problem (1.16). Eventually, all the chapters tie together towards the same goal. We see that in Chapter 6. All the published work deals with the locomotion of multi-body systems, at different levels of the problem.

The following articles were published in peer-reviewed conferences:

- **Budhiraja, R.**, Carpentier, J., & Mansard, N. (2019). Dynamics Consensus between Centroidal and Whole-Body Models for Locomotion of Legged Robots. In ICRA 2019 - IEEE International Conference on Robotics and Automation. Montreal, Canada.
  `https://hal.laas.fr/hal-01875031`

The work from this publication appears jointly in Chapters 1 and 5

- **Budhiraja, R.**, Carpentier, J., Mastalli, C., & Mansard, N. (2018). Differential Dynamic Programming for Multi-Phase Rigid Contact Dynamics. In 2018 IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids) (pp. 19). IEEE.
  `https://doi.org/10.1109/HUMANOIDS.2018.8624925`

The work from this publication appears jointly in Chapters 3 and 4

- Carpentier, J., **Budhiraja, R.**, & Mansard, N. (2017). Learning Feasibility Constraints for Multicontact Locomotion of Legged Robots. In Robotics: Science and Systems XIII. Robotics: Science and Systems Foundation.
  `https://doi.org/10.15607/RSS.2017.XIII.031`

The work from this publication appears in Chapter 2

- Mastalli, C., **Budhiraja, R.**, ... Mansard, N. (2020). Crocoddyl: An Efficient and Versatile Framework for Multi-Contact Optimal Control. In ICRA 2020 - IEEE International Conference on Robotics and Automation. Paris, France

The work from this publication appears in Chapters 4

- Stasse, O., Flayols, T., **Budhiraja, R.**, Giraud-Esclasse, K., Carpentier, J., Mirabel, J., ... Ferro, F. (2017). TALOS: A new humanoid research platform targeted for industrial applications. In 2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids) (pp. 689695). IEEE.
  `https://doi.org/10.1109/HUMANOIDS.2017.8246947`

Part of the work from this publication appears in Chapter 2

- Carpentier, J., Del Prete,A., Tonneau, S., Flayols, T., Forget, F., Mifsud, A., Giraud, K., Atchuthan, D., Fernbach, P., **Budhiraja, R.**, Geisert, M., Solà, J., Stasse, O., & Mansard N. (2017b). Multi-contact Locomotion of Legged Robots in Complex Environments The Loco3D project. In: RSS Workshop on Challenges in Dynamic Legged Locomotion. Boston, United States.
  `https://hal.laas.fr/hal-01543060/`

This work is partly described in Chapters 1, 2 and 3

2 Learning Feasibility Constraints for Multicontact Locomotion of Legged Robots

> *Le poète ne doit avoir qu'un modèle, la nature ; qu'un guide, la vérité. Il ne doit pas écrire avec ce qui a été écrit, mais avec son âme et avec son cœur.*
> *(The poet must have only one model, the nature; and only one guide, the truth. He must not write with what has been written, but with his soul and with his heart)*
>
> — **Victor Hugo**
> (Odes et ballades)

Carpentier, Justin, **Rohan Budhiraja**, and *Nicolas Mansard* (2017a). Learning Feasibility Constraints for Multicontact Locomotion of Legged Robots. In: Robotics: Science and Systems XIII. Robotics: Science and Systems Foundation.

2.1 Introduction

From the discussion in Chapter 1 and from the contact-constrained optimization problem (1.16), we can formulate the reduced problem of finding the centroidal trajectory by the following:

> *From a given contact sequence and an initial centroidal state, find a feasible centroidal trajectory, satisfying the Newton-Euler dynamics, respecting the contact constraints and leading to a viable state.*

This formulation is already present in the decoupling of (1.16). Here, we can explicitly transcribed it as an optimal control problem of the form:

$$
\begin{aligned}
\underset{\underline{\mathbf{d}}_c:=[\mathbf{c},\dot{\mathbf{c}},\boldsymbol{L},\boldsymbol{\lambda}]}{\text{minimize}} \quad & \sum_{s=1}^{S} \int_{t_s}^{t_s+\Delta t_s} \ell_c(\mathbf{d}_c|\mathbf{S})\, dt \\
\text{subject to} \quad & \forall t \quad \boldsymbol{\lambda} \in \mathcal{K} && \text{(2.1a)} \\
& \forall t \quad \dot{\mathbf{x}}_c = f_c(\mathbf{d}_c) && \text{(2.1b)} \\
& \quad \text{Coupling Constraints (1.15) are satisfied} && \text{(2.1c)} \\
& \quad \mathbf{x}_c(0) \text{ is given}, \mathbf{x}_c(T) \text{ is viable} && \text{(2.1d)}
\end{aligned}
$$

Constraint (2.1d) constrains the trajectory to start with a given state (typically estimated by the sensor of the real robot). Terminal constraint (2.1d) is difficult to exactly represent (Wieber, 2008) and is replaced in practice by zero terminal movement $\ddot{\mathbf{c}}(T) = \dot{\boldsymbol{L}}(T) = 0$ and $x(T) = (\mathbf{c}^*, \mathbf{0}, \mathbf{0})$. Finally, ℓ_s is the cost function which enforces the smoothness of both the state and control trajectories. Various ℓ_s can be discussed and implemented. The one used in the experiments requires some additional definitions and is given in Section 2.3.

Coupling Constraints (1.15) are difficult to represent in this optimization problem. The coupling constraints depend on $\mathbf{q}, \dot{\mathbf{q}}, \boldsymbol{\tau}$, none of which are variables of the above optimization problem. Since the centroidal problem is solved first, the centroidal solver does not have information on the whole-body variables. Ideally, the constraints (1.15) would take the following form:

$$
\begin{aligned}
\forall t\ \exists\ (\mathbf{q}^*, \dot{\mathbf{q}}^*, \boldsymbol{\tau}^*) \text{ such that:} \quad \mathbf{c} &= CoM(\mathbf{q}^*) && \text{(2.2a)} \\
\begin{bmatrix} m\dot{\mathbf{c}} \\ \boldsymbol{L} \end{bmatrix} &= \boldsymbol{A}_g(\mathbf{q}^*)\, \dot{\mathbf{q}}^* && \text{(2.2b)} \\
\boldsymbol{\lambda} &= g_\lambda(\mathbf{q}^*, \dot{\mathbf{q}}^*, \boldsymbol{\tau}^*) && \text{(2.2c)} \\
(\mathbf{q}^*, \dot{\mathbf{q}}^*, \boldsymbol{\tau}^*) &= \underset{\mathbf{q},\dot{\mathbf{q}},\boldsymbol{\tau}}{\arg\min} \sum_{s=1}^{S} \int_{t_s}^{t_s+\Delta t_s} \ell_l(\mathbf{q}, \dot{\mathbf{q}}, \boldsymbol{\tau}|\mathbf{S})\, dt \\
& \quad \text{subject to constraints} && \text{(2.2d)}
\end{aligned}
$$

Thus, the coupling constraints (1.15) can be re-written in the centroidal problem by two separate sets of equations:

Feasibility constraints: (2.2a), (2.2b) and (2.2c) ensure that the centroidal variables are feasible with respect to the whole-body problem. Indeed, without this feasibility, our decoupled approach from Sec 1.2 would struggle to find a proper solution.

Whole-body Optimality constraint: (2.2d) ensures that the decoupled approach indeed matches exactly the solution of the full contact-constrained optimization problem (1.16). For clarity reasons, we don't list the constraints to the nested optimization problem (2.2d). Normally, they ensure collision avoidance, non-slippage and so on. These constraints are listed properly in Chapter 3.

The feasibility constraints (2.2a)-(2.2c) are already difficult to handle in the reduced problems. In addition, the Whole-body Optimality Constraint (2.2d) makes the problem completely intractable.

For now, we choose to let go of the optimality constraint (2.2d). We will try to handle the Optimality Constraint in Chapter 5, which will (to an extent) make sure that our centroidal solution ensures optimality of the whole-body problem.

Instead, in this chapter, we propose a systematic approach to handle kinematic feasibility constraint (2.2a) in the context of trajectory optimization for reduced models, leading to efficient resolution on the real robot. We try to maximize the set of feasible $\mathbf{q}$ values, from which we can try to find $\mathbf{q}^*$ in the whole-body optimization. The resulting constraint formulation could be employed in most of the optimal control solvers based on centroidal dynamics (Mordatch et al., 2012b; Deits and Tedrake, 2014; Kudruss et al., 2015; Herzog et al., 2015), although we implement it inside a multiple-shooting solver (Carpentier et al., 2016).

2.1.1 Kinematic Feasibility as a Proxy Constraint

The coupling constraints (1.15) can be represented at the level of the reduced model by using so-called proxy constraints (Zaytsev, 2015). Proxy constraints are an old approach to encode information within a black box which provides some metric as its output. The idea is simple, the proxy constraints try to encode the feasibility criterion in a different way, and thus we avoid the requirement to handle the whole-body variables.

In most of previous works, proxy constraints are defined by some rough approximations (box constraints, elliptic bounds, etc) leading to a certain conservatism; or it is simply ignored inside the reduced problem formulation. Footstep limits have been encoded by hyper-plane based on a dataset of robot success and failure inside a dynamic simulator (Perrin et al., 2012). Similar constraints can be obtained by training a neural network Orthey and Stasse, 2013. In Zaytsev, 2015, similar bounds are obtained by trial and errors based on stability analysis of the whole-body system.

In this chapter, we talk about using proxy constraints in order to encode the CoM position constraint (1.15a)

Ideally, constraining only the CoM position is not sufficient. It is also necessary to consider the constraints related to the contact forces Wieber, 2002 which must lie inside so-called friction cones, the capacity of robot to generate such value of angular momentum, etc. The main problem lies in the fact that it is hard to find analytic formulas to represent and express those constraints. The CoM constraint (1.15a), on the other hand, is static in nature, and thus has the potential to be encoded.

The central idea behind our approach is to represent proxy constraints by occupancy measures, whose corresponding cost of transport is optimized in the optimization problem. In Section 2.3, we propose a complete solution to learn the CoM feasibility constraint by off-line sampling the robot motion capabilities. The effectiveness of the approach is highlighted with two real experiments on the HRP-2 robot climbing stairs with or without using guardrail and one in simulation with the TALOS humanoid robot climbing stairs using guardrail, reported in Section 5.5.

2.1.2 Contact model

The interaction between a robot and the environment is defined through a set of contact points $\{\boldsymbol{p_k} \in \mathbb{R}^3, k = 1...K\}$. For instance, for a humanoid robot equipped with rectangular feet, the contact points correspond to the four vertices of the rectangular shape. At each contact point $\boldsymbol{p}_k$ is defined a contact force $\boldsymbol{f}_k$. In the case of unilateral contacts, $\boldsymbol{f}_k$ must lie inside a 3-dimensional quadratic friction cone $\mathcal{K}_k^3$, characterized by a positive friction coefficient μ_k. Fig. 2.1 depicts humanoid robots making contact with the environment.

We only consider here rigid contact interaction (contacting bodies are fixed) which is a reasonable assumption for modern legged robots which are mostly equipped with rigid soles.

A *contact phase* is defined by a constant set of contact points. In the context of bipedal walking, two examples of contact phases are the single and double support phases. As soon as a creation or a rupture of contact point occurs, the contact set is modified, defining a new contact phase. The concatenation of contact phases describes what we name a *contact sequence*, inside which all the contact phases have their own duration.

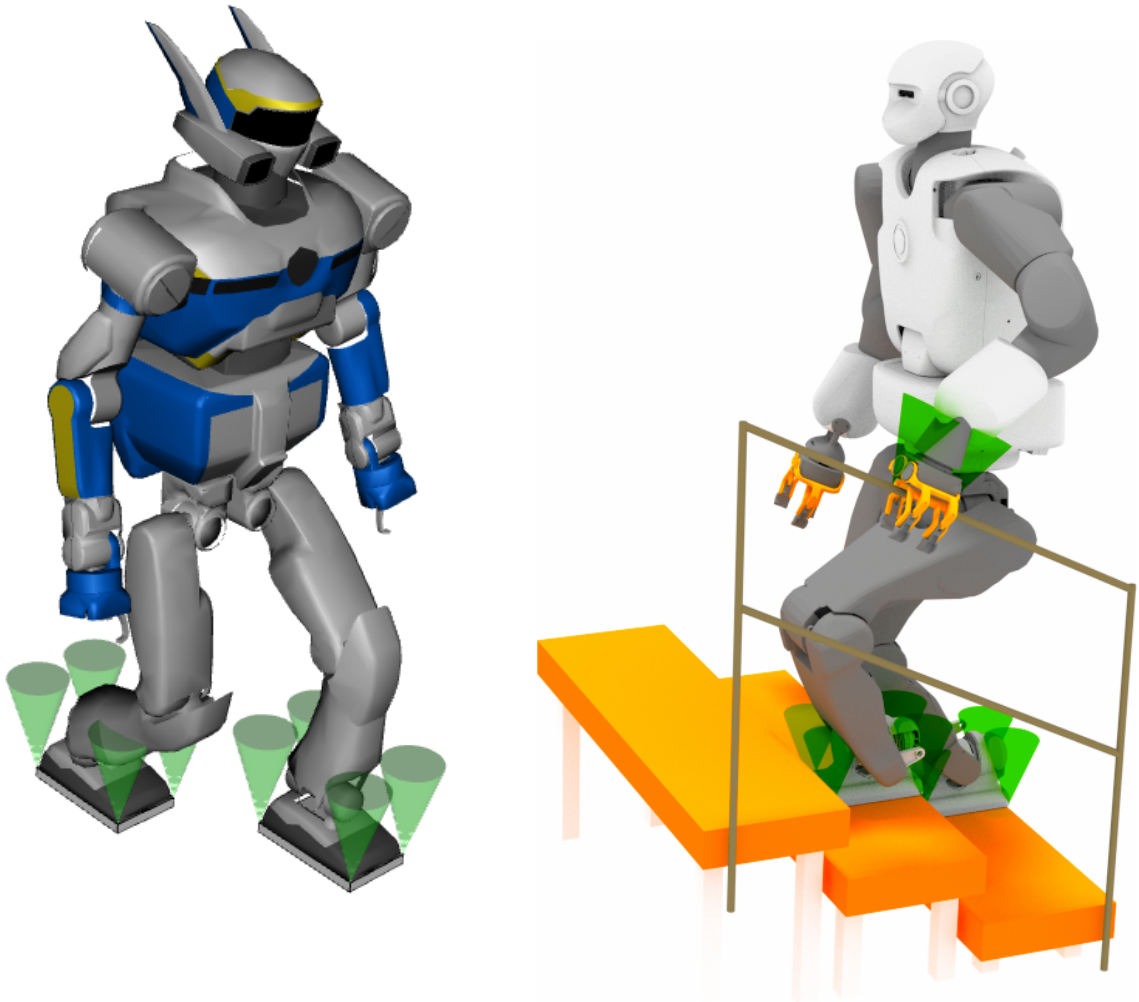


Fig. 2.1: Illustration of HRP-2 robot and TALOS robot making contacts with their environment. The green "ice-cream" cones are dispatched on the 4 vertices of the feet, symbolizing the friction cones with friction coefficient of value 0.3.

Computing automatically the contact sequence is a difficult problem (Bretl, 2006), but efficient contact planners now exist to compute it in a short amount of time (Escande et al., 2006; Tonneau et al., 2018b).

2.2 Feasibility of the centroidal problem

In this section, we present a mathematical coding of the feasibility constraints as probability measures. We then discuss the interest of this representation with respect to more-classical set-membership and show how it can be used to efficiently implement (1.15a) in the OCP. This section introduces the abstract definitions, that next section section uses to build the complete implementation.

2.2.1 Mathematical representation of feasibility constraints

Our objective is to efficiently implement the CoM coupling constraint (1.15a) in our OCP. This constraint explicitly depends on the robot configuration $\mathbf{q}$, which is not a variable of the centroidal OCP. A straight-forward implementation is to add the robot configuration in the variables of the OCP (Dai et al., 2014). However, this would surely lead the OCP to optimize the whole-body trajectory in order to handle all the robot constraints, which is yet not tractable especially if targeting real-time performances. We rather believe that it is possible to represent this constraint by an equivalent "proxy" constraint not dependent on the robot configuration.

Various ways to encode proxy constraints have been proposed in the literature. Most of them rely on set-membership. Denoting by γ the centroidal projection function:

$$\gamma : (\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}) \to (\boldsymbol{x}, \dot{\boldsymbol{x}}) = \gamma(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})$$

the proxy can be written as the constraint to have the state variables in the range space of γ. Set-membership proxies are used for instance in Herdt et al., 2010a; Deits and Tedrake, 2014 to encode maximal step size in biped walking, or in Dai and Tedrake, 2016 to bound the CoM position by simple geometric shape. In all these cases, the set boundaries are represented by very simple mathematical structures (typically linear inequalities) in order not to burden the OCP solver. Remarkably, there are few papers about the automatic synthesis of the set boundaries (Perrin et al., 2012; Orthey and Stasse, 2013; Zaytsev, 2015).

Despite its popularity, the set-membership representation has important drawbacks. First, it is often difficult to handle by the OCP solver, in particular when the feasible set is not convex. The boundary, which is a singular mathematical object, is also complex to describe or numerically approximate. Finally, the OCP solver often tends to saturate the set boundary, where the inverse kinematics γ^{-1} is likely to fail. Consequently, the set is often arbitrarily reduced to improve the robustness of the whole-body solution.

2.2.2 Proxy as occupancy measure

In this paper, we rather state that the proxy is best represented by the occupancy measure over $\boldsymbol{x}, \dot{\boldsymbol{x}}$.

Consider a trajectory $\underline{\mathbf{c}}$. With (1.15a), we want to maximize the likelihood that the inverse-kinematics solver converges on a trajectory $\underline{\mathbf{q}}$ such that $\underline{\mathbf{c}}$ is the image of $\underline{\mathbf{q}}$ by γ. For that purpose, it is desirable that to any state $\mathbf{c}$ corresponds as many robot

configurations as possible, so that the inverse kinematics is likely to converge to a solution $\mathbf{q}$ meeting continuity constraints.

We define the occupancy measure as the image of the uniform distribution in configuration space through the centroidal projection γ:

$$\mu_o(\tilde{\mathbf{c}}) \stackrel{\text{def}}{=} \int_{\tilde{\boldsymbol{q}} \text{ subject to } \gamma(\tilde{\boldsymbol{q}})=\tilde{\mathbf{c}}} d\tilde{\boldsymbol{q}} = \int_{\mathcal{Q}} \mathbb{1}_{\gamma(\tilde{\boldsymbol{q}})=\tilde{\mathbf{c}}} d\mu_{\mathcal{Q}}$$

where $\tilde{\mathbf{c}} \stackrel{\text{def}}{=} (\mathbf{c}, \dot{\mathbf{c}})$, $\tilde{\boldsymbol{q}} \stackrel{\text{def}}{=} (\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})$, $\mathcal{Q}$ is the whole-body motion range, $\mathbb{1}_a$ is the indicator function (i.e. 1 when the assertion a is true, 0 otherwise) and $\mu_{\mathcal{Q}}$ is the uniform distribution on $\mathcal{Q}$.

Measure μ_o has several properties of the set-membership representation. First, its support is the feasibility set, which means that μ_o contains at least as much information as the set boundaries. It indeed contains more information, as for example the level sets of μ_o can be used as boundaries of the inner of the feasibility set, used to improve the robustness. We will see in the experimental results that the resulting centroidal trajectories correspond to whole-body configurations which remain feasible with respect to the kinematic limits of the robot.

However, in practice, it is desirable that OCP (2.1) promotes centroidal states $\tilde{\mathbf{c}}$ where μ_o is the highest, not only feasible. First, it makes it easier to then compute a corresponding configuration $\tilde{\boldsymbol{q}}$. Second, the configuration is well inside the kinematic feasibility set, where redundancy will help the robot to handle disturbances.

Finally, the measure also eases the life of the OCP solver, compared to handling directly the feasibility set membership, as explained next.

2.2.3 Maximizing the occupancy measure

Before deriving an effective solution to represent μ_o for the specific case of the kinematic feasibility, we quickly show how μ_o can be integrated in the OCP (2.1).

In practice, the measure can be normalized and represented by the corresponding probability density function (PDF), denoted by $p(\mathbf{c}, \dot{\mathbf{c}})$. It is then possible to directly exploit the measure to represent the set-membership constraint (by imposing the integral of the measure to be positive on any small neighborhood around the trajectory). In addition, we could use the PDF to directly optimize the robustness, either by optimizing over a level set of the PDF, or by maximizing the neighborhood around the trajectory where the measure is nonzero.

However, adding a PDF as a constraint of an OCP is not straightforward. Therefore, we propose to remove the hard constraint (1.15a) and penalize the OCP cost with the log PDF.

In practice, the logarithm prevents the solver from selecting non-feasible $\mathbf{c}$ states. Constraints (1.15a) is always satisfied. It also penalizes non-robust behavior where no redundancy $\mathbf{q}$ is available, and avoids saturation of the hard constraint. Finally, the OCP solver is gently pushed away from the constraint, instead of searching for a solution living on the boundaries, which greatly improves its efficiency. Furthermore, it is unlikely that the OCP solver is trapped in local minima of μ_o, as it manipulates a full trajectory $\underline{\mathbf{c}}$ and not a single state $\mathbf{c}$. Experimentally, we observed that our OCP solver robustly computes a good local minimum when optimizing over a cost penalizing the log-PDF, while it is unlikely to converge to a solution when optimizing over set-membership.

2.3 Learning the CoM reachability proxy

We now present a complete solution to efficiently approximate the CoM feasibility, i.e. for any time t, there exists a joint configuration $\mathbf{q}(t)$ such that (i) the contact placements are respected and (ii) the CoM of the poly-articulated system matches $\mathbf{c}(t)$. Handling this sole constraint first is a proper way of validating our approach. It is also interesting in practice, as the feasibility of the CoM is the most limiting constraint. Generalization to velocity and acceleration of the CoM with respect to joint velocity and acceleration limits would be straight-forward. Extension to the construction of the proxy on the torque limits is left as a perspective.

2.3.1 Probabilistic model

The geometric condition can be stated as the *conditional probability* of the CoM to be at the position $\mathbf{c}$ given the current set of K contact points $\{\boldsymbol{p_k} \in \mathbb{R}^3, k = 1...K\}$. This probability is denoted by $p(\mathbf{c}|\boldsymbol{p}_k, k = 1...K)$. It lives in the high dimensionality domain $\mathbb{R}^{3(K+1)}$ and it is hard to compute in general.

The probability domain can be exactly reduced by gathering together the contact points belonging to the same rigid end-effector (e.g., the 4 vertices of the humanoid foot belongs to the same end-effector). We denote by $M_i = (R_i, p_i) \in SE(3)$ the placement (position and orientation) of the contact body i. The conditional probability is then reduced to $p(\mathbf{c}|M_i, i = 1...K_c)$ where K_c is the number of end-effectors in contact.

We now assume that variables M_i are all independent. This assumption is clearly abusive, however is a reasonable approximation under knowledge of $\mathbf{c}$. It is later discussed. Under this assumption, the conditional probability reads:

$$p(\mathbf{c}|M_i, i = 1...K_c) \propto \prod_{i=1}^{K_c} p_i(\mathbf{c}) \tag{2.3}$$

where $p_i(\mathbf{c})$ stands for $p(\mathbf{c}|M_i)$ and $\propto$ stands for "is proportional to". $p_i(\mathbf{c})$ is nothing more than the probability distribution of the CoM to be at position $\mathbf{c}$ w.r.t the frame defined by M_i.

The assumption of independence of the M_i is commonly employed inside the machine-learning community as a trick to make the problem numerically tractable. In this particular case, it simplifies a lot the learning process: instead of working in a high dimensional space, the problem is restricted to a subset of $\mathbb{R}^3$. In addition, the independence of end-effector placements plays the role of an upper-bound for the real probability: if a CoM is not feasible for at least one of the end-effectors (i.e. one of the $p_i(\mathbf{c})$ is equal to 0), then the joint probability is also zero. The converse is not true. We show in next section that this approximation, although intuitively rough, is quite reasonable in practice and leads to good experimental results.

2.3.2 Kernel density estimation by CoM sampling

There is in general no closed form to encode $p_i(\mathbf{c})$ for a particular legged robot. Nevertheless, this conditional probability can be easily approximated by extensive dataset of the CoM position expressed in the end-effector frames.

Computing a dataset of the CoM position expressed in the frame M_i does not raise particular difficulties. A configuration q_a of the actuated joints is randomly sampled and the corresponding CoM position is computed (expressed in placement frame) by forward kinematics. The sample is rejected if joint limits or self collision are violated. Thus, $\mathrm{N}_{\mathrm{samples}}$ successful joint configuration samples correspond to each of the points inside the CoM dataset.

The probability distribution can be approximation from the cloud of CoM points by the kernel density estimators (KDE) (Parzen, 1962). KDE are in some sense the analogues of histograms but for continuous domains: for each point of the data set, it associates one kernel centered on the point and all kernels share the same parameters. In the present work, we use isotropic Gaussian kernel.

2.3.3 Reduction of dimension

One drawback of the KDE representation is its computational complexity: evaluating the exponential function contained in the Gaussian kernel takes around $10\,$ns on modern CPU. So, roughly speaking, evaluating the PDF of the KDE takes approximately $10.\mathrm{N}_{\text{samples}}$ ns which becomes rapidly a bottleneck when the number of points is huge ($\mathrm{N}_{\text{samples}}$ greater than 100 points).

We propose to then approximate the KDE by a Gaussian mixture model (GMM) Bishop, 2006. GMMs are particularly suited to approximate a PDF with only few Gaussians in the mixture. The GMMs are learned for each end-effector from the corresponding cloud of samples by means of the expectation-maximization (EM) algorithm Dempster et al., 1977.

The quality of the GMM approximation can be estimated using the Kullback-Leibler (KL) divergence between the KDE (ground-truth) and the learned GMM (approximation) using the Monte Carlo estimator proposed in Hershey and Olsen, 2007. Depending on the number of Gaussians in the mixture, the divergence can reveal under or over fitting effects. The optimal number of Gaussians is easily selected for each end effector by dichotomy, as exemplified in next section.

2.3.4 Summary of the learning procedure

In summary, for each end effector, N_{samples} configurations are sampled and the corresponding CoM is computed in the end-effector frame. The resulting KDE is approximated by fitting a GMM using EM. Finally, the probability of CoM occupancy is approximated as the product of $p_i(\mathbf{c})$, for i the end effectors in contact with the environment.

2.3.5 Proposed optimal control formulation

We can now express the complete formulation of the cost function ℓ_s.

$$\ell_c(\boldsymbol{x}, \boldsymbol{u}) = w_x \|\dot{\boldsymbol{x}}\|^2 - \sum_{i=1}^{K_c} \log(p_i(\mathbf{c})) \tag{2.4}$$

where the first term[1] enforces a smooth trajectory, the second term is the cost of transport for the approximate CoM occupancy measure, and w_x weights their relative importance. The first term can similarly be interpreted as a weak formulation

[1] $\|\dot{\boldsymbol{x}}\|$ is a function of $\boldsymbol{x}$ and $\boldsymbol{u}$ through $\dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u})$

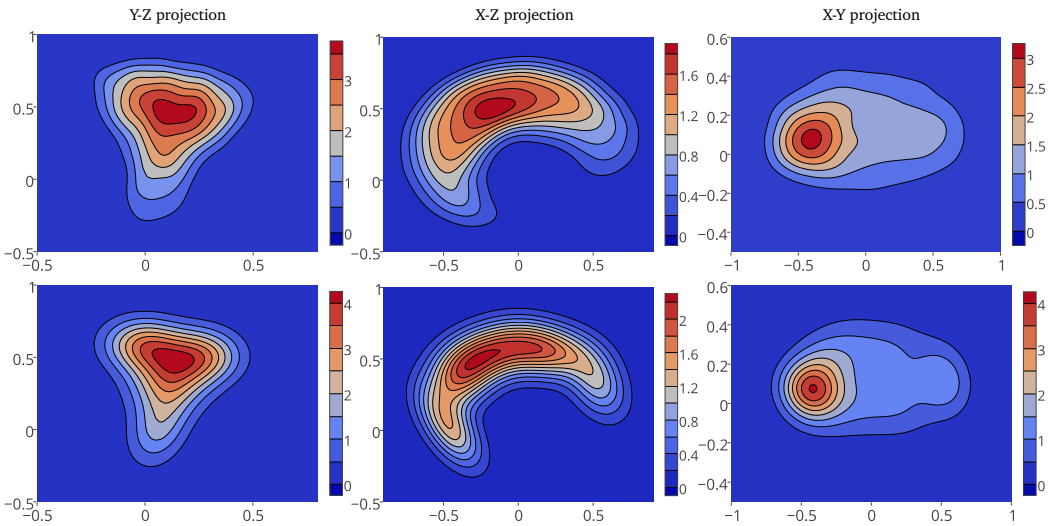


Fig. 2.2: Illustration of the probability density distribution of the CoM w.r.t. the right foot frame of HRP-2. The PDF are projected along the three axis X,Y,Z and represented by the means of color map: the low values are closed to the blue colour while the high values tend to be more red. The first row corresponds to the ground truth distribution estimated through KDE. The KDE is composed of 20000 points. The second row is the colour map of the GMM used in the OCP and composed of 7 Gaussian kernels.

of the occupancy measures for the second order terms ($\dot{\mathbf{c}}$ and $\boldsymbol{L}$) and their derivatives, through centred Gaussian measures (i.e. no prior on occupancy distribution). If the complete occupancy measure μ_o is available, the first term would become useless.

2.4 Results

2.4.1 Illustration of the learning procedure

We first illustrate the learning procedure exposed in Sec. 2.3 on the HRP-2 robot. For that purpose, we only expose for space reasons the learning of the accessibility space of the CoM w.r.t. the right foot (RF). A similar study can be conducted on the three other end-effectors.

The learning process is made from a set of 20000 points sampled uniformly in the configuration space. The KDE of this set is represented on the first row of Fig. 2.2. The first observation is that the PDF of the RF is not convex and follows a kind of banana distribution on the X-Z plane. In other words, this means that the distribution cannot be approximated by a single normal distribution but must be composed of several ones. The second row of Fig. 2.2 represents the colour map of the GMM

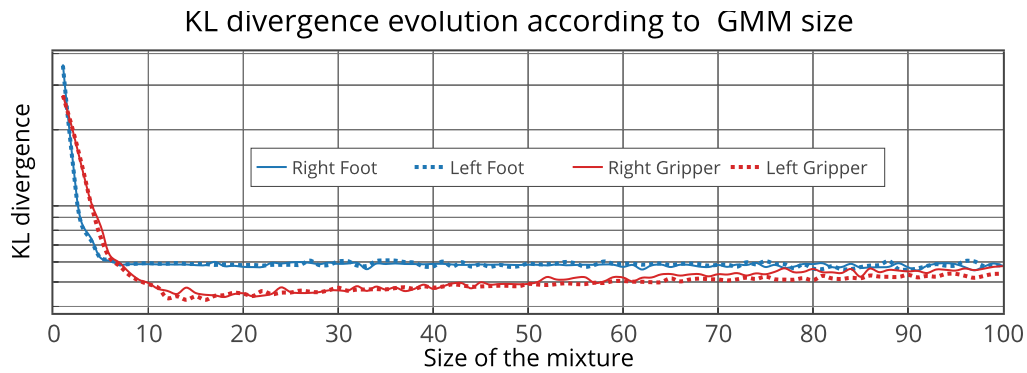


Fig. 2.3: Evolution of the KL divergence between the KDE distribution and GMMs of different sizes for the four end-effectors of the HRP-2 robot.

used inside the OCP. At this stage, it is important to notice that the approximation with GMMs does not fit perfectly the maximal values of the real distribution. However, this approximation is conservative with respect to the support and the level sets of the original distribution.

Fig. 2.3 highlights the experimental procedure suggested in Sec. 2.3.3 and shows the evolution of the KL-divergence with respect to the size of the GMMs. For the right and left feet, the KL-divergence stagnates from 7 kernels in the mixture. In other words, it is sufficient to takes a GMM of size 7 to represent the CoM distribution in the foot frames. For the right and left grippers, it is a little bit different. The KL-divergence first decreases and then increases from 14 kernels. This behaviour can be explained by the fact that the EM algorithm does not optimize the KL divergence but the likelihood of observation (expectation). We chose to represent the CoM distribution w.r.t. the grippers with a GMM of size 14.

A similar study has been done on the TALOS humanoid robot, which is larger than HRP-2 and as different leg and arm kinematics. The distributions for the right foot of TALOS is depicted in Fig. 2.4.

2.4.2 Experiments on HRP-2 and Talos robots

This part reports the experiments achieved on the HRP-2 robot in real conditions, and Talos in simulation. We show two real experiments of multicontact locomotion with the HRP-2 robot inside an environment similar to what can be found in the industry. Simulation on Talos are also done in the same environment.

We use the Loco3D Pipeline, as explained in Sec 1.4, in order to produce both real and simulated motions on the two different robots. In these experiments, we use a second order IK solver similar to Saab et al., 2013 to get the whole-body resolution.

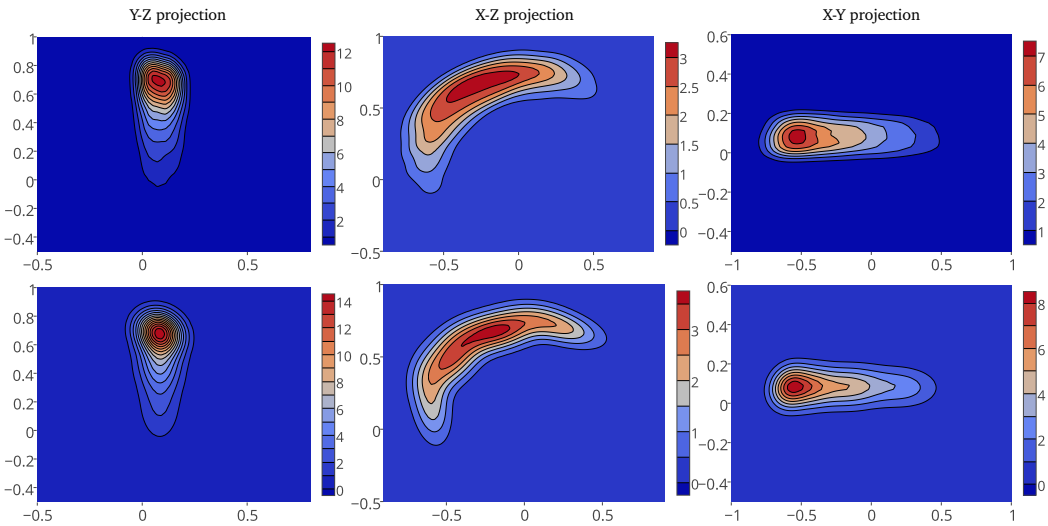


Fig. 2.4: Illustration of the probability density distribution of the CoM w.r.t. the right foot frame of TALOS robot. The PDF are projected along the three axis X,Y,Z and represented by the means of color map: the low values are closed to the blue colour while the high values tend to be more red. The first row corresponds to the ground truth distribution estimated through KDE. The KDE is composed of 20000 points. The second row is the colour map of the GMM used in the OCP and composed of 4 Gaussian kernels. The axes have the same scale as in Fig 2.2.

In the following Chapter 3 and Chapter 4, we will develop a whole-body solver based on the principle of trajectory optimization for multi-contact scenarios.

Experiment 1 - climbing up 10-cm high steps

The experimental setup is an industrial stairs made of six 10-cm high steps. The steps have a length of 30 cm. The duration of the single and double support phases are 1.4 s and 0.2 s respectively. The resulting motion is depicted in Fig. 2.6. During execution, the reference posture is tracked as well as the reference foot forces using the robot low-level control system (named HRP "stabilizer").

Computing the 25s of motion takes 42 iterations of the multiple-shooting algorithm, costing about 8s in total. In average, each iteration takes approximately 0.2s for 25s of motion. About 70% of the computation time is spent solving the underlying quadratic program of the multiple-shooting algorithm and other 20% are dedicated to the numerical integration of the dynamics together with the computations of sensitivities (derivatives).

Fig. 2.5 shows two trajectories of the CoM projected in the right foot frame: the black curve takes into account the log-pdf term in the cost function, while the green one does not. The figure also includes the level sets of the GMM of right foot (depicted in Fig. 2.2). It appears that the OCP tends to maximize the inclination of

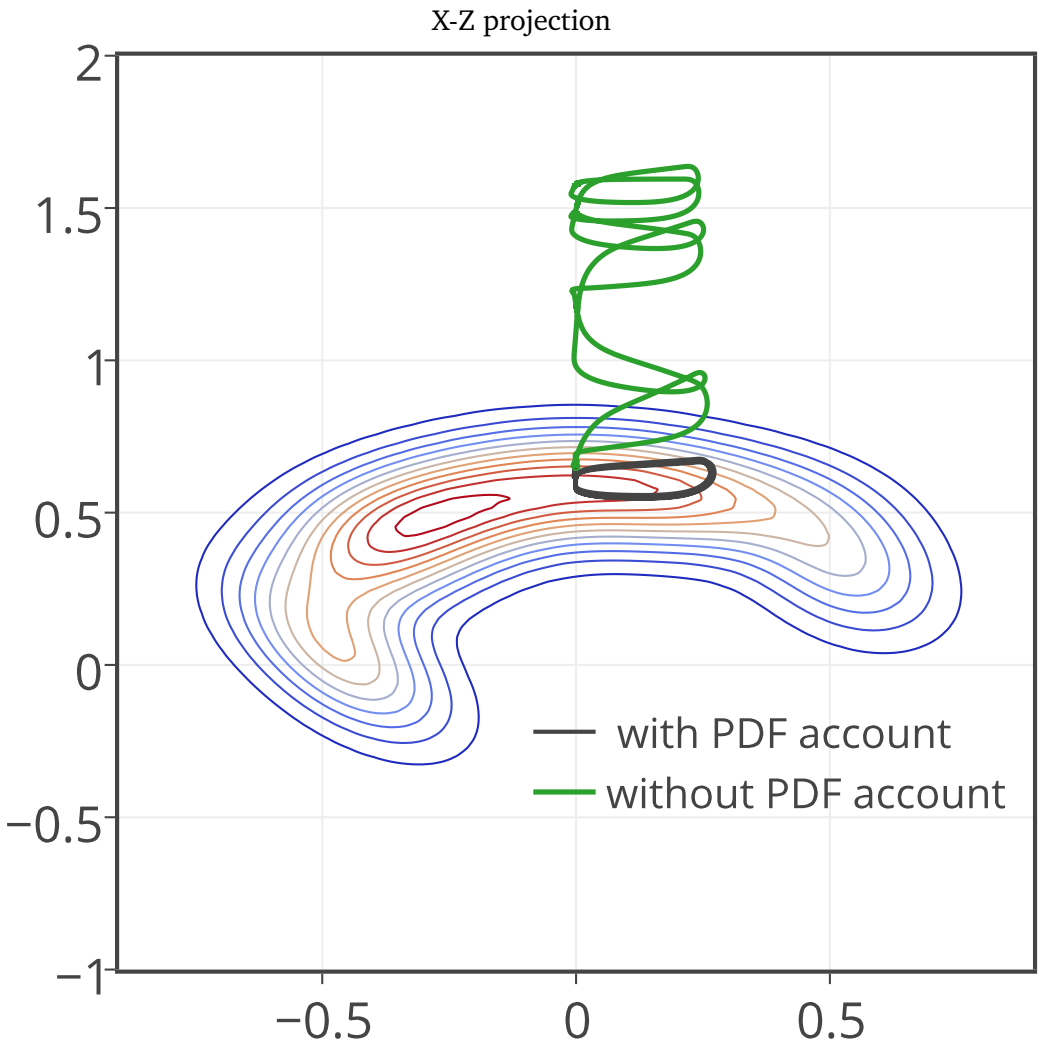


Fig. 2.5: Projection of the CoM trajectory inside the right foot frame with and without taking into account the log-pdf term in the cost function. The level set corresponds to the GMM distribution used in our OCP.

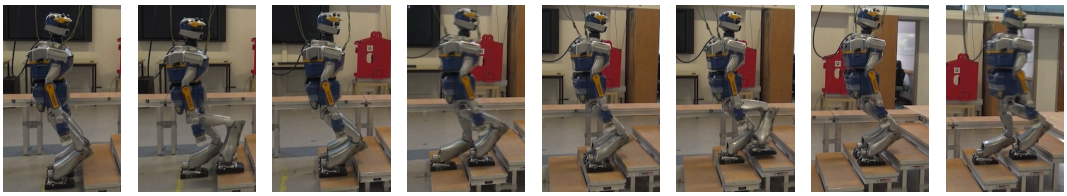


Fig. 2.6: Snapshots of the climbing up 10-cm high steps motion with the HRP-2 robot.

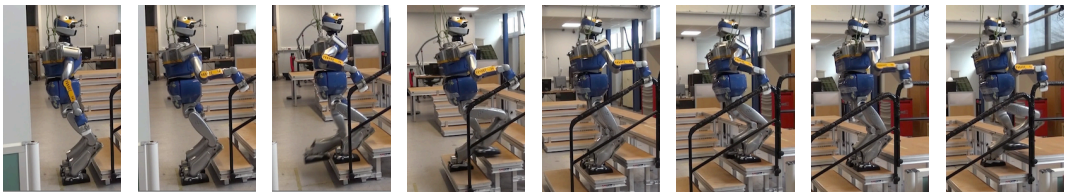


Fig. 2.7: Snapshots of the climbing up 15-cm high steps motion with the HRP-2 using the guardrail.

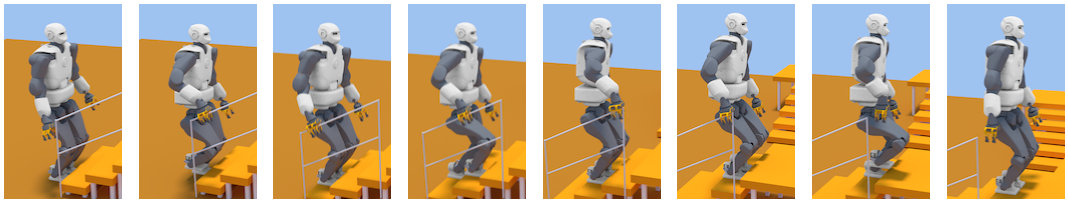


Fig. 2.8: Snapshots of the climbing 15-cm high steps motion with guardrail by the TALOS robot in simulation.

the CoM to stay in the most feasible region, i.e. closed to the maxima of the PDF. On the contrary, if we do not add the log-pdf term, the CoM tends to be infeasible.

Experiment 2 - climbing up 15-cm high steps with guardrail support

The experimental setup is another industrial stairs made of four 15-cm high steps and equipped with a guardrail. The steps have a length of 30 cm too. The duration of the double and triple support phases are 1.8 s and 0.4 s respectively. Here, the double support phases correspond either to the case of two feet on the steps or one feet plus the right gripper on the handrail. Snapshots of the entire motion are shown in Fig. 2.7.

2.4.3 Experiments in simulation

We reproduce the climbing stairs with guardrail scenario, but this time with the TALOS robot in simulation. Compared to HRP-2, TALOS is a 1.78m high humanoid robot weighting around 100kg. For this experiment, only the end-effector trajectories and the GMMs are different: the cost function remains the same. The complete motion is depicted in Fig. 2.8.

2.5 Conclusion and Perspective

In this chapter, we propose a method to solve the first half of our contact-constrained problem (1.16), i.e., the part on the reduced variables $\mathbf{c}, \dot{\mathbf{c}}, \boldsymbol{\lambda}$ that contains information on underactuated which is the main difficulty of the locomotion problem.

We introduce a systematic approach to include feasibility constraints inside the optimal control formulation as occupancy measure. In particular, we propose an effective way to learn the Center Of Mass feasibility constraint by learning the probability density of the Center Of Mass positions with respect to the end-effector locations.

This allows us to use our centroidal pattern generator to run before the whole-body optimal control problem, thanks to the encoded proxy constraint.

In this first chapter, we have used a second order IK solver (Saab et al., 2013) for whole-body resolution, and we demonstrate the validity of the methods with two real experiments on the HRP-2 which was asked to climb industrial stairs with or without handrails and one experiment in simulation with the TALOS platform which was asked to achieve multicontact stairs climbing.

Even though second order IK lack the benefits provided by trajectory optimization, our experiments show that this approach can be used to generate robust whole-body trajectories. While the whole-body resolution is limited in its ability (for e.g. to generate proper angular momentum trajectory), the robust trajectories provided by the centroidal solver are still able to be applied on the robot.

The methodology requires a systematic learning procedure to be executed off-line in simulation. On-line, the resulting optimal control is solved in a very efficient way (with each iteration running about 100 times faster than execution time) and it leads to smooth centroidal trajectory easily tracked by the robot whole body.

We have defined our proxy to be an occupancy measure over the whole centroidal state and contact forces, although only the measure over the Center Of Mass was approximated. If proxy constraints are used in the centroidal problem, it is possible to completely account for the coupling constraints (1.15) and not have to worry about the problem of solution feasibility. Introducing the occupancy measure over all centroidal variables would reduce the centroidal locomotion problem to a simple optimal control problem composed of a single cost function, with only initial constraints.

Defining and learning such proxy constraints for angular momentum and contact forces is not easy because of their dynamic nature, and thus we only learn the Center Of Mass kinematic feasibility. However, even with Center Of Mass feasibility, we have seen good performance of on a real robot. Even though we are not ensuring that the angular momentum and contact forces constraints are satisfied, maximization of the Center Of Mass expectancy gives latitude to the whole-body solver to find angular momentum trajectories and still be valid.

We will see in Chapter 5, how to consider not just the Center Of Mass, but all the coupling constraints (1.15) for our solution. But even in that case, we will use the Center Of Mass proxy that we define in this chapter to provide a robust initial centroidal trajectory. In this way, we will make sure that our initial guess for the full solver of Chapter 5 is good and robust.

3

Differential Dynamic Programming for Multi-Phase Rigid Contact Dynamics

सच है, विपत्ति जब आती है,
कायर को ही दहलाती है,
सूरमा नही विचलित होते,
क्षण एक नहीं धीरज खोते,
विघ्नों को गले लगाते हैं,
काँटों में राह बनाते हैं।
(It is true, that when obstacles arise, only the weak falter. The brave embrace the obstacles, and find a way forward even amongst the thorns.)

— **Ramdhari Singh 'Dinkar'**
(Rashmirathi, Third canto)

Budhiraja, Rohan, *Justin Carpentier*, *Carlos Mastalli*, and *Nicolas Mansard* (2018). Differential Dynamic Programming for Multi-Phase Rigid Contact Dynamics. In: 2018 IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids). IEEE, pp. 19.

3.1 Introduction

Let us recall that in Chapter 1, we decoupled our locomotion problem into its reduced dynamics ($\mathbf{d}_c$), and whole-body dynamics ($\mathbf{d}_l$). In Chapter 2, we dealt with the first half of the problem, and showed how $\mathbf{d}_c$ can be handled. Let's now consider the second half of the problem, the variables $\mathbf{d}_l$.

In this chapter, we propose to combine the advantages of centroidal dynamics optimization (to decide the gait, the timings and the main shape of the centroidal trajectory) with a whole-body trajectory optimizer based on multi-phase rigid contact dynamics.

An acute observer might say that we have already shown in Chapter 2 that whole-body variables can be decided instantaneously, using inverse kinematics(i.e. a trivialized form of the optimal control problem that we later present in (3.2), and in which the horizon length has been collapsed to zero). So why should we now consider the much more complex optimal control problem over a time horizon? We argue in this chapter that combining the reduced OCP with inverse kinematics or inverse dynamics is not satisfactory to handle the AM. In what follows, we first discuss the importance of properly handling the angular momentum during locomotion, before introducing the DDP method.

3.1.1 On the importance of angular momentum

Consider an astronaut, floating in space, without any external forces. If he/she mimics the normal human walk, he/she will start spinning in his/her sagittal plane. The reason for this is simple, the forward gait is not symmetric with the backward gait. Thus, while the instantaneous angular momentum is conserved, the net velocity change is not zero. Indeed, as seen in Fig. 3.1, an astronaut is able to re-orient himself just by utilizing this mismatch between the upper and lower limbs. Cats are capable of a change of orientation when falling. This is explained more in Fig. 3.2.

Contact forces are not the only way to change the robot orientation. It is known from Wieber, 2006a that robot orientation can be controlled without the need of contact forces (i.e. only with the internal joint actuators). Under the action of only internal forces, the AM conservation can be seen as a non-holonomic constraint on the robot orientation. Of course, one can design a control law that counterbalances the lower-body AM. However this will create tracking errors (and potentially instabilities) without mentioning the cases where the arms need to be used for multi-contact locomotion. In fact, as shown in Brockett, 1983, a system under non-holonomic constraints cannot be controlled with a time-invariant feedback law. Thus AM requires a preview control strategy to be correctly regulated or triggered.

[1]Wieber, 2006a showed to robotics community the work by Frizot, 2001, where he captures the movement of a cat falling to the ground. In this thesis, we give the same explanation as Wieber, 2006a, but instead we go to contemporary youtube for better camera resolution and lighting :)

Fig. 3.1: An astronaut rotating himself in the transverse plane. There is a mismatch in the inertia properties of the upper and the lower limbs. As a result, the astronaut can create different velocities for the upper and lower limbs, while maintaining the angular momentum conservation constraint. Eventually, this changes the orientation of his body. Video thanks to NASA, 1988

Fig. 3.2: A cat falling and correcting its orientation before he lands. The cat is able to create two distinct groups of motion along its spine. During the first half, it creates a mismatch between its upper and lower body by closing its arms and elongating its legs. Then it rotates them in different directions, thus maintaining the angular momentum constraint (3.1), but changing the orientation of the spine. It completes its rotation by repeating the same movement, but in the opposite sense (Video thanks to SmarterEveryDay, 2012)[1]

It is often (wrongly) understood that centroidal optimization provides the answer to this problem. The centroidal optimizer can neither anticipate nor modify the limb movements in order to change the AM as needed. For instance, the centroidal optimizer cannot anticipate a high demand of the linear part (CoM) by delaying the limb movement, or exploit the movement of the arms to compensate for large forces acting for a short duration. Nonetheless, these methods are still valid since they provide an efficient way to compute the CoM motion while keeping balance and avoiding slippage. Indeed, AM of a body is accounted by both, as a result of the contact forces, and by the limb movement. Consider a floating-base robot. With no external forces acting on the robot, a constant AM is maintained by the non-holonomic constraint on the joint velocities (Wieber, 2006a):

$$\sum_{k=0}^{n_j} m_i[\mathbf{r}_k - \mathbf{r}]_\times \dot{\mathbf{x}}_k + \mathbf{R}_k \mathbf{I}_k \boldsymbol{\omega}_k = \text{Constant}, \tag{3.1}$$

where k denotes the index of a rigid limb and $\mathbf{I}_k$ corresponds to its inertia matrix expressed in the body's CoM frame. $\dot{\mathbf{x}}_k$ and $\boldsymbol{\omega}_k$ are the linear and angular velocities of the links.

In order to produce a relevant movement of the body, this "gesticulation" of the limbs needs to be accounted for in the contact forces. This can only be done during the centroidal optimization (Wieber, 2006a), since contact force optimization is in the domain of the centroidal optimization. However, joint movement optimization is in the whole-body domain, and this makes it a difficult problem to solve in real-time.

In the past, we have assumed that this effect is small Carpentier et al., 2017a, but yet important to consider, and we track it with a whole-body Model Predictive Control (MPC). DDP (Mayne, 1973) is a reasonable choice between the two, because it allows us to generate additional AM using (3.1), while efficiently tracking the reference trajectories provided by the centroidal solver. DDP has already been shown (Tassa et al., 2012) to be efficient in solving online Optimal Control (OC) in legged systems.

3.1.2 Multi-contact Motion Generation

In this chapter, we consider the problem related to the wholebody variables, $\mathbf{d}_l := [\mathbf{q}, \dot{\mathbf{q}}, \boldsymbol{\tau}]$. More precisely, we consider that the contact sequence $\mathbf{S}$ is already given; thus we know the surfaces which are in contact with the environment, the order of the contacts, the timing of each phase of contact etc. We are dealing with the clas-

sical hypothesis defined by hybrid dynamics, that encompasses gaited/non-gaited locomotion as long as the contact dynamics, impact dynamics etc are defined.

Instead of relying on instant linearization using Inverse Kinematics (IK)/Inverse Dynamics (ID), we propose in this chapter to rely on optimal control (Lengagne et al., 2013; Schultz and Mombaur, 2010). Namely we rely on DDP, to compute the whole-body motion while tracking the centroidal trajectory.

Generating a whole-body trajectory requires finding a trajectory which is subject to dynamic-consistency, the friction-cone constraints, the self-collision avoidance and the joint limits. While dynamic-consistency creates dependence between the centroidal solver and the whole-body solver, the other constraints are dependent only on the whole-body state. If we look at the contact-constrained formulation of our optimization problem (1.16) from Chapter 1, the whole-body part of our optimization problem is given by:

$$
\begin{aligned}
\underset{\underline{\mathbf{d}}_l := [\mathbf{q}, \dot{\mathbf{q}}, \boldsymbol{\tau}]}{\text{minimize}} \quad & \sum_{s=1}^{S} \int_{t_s}^{t_s + \Delta t_s} \ell_l(\mathbf{d}_l | \mathbf{S})\, dt \\
\text{subject to} \quad & \forall t \quad g_\lambda(\mathbf{q}, \dot{\mathbf{q}}, \boldsymbol{\tau}) \in \mathcal{K} && \text{(3.2a)} \\
& \forall t \quad \mathbf{q} \in \mathcal{Q} && \text{(3.2b)} \\
& \forall t \quad \dot{\mathbf{x}}_l = f_l(\mathbf{d}_l) && \text{(3.2c)} \\
& \quad\quad \text{Coupling Constraints (1.15) are satisfied} && \text{(3.2d)} \\
& \quad\quad \mathbf{x}_l(0) \text{ is given}, \mathbf{x}_l(T) \text{ is viable} && \text{(3.2e)}
\end{aligned}
$$

$\mathcal{Q}$ here denotes the admissible set of non-colliding, joint safe configurations. Note that while constraint (3.2b) is not mentioned in the optimization problem (1.16) for the sake of simplicity and clarity, adding it here does not alter in any way our original explanations of decoupling (Sec 1.2), coupling constraints (Sec 1.3.2), and how to handle them (Sec 1.3.3).

The rest of the chapter is organized as follows: Sec 3.2 briefly introduces the DDP algorithm, we then describe our novel DDP formulation for rigid contact dynamics in Sec 3.3. Then, in Sec 3.4 we show experimental trials and realistic simulation with the HRP-2 robot and compare them against a whole-body IK solver. Lastly, Sec 3.5 summarizes the work conclusions.

3.2 Differential Dynamic Programming

In this section, we give a formal description of the DDP algorithm for completeness. The reader is referred to Mayne, 1973 for full analysis.

DDP belongs to the family of OC. It is a sparse algorithm, and deals specifically with sparsity provided by the Markovian structure of our dynamics constraint (3.2c). It does so by using the Bellman's principle to convert our optimization problem (3.2) into a recursive sequence of mini-optimizations. Our interest in DDP lies in its ability to do highly sparse resolution of our optimization problem.

3.2.1 Relaxation of Constraints

Constrained optimization like (3.2) poses challenges at arriving at fast solutions. DDP is more suited to problems with only initial value and dynamics constraints. In order to exploit the true potential of DDP, we'll try to relax these constraints into something manageable. We'll handle the coupling constraints (1.15) as a trajectory tracking task. This is straightforward, since the whole-body optimization would be following on the heels of centroidal optimization. In the upcoming Sec 3.3, we'll see how to modify our dynamics to track a contact force reference trajectory. Thus, we'll remove (3.2a) as well. (3.2b) would be implemented by a penalty cost. (3.2c) would be handled by our DDP optimization algorithm itself. We replace the viability constraint on $\mathbf{x}_l(T)$ with a penalty on zero terminal velocity. With this adjustment, we drop the viability constraint from (3.2e).

Moreover, in order to do numerical optimization, we discretize our dynamics (3.2c). The control trajectory, which is $\underline{\boldsymbol{\tau}}$, is divided into a grid of T points, and the trajectory between any two consecutive grid points t and $t+1$ is taken to be $\boldsymbol{\tau}(t)$. For conformity with state and control descriptions in the literature, we'll be using $\mathbf{x}_t := \mathbf{q}(t), \dot{\mathbf{q}}(t)$ and $\mathbf{u}_t := \mathtt{S}\boldsymbol{\tau}(t)$ to define the discretized whole-body state and control variables at time t. $\mathtt{S}$ here is the selection matrix that maps to the actuated part of (1.12). We'll define f_t as the discretized version of our continuous dynamics at a given grid point t.

Thus our optimization problem takes the following generic form:

$$
\begin{aligned}
\underset{\underline{\mathbf{u}}}{\text{minimize}} \quad & \sum_{t=0}^{T-1} \ell(\mathbf{x}_t, \mathbf{u}_t) + \ell(\mathbf{x}_T) \\
\text{subject to} \quad & \mathbf{x}_0 \text{ is given} \\
\forall t \in \{0 \dots T-1\} \quad & \mathbf{x}_{t+1} = f_t(\mathbf{x}_t, \mathbf{u}_t) \qquad (3.3a)
\end{aligned}
$$

(3.3a) defines our discretized dynamics at time t. Notice that there is also change in the variable of optimization between (3.2) and (3.3). We are only considering optimization in $\mathbf{u}$. Since the first state $\mathbf{x}_0$ is defined, the optimal $\mathbf{u}_t$ decisions would result in optimal state values $\mathbf{x}_t$ by using the discrete dynamics (3.3a). This is why DDP is called a "shooting" method (we are "shooting" from initial $\mathbf{x}_0$).

3.2.2 Understanding Bellman's Principal of Optimality

Bellman's principle of optimality states the following:

Principle of Optimality

An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision. (Bellman1962)

Thus, the principle of optimality gives us two tools.

The first tool is that we don't consider the past when optimizing our problem from a given state. Consider that we are currently at state $\mathbf{x}_i$. Let us take some decisions $\mathbf{u}_{i:T-1}$, to arrive at the final state. We can thus define a cost-to-go, which is the net objective cost of (3.3) because of these decisions:

$$
J_i(\mathbf{x}_i, \mathbf{u}_{i:T-1}) = \sum_{t=i}^{T-1} \ell(\mathbf{x}_t, \mathbf{u}_t) + \ell(\mathbf{x}_T) \qquad (3.4)
$$

This is possible through a forward simulation of the system dynamics (3.3a). Bellman's principle states that the optimal trajectory from state $\mathbf{x}_i$ only depends on

finding the optimal $\mathbf{u}_{i:T-1}$. Thus, we can define the value function V_i, which describes the minimum cost-to-go:

$$V_i(\mathbf{x}_i) = \min_{\mathbf{u}_{i:T-1}} J_i(\mathbf{x}_i, \mathbf{u}_{i:T-1}). \tag{3.5}$$

Another tool is the observation that if we consider the optimal trajectory that starts at $\mathbf{x}_0$ and goes to $\mathbf{x}_T$, any sub trajectory $\mathbf{x}_i \dots \mathbf{x}_T$ of this optimal trajectory is optimal.

Thus, instead of finding the entire optimal trajectory (3.5), we make recursive decisions:

$$V_i(\mathbf{x}_i) = \min_{\mathbf{u}_i} [\ell(\mathbf{x}_i, \mathbf{u}_i) + V_{i+1}(\mathbf{f}(\mathbf{x}_i, \mathbf{u}_i))], \tag{3.6}$$

Each feasible state $\mathbf{x}_i$ corresponds to a Value V_i which is related to the optimization problem (3.3). We are looking for the Value V_0 which is the solution of (3.3) at state $\mathbf{x}_0$.

3.2.3 Backward Pass

DDP searches locally the optimal state and control sequences of the above problem (3.6). Let $\mathbf{Q}(\delta\mathbf{x}, \delta\mathbf{u})$ be the variation of $\ell(\mathbf{x}_i, \mathbf{u}_i) + V_{i+1}(\mathbf{f}(\mathbf{x}_i, \mathbf{u}_i))$ around the current state control pair $\mathbf{x}_i, \mathbf{u}_i$. i.e.

$$\mathbf{Q}(\delta\mathbf{x}, \delta\mathbf{u}) = \ell(\mathbf{x}_i + \delta\mathbf{x}, \mathbf{u}_i + \delta\mathbf{u}) - \ell(\mathbf{x}_i, \mathbf{u}_i) + V_{i+1}(\mathbf{f}(\mathbf{x}_i + \delta\mathbf{x}, \mathbf{u}_i + \delta\mathbf{u})) - V_{i+1}(\mathbf{f}(\mathbf{x}_i, \mathbf{u}_i)) \tag{3.7}$$

If we take the quadratic approximation of this quantity, and drop the higher order terms, we get the following relation:

$$\mathbf{Q}(\delta\mathbf{x}, \delta\mathbf{u}) \approx \begin{bmatrix} 1 \\ \delta\mathbf{x} \\ \delta\mathbf{u} \end{bmatrix}^\top \begin{bmatrix} \mathbf{0} & \mathbf{Q_x}^\top & \mathbf{Q_u}^\top \\ \mathbf{Q_x} & \mathbf{Q_{xx}} & \mathbf{Q_{xu}} \\ \mathbf{Q_u} & \mathbf{Q_{ux}} & \mathbf{Q_{uu}} \end{bmatrix} \begin{bmatrix} 1 \\ \delta\mathbf{x} \\ \delta\mathbf{u} \end{bmatrix} \tag{3.8}$$

where

$$
\begin{aligned}
\mathbf{Q_x} &= \ell_\mathbf{x} + \mathbf{f_x}^\top \mathbf{V}'_\mathbf{x}, \\
\mathbf{Q_u} &= \ell_\mathbf{u} + \mathbf{f_u}^\top \mathbf{V}'_\mathbf{x}, \\
\mathbf{Q_{xx}} &= \ell_\mathbf{xx} + \mathbf{f_x}^\top \mathbf{V}'_\mathbf{xx} \mathbf{f_x} + \mathbf{V}'_\mathbf{x} \mathbf{f_{xx}}, \\
\mathbf{Q_{uu}} &= \ell_\mathbf{uu} + \mathbf{f_u}^\top \mathbf{V}'_\mathbf{xx} \mathbf{f_u} + \mathbf{V}'_\mathbf{x} \mathbf{f_{uu}}, \\
\mathbf{Q_{ux}} &= \ell_\mathbf{ux} + \mathbf{f_u}^\top \mathbf{V}'_\mathbf{xx} \mathbf{f_x} + \mathbf{V}'_\mathbf{x} \mathbf{f_{ux}},
\end{aligned}
\tag{3.9}
$$

The primes denote the values at the next time-step. The subscripts $\{\}_\mathbf{x}$, $\{\}_\mathbf{u}$, $\{\}_\mathbf{ux}$, $\{\}_\mathbf{uu}$ and $\{\}_\mathbf{xx}$ denote the first and second order derivatives with respect to state and control variables. Note that $\mathbf{f_{ux}}, \mathbf{f_{uu}}, \mathbf{f_{xx}}$ are tensors, and thus calculation and multiplication of these quantities is computationally costly. In practice, we'll drop these quantities, and perform what is called a Gauss-Newton step.

The quadratic approximation of Q is quite useful. We can approximate our equation (3.6) to instead recursively find the optimal $\delta\mathbf{u}$ that optimizes our approximation. This leads to a recursive solution in the unconstrained setting:

$$
\delta\mathbf{u}^* = \arg\min_{\delta\mathbf{u}} \mathbf{Q}(\delta\mathbf{x}, \delta\mathbf{u}) = \mathbf{k} + \mathbf{K}\delta\mathbf{x}, \tag{3.10}
$$

where $\mathbf{k} = -\mathbf{Q}_\mathbf{uu}^{-1}\mathbf{Q_u}$ and $\mathbf{K} = -\mathbf{Q}_\mathbf{uu}^{-1}\mathbf{Q_{ux}}$ are the feed-forward and feedback terms. Recursive updates of the derivatives of the value function, which we denote by $V_\mathbf{x}(i)$ and $V_\mathbf{xx}(i)$, can be done as follows:

$$
\begin{aligned}
\mathbf{V_x}(i) &= \mathbf{Q_x} + \mathbf{K}^\top \mathbf{Q_{uu}}\mathbf{k} + \mathbf{K}^\top \mathbf{Q_u} + \mathbf{Q}_\mathbf{ux}^\top \mathbf{k}, \\
\mathbf{V_{xx}}(i) &= \mathbf{Q_{xx}} + \mathbf{K}^\top \mathbf{Q_{uu}}\mathbf{K} + \mathbf{K}^\top \mathbf{Q_{ux}} + \mathbf{Q}_\mathbf{ux}^\top \mathbf{K}.
\end{aligned}
\tag{3.11}
$$

3.2.4 Forward pass

The forward pass determines the step size along the search direction by adjusting the line search parameter α. It computes a new trajectory by integrating the dynamics along the computed feed-forward and feedback commands $\{\mathbf{k}_i, \mathbf{K}_i\}$:

$$
\begin{aligned}
\hat{\mathbf{u}}_i &= \mathbf{u}_i + \alpha\mathbf{k}_i + \mathbf{K}_i(\hat{\mathbf{x}}_i - \mathbf{x}_i), \\
\hat{\mathbf{x}}_{i+1} &= \mathbf{f}(\hat{\mathbf{x}}_i, \hat{\mathbf{u}}_i),
\end{aligned}
\tag{3.12}
$$

in which $\hat{\mathbf{x}}_1 = \mathbf{x}_1$, and $\{\hat{\mathbf{x}}_i, \hat{\mathbf{u}}_i\}$ is the new state-control pair. Note that if $\alpha = 0$, neither the state nor the control trajectories are modified.

3.2.5 Line search and regularization

We perform a *backtracking* line search by trying the full step ($\alpha = 1$) first. The choice of α is dual to the choice of regularization terms, and both are updated between subsequent iterations to ensure a good progress toward the (local) optimal solution. We use two regularization schemes: the Tikhonov regularization (over $\mathbf{Q_{uu}}$) and its update using the Levenberg-Marquardt algorithm are typically used (Toussaint, 2017). Tassa et al., 2012 propose a regularization scheme over $\mathbf{V_{xx}}$, which is equivalent to adding a penalty in the state changes.

3.2.6 DDP as an evolution of Newton Step

The backward pass of DDP produces the inverted Hessian of the objective function. DDP uses the Value Function and its derivatives $\mathbf{V}, \mathbf{V_x}, \mathbf{V_{xx}}$ in order to iteratively invert the block diagonal components. The Newton descent direction is also characterized by the inverted Hessian of the objective function.

While the Hessian computation by the backward pass is not straightforward as a simple inverse of the Hessian, both are equivalent. The comparison, as shown by Dunn and Bertsekas, 1989, becomes much easier to understand when we use an iterative scheme for the Newton Method as well.

Once the descent direction is calculated, the line-search computes the next best guess for optimal solution. In a typical Newton line search, the search around the current solution $\underline{x}$, $\underline{u}$ is done by approximating that the dynamics $x_{t+1} = f(x, u)$ is linear. In the case of DDP, we perform the forward pass on the exact non-linear dynamics, not its linearized version. This ensures that the DDP solution remains in the feasible domain, unlike in the classical case where the linearized dynamics would produce discontinuities in the solution. This non-linear roll-out is indeed the only difference between DDP and the classical Newton step. If the dynamics is considered to be linear, their difference disappears. DDP solution becomes exactly the same as that of the Newton Step.

Indeed, the power of DDP comes from its ability to iteratively invert N smaller dimension matrices in order to find the inverse of a huge N times larger matrix. The forward pass of the DDP, while it ensures that the solution is feasible, is optional. If the non-linear rollout of the exact dynamics is not used in DDP, discontinuities in the state space would appear, and we need to be able to find strategies to account for these discontinuities in the line search. This grants more power to the solver to explore the solution space than in the case of the exact dynamics roll-out.

In this chapter, we deal with the idea of converting the contact-constraints into holonomic constraints, that can be tracked using DDP. The idea of updating DDP line search for exploring the non-feasibility space is further explained in Chapter 4, and discussed in detail in Mastalli et al., 2020; Mansard, 2019

3.3 DDP with Constrained Robot Dynamics

3.3.1 Contact dynamics

Let's consider the case of rigid contact dynamics with the environment. Given a predefined contact sequence, rigid contacts can be formulated as holonomic scleronomic constraints to the robot dynamics (i.e. equality-constrained dynamics). The unconstrained robot dynamics (similar to (1.12)) is typically represented as:

$$\mathbf{M}\ddot{\mathbf{q}}_{free} = \mathtt{S}\boldsymbol{\tau} - \mathbf{b}, \tag{3.13}$$

Here, $\ddot{\mathbf{q}}_{free}$ is the unconstrained joint acceleration vector, and $\mathtt{S}$ is the selection matrix of the actuated joint coordinates.

We can account for the rigid contact constraints by applying the Gauss principle of least constraint (Udwadia and Kalaba, 1992; Wieber, 2006a). Under this principle, the constrained motion evolves in such a way that it minimizes the deviation in acceleration from the unconstrained motion $\ddot{\mathbf{q}}_{free}$, i.e.:

$$\begin{aligned} \arg\min_{\ddot{\mathbf{q}}} \quad & \frac{1}{2}\|\ddot{\mathbf{q}} - \ddot{\mathbf{q}}_{free}\|_{\mathbf{M}} \\ \text{subject to} \quad & \mathbf{J}_c\ddot{\mathbf{q}} + \dot{\mathbf{J}}_c\dot{\mathbf{q}} = \mathbf{0} \end{aligned} \tag{3.14}$$

Note that we differentiate twice the holonomic contact constraint $\phi(\mathbf{q})$ in order to express it in the acceleration space. In other words, the rigid contact condition is expressed by the second-order kinematic constraints on the contact surface position. $\mathbf{J}_c = \begin{bmatrix}\mathbf{J}_{c_1} & \cdots & \mathbf{J}_{c_f}\end{bmatrix} \in \mathbb{R}^{cp\times n}$ is a stack of the f contact Jacobians.

3.3.2 KKT conditions

The Gauss minimization in (3.14) corresponds to an equality-constrained convex optimization problem[2], and it has a unique solution if $\mathbf{J}_c$ is full-rank. The primal

[2] $\mathbf{M}$ is a positive-definite matrix.

and dual optimal solutions $(\ddot{\mathbf{q}}, \lambda)$ must satisfy the so-called KKT conditions given by

$$\begin{bmatrix} \mathbf{M} & \mathbf{J}_c^\top \\ \mathbf{J}_c & \mathbf{0} \end{bmatrix} \begin{bmatrix} \ddot{\mathbf{q}} \\ -\lambda \end{bmatrix} = \begin{bmatrix} \mathtt{S}\boldsymbol{\tau} - \mathbf{b} \\ -\dot{\mathbf{J}}_c \dot{\mathbf{q}} \end{bmatrix} \tag{3.15}$$

These dual variables $\lambda^k \in \mathbb{R}^p$ render themselves nicely in mechanics as the external forces at the contact level. This relationship allows us to express the contact forces directly in terms of the robot state and actuation. As compared to previous approaches which would introduce the contact constraints in the whole-body optimization (Carpentier et al., 2016; Saab et al., 2013), here we solve for the contact constraints at the level of the dynamics, and not the solver. In other words, this would free the solver to find an unconstrained solution to the KKT dynamics (3.15), without worrying about the contact constraint.

We need the derivatives of (3.15) for applying fast iterative Newton and quasi-Newton methods to achieve real-time performance. Highly efficient implementation (Carpentier and Mansard, 2018a; Carpentier et al., 2019) of the analytical derivatives are available for the derivatives of the unconstrained dynamics (3.13) (Featherstone, 2008). This implementation can easily be extended to compute the derivatives of the augmented KKT dynamics (3.15).

If we consider the derivative of (3.15) with respect to a variable z, where z could be defined by $\mathbf{q}$, $\dot{\mathbf{q}}$, or $\boldsymbol{\tau}$, the derivatives of the dynamics defined by (3.15) are given by:

$$\begin{bmatrix} \frac{\partial \ddot{\mathbf{q}}}{\partial z} \\ -\frac{\partial \lambda}{\partial z} \end{bmatrix} = -\begin{bmatrix} \mathbf{M} & \mathbf{J}_c^\top \\ \mathbf{J}_c & \mathbf{0} \end{bmatrix}^{-1} \begin{pmatrix} \frac{\partial \mathbf{M}}{\partial z}\ddot{\mathbf{q}} - \frac{\partial \mathbf{J}_c^\top}{\partial z}\lambda + \frac{\partial \mathbf{b}}{\partial z} - \frac{\partial \boldsymbol{\tau}}{\partial z} \\ \frac{\partial \mathbf{J}_c}{\partial z}\ddot{\mathbf{q}} + \frac{\partial (\dot{\mathbf{J}}_c \dot{\mathbf{q}})}{\partial z} \end{pmatrix} \tag{3.16}$$

The derivative computation in (3.16) contain tensor products, and normally these would be computationally intensive. However, it is possible to consider these tensor products as simple kinematic and dynamic quantities, and the whole computation (3.16) can be done really efficiently without any such computational burden. The inverse of the augmented mass matrix in (3.16) can be found by a simple schur complement inverse. The top row on the right hand side, which computes the derivatives of the dynamics with respect to the state variables ($\mathbf{q}$, $\dot{\mathbf{q}}$) or the control variable ($\boldsymbol{\tau}$), is already implemented in the dynamics library *Pinocchio* (Carpentier et al., 2019). The bottom row is simply the derivative of the end-effector kinematics, which is also available in *Pinocchio*. We can thus use analytical implementation of the dynamics to improve our efficiency (as we will see later in Chapter 4).

3.3.3 KKT-based DDP algorithm

From (3.15), we can see the augmented KKT dynamics as a function of the state $\mathbf{x}_i$ and the control $\mathbf{u}_i$. However, the dynamics is continuous, and the decision variables ($\underline{\mathbf{x}}$, $\underline{\mathbf{u}}$) of continuous dynamics over a time horizon are infinite. We approximate this problem using a discrete version of it, by following the so-called direct (discretize first, solve second) approach. We take the control $\mathbf{u}_u$ as constant over a discrete interval, and use numerical integration schemes (like n-order *Runge-Kutta*, or *Euler* integration) in order to move from one state to another (Press et al., 2007). Our continuous dynamics from (3.15) can then be written as:

$$\begin{aligned} \mathbf{x}_{i+1} &= \mathbf{f}(\mathbf{x}_i, \mathbf{u}_i), \\ \lambda_i &= \mathbf{g}(\mathbf{x}_i, \mathbf{u}_i), \end{aligned} \tag{3.17}$$

where the state $\mathbf{x} = (\mathbf{q}, \dot{\mathbf{q}})$ is represented by the configuration vector and its tangent velocity, $\mathbf{u}$ is the torque-input vector, and $\mathbf{g}(\cdot)$ is the dual solution of (3.15). In case of legged robots, the placement of the free-floating link is described using the special Euclidean group $SE(3)$.

Given a reference trajectory for the contact forces, the DDP backward-pass cost and its respective Hessians (see (3.6) and (3.9)) are updated as follows:

$$J_i(\mathbf{x}_i, \mathbf{U}_i) = l_f(\mathbf{x}_N) + \sum_{k=i}^{N-1} l(\mathbf{x}_k, \mathbf{u}_k, \lambda_k), \tag{3.18}$$

where $\mathbf{U}_i = \{\mathbf{u}_i, \mathbf{u}_{i+1}, \cdots, \mathbf{u}_{N-1}\}$ is the tuple of controls that acts on the system dynamics at time i, and the Gauss-Newton approximation of the $\mathbf{Q}$ coefficients (i.e. first-order approximation of $\mathbf{g}(\cdot)$ and $\mathbf{f}(\cdot)$) are

$$\begin{aligned} \mathbf{Q_x} &= \mathbf{l_x} + \mathbf{g_x}^\top \mathbf{l}_\lambda + \mathbf{f_x}^\top \mathbf{V}'_\mathbf{x}, \\ \mathbf{Q_u} &= \mathbf{l_u} + \mathbf{g_u}^\top \mathbf{l}_\lambda + \mathbf{f_u}^\top \mathbf{V}'_\mathbf{x}, \\ \mathbf{Q_{xx}} &\approx \mathbf{l_{xx}} + \mathbf{g_x}^\top \mathbf{l}_{\lambda\lambda} \mathbf{g_x} + \mathbf{f_x}^\top \mathbf{V}'_{\mathbf{xx}} \mathbf{f_x}, \\ \mathbf{Q_{uu}} &\approx \mathbf{l_{uu}} + \mathbf{g_u}^\top \mathbf{l}_{\lambda\lambda} \mathbf{g_u} + \mathbf{f_u}^\top \mathbf{V}'_{\mathbf{xx}} \mathbf{f_u}, \\ \mathbf{Q_{ux}} &\approx \mathbf{l_{ux}} + \mathbf{g_u}^\top \mathbf{l}_{\lambda\lambda} \mathbf{g_x} + \mathbf{f_u}^\top \mathbf{V}'_{\mathbf{xx}} \mathbf{f_x}. \end{aligned} \tag{3.19}$$

The set of equations (3.19) takes into account the trajectory of the rigid contact forces inside the backward-pass. The system evolution needed in the forward-pass is described by (3.17).

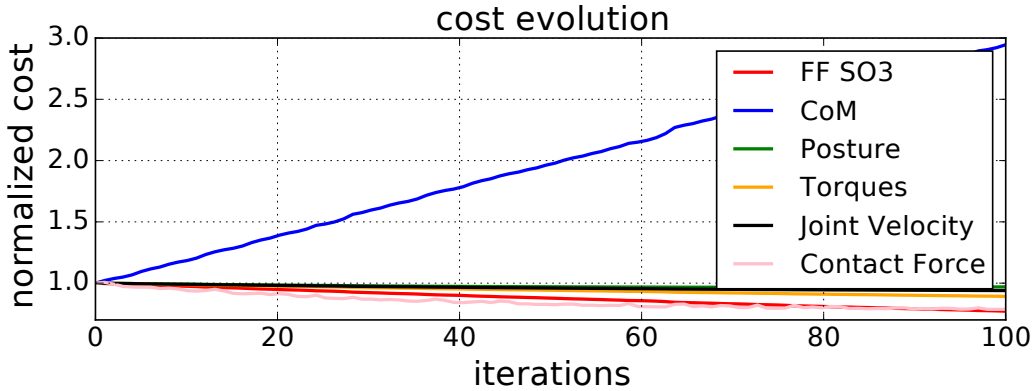


Fig. 3.3: Evolution of the different cost functions (normalized) with respect to iterations. DDP reduces the applied torques by recalculating the CoM tracking. It improves the contact force by taking into account the whole-body angular momentum. The result is a continuous improvement in the performance as compared to IK. We stop after 100 iterations. ("FF SO3" refers to the free-flyer orientation cost; CoM refers to Center of Mass tracking cost, and so on...)

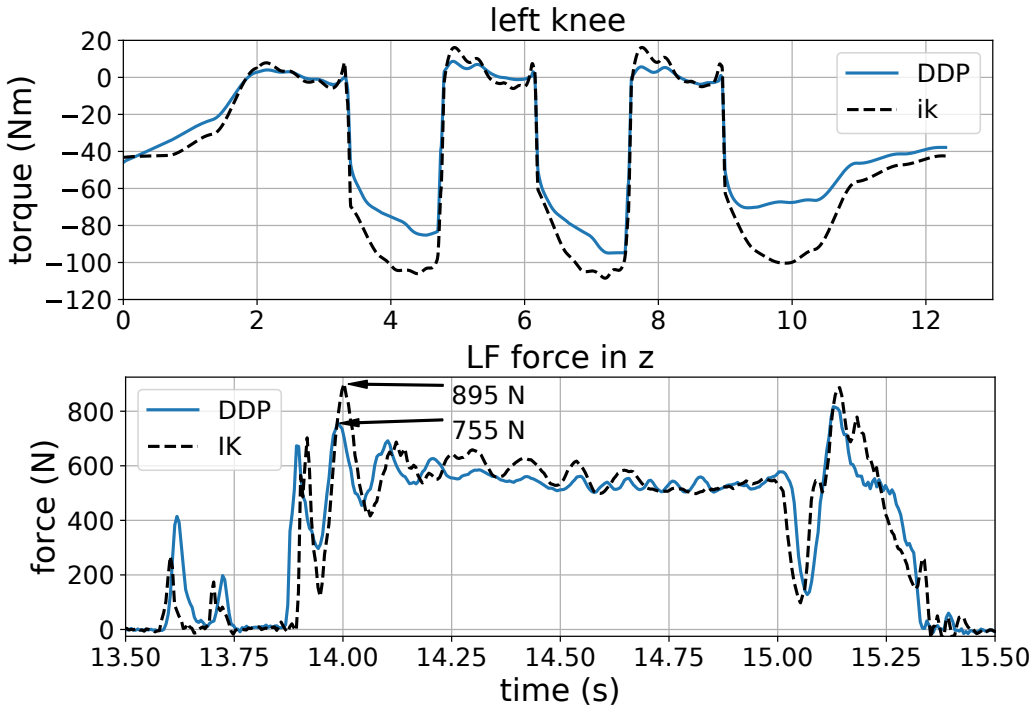


Fig. 3.4: Comparison between the IK and DDP trajectory for 100 cm stride on the HRP-2 robot. *Top*: Knee torques generated in the left leg. *Bottom*: Ground Reaction Forces (GRFs) generated in the left foot. The DDP formulation allows us to utilize the angular momentum of the upper body, which reduces the requirement on the lower body to create a counterbalancing motion. This results in a lower torque in the lower body, as well as lower GRFs. Around $t = 14s$ we can see high peaks for the IK and DDP trajectories of 895 N and 755 N, respectively.

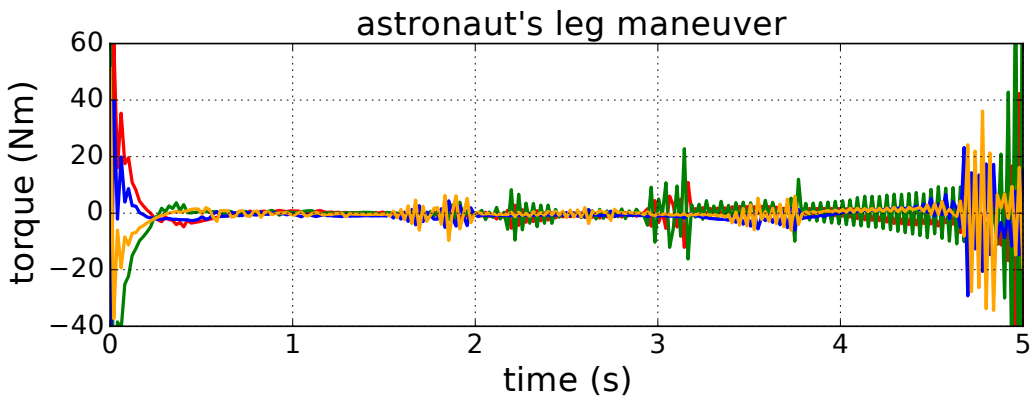


Fig. 3.5: Joint torques for the astronaut maneuver. Our method plans a smart strategy by kick-starting the rotation, and then tries to maintain the velocity by small bang-bang control signals. Towards the end, it changes again the velocity of the lower legs in order to bring the system to a stop.

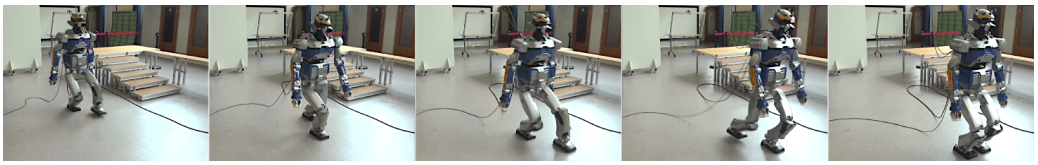


Fig. 3.6: Snapshots of 100 cm stride on a flat terrain used to evaluate the performance of our whole-body trajectory optimization method. The DDP trajectory reduces significantly the normal forces peaks compared with classic whole-body IK.

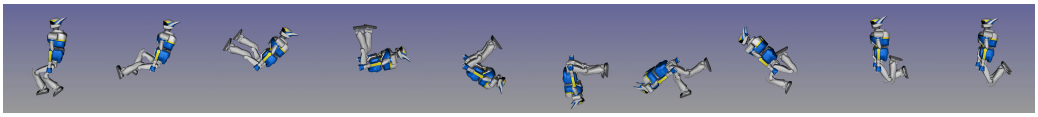


Fig. 3.7: Attitude adjustment maneuver conducted by the robot in gravity free space. DDP solver takes into account the non-holonomic angular momentum constraint and uses internal actuation to rotate 360° without the need for contact forces.

3.4 Results

In this section, we show that our DDP formulation can generate whole-body motions which require regulation of the angular momentum. The performance of our algorithm is assessed on realistic simulations and aggressive experimental trials on the HRP-2 robot. First, we perform very large strides (from 80 to 100 cm) which require large amount of angular momentum (due to the fast swing of the 6-kg leg) and reach the HRP-2 limits. This demonstrates the ability of the solver to handle contact constraints, as well as to *generate* excessive AM required by the leg motions. Then, to again emphasize the need for a horizon based optimizer such as DDP, and the ability of our solver to handle these AM requirements, we regulate the robot attitude in absence of contact forces and gravitational field. These motions cannot be generated through a standard time-invariant IK/ID solver, as the system becomes non-holonomic as shown in (3.1).

All the motions were computed offline. Contact sequence (Tonneau et al., 2018b) and the centroidal trajectory (Carpentier and Mansard, 2018b) are precomputed and provided to the solver for the large stride experiments. We used the standard controller OpenHRP (Kanehiro et al., 2004) for tracking the motions on the real robot. The large strides produced by DDP are compared with those produced by an IK solver (Saab et al., 2013), showing the benefit of our approach.

3.4.1 Large stride on a flat ground

In these experiments, we generate a sequence of cyclic contact for 80 cm to 100 cm stride. These are very big steps for HRP-2 compared to its height (160 cm). For the contact location, we use the OC solver reported in Carpentier et al., 2016 to compute the contact timings and the centroidal trajectory. As the centroidal solver is able to provide feasible contact forces for individual contacts in the phase, we use a damped cholesky inverse of the KKT matrix to deal with the rank deficient $\mathbf{J}_c$ matrix. Then we use our proposed DDP to generate the full robot motion.

The cost function is composed of the weighted square norm (weighted by $\mathbf{Q}_i$) of various quadratic residues $\mathbf{r}_i$ (i.e. $\|\mathbf{r}_i(\mathbf{x}, \mathbf{u}, \boldsymbol{\lambda})\|^2_{\mathbf{Q}_i}$) in order to keep balance and to increase efficiency and stability: (a) CoM, foot position and orientation and contact forces tracking of centroidal motion, (b) torque commands minimization and (c) joint configuration and velocity regularization. The evolution of the different normalized task costs with iterations is shown in Fig. 3.3. Our method adapts the CoM to create a more efficient torque and contact force trajectory.

Increasing the upper-body angular momentum helps to counterbalance the swing leg motion, this in turn reduces ground reaction forces (GRFs) and improves the locomotion stability. Our experimental results show a reduction on the GRFs peaks compared to the IK solver. Fig. 3.4 shows the measured normal contact forces and the knee torques in case of DDP solver and IK solvers. Our DDP reduced the normal forces peaks of the IK solver from 895 N to 755 N. This represents a significant improvement, considering that the minimum possible contact forces are 650 N (the total mass of the HRP-2 robot is 65 kg) and the maximum safe force allowed by the sensors on the foot is 1000 N. An overview of the motion is shown in Fig. 3.6.

3.4.2 Attitude regulation through joint motion

The angular momentum equation (3.1) shows that it is possible to regulate the robot attitude without the need of contact forces Wieber, 2006a. It can be seen that the gravity field does not affect this property. Thus, we analyze how our DDP solver regulates the attitude in zero-gravity condition, we named this task *astronaut reorientation*. The astronaut reorientation (similar to cat falling) is an interesting motor task due to fact that it depends on a proper exploitation of the angular momentum based on the coordination of arms and legs motions. Fig. 3.7 demonstrates the motion found by the solver to rotate the body 360°. Unlike an instantaneous tracking solver like IK, the solver is willing to bend in the opposite direction, in order to obtain an ability to create sufficient angular momentum by the legs. It is important here to note that such motion cannot be obtained by a time-invariant control law which does not take the future control trajectory into account.

The cost matrices for this problem require a barrier function on the robot configuration to avoid self-collision. Final cost on the body orientation provides the goal, and a running cost on the posture is added for regularization. No warm start is given to the solver, the initial control trajectory is a set of zero vectors. For the ease of demonstration, we used only the leg joints in the sagittal plane. Fig. 3.5 shows the torques produced by the hip and the knee joints. Our method creates a rotation of the upper body by a quick initial motion in the legs. Then it maintains the angular velocity by small correctional torque inputs during the rest of the trajectory. At the end, to bring the rotation to a halt, the same behavior is repeated in the reverse.

3.5 Conclusion and Perspective

Typically, reduced centroidal trajectory optimization does not take into account the angular momentum produced by the limb motions. Proper regulation of the angular momentum exploits the counterbalancing effect of limb motion in order to

reduce the contact forces and torque inputs. It also improves the stability during flight phases where the momentum control can only be done through joint motions. Excessive angular momentum cannot be produced by a simple IK/ID solver. OC provides the required tools for solving it. Our proposed solver is an extension of our previous work (Carpentier et al., 2016).

In this chapter, we have explained our DDP formulation based on the augmented KKT dynamics (see (3.17)) which is a result of holonomic contact constraints. It represents the first application of motion generated by DDP solver on a real humanoid locomotion. Our whole-body motion generation pipeline enables us to potentially regulate angular momentum dynamics during the whole-body motion in real-time. We have observed a reduction of the contact forces compared to the IK solver, even though we had to restrict the angular momentum in the sagittal plane for the stride on flat ground task due to robot limits. A more revealing experiment, the *astronaut reorientation*, demonstrates further the limits of the previous approaches and the advantages of using DDP. The solver generates the desired motion from scratch in this case by manipulating joint velocities within the non-holonomic angular momentum constraints.

Compared to a simple newton step, DDP does line search using the non-linear dynamics (f) of the system. While this keeps the solution within the feasible space, it loses the convergence guarantees (for convex optimization problems) of a linear newton step. On the other hand, linear newton steps introduce discontinuities in the solutions, and these need to be taken care of before a feasible solution can be produced (by a merit function, or other methods). Thus, the choice of line search with DDP must be made with caution, and this has important consequences on the convergence properties of the solver. This issue is further discussed in Mansard, 2019.

While the solver is able to generate angular momentum, and track the centroidal trajectories, the current approach still lacks an assurance that the additional angular momentum generated by the DDP solver is accounted for in the centroidal optimization, and vice versa. In the upcoming Chapter 5, we focus on this guarantee. In Chapter 5, we ensure that the output from this current chapter is indeed matched by the centroidal dynamics. However, handling constraints (e.g. the non-slippage contact constraint), along with generating dynamic movements is difficult in the whole-body solver. In Chapter 5, we will see how we can use our DDP solver along with the centroidal solver to divide the jobs of constraint handling and dynamic motion generation.

This chapter introduced the ideas behind our constrained-dynamics DDP algorithm. In the next Chapter 4, we will explain how to implement this in an efficient manner

by using proper computational and mathematical tools and analytical derivatives (Sohl and Bobrow, 2001; Carpentier and Mansard, 2018a) of the robot dynamics. Moreover, we will see in Chapter 4 an implementation of DDP which uses the linear newton step while avoiding the merit-function (Mansard, 2019; Mastalli et al., 2020), in order to begin its search from the infeasible solution space.

4

Crocoddyl: **C**ontact **Ro**bot **Co**ntrol by **D**ifferential **Dy**namic Programming **L**ibrary

For our path in life . . . is stony and rugged now, and it rests with us to smooth it. We must fight our way onward. We must be brave. There are obstacles to be met, and we must meet, and crush them!

— **David, to Dora**
(Charles Dickens, David Copperfield)

Mastalli, Carlos, **Rohan Budhiraja**, *Wolfgang Merkt*, *Guilhem Saurel*, *Bilal Hammoud*, *Maximilien Naveau*, *Justin Carpentier*, *Sethu Vijayakumar*, and *Nicolas Mansard* (2019). Crocoddyl: An Efficient and Versatile Framework for Multi-Contact Optimal Control. Submitted to IEEE Robotics and Automation Letters. In: arXiv:1909.04947

Crocoddyl is an optimal control library in python and C++ for robot trajectory optimization with predefined contact phases. Its solver is based on an efficient Differential Dynamic Programming (DDP) algorithm (Budhiraja et al., 2018) that we proposed earlier in Chapter 3.

4.1 Features of Crocoddyl

Over the last decade, robot hardwares have improved their capabilities by leaps and bounds. From the human-like walking of Honda ASIMO, we have arrived at the more-than-human backflips of the current ATLAS robot. In order to keep up, the robot controllers need to be able to find a control policy in a fast and efficient

manner. Such a control mandates that the controller be based on a strong mathematical background, and use algorithms that are able to consider the maximum amount of model information in order to generate the control policy. In Crocoddyl, we try to take into account these requirements in order to increase the control bandwidth as much as possible. As a result, the following features are central to Crocoddyl's design:

- **Transcription vs Resolution**: In Crocoddyl, problem transcription is separately handled from problem resolution. We understand that while the problem structure remains the same, choice of solvers might differ based on requirements. Thus, any of the solvers in Crocoddyl can be plugged and used with the same transcription of the problem. This simplifies the API and makes it intuitive.

- **Efficient Rigid Body Algorithms**: Crocoddyl is centered around Pinocchio (Carpentier et al., 2019) as its dynamics engine. Pinocchio implements highly fast and efficient implementations of the Rigid Body Algorithms (Featherstone, 2008) based on Lie Algebra. Crocoddyl is able to leverage Pinocchio in order to reduce its own computation time.

- **Analytical Derivatives**: We use analytical derivatives of the dynamics in order to find the descent direction. This makes the solver much faster when compared to numerical finite differencing (Carpentier and Mansard, 2018a).

- **Multi-threading**: In Crocoddyl, we use multi-threading to reduce the computation time of finding the derivatives. As a result, we can improve multi-fold our performance, as discussed in Sec 4.4

- **Feasibility-prone DDP**: We propose a variant of the DDP algorithm that exactly matches the behaviour of direct multiple-shooting methods. It has the ability to handle infeasible guesses that occur whenever there is a gap between subsequent nodes in the trajectory. As a result FDDP has the ability to expand its search and improve its solution.

- **Memory Management**: Crocoddyl reduces any dynamic allocation of memory. Moreover, we structure our memory in constant variables descriptions called `Models` and problem specific variables called `Datas`, which helps with efficient caching of the memory.

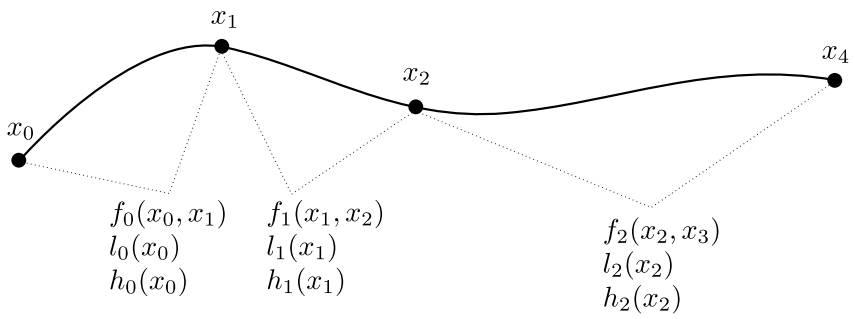


Fig. 4.1: Illustrating the First Order Markov structure which guides our problem. The constraints and costs of a node are dependent only on the current state, and the current state is only dependent on the previous state. (Toussaint, 2017)

The Problem Transcription and Problem Resolution are further discussed in Sec 4.2 and Sec 4.3. The above features, which are either a property of the transcription or the resolution of a problem, are also explained in these sections.

4.2 Problem Transcription

Easy problem transcription is an essential feature of a solver. In crocoddyl, we recognize that

- discretized robot dynamics is Markovian in nature;
- robot description and dynamics remains unchanged for a problem.

The 1st order Markovian dynamics is explained further in Fig. 4.1. Each node in the figure represents a time instant, and only the successive nodes are connected together to form the entire horizon. The discretized dynamics($\mathbf{f_i}$) moves the state (x_i) from one node to the next (x_{i+1}). The dynamics, the cost function ($\mathbf{l_i}$), and the constraints ($\mathbf{h_i}$) remain only the property of the node i, and as a result, the problem has the potential to be separated into N different subproblems, where N is the number of nodes.

4.2.1 `Action Models`: Unification of Dynamics, Costs and Constraints

A look at the structure of the Markovian Dynamics in Fig. 4.1 reveals that dynamics, cost, and constraints of a system are inter-related. For e.g., consider a typical end-effector tracking problem. If we try to minimize the error in the position of the end-

effector while avoiding self-collisions, both the cost and constraints are dependent on the kinematics and dynamics of the multi-body system.

Thus, it makes sense to store and use the dynamics along with the cost and constraints, and reduce any extra and redundant computations resulting from a disparate structure. Once f, l, h are defined in a single `Action` class, we can simply define our Markovian problem as a series of Actions that are linked together by the dynamics of our problem. This makes it easy to identify the constant features inside a node, and the features which are dependent on the problem.

One of the main features of the crocoddyl structure is this distinction between the invariant description of the node (`Models`), and the problem-specific variables(`Datas`) which are a result of the computations and the algorithms.

Action Model: Action Models consist of the invariant parameters, such as the description of the dynamics ($\mathbf{f_i}$), the costs($\mathbf{l_i}$), and the constraints (in our case, the contact constraints) ($\mathbf{h_i}$), that define the particular node. The `Data` classes , on the other hand, store the values, the derivatives, and other problem dependent quantities.

Differential Action Model: The continuous robot dynamics is typically specified in terms of differential equations. The discretization is done at this level, and then an integration scheme is used that defines the transition from one state to the other. Taking this into account, we implement the following differential models under the name of `Differential Action Model`:

- Full-Body Dynamics;
- Contact-Constrained Full-Body Dynamics (Budhiraja et al., 2018);

Integrated Action Model: Each of these provides the derivative of the state variable ($\dot{x}_i$). Thus, we need an integration scheme $\mathbf{I}_i$ that implements $\mathbf{I}_i(\dot{x}_i, x_i) \rightarrow x_{i+1}$. In Crocoddyl, there are two separate integration schemes implemented:

- Symplectic Euler (Press et al., 2007);
- Runge-Kutta4 (Press et al., 2007);

As can be seen, the integration scheme $\mathbf{I}_i$ along with the `DifferentialActionModel` makes up an `ActionModel` that defines the dynamics of a node. In addition to the above, Crocoddyl also has implementation of `ImpactDynamics` which implements the state transition in case of a sudden impact absorbed in a small amount of time.

List of Activation Models	
Activation Model Name	*Description*
QuadModel	$\mathbf{a}(r) = 0.5r^T r$
WeightedQuadModel	$\mathbf{a}^W(r) = 0.5r^T W r$
InequalityModel	$\mathbf{a}_l^u(r) = \left\{ \begin{array}{l} 0 \text{ if } l \leq r \leq u \\ 0.5(l-r)^T(l-r) \text{ if } r < l \\ 0.5(r-u)^T(r-u) \text{ if } r > u \end{array} \right\}$
SmoothAbsModel	$\mathbf{a}(r) = \sqrt{1 + r^T r}$

Tab. 4.1: Activation Models in Crocoddyl

List of Cost Models	
Cost Model Name	*Description*
`CostModelFramePlacement`	*Tracking the placement of a frame on the kinematic tree*
`CostModelCoM`	*Tracking the Center of Mass*
`CostModelState`	*Tracking the state* x
`CostModelControl`	*Tracking the control* u
`CostModelForce`	*Tracking the forces, only for Contact-Constrained dynamics*
`CostModelZmpBound`	*Soft inequality constraint on the zmp position with respect to bounds*
`CostModelMomentum`	*Tracking the centroidal momentum, as described in Orin et al., 2013*
`CostModelForceLinearCone`	*Soft inequality constraint on Contact Force Vector, with respect to a linear cone. Only for contact-constrained dynamics*
`CostModelFrameTranslation`	*Tracking the translation of a frame on the kinematic tree*
`CostModelFrameVelocity`	*Tracking the velocity of a frame on the kinematic tree*
`CostModelFrameVelocityLinear`	*Tracking the linear velocity of a frame on the kinematic tree*

Tab. 4.2: Cost Models in Crocoddyl

Cost and Activation Model: Along with the dynamics and the contact constraints, cost terms ($\mathbf{l}_i$) are also saved inside an `ActionModel`. In Crocoddyl, we implement costs in the forms of errors (given by residue r) during tracking of a feature (e.g. the position of an end-effector with respect to a reference), and the activation model ($a(r)$), which defines the effect of the residue on the net cost. The activation models, implemented in Crocoddyl are described in Table 4.1.

Each of these activation models takes the input (r) from a cost model. The implemented cost models are dependent on the robot dynamic and control parameters, and thus the `DifferentialActionModels`. Different cost models are implemented for the Full-body and Contact-Constrained Full-body dynamics, and are described in Table 4.2

Calc and Calcdiff Functions: Each of the above classes (`DifferentialActionModel`, `IntegratedModel`, `CostModel`, `ActivationModel`) contain a `calc` and a `calcDiff` function. The `calc` function performs the computations for a given state and con-

trol, as described above. The `calcDiff` computes the sensitivities of `calc` with respect to the state and the control. As a result, the sensitivities are transmitted to the `ActionModels`. Thus, each action model implements

$$\begin{aligned} x_{i+1}, l_i = f(x_i, u_i) &= \texttt{ActionModel.calc}(model, data, x_i, u_i), \\ f_x, f_u, l_x, l_u, l_{xx}, l_{xu}, l_{uu} &= \texttt{ActionModel.calcDiff}(model, data, x_i, u_i), \end{aligned} \tag{4.1}$$

where x_i, u_i represent the state and control at interval i. f represents the dynamics, and l represents the local cost during the interval. All the outputs are stored inside the data object for future access. No new memory is allocated after initialization.

The robot dynamics and the cost models which are based on this dynamics are implemented efficiently using the software framework Pinocchio (Carpentier et al., 2019). Moreover, the sensitivities of the the action models (and all its objects) are also computed using the same framework (Carpentier and Mansard, 2018a). The analytical derivation, instead of using finite-differences, makes the solution much faster to obtain.

4.2.2 Non-Euclidean State Models

The state of a system lies in a manifold X, while its sensitivities are in the tangent space $T_x X$. For N-dimensional euclidean states, like the joint space of a serial manipulator, the computation of T_x is straightforward. However, floating based systems contain the transformation of the floating joint in their state space, and as a result, the joint space is non-euclidean.

Using Lie Algebra, it is possible to represent the floating joint by an element of $\mathbb{SE}(3)$ Lie group, and thus the tangent space T_x by its associated Lie Algebra. As a result, we need to be able to accommodate different `State Models` in order to accurately represent state in different manifolds and compute their sensitivities. Each manifold obeys its own algebra, i.e., there are specific rules for the following operations:

$$\begin{aligned} \mathbf{x}' &= \mathbf{x} \oplus \delta\mathbf{x}, \\ \delta\mathbf{x} &= \mathbf{x}_1 \ominus \mathbf{x}_0, \end{aligned} \tag{4.2}$$

where $\mathbf{x} \in X$ and $\delta\mathbf{x} \in T_{\mathbf{x}} X$ and can be described by a n_x- and n_{dx}-tuples, respectively. These operations are encoded by `integrate` and `difference`, respectively.

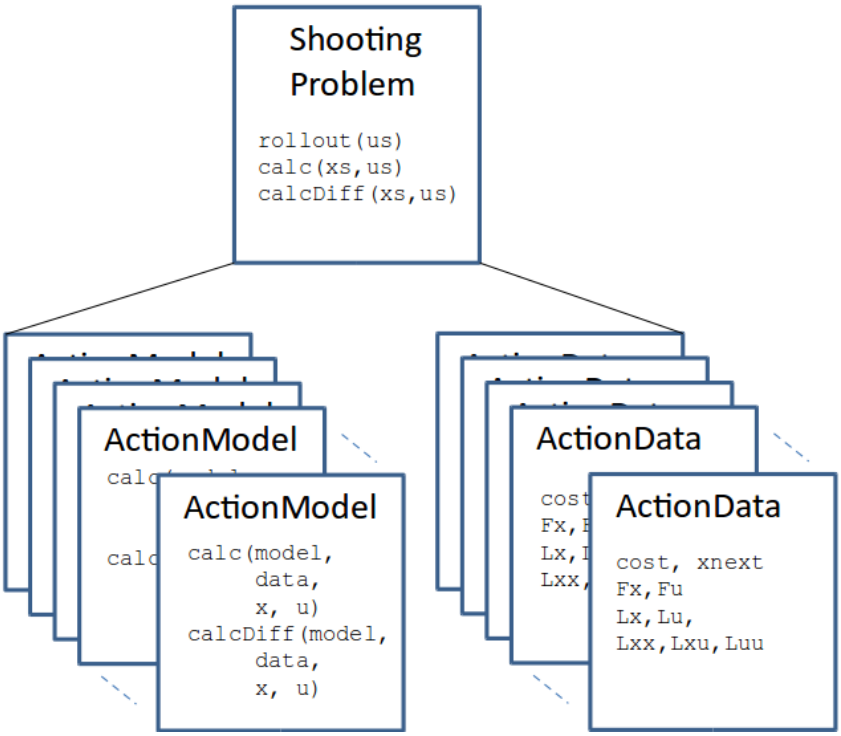


Fig. 4.2: Creating a shooting problem using crocoddyl

In addition, we need to implement the sensitivities of the above operations. These are defined as the left and right jacobians of the above operators, and implemented as follows:

$$\begin{aligned} \frac{\partial(\mathbf{x} \oplus \delta\mathbf{x})}{\partial\mathbf{x}}, \frac{\partial(\mathbf{x} \oplus \delta\mathbf{x})}{\partial\delta\mathbf{x}} &= \texttt{Jintegrate}(\mathbf{x}, \delta\mathbf{x}), \\ \frac{\partial(\mathbf{x}_1 \ominus \mathbf{x}_0)}{\partial\mathbf{x}_0}, \frac{\partial(\mathbf{x}_1 \ominus \mathbf{x}_0)}{\partial\mathbf{x}_1} &= \texttt{Jdifference}(\mathbf{x}_0, \mathbf{x}_1), \end{aligned} \tag{4.3}$$

4.2.3 Shooting Problem

The global problem is thus described by a stack of `Action Models`, which are linked with each other by the dynamics implemented in each of them, along with initial and final constraints. This is written in the form of a `Shooting Problem`:

$$\texttt{ShootingProblem}(x_0, [\texttt{runningModel}_0 ... \texttt{runninModel}_{T-1}], \texttt{terminalModel})$$

Thus, the input to a shooting problem is the initial state x_0, and a list of action models. This structure is shown in Fig. 4.2. Further API on shooting problem simplifies the computation of dynamics and derivatives[1]. This API is shown in Table 4.3.

[1]Note that for all variables, underlines denote a trajectory of the variable over time. e.g. $\underline{x} = [x_0 ... x_T]$

Shooting Problem Member Functions	
Function Name	*Description*
`ShootingProblem.calc(x, u)`	Compute the dynamics and cost given by `calc(`x_i,u_i`)` for each action model i in the stack.
`ShootingProblem.calcDiff(x, u)`	Compute the derivatives given by `calcDiff(`x_i,u_i`)` for each action model i in the stack.
`ShootingProblem.rollout(u)`	Iterative integrate the dynamics from $t = 0$ to $t = T$ for each action model in the stack

Tab. 4.3: Shooting Problem API

4.3 Problem Resolution

Once the problem is defined inside `ShootingProblem` class, different solvers can be implemented to find the solution. In crocoddyl, all the solvers share the following features: 1) Multi-threading for derivative computations, and 2) Using Analytical Derivatives for speed and efficiency.

4.3.1 Multi-threading

For a given state trajectory $\underline{x}$ and control trajectory $\underline{u}$, the `ShootingProblem` contains `ActionModels` whose sensitivities can be computed independently from each other. As a result, `ShootingProblem.calcDiff` is an embarrassingly parallel `for` loop which can be solved in parallel by a pool of threads.

```
1)  Launch T_N threads in pool P(N)
2)  for m_i in ShootingProblem.[runningModels+terminalModel] :
      send m_i.calcDiff(x,u) asynchronously to P(N)
```

Since `ActionData` objects which correspond to the above `ActionModel` objects are separate blocks of memory, and there are no memory sharing or return values, the step (2) in the above pseudo-code is truly independent. The results of multi-threading are further discussed in Sec 4.4.

4.3.2 Analytical Derivatives

Crocoddyl uses Pinocchio (Carpentier et al., 2019) to efficiently compute analytical and sparse derivatives (Sohl and Bobrow, 2001; Carpentier and Mansard, 2018a). Pinocchio has dedicated algorithms to compute analytical derivatives of rigid-body algorithms (e.g. RNEA and Articulated Body Algorithm (ABA)) using spatial alge-

Solver Implementations	
Solver	*Description*
Contact-Constrained DDP Solver	See Chapter 3
KKT based Solver	See *Mansard, 2019*
Feasibility-Prone DDP solver	See *Mansard, 2019*
Control-limited box-DDP solver	See *Tassa et al., 2014*
Box-KKT solver	*KKT based control-limited resolution of the full horizon*

Tab. 4.4: List of Solvers in Crocoddyl

bra. Crocoddyl employs these routines to derive the analytical derivative of contact dynamics as described in Chapter 3, as well as the ones of cost functions, e.g., Center of Mass (CoM) tracking, frame placement tracking, etc.

4.3.3 Solvers

In Crocoddyl, the central idea is to solve for problems by converting the rigid contact constraints into holonomic constraints (as explained in Chapter 3). Based on this principle, a number of solvers are available in Crocoddyl. These are listed in Table 4.4.

Different solvers are homogenized into a single API. The API is meant for shooting-based solvers, and is independent of the implementation. It is listed in Table 4.5

Contact-Constrained DDP Solver: Implements the DDP Solver with contact constraints, as described in Chapter 3.

KKT-based Solver: The Karush-Kuhn-Tucker conditions of optimality (Boyd and Vandenberghe, 2004) give a set of equations which must be satisfied by the solution of an optimization problem. Using these principles, and an initial guess x, a KKT-based solver finds the best ∂x based on a quadratic estimate of the KKT conditions.

The KKT-based solver inverts a giant block-diagonal matrix, and thus is not suitable for fast and real-time computations. However, KKT-based resolution of the full problem, while slow in practice, gives the ground truth against which all other solvers can be tested since the behaviour of the KKT-solver is the same as that of an SQP. Full details on the implementation of the solver can be found in Mansard, 2019

Feasibility-prone DDP: Multiple-shooting formulation introduces intermediate gaps as additional decision variables, for which an additional metric needs to be defined

Shooting Problem Member Functions	
Function Name	Description
`Solver.__init__(*args)`	Initialize the solver with the shooting problem.
`Solver.solve(*args)`	Solve the shooting problem. The API takes meta-parameters related to numerical optimization, such as the maximum number of solver iterations, the warm start, and regularization.
`Solver.computeDirection(*args)`	Compute the optimal search direction (based on the sensitivities of the action models), for the current state and control.
`Solver.tryStep(*args)`	Try a step of a given length in the search direction provided by computeDirection.
`Solver.stoppingCriteria(*args)`	A set of values that define the optimal point of termination of the solver, such as the norm of the sensitivities.
`Solver.allocateData(*args)`	Create the `Data` elements to all the `Model` classes in order to store the variables. This is where all the memory is allocated in the solver.
`Solver.setCandidate(*args)`	Define the initial warm guess for the state and the control.
`Solver.models(*args)`	List of all the Action Models inside the shooting problem.
`Solver.datas(*args)`	List of all the Action Datas inside the shooting problem.
`Solver.setCallbacks(*args)`	A set of user-defined callback functions for accessing/-monitoring the solver's performance.

Tab. 4.5: Solver API in Crocoddyl

(the gaps should be zero for the solution to be feasible). However, such a merit function can be avoided if the line search in DDP can be used to encode the gaps in the consecutive states. As a result, FDDP has the ability to start its search from infeasible space, and slowly converge to the full line search (and thus zero gaps) as the convergence is achieved. Full details of the implementation can be seen in Mansard, 2019.

Control-limited Box-DDP solver: DDP Solver doesn't handle control constraints in itself. However, in Tassa et al., 2014, a small modification of the DDP algorithm was proposed, in which (3.10) is solved while keeping in account the contact constraints. In addition, a minor modification is made in the line search, to ensure that the final contact solution also does not surpass the control constraints. Please see Tassa et al., 2014 for full details of the implementation.

Box-KKT solver: Implements the KKT version of the Box-DDP solver described in Tassa et al., 2014.

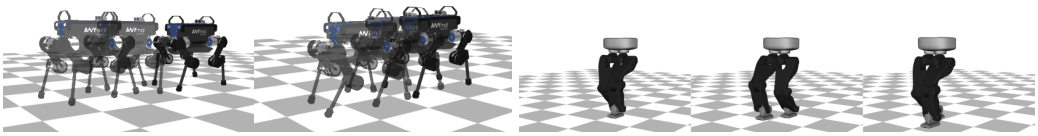


Fig. 4.3: Different legged gaits optimized by our FDDP algorithm. (top) from left to right, walking and trotting gaits on the ANYmal robot, respectively; (bottom) biped walking gait using the Talos' legs. You can run these results in our repository

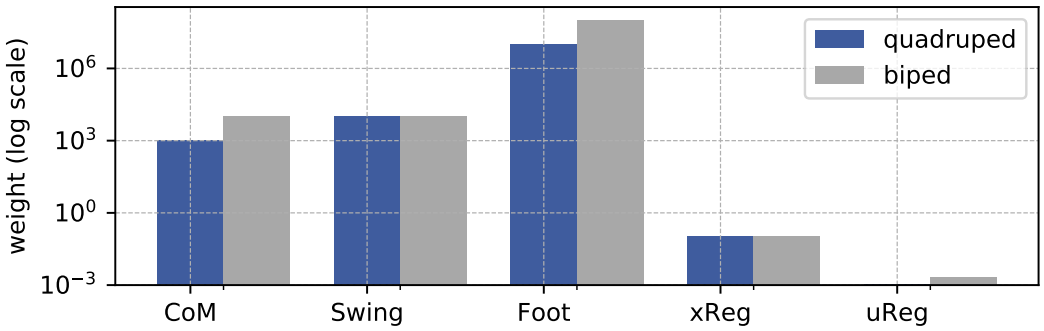


Fig. 4.4: Weight values for each cost function used in the generation of the legged gaits. The cost functions are (a) CoM: CoM tracking of the interpolated postures; (b) Swing: swing tracking of the reference foot trajectory; (c) footstep deviation from the predefined placement; (d) state regularization; (e) control regularization. Note that the values are in logarithmic scale.

4.4 Results

Crocoddyl library was used to generate gaits for both quadruped and biped robots. Since FDDP can be used to infeasibly start the DDP solver, we can easily initialize the problems with infeasible guesses. We analyzed the convergence patterns, and find the computation times for our Model-based Optimal Control problems over a horizon.

4.4.1 Simulation Examples

Some of the gaits generated by Crocoddyl are shown in Fig. 4.3. We computed different gaits walking, trotting, pacing, and bounding with our FDDP algorithm in the order of milliseconds. All these gaits are a direct outcome of our algorithm, we only need to predefine the sequence of contacts and the step timings. These motions are computed in around 12 iterations with the exception of the bounding gait which takes at least 18 iterations. We used the same weight values and cost functions for all the quadrupedal gaits, and similar weight values for the bipedal walking. Indeed, we noticed that these weight values and cost functions might work out of the box for other legged robots, e.g., the HyQ and the ICub robots. We report the used values in Fig. 4.4.

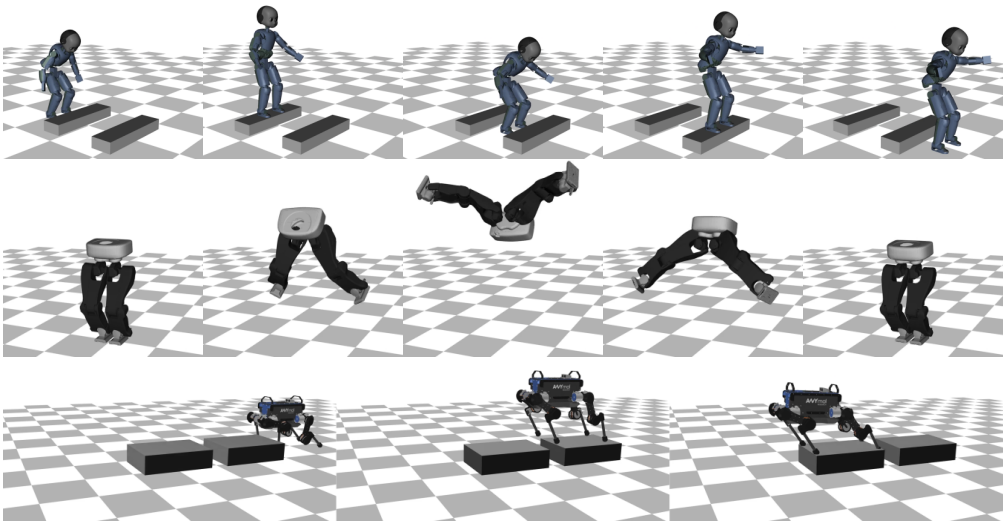


Fig. 4.5: Snapshots of highly-dynamic maneuvers in legged robots using the FDDP algorithm in Crocoddyl. (top) jumping obstacles in a humanoid robot; (middle) front-flip maneuver in a biped robot; (bottom) jumping obstacles in a quadruped robot.

The cost function is composed of the CoM and the foot placement tracking costs together with regularization terms for the state and control. We warm-start our solver using a linear interpolation between the nominal body postures of a sequence of contact configurations. This provides us a set of body postures together with the nominal joint postures as state warm-start $\underline{x}_0$. Then, the control warm-start $\underline{u}_0$ is obtained by applying the quasi-static assumption along $\underline{x}_0$ (i.e., gravity compensating control torque as initial warm-start). During contact switches, we use the impulse dynamics to ensure that the impulse is minimized. We observed that the use of impulse models improves the algorithm convergence compared to penalizing the contact velocity. We used a weighted least-square activation function to regularize the state with respect to the nominal robot posture, and quadratic activation functions for the tracking costs and control regularization.

In addition to the walking gaits, highly dynamic maneuvers like jumps and flips were also computed using Crocoddyl and the FDDP solver. These are shown in Fig. 4.5. One major advantage of FDDP algorithm is clearly evident in the generation of highly-dynamic maneuvers. As a result of gravity, feasible rollouts might produce trajectories that are numerically unstable and far from the solution. The single-shooting nature of the classical DDP means that feasible rollouts in the first iterations would make the robot fall in the empty space during jumps; it would struggle to find the appropriate launching and landing in order to solve these kind of problems. To overcome this limitation in the DDP, a proper warm-start is needed. This limits our solver to problems for which a suitable warm-start can be provided. FDDP does not suffer from these limitations.

4.4.2 Convergence

We analyzed the gaps contraction and convergence rates for all the presented motions: quadrupedal and bipedal gaits, and highly-dynamic maneuvers. To easily compare the results, we normalize the gaps and cost values per each iteration as shown in Fig. 4.6. For the case of the gaps plot Fig. 4.6 (top), we use the L2-norm of the total gaps. These results show that keeping the gaps open is particularly important for highly-dynamic maneuvers such as jumping. Indeed, the DDP solver immediately closes the gaps for all the legged gaits. The reason is that the solver can apply a full-step in the first iteration with simply using a big initial regularization value. A big regularization value changes the search-direction from Newton to steepest-descent. ANYmal and ICub jumps are computed with a sequence of Optimal Control problems, where each jump formulates a single problem. Additionally, we observed in practice super-linear convergence of FDDP algorithm after closing the gaps. This is expected since FDDP behaves as DDP when the gaps are closed. Highly-dynamic maneuvers have a lower rate of improvement in the first iterations Fig. 4.6 (bottom). The same occurs in the quadrupedal walking case (walk-4f), in which the four-feet support phases have a very short duration ($\Delta t = 2$ms).

The motions converge within 10 to 34 iterations, with an overall computation time of less than 0.5s. The numerical integration step size is often $\delta t = 10^{-2}$s, with the exception of the biped walking $\delta t = 5 * 10^{-3}$s, and the number of nodes are typically between 60 to 115. Therefore, the optimized trajectories have a horizon of between 0.6-3s.

4.4.3 Computational Benchmarks

Without multi-threading, our efficient implementation of contact dynamics achieves computation rates up to 859.6Hz for the quadrupedal gaits with 60 nodes (jump-4f on i9-9900K). We parallelize only the computation of the derivatives. For this case, roughly speaking, we reduce the computation time in half using four to eight threads (cf. Fig. 4.7). To understand the performance of Crocoddyl, we have run 5000 trials for each of the motions presented in this chapter on four different Intel PCs with varying levels of parallelization:
PC1: i7-6700K @ $4.00\text{GHz} \times 8$ with 32 GB 2133MHz RAM,
PC2: i7-7700K @ $4.20\text{GHz} \times 8$ with 16 GB 2666MHz RAM,
PC3: i9-9900K @ $3.60\text{GHz} \times 16$ with 64 GB 3000MHz RAM, and
PC4: i7-9900XE @ $3.00\text{GHz} \times 36$ with 128 GB 2666MHz RAM.

We used the optimal number of threads for each PC as identified in Fig. 4.7. The computation frequency per one iteration is reported in Fig. 4.8.

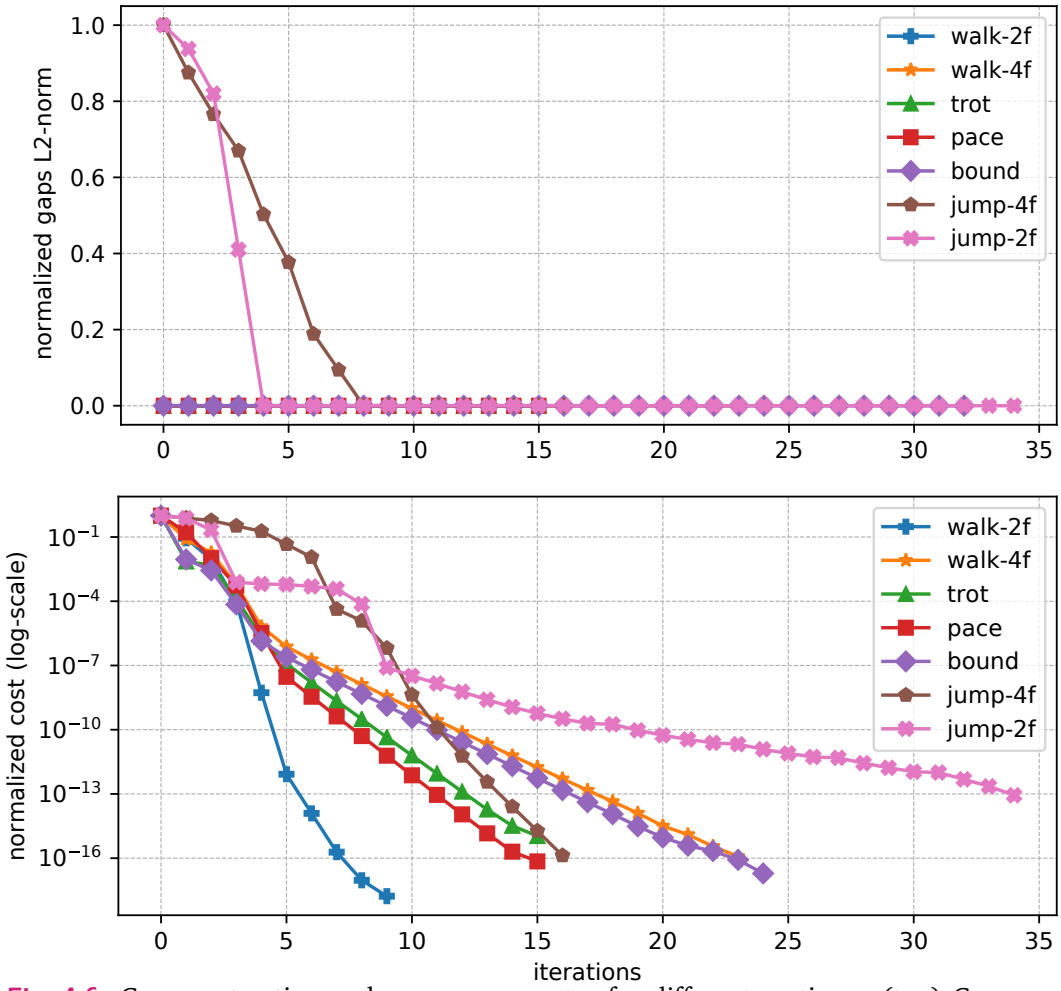


Fig. 4.6: Gaps contraction and convergence rates for different motions. (top) Gaps are closed in the first iteration for simpler motions such as biped walking and quadrupedal gaits. Instead, the FDDP solver chooses to keep the gaps open for the early iterations for highly-dynamic maneuvers. Here we use the L2-norm of the total gaps, i.e., gaps for all the nodes of the trajectory. (bottom) The required iterations increases mainly with the dynamics of the gait and numbers of nodes. For instance, we can see lower rate of improvement in the first nine iterations in the ANYmal (jump-4f) and ICub jumps (jump-2f). In case of the quadrupedal walking, we have very short durations in the four-feet support phases, making it a *dynamic* walk.

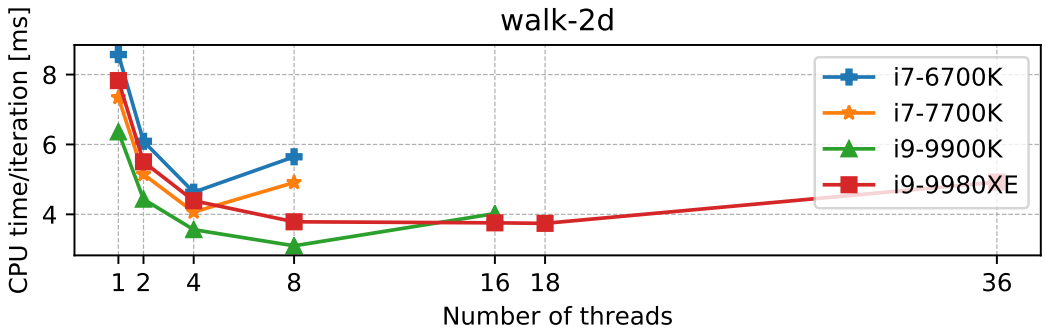


Fig. 4.7: Computation time per iteration for different CPUs and level of parallelism. Note that the use of hyper-threading decreases the computation frequency for all tested CPUs.

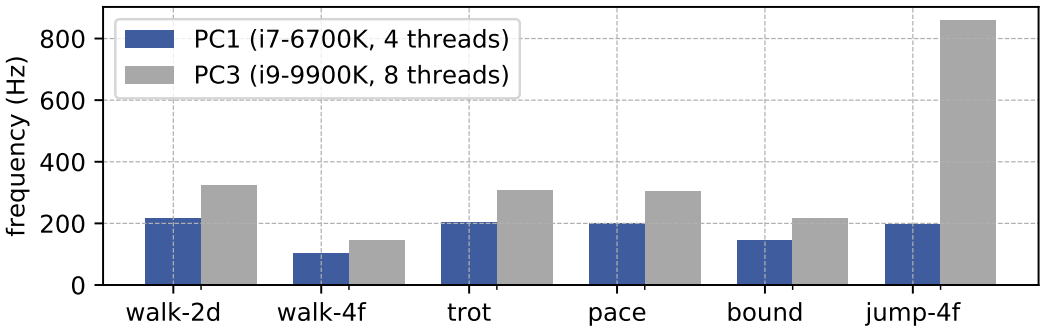


Fig. 4.8: Computation frequency per iteration for different motions for different PCs. PC1 has specifications typical for on-board computers found on robots, while PC3 uses high-performance CPU and RAM. The reported values use the optimal number of threads as identified in Fig. 4.7.

4.5 Conclusion and Perspective

In this chapter, we have introduced Crocoddyl, a solver meant for multi-body locomotion generation. During the development of Crocoddyl, we have focussed on reducing the processing time which is usually required for solving large dimension problems. Our goal with crocoddyl is using trajectory optimization in order to do MPC on the real robot. All the features presented in this chapter are a step towards that goal.
1) The contact-constrained DDP solver (introduced in Chapter 3) uses the sparsity of the optimization problem to find the solution in a recursive manner.
2) The analytical derivatives of the dynamics have a big effect on the computation time (see benchmarks in Carpentier et al., 2019).
3) The multi-threading on the backward pass uses the best possible speed offered by processors in order to compute the derivatives.
4) The memory-conscious problem transcription avoids unnecessary memory allocations.
5) FDDP (Mansard, 2019) helps to start the search in the infeasible domain, while avoiding the merit functions needed to remove infeasibility.

We use DDP as the base algorithm in the solvers of crocoddyl. The reason is simple, our Hessian matrix is block diagonal, and DDP can invert the Hessian by iteratively inverting each block of the block diagonal matrix. Adding control constraints does not effect this block-diagonal structure. Thus, we were able to introduce the control-limited formulation proposed by Tassa et al., 2014 as one of the solvers. However, this block diagonal structure is ruptured when state constraints are added to DDP. Indeed, DDP has struggled when dealing with state constraints, and active set methods such as the one proposed by Xie et al., 2017 struggle with regards to performance. Similar issues can be observed when doing self-collision avoidance on DDP, as it is completely dependent on the current joint configuration. We convert

the state dependent constraints such as joint configuration limits to cost terms with a high weight, and this helps to produce the required motions. But this is not the best-possible approach to deal with state constraints, and a different approach is definitely needed for completeness.

A similar limitation of our contact-constrained formulation can be expected when we aim to do admittance control on the robot with Crocoddyl as the solver. Our contact-constrained DDP (from Chapter 3) defines the contact forces as a function of the instantaneous state and the control. While this is good for the computations, this formulation is ill-equipped to handle the force feedback. In order to do force-based control with DDP as an MPC controller, contact force needs to be somehow a part of the state vector, whether explicitly, or implicitly. One such approach has been used by Neunert et al., 2018, where they use the spring-damper contact model. Unfortunately, such models introduce an additional stiffness in the numerical integration schemes (Press et al., 2007). Since DDP is a shooting solver which integrates the dynamics at each node, this leads to a deterioration in the performance of the solver. A way forward which does not affect the computational efficiency while allowing force-control on the robot is another way that crocoddyl can be improved in the future.

5 Alternating Direction Method of Multipliers for locomotion

Suis-je tellement dépendant du corps et des sens, que je ne puisse être sans eux?
(Am I so tied to a body and senses that I am incapable of existing without them?)

— **René Descartes, on mind-body dualism**
(Méditations métaphysiques)

Budhiraja, Rohan, *Justin Carpentier*, and *Nicolas Mansard* (2019). Dynamics Consensus between Centroidal and Whole-Body Models for Locomotion of Legged Robots. In: ICRA 2019 - IEEE International Conference on Robotics and Automation. Montreal, Canada

5.1 Introduction

We look at the problem of generating consistent and coherent momentum (CoM and AM) and forces at both centroidal and whole-body levels. In Chapter 2, we saw that proxies could be used to deal with the coupling constraints between the centroidal and whole-body problems. However, due to the dynamic nature of AM and Contact Forces, it wasn't possible to use machine learning techniques to learn these constraints into a proxy. In Chapter 3, we saw how to update the rigid body dynamics to incorporate the contact constraints within the DDP framework, and in Chapter 4, we saw our implementation. In Chapter 4, we show that our implementation is capable of providing real-time control at a high bandwidth. We have already discussed our centroidal solver in Carpentier and Mansard, 2018b.

In this chapter, we claim that given the efficacy of our solvers, it is possible to incorporate a feedback from the whole-body dynamics solver towards the centroidal

problem. This would improve the consistency of the global locomotion solution within a few iterations. We will use an alternating scheme to ensure that our two problems don't diverge.

5.1.1 Why should we alternate?

In a first implementation, it is often proposed to first compute the centroidal pattern and then track it by solving the Lagrangian dynamics. The whole-body movement is usually computed with an IK/ID, which are theoretically equivalent to solving the Lagrangian part of (1.16) but with a void horizon $T = 0$. With such a simplification, it is desired that the *AM output* from the centroidal problem must match near perfectly to the *AM requirement* by the Lagrangian part. However, this is not possible without some form of feedback (in the form of alternation) on the AM value. While this requirement is known by many teams, we believe that it is not sufficiently documented and explain why alternating is important.

Momentum variations are caused by the forces exerted by the environment at the contact level, and they result in the motion of the limbs (also called "gesticulation" by Wieber, 2006a).

As the centroidal model does not understand gesticulation(which depends on the kinematic and dynamic structure of the robot), it is not possible to get the correct momentum estimate when considering only the centroidal quantities. Consider the example of the biped locomotion gait: an astronaut mimicking walk in deep space would rotate (pitch rotation) on the spot. We showed a similar rotation, but in the transverse plane, for an astronaut in Fig. 3.1. However, we don't have to go to outer space to find an example. A cat falling on the ground demonstrates the same principle as our astronauts (see Fig. 3.2).

This is due to the asymmetry in the movements of the limbs during the forward and backward motions. As we are not rotating during locomotion, we can conclude that we exert some contact forces to counterbalance this rotation effect. These extra forces, which are required due to gesticulation, cannot be decided from the centroidal model alone. Consequently, trying to approximately match the AM computed by the centroidal solver is a bad idea. The same is also true for CoM (linear momentum) trajectory, which should change to account for the change in forces, according to (1.13).

A pragmatic solution is to compute the centroidal pattern by trying to match the AM that the limbs will generate (Herzog et al., 2016b). This implicitly suggests that we are not expecting to use the AM variations to improve the walk, but we are just

trying to compensate for it. This is the standard implementation of the table-cart pattern generator, by adding a second stage of ZMP-CoM computation (Kajita et al., 2003). It has also been proposed to couple an IK with a centroidal solver (Dai et al., 2014; Herzog et al., 2015). In both cases, it has been experimentally observed that alternating twice is enough to obtain a consensus. However, no theoretical basis has yet been provided. In Sec 5.4, we propose to alternate using an existing theoretical framework which forces consensus as an output of the optimization.

None of the aforementioned methods *ensure* convergence of the 2 subproblems to a common and same solution (a consensus), notably for the angular momentum. Rather, they rely on the ability of the individual solvers to produce mutually feasible solutions without properly considering the global structure of the problem. This forces the solvers to have additional robustness in order to account for the lack of structure in the simplified subproblems.

5.1.2 Outline of the chapter

Previously, in Sec 1.2, we explained the decoupled approach to solving the locomotion problem, and in Sec 1.3 we explained the optimal control problem (OCP) dedicated for predefined contacts and which exhibit a complete splitting between centroidal and Lagrangian dynamics. Consequently, in Chapter 2 and Chapter 3, we made obvious the cost implied by solving both subproblems. However, in the previous chapters, we solved our problems without maintaining consensus between the two subproblems.

Our main contribution is to introduce a well-posed mathematical formulation that properly enforces this consensus. Rather than giving the solution of the whole-body problem directly to the centroidal optimizer as done by (Herzog et al., 2016b), we rely on the ADMM technique to handle this feedback communication.

ADMM is an old but well established method for solving convex problems in which the objective is separable into two mutually exclusive cost functions along a set of problem variables. While the method has been around for decades, it was recently reintroduced (Boyd et al., 2010) to solve large scale distributed optimization problems subject to constraints. ADMM provides a feedback to the subproblems in the form of the sum of the residues on the constraints, and in that fashion, it behaves similar to an integral controller. For example, the feedback property has been exploited in (Mordatch et al., 2014) to alternate between trajectory and policy optimization.

We find that the robustness and simplicity of this technique makes it an ideal candidate for solving the global optimization problem of locomotion as well.

In Section 5.2, we detail how ADMM can be exploited to solve the complete OCP by alternatively solving the two subproblems. We gather the details of implementation in Section 5.4, used to obtained experimental results on the HRP-2 robot in Section 5.5.

5.2 Short overview of ADMM

In this section, we give a short description of the ADMM technique. We then apply it on the global OCP for locomotion (1.16).

ADMM is a simple optimization technique to solve constrained problems of the form:

$$\begin{aligned} \underset{\boldsymbol{x},\boldsymbol{z}}{\text{minimize}} \quad & l_1(\boldsymbol{x}) + l_2(\boldsymbol{z}) \\ \text{subject to} \quad & A\boldsymbol{x} + B\boldsymbol{z} = \boldsymbol{c} \end{aligned} \tag{5.1}$$

where the cost function is composed of two separable objectives $l_1(\boldsymbol{x})$ and $l_2(\boldsymbol{z})$. The main idea behind ADMM is to exploit this splitting between cost terms in a recursive manner, allowing to solve simpler problems than the original one (Boyd et al., 2010). This precise point can be highlighted by writing the augmented Lagrangian associated with the constrained optimization problem (5.1):

$$\mathcal{L}_\rho(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{y}) \;=\; l_1(\boldsymbol{x}) + l_2(\boldsymbol{z}) + \boldsymbol{y}^T(A\boldsymbol{x} + B\boldsymbol{z} - \boldsymbol{c}) + \frac{\rho}{2}\|A\boldsymbol{x} + B\boldsymbol{z} - \boldsymbol{c}\|_2^2 \tag{5.2}$$

where $\boldsymbol{y}$ is the vector of dual variables associated with the constraint $A\boldsymbol{x} + B\boldsymbol{z} = \boldsymbol{c}$ and $\rho > 0$ is the penalty parameter which penalizes the violations of this constraint. The solution is then found by the following steps recursions[1]:

$$\boldsymbol{x}^{k+1} = \arg\min_{\boldsymbol{x}} \mathcal{L}_\rho(\boldsymbol{x}, \boldsymbol{z}^k, \boldsymbol{y}^k) \tag{5.3a}$$

$$\boldsymbol{z}^{k+1} = \arg\min_{\boldsymbol{z}} \mathcal{L}_\rho(\boldsymbol{x}^{k+1}, \boldsymbol{z}, \boldsymbol{y}^k) \tag{5.3b}$$

$$\boldsymbol{y}^{k+1} = \boldsymbol{y}^k + \rho(A\boldsymbol{x} + B\boldsymbol{z} - \boldsymbol{c}) \tag{5.3c}$$

Problem (5.3a) and (5.3b) are minimization over l_1 and l_2 respectively (with an additional quadratic term). It is also worth noticing that the dual variable $\boldsymbol{y}$, through the update (5.3c) acts as an integral term by collecting the residues on the consensus between the two subproblems, and forces the residual to converge to 0 along the iterations.

[1]Throughout this chapter, superscripts are used to refer to the current iteration of the solver

5.3 Rational for our alternative scheme

ADMM provides a way for us to exploit the splitting of dynamic variables exposed in Sec 1.3.4, and defines a mathematical framework to feedback and optimize the AM variable inside the centroidal OCP. OCP (1.16) does not match exactly the pattern of (5.1). For convenience, we recall OCP (1.16) here, along with the coupling constraints (1.15). The structure of the full optimization problem, as explained in Chapter 1 (Eq 1.16), is thus shown by:

$$
\begin{aligned}
\underset{\substack{\underline{\mathbf{d}}_c:=[\mathbf{c},\dot{\mathbf{c}},\boldsymbol{L},\boldsymbol{\lambda}] \\ \underline{\mathbf{d}}_l:=[\mathbf{q},\dot{\mathbf{q}},\boldsymbol{\tau}]}}{\text{minimize}} \quad & \sum_{s=1}^{S} \int_{t_s}^{t_s+\Delta t_s} \ell_c(\mathbf{d}_c|\mathbf{S})\, dt + \sum_{s=1}^{S} \int_{t_s}^{t_s+\Delta t_s} \ell_l(\mathbf{d}_l|\mathbf{S})\, dt && \text{(5.4a)} \\
\text{subject to} \quad & \forall t \quad \boldsymbol{\lambda} \in \mathcal{K} && \text{(5.4b)} \\
& \forall t \quad g_\lambda(\mathbf{q}, \dot{\mathbf{q}}, \boldsymbol{\tau}) \in \mathcal{K} && \text{(5.4c)} \\
& \forall t \quad \dot{\mathbf{x}}_c = f_c(\mathbf{d}_c) && \text{(5.4d)} \\
& \forall t \quad \dot{\mathbf{x}}_l = f_l(\mathbf{d}_l) && \text{(5.4e)} \\
& \forall t \quad \mathbf{c} = CoM(\mathbf{q}) && \text{(5.4f)} \\
& \forall t \quad \begin{bmatrix} m\dot{\mathbf{c}} \\ \boldsymbol{L} \end{bmatrix} = \boldsymbol{A}_g(\mathbf{q})\, \dot{\mathbf{q}} && \text{(5.4g)} \\
& \forall t \quad \boldsymbol{\lambda} = g_\lambda(\mathbf{q}, \dot{\mathbf{q}}, \boldsymbol{\tau}) && \text{(5.4h)} \\
& \quad \mathbf{x}_c(0) \text{ is given}, \mathbf{x}_c(T) \text{ is viable} && \text{(5.4i)} \\
& \quad \mathbf{x}_l(0) \text{ is given}, \mathbf{x}_l(T) \text{ is viable} && \text{(5.4j)}
\end{aligned}
$$

While (5.1) has a linear coupling constraint, (5.4) has three nonlinear coupling constraints((5.4f), (5.4g), (5.4h)). In addition, there are decoupled constraints for the whole-body and the centroidal problems, which must be satisfied by each problem.

Non-linearity is a theoretical issue and it makes the problem non-convex. Convergence guarantee with ADMM are yet only obtained for convex problems with linear constraints Eckstein and Bertsekas, 1992. Yet ADMM with non-convex problems would act as just another local optimizer. With ADMM, we at least know that the linearization of (5.4) at the current estimate will converge. In practice, ADMM is often used, with good empirical results, for solving problems with non-convex objectives (Boyd et al., 2010; Wang et al., 2019), and for non-linear constraints (Benning et al., 2016).

In our case, for the 3 semi-infinite (i.e. defined $\forall t$) coupling constraints, we have to introduce 3 multipliers functions of time. For the additional decoupled constraints, we handle them in the solvers of each subproblems. Let us remember that the partial solutions $\underline{\mathbf{d}}_c$ and $\underline{\mathbf{d}}_l$ of each subproblems should respect these additional constraints.

In the next section, we first explain the alternating algorithm with the hypothesis that some oracles can be called to provide the optimum of the two subproblems. Then, we describe with more details which centroidal and whole-body solvers we used for the experiments.

5.4 Application of ADMM for solving the locomotion problem

Let us associate with each coupling constraint a residual function:

$$\forall t \quad \boldsymbol{r}_c(\mathbf{c}, \mathbf{q}) = \mathbf{c} - CoM(\mathbf{q}) \tag{5.5a}$$

$$\forall t \quad \boldsymbol{r}_m(\dot{\mathbf{c}}, \boldsymbol{L}, \mathbf{q}, \dot{\mathbf{q}}) = \begin{bmatrix} m\dot{\mathbf{c}} \\ \boldsymbol{L} \end{bmatrix} - \boldsymbol{A}_g(\mathbf{q})\,\dot{\mathbf{q}} \tag{5.5b}$$

$$\forall t \quad \boldsymbol{r}_\lambda(\lambda, \mathbf{q}, \dot{\mathbf{q}}, \boldsymbol{\tau}) = \lambda - g_\lambda(\mathbf{q}, \dot{\mathbf{q}}, \boldsymbol{\tau}) \tag{5.5c}$$

Residuals $\underline{\boldsymbol{r}}_c, \underline{\boldsymbol{r}}_m, \underline{\boldsymbol{r}}_g$ respectively corresponds to constraints (1.15a), (1.15b), (1.15c). We also respectively define $\underline{\boldsymbol{y}}_c, \underline{\boldsymbol{y}}_m, \underline{\boldsymbol{y}}_g$ as the multipliers corresponding to these 3 constraints.

For convenience, let us define $\underline{\boldsymbol{r}}$ as the augmentation of (1.16) (i.e. sum of linear and quadratic penalization):

$$\underline{\boldsymbol{r}}(\underline{\mathbf{d}}_c, \underline{\mathbf{d}}_l, \underline{\boldsymbol{y}}) = \sum_{k=m,c,\lambda} \int_0^{T_f} \boldsymbol{y}_k^T(t)\boldsymbol{r}_k(t) + \frac{\rho_k(t)}{2} \|\boldsymbol{r}_k(t)\|_2^2 \, dt \tag{5.6}$$

We can now separate the augmented Lagrangian of the global OCP into centroidal and full body parts:

$$\begin{aligned}\mathcal{L}_\rho^c(\underline{\mathbf{d}}_c, \underline{\mathbf{d}}_l, \underline{\boldsymbol{y}}) &= \sum_{s=1}^{S} \int_{t_s}^{t_s+\Delta t_s} \ell_c(\mathbf{d}_c)\, dt + \underline{\boldsymbol{r}} \\ \mathcal{L}_\rho^l(\underline{\mathbf{d}}_c, \underline{\mathbf{d}}_l, \underline{\boldsymbol{y}}) &= \sum_{s=1}^{S} \int_{t_s}^{t_s+\Delta t_s} \ell_l(\mathbf{d}_l)\, dt + \underline{\boldsymbol{r}}\end{aligned} \tag{5.7}$$

where $\boldsymbol{y}$ is the stack of the 3 multipliers. Note that the multipliers are trajectories of vectors, while $\rho_c, \rho_m, \rho_\lambda$ are trajectories of scalars.

Using the definition of ADMM from Sec 5.2, the global OCP can thus be solved by the following iterations:

$$\begin{aligned}
&\underline{\mathbf{d}}_c^{k+1} = \arg\min_{\underline{\mathbf{d}}_c} \mathcal{L}_\rho^c(\underline{\mathbf{d}}_c, \underline{\mathbf{d}}_l^k, \underline{\boldsymbol{y}}^k) \text{ subject to } (1.16b), (1.16d), (1.16g) \\
&\underline{\mathbf{d}}_l^{k+1} = \arg\min_{\underline{\mathbf{d}}_l} \mathcal{L}_\rho^l(\underline{\mathbf{d}}_c^{k+1}, \underline{\mathbf{d}}_l, \underline{\boldsymbol{y}}^k) \text{ subject to } (1.16c), (1.16e), (1.16h) \\
&\forall t \quad \boldsymbol{y}_c^{k+1} = \boldsymbol{y}_c^k + \rho_c(\mathbf{c}^{k+1} - CoM(\mathbf{q}^{k+1})) \\
&\forall t \quad \boldsymbol{y}_m^{k+1} = \boldsymbol{y}_m^k + \rho_m(\begin{bmatrix} m\dot{\mathbf{c}}^{k+1} \\ \boldsymbol{L}^{k+1} \end{bmatrix} - \boldsymbol{A}_g\left(\mathbf{q}^{k+1}\right)\dot{\mathbf{q}}^{k+1}) \\
&\forall t \quad \boldsymbol{y}_\lambda^{k+1} = \boldsymbol{y}_\lambda^k + \rho_\lambda(\lambda^{k+1} - g_\lambda(\mathbf{q}^{k+1}, \dot{\mathbf{q}}^{k+1}, \boldsymbol{\tau}^{k+1}))
\end{aligned} \tag{5.8}$$

5.4.1 On the advantages of scaling the dual variables

Scaling the dual variables through $\boldsymbol{z} = \frac{\boldsymbol{y}}{\rho}$ is more convenient to implement as it combines under a single norm minimization both the linear and quadratic terms in the augmented Lagrangian (5.7). The subproblems are then reduced to simply minimize the residual squared norm of the constraints linking the two problems Boyd et al., 2010.

5.4.2 The ADMM solver for locomotion

By using $\underline{z}$, we can simplify the notation further by collecting the constraints (1.15a), (1.15b) and (1.15c) into a single quantity. For that aim, let us define Φ_c to contain the elements $(\boldsymbol{c}, m\dot{\mathbf{c}}, \boldsymbol{L}$ and $\boldsymbol{\lambda})$. Similarly, let us define Φ_l to contain the mappings $(CoM(\mathbf{q}), \boldsymbol{A}_g(\mathbf{q})\dot{\mathbf{q}}$ and $g_\lambda(\mathbf{q}, \dot{\mathbf{q}}, \boldsymbol{\tau}))$. Let us define addition/subtraction operations on Φ to be the addition/subtraction on its corresponding elements and a function $DMap$ which maps the centroidal($\mathbf{d}_c$) and whole-body($\mathbf{d}_l$) variables onto these el-

ements. The iterations (5.8) can be then written equivalently as Algorithm 1. In Algorithm 1, the first input to the solvers is the solution of the previous iteration (as a warm start), and the second input is the reference to be tracked for the augmented Lagrangian quadratic costs.

Algorithm 1: ADMM solver for locomotion

Data: $\underline{\mathbf{d}}_c^0, \underline{\mathbf{d}}_l^0$

$\underline{\mathbf{d}}_c \leftarrow \underline{\mathbf{d}}_c^0, \qquad \underline{\mathbf{d}}_l \leftarrow \underline{\mathbf{d}}_l^0;$
$\underline{\Phi}_c \leftarrow DMap(\underline{\mathbf{d}}_c), \quad \underline{\Phi}_l \leftarrow DMap(\underline{\mathbf{d}}_l);$
$\underline{\Phi}_d \leftarrow \underline{\Phi}_c - \underline{\Phi}_l;$
repeat
  $\underline{\mathbf{d}}_c, \underline{\Phi}_c \leftarrow CentroidalSolver(\underline{\mathbf{d}}_c, \underline{\Phi}_l - \underline{\Phi}_d);$
  $\underline{\mathbf{d}}_l, \underline{\Phi}_l \leftarrow WholebodySolver(\underline{\mathbf{d}}_l, \underline{\Phi}_c + \underline{\Phi}_d);$
  $\underline{\Phi}_d \leftarrow \underline{\Phi}_d + \underline{\Phi}_c - \underline{\Phi}_l;$
until *convergence*;
$\underline{\mathbf{d}}_c^* \leftarrow \underline{\mathbf{d}}_c, \qquad \underline{\mathbf{d}}_l^* \leftarrow \underline{\mathbf{d}}_l;$

Result: $\underline{\mathbf{d}}_c^*, \underline{\mathbf{d}}_l^*$

5.4.3 Initializing the dual variables

Many of the currently available centroidal and whole-body dynamics solvers are tailored for producing solutions which are almost always mutually feasible (Herzog et al., 2016a Budhiraja et al., 2018 Carpentier and Mansard, 2018b). As a result,with this assumption of feasibility, a simple feedback from the whole-body solver to the centroidal solver generates acceptable result within the second iteration (Herzog et al., 2016a). On the other hand, since ADMM transmits the residual of the two subproblems and not the output, an *uninitialized* ADMM solver would overshoot and only promise convergence from the third iteration, as we later see in Sec 5.5. There is indeed a minimum number of iterations to synchronize the two subproblems.

However, this extra iteration of the solver could be avoided by setting the dual after the first iteration as $\underline{\Phi}_d^1 \leftarrow \underline{\Phi}_c^1$. This change makes the feedback from the first iteration to be equal to the output, and the knowledge of individual solvers can then be exploited to converge within two iterations. Indeed, the second iteration of the ADMM solver then becomes equivalent to (Herzog et al., 2016a), and would provide similar results without affecting convergence. While not used in this paper, this trick needs to be evaluated further.

5.4.4 Key observations for whole-body solver

During the full-body step of (5.8), the knowledge of λ^{k+1} from the centroidal step can be exploited to ensure that the Constraint (1.16c) is never violated. In the

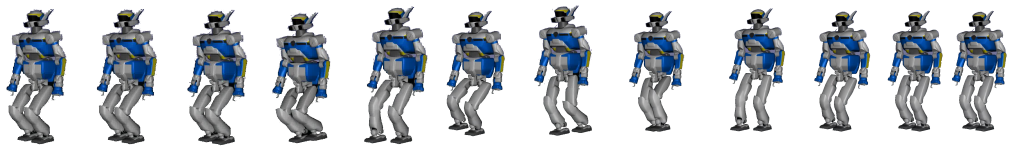


Fig. 5.1: Walking sequence generated for HRP-2 robot using the proposed ADMM solver.

full-body step (by the mapping g_λ) $\boldsymbol{\lambda}$ is dependent on $\mathbf{q}, \dot{\mathbf{q}}, \boldsymbol{\tau}$. A predefined $\boldsymbol{\lambda}_{ref}^{k+1}$ provides a good reference for the contact forces. Moreover, since the $\boldsymbol{\lambda}_{ref}^{k+1}$ was generated by the centroidal optimizer, we are sure that this reference always satisfies the force friction cone constraint.

We can exploit this in the whole-body solver. Since the friction cone constraint has already been satisfied by the centroidal optimizer, we can avoid this constraint for the whole-body problem. With a good tracking, Constraint (1.16c) can be relaxed.

An implementation of the full-body step with rigid contacts which exploits such a reference tracking was proposed in Chapter 3. Note that in Chapter 3 $\mathbf{c}$ and $\boldsymbol{\lambda}$ were being tracked in the full-body OCP, while in the present formulation they are updated during the dual ascent.

5.5 Experimental results

In this section, we describe the practical details of our implementation, enabling us to simply add the alternating descent on top of our existing frameworks for solving centroidal and whole-body dynamics OCP, with only a minor additional cost in terms of development.

In addition, we validate and highlight the efficiency of the proposed approach in simulation. For that purpose, we study the convergence properties of our solver on a simplified version of the humanoid robot HRP-2 only composed of its lower limbs. We generate a walking motion composed of 3 steps of 40 cm step length, depicted in Fig. 5.1.

5.5.1 Locomotion Pipeline

Our locomotion framework (Loco3D) was previously explained in Sec 1.4. For these experiments, and for coupling, we use the following solvers in the pipeline:

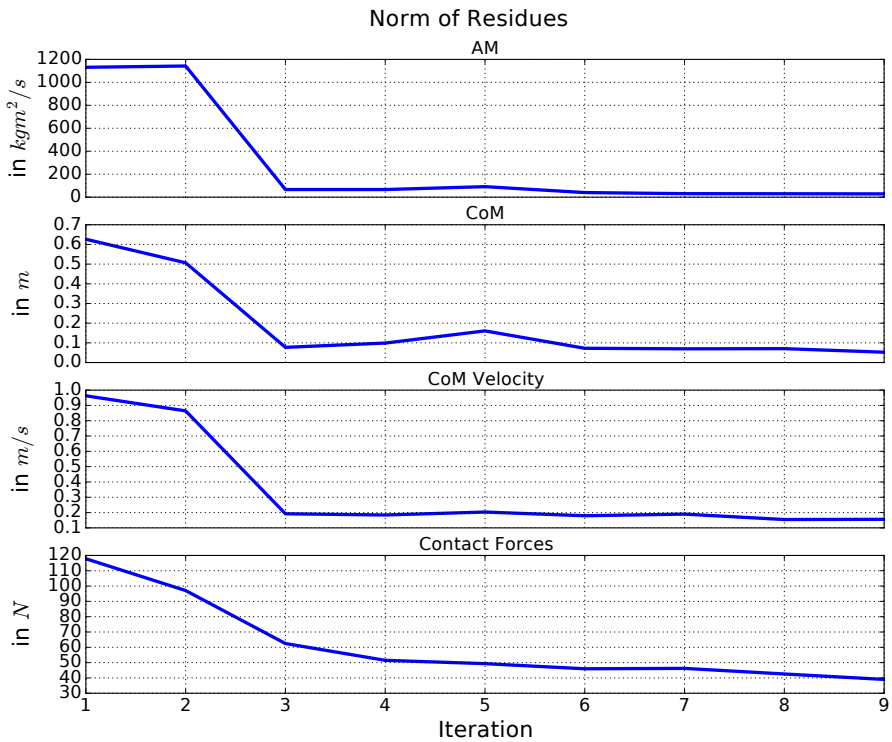


Fig. 5.2: Evolution of the total norm of the constraint residual along the iterations of the ADMM solver.

The Contact Sequence Generator: We use the randomized motion planner described in Tonneau et al., 2018b which ensures the feasibility of the sequence at the kinematic level, with real-time performances;

The Centroidal Optimizer: As explained in Chapter 2, we solve the trajectory optimization problem over the centroidal dynamics using feasibility measures to enforce the kinematic feasibility of the solution with respect to the whole-body problems. We use MUSCOD-II (Leineweber et al., 2003), which implements an efficient multiple-shooting algorithm particularly suited for the multiple contact phases of the locomotion problem. The output of the solver is the centroidal trajectories $\underline{\mathbf{d}}_c$;

The Whole-Body Optimizer: We solve the full-body trajectory optimization problem subject to the dynamics constraints by the DDP algorithm explained in Chapter 3, which accounts for the contact constraints in the dynamics of the problem. The output of the solver is the whole-body trajectories $\underline{\mathbf{d}}_l$.

While we use our own solvers in these experiments, it should be noted that the framework described in chapter is independent of the individual solvers that are used.

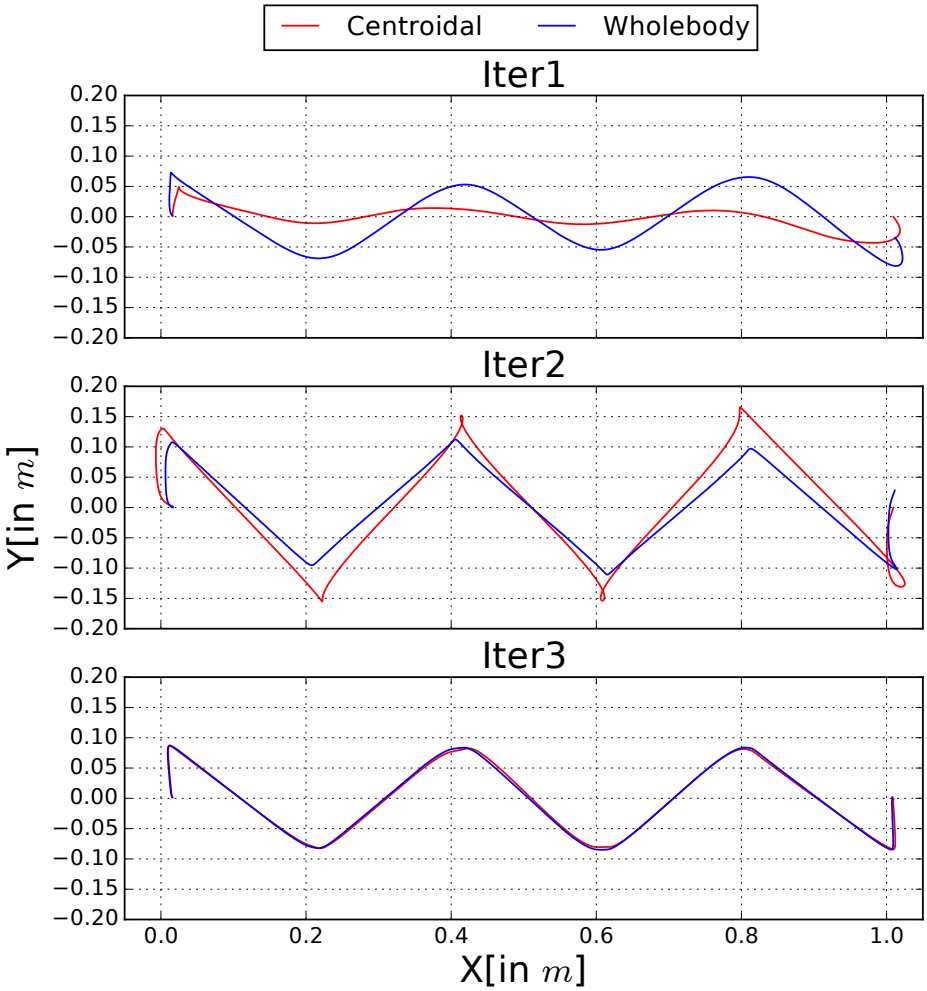


Fig. 5.3: CoM Trajectory in the XY plane for the first three iterations of the ADMM solver.

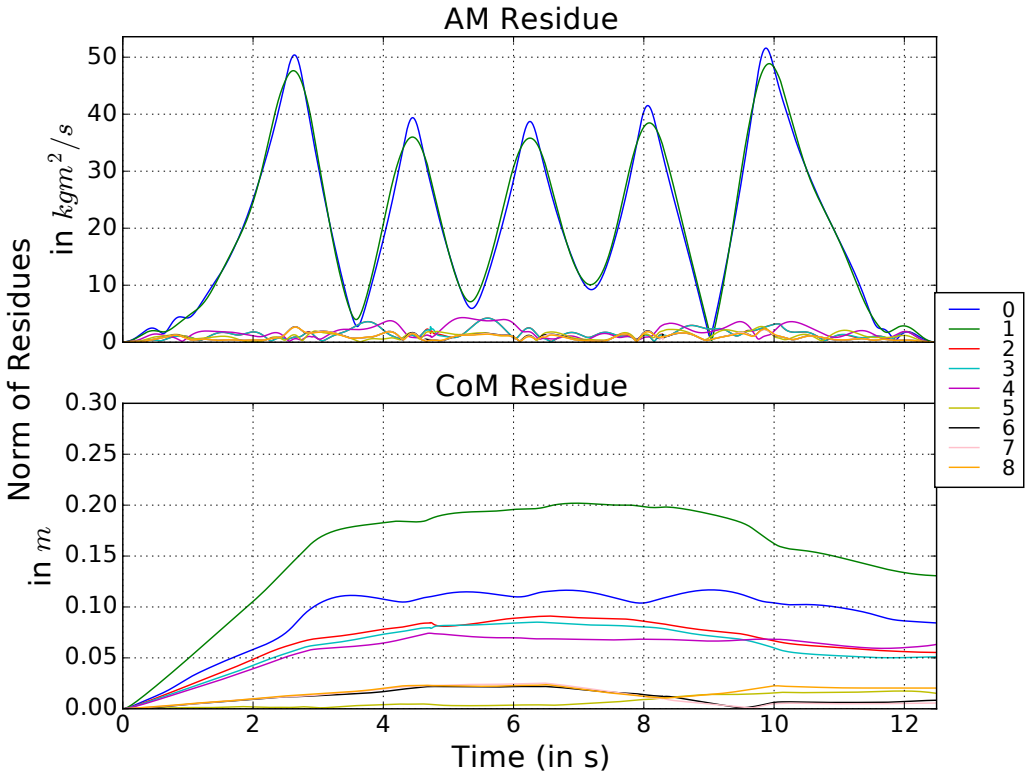


Fig. 5.4: Residue between the centroidal and whole-body trajectory of AM and CoM over the iterations.

5.5.2 Cost functions

The centroidal cost function

For the first iteration of the centroidal solver, we minimize the log barrier on the feasibility measure of the CoM (explained in Chapter 2), and we regularize the CoM velocity (weight: 10) and contact forces (weight:10^{-4} for linear and 10^{-2} for angular) with quadratic costs. For subsequent iterations, we solely minimize the dual terms with $\rho_c = 1.0, \rho_m = 10^{-2}$ and $\rho_\lambda = 10^{-4}$.

The whole-body cost function

For the whole-body solver, we use only quadratic costs to regularize the posture (weight: 10^{-6}) and the free-flyer orientation (weight: 20) in addition to the augmented Lagrangian terms.

5.5.3 Convergence analysis

We stopped the alternating resolution after only 10 iterations of the solver described in Algorithm 1. While 3 iterations were empirically sufficient to compute a feasible motion in simulation, we show here the continuing convergence. As shown in Fig. 5.2, the total residuals of the matching constraints decrease rapidly over the first three iterations and more slowly after. This really means that the two problem are able to find a consensus in very few iterations. This behavior is also well depicted in Fig. 5.3, where one can observe the CoM trajectories of both centroidal and whole-body subproblems. Already at iteration 3, the two trajectories match almost perfectly.

The mismatching of CoM and AM quantities is also reflected in Fig. 5.4. The two first iterations show a large mismatch between the centroidal and the whole-body problems, but the residual decreases rapidly towards the value 0. This phenomena can be explained by the delay induced by the ADMM approach: the solver overshoots and then builds and *maintains* a consensus between the two subproblems as discussed in Sec 5.4.3.

5.6 Conclusion and Future Work

In this chapter, we introduced a systematic approach to build a consensus on the dynamics constraints between the centroidal and whole-body optimization problems. Based on previous observation between the nice articulation between underactuated and actuated dynamics of legged robots, we have given a mathematical framework to separate the two problems, and proposed a solution which iterates between the two subproblems and maintains consensus in the solutions. Finally, we demonstrate the performance of the solver with a walking sequence on HRP-2, and with jumping simulations on the Talos robot.

While some heuristics are available allowing similar behaviors, the ADMM solver encompasses the dynamics constraints within the framework of the solver itself. In this way, the current method deals with not only individual and separated subproblems, but tackles the global problem defined in (1.16). To the best of our knowledge, this is the first time that a consensus between centroidal and Lagrangian dynamic solvers is obtained based on theoretical grounding.

The dual variable in ADMM contains information about the difference in the dynamics models of the centroidal and whole-body subproblems, and the optimization scheme. For a given robot, the whole-body dynamics remains the same. Thus,

it is possible that if the centroidal solution is known, and for a given robot, we can learn the dual variable that corresponds to our optimization problem. This approach holds potential: The problem of feasibility makes us alternate with the dual. If the dual optimal is learned , we know the solution of the centroidal problem, and we know how far this solution is from the feasible reference that should be used in the whole-body problem. We then just have to solve our whole-body problem once with the updated centroidal references, and we can be sure that these would produce a consensus in the first iteration.

Even if our solver currently solves the full-body dynamics problem, it assumes a dependence on the upstream contact planner to provide feasible contact planning. In order to be fully optimal, we need to ensure a similar consistency between the upstream contact planner and the full body optimizer as well. However, this problem is not straightforward. Contact planning is a combinatorial problem. A change of contact sequence changes the manifold on which our full-body optimization problem is solved, and it this change is not easy to encode. This decoupling has not yet been explicitly explained in literature, and this would be another amazing field of research for the future.

Generation of dynamic motions

6

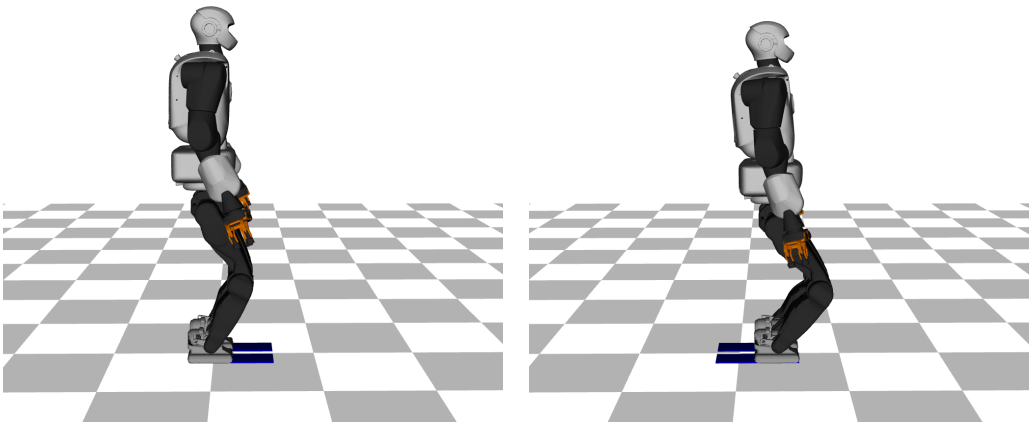
> *- Et maintenant, messieurs, dit d'Artagnan sans se donner la peine d'expliquer sa conduite à Porthos, tous pour un, un pour tous ; c'est notre devise, n'est-ce pas ?*
> *("And now, gentlemen" said D'Artagnan, without bothering to explain his conduct to Porthos, all for one, one for all; it's our motto, isn't it?)*
>
> — **Alexandre Dumas**
> (Les Trois Mousquetaires)

Each chapter in this thesis deals with a particular aspect of the same problem. We've broken our problem into sub-problems, and then re-connected those subproblems in the previous Chapter 5. The examples presented in each chapter deal with the particular sub-problem, and show its features.

In this chapter, we'll show with an example, what each of our contributions brings to the table. We'll be looking at a difficult dynamic problem, and evaluate how each of our works helps in making the resolution easier. We'll see how all the chapters connect together to provide solutions which take into account the full dynamics of the problem.

6.1 Definition of the Motion Problem

To demonstrate the various components of this thesis, we take the example of our robot jumping forward. This is a highly dynamic manoeuvre, which first requires the robot to create enough momentum in order to jump, then asks him to land in a controlled manner so that the impact of landing remain within limits. We will assume the contact placements to be pre-decided. Thus the robot is aware of the landing position. We will concern ourselves with finding the control trajectory that makes the jump possible. This problem is shown in Fig. 6.1.



(a) Before jump (b) After jump

Fig. 6.1: *Problem Definition*: The robot is asked to make a jump forward. The current feet contacts, and the feet contacts on landing, are provided. The robot has to find the state and control trajectory of motion that satisfy its kinematic and dynamic constraints.

6.2 Feasibility constraints for warm-starting centroidal solver

6.2.1 Centroidal Optimization without feasibility constraint

We consider the associated centroidal problem for jumping. In order to see the effect of the kinematic feasibility constraint from Chapter 2, we first consider the optimization problem without these feasibility constraint. Then we'll see what effect the feasibility constraints have on the solution. Thus, we look at the problem (2.1), and implement it as follows:

Discretization: Let K be the set of active contacts. Since the contact positions $(\boldsymbol{p}_k, k \in K)$ are defined before, they become a part of the dynamics. First, we discretize the dynamics the same way we discretized the whole-body dynamics in Sec 3.2. We consider the control trajectory, which is $\boldsymbol{\lambda}$, as the concatenation of contact forces $\boldsymbol{\lambda}_k$ acting on each contact surface. We divide it into a grid of T points, and take the trajectory between any two consecutive grid points t and $t+1$ to be equal to $\boldsymbol{\lambda}(t)$. We obtain the state at a given node point by integrating the control trajectory from the previous node point, using the underactuated dynamics of (1.13) and an integration scheme. Thus, let's define this discretized centroidal dynamics by F_K, where the dynamics depends on the set of active contacts, the contact forces, and the current state. Since we are not considering the coupling constraints for this first example, we drop the coupling constraints (2.1c), and replace the viability constraint of the final position with a cost that forces the velocity

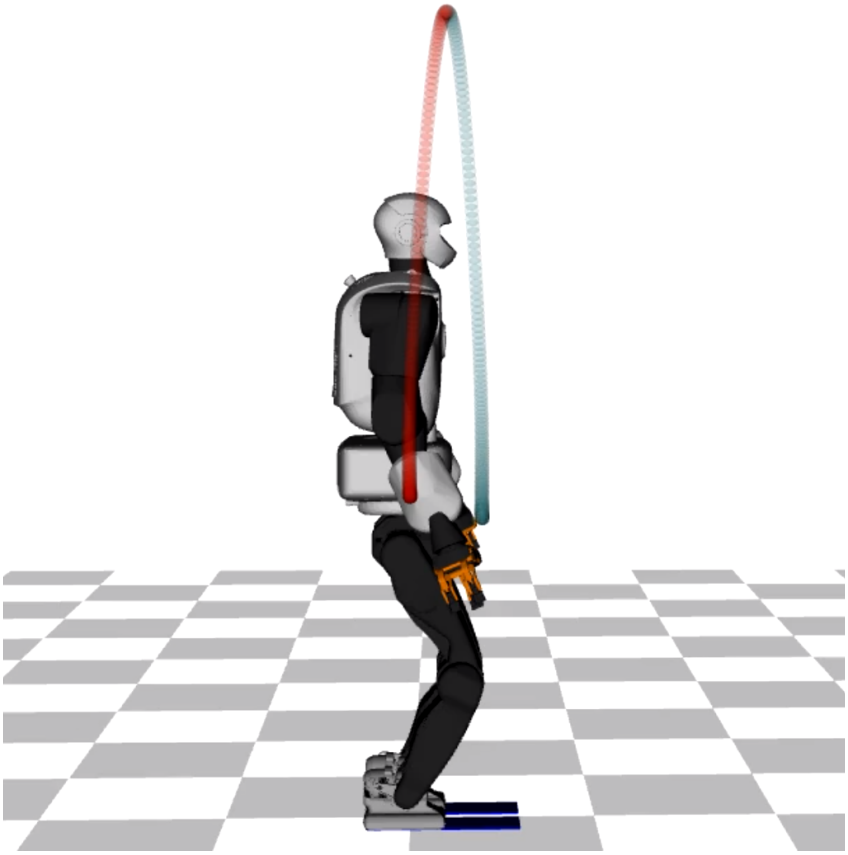


Fig. 6.2: Solution of the Centroidal Optimization Problem (6.1). The ball shows the trajectory followed by the center of mass, when we obtain the solution of our optimization problem (6.1). Since we did not consider the feasibility constraints, the solution assumes really long robot legs. This trajectory can be improved by taking kinematic information in our optimization.

and angular momentum to be zero. This first optimization problem is defined in (6.1).

$$
\begin{aligned}
\underset{\mathbf{c},\dot{\mathbf{c}},\boldsymbol{L},\boldsymbol{\lambda}}{\text{minimize}} \quad & \sum_{t=0}^{T} \mathtt{w}_{\lambda} \left\|\boldsymbol{\lambda}\right\|^2 + \mathtt{w}_{\dot{c}} \left\|\dot{\mathbf{c}}\right\|^2 \\
\text{subject to} \quad & \\
\forall t \in \{0 \dots T-1\} \quad & \boldsymbol{\lambda}(t) \in \mathcal{K} && \text{(6.1a)} \\
\forall t \in \{0 \dots T-1\} \quad & \begin{bmatrix} \mathbf{c}(t+1) \\ \dot{\mathbf{c}}(t+1) \\ \boldsymbol{L}(t+1) \end{bmatrix} = F_k \left(\begin{bmatrix} \mathbf{c}(t) \\ \dot{\mathbf{c}}(t) \\ \boldsymbol{L}(t) \end{bmatrix}, \boldsymbol{\lambda}(t) | p_k \right) && \text{(6.1b)} \\
& \mathbf{c}(0), \dot{\mathbf{c}}(0), \boldsymbol{L}(0) \text{ is given} && \text{(6.1c)}
\end{aligned}
$$

As shown, the cost is simply a minimization of the weighted square norm of the center of mass velocity and the contact forces (weighted by $\mathtt{w}_{\lambda}, \mathtt{w}_{\dot{c}}$). Both these objectives enforce a smooth trajectory. The positions of the contacts, and thus the contact transitions are given, and we don't need to specify the final position. The solution of the problem (6.1) finds the final center of mass position where constraint (6.1a) is satisfied, and the center of mass velocity is zero.

By solving (6.1), we get the solution of our optimization problem. This is shown in Fig. 6.2. On first glance, Fig. 6.2 seems to be a good centroidal trajectory. However, we can see that Fig. 6.2 assumes very long legs for the robot, which is not the case. This is expected, since we do not provide any information about the kinematic limits of the robot to our optimization problem. The solution in Fig. 6.2 corresponds to a trajectory that does not respect these kinematic limits.

6.2.2 Learning and optimizing with feasibility constraints

In order to improve this first solution, we look at the formulation defined in Chapter 2. We sample joint configurations, and learn the CoM positions that correspond to non self-colliding joint configurations. An example of the generated CoM positions with respect to the Left Foot is shown in Fig. 6.3. Once we've encoded these samples in a Gaussian Mixture Model, we use this learned model inside our optimization problem (6.1). This was discussed in Sec 2.3, and we replicate (2.4) here for completeness. Thus, we update the objective function of (6.1) by the following: $\mathtt{w}_{\lambda} \left\|\boldsymbol{\lambda}\right\|^2 + \mathtt{w}_{\dot{c}} \left\|\dot{\mathbf{c}}\right\|^2 - \sum_{p=1}^{K} \log p_i$. With the new feasibility constraint in our optimization problem, we see again the centroidal solution in Fig. 6.4

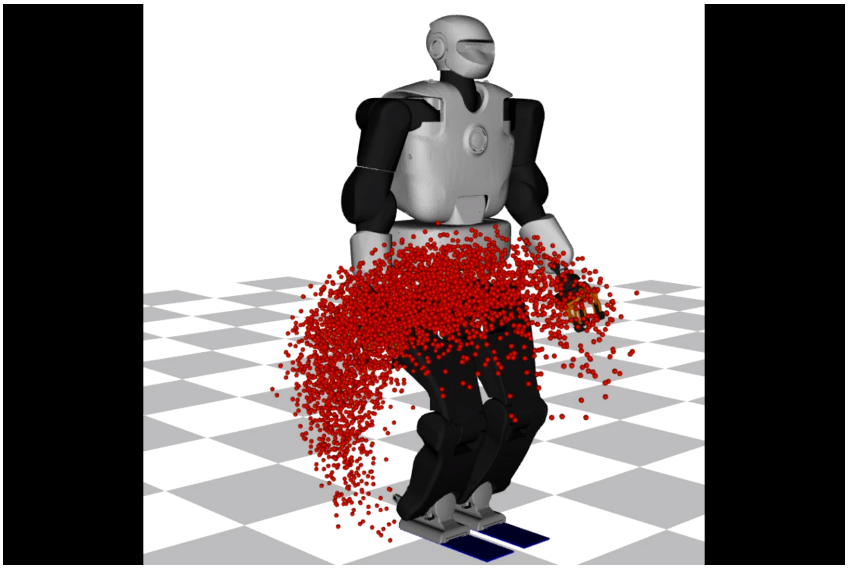


Fig. 6.3: Center of Mass points with respect to left foot placement. Joint configurations are sampled, and for each non self-colliding joint configuration, the center of mass position is computed and learned with respect to the end-effectors.

6.3 Wholebody optimization using centroidal trajectories

After obtaining the centroidal trajectories in Sec 6.2, we use our DDP solver *Crocoddyl* to form a wholebody optimization problem. We use the problem formulation that we defined in (3.3). We use the following costs in our problem: tracking, regularization, and penalty. The tracking costs track the trajectories given by the centroidal solver, and the regularization costs are used to remove redundancy from our problem.

Tracking Costs:

- Center of Mass tracking $\mathtt{w}_c \left\| c_{ref} - CoM(\mathbf{q}) \right\|^2$
- Momentum tracking $\mathtt{w}_{\boldsymbol{L}} \left\| \boldsymbol{L}_{ref} - Ag(\mathbf{q}, \dot{\mathbf{q}}) \right\|^2$
- Contact Force tracking $\mathtt{w}_\lambda \left\| \lambda_{ref} - g(\mathbf{q}, \dot{\mathbf{q}}, \tau) \right\|^2$

Regularization Costs:

- State regularization $\mathtt{w}_x \left\| x \right\|^2$
- Control regularization $\mathtt{w}_{\boldsymbol{\tau}} \left\| \boldsymbol{\tau} \right\|^2$

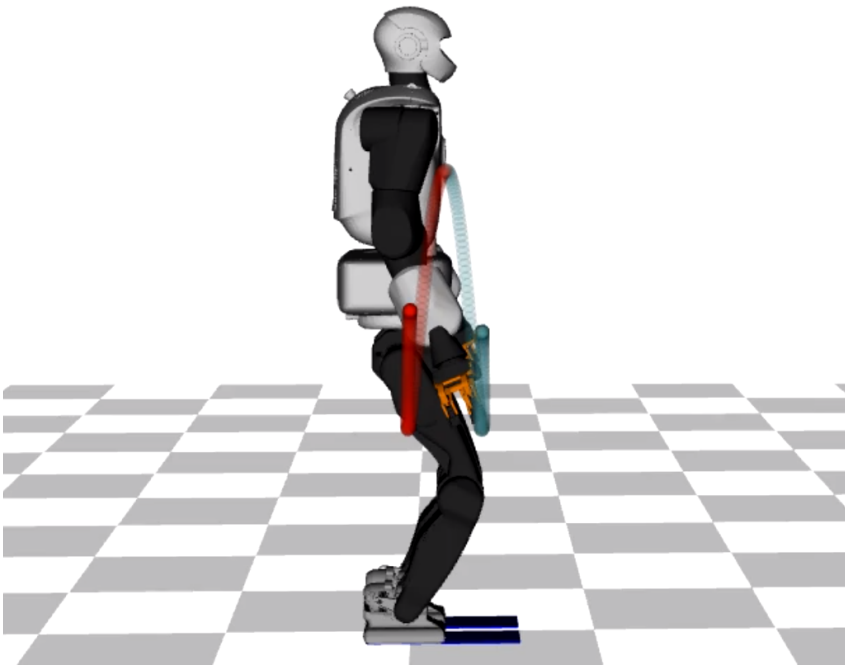


Fig. 6.4: Solution of the Centroidal Optimization Problem (6.1) with feasibility constraints, as discussed in Sec 2.3. The ball shows the trajectory followed by the center of mass. We see that the new solution takes into account the kinematic limits of the robot. Instead of simply ascending, it first descends in order to create space of motion. After landing, we see similar behaviour, where it descends in order to distribute the impact to a longer duration.

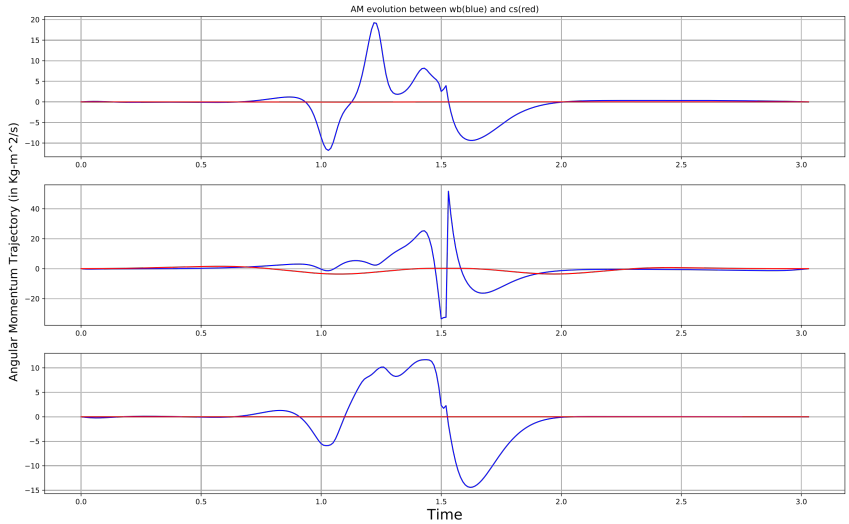


Fig. 6.5: Comparison of the Angular Momentum trajectory, after the centroidal (red) and whole-body(blue) optimization. The angular momentum output from the centroidal optimization is close to zero. However, the angular momentum required by the whole-body optimization is high, because of the dynamic nature of the problem. As a result, tracking the centroidal references does not result in a good whole-body solution.

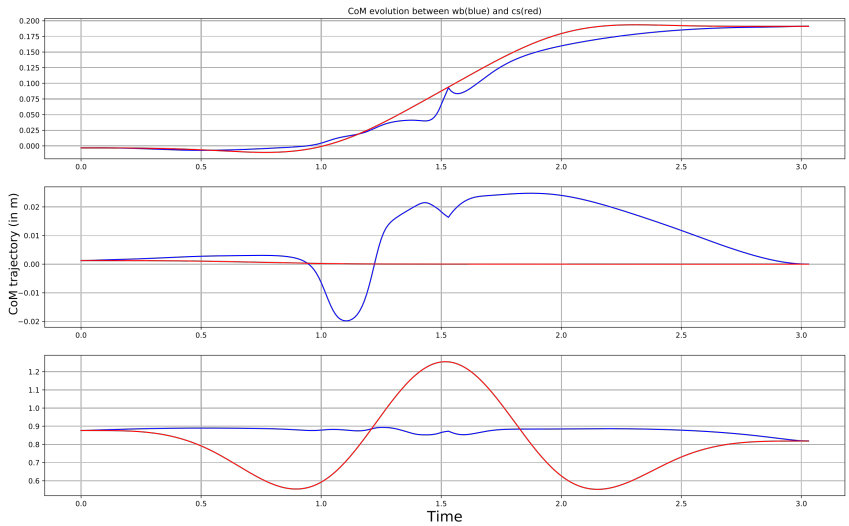


Fig. 6.6: Comparison of the Center of Mass trajectory, after the centroidal (red) and whole-body(blue) optimizations. We took the CoM kinematic feasibility into account in our centroidal problem. However, as a result of the tracking error in angular momentum, the whole-body solver cannot properly track the center of mass trajectories.

In addition to the above, we use high penalty costs that replace our kinematic and dynamic constraints by high weight penalties. The following penalties are used to replace inequality conditions that are needed to ensure a smooth take-off and landing trajectories. Implementation-wise, they are implemented using the `ActivationModelInequalityModel` that we defined in Table 4.1.

- Joint torque limits
- Joint configuration limits
- Positive foot velocity at take-off
- Negative foot velocity when landing
- Zero foot velocity for first node of the phase

With the above objective function formulation, we try to solve optimization problem (3.3). However, we are unable to produce a solution. On inspection, we see that the centroidal tracking references for the angular momentum cannot be directly used in our whole-body optimization, as seen in Fig. 6.5. We require highly dynamic motions, while the reference being produced by the centroidal reference is closer to zero. As a result of this discrepancy, even our CoM tracking suffers, as seen in Fig. 6.6.

6.4 Motion generation using ADMM

In order to properly take into account the coupling constraints (1.15) that link centroidal and whole-body problems together, we solve our two problems iteratively using the ADMM scheme, as discussed in Sec 5.4. As a result, the dual variable transmits the discrepancy to each of our problems, and both of them arrive at a solution that takes care of the dynamics of our robot. Fig. 6.7 shows the resulting jumping motion that takes into account the kinematic dynamic constraints of the system.

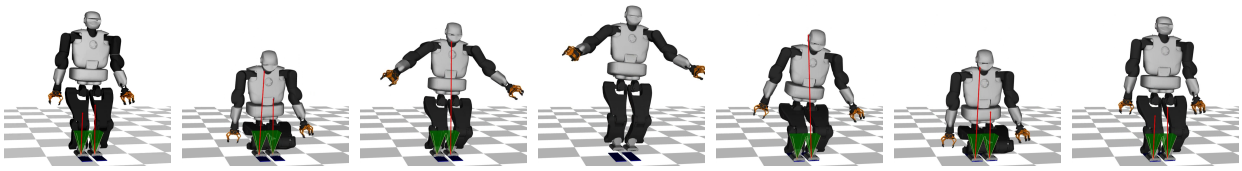


Fig. 6.7: Snapshots that show the final jumping motion as a result of ADMM iterations on our centroidal and whole-body optimization formulation. Our whole-body solver (DDP) does not take into account the constraints on the contact forces. On the other hand, our centroidal solver does not take into account the information about the dynamics of our multi-body robot. However, both these solvers work together and are able to arrive at the common solution which is both highly dynamic, and dynamically feasible.

7 Conclusion

> *The only people who see the whole picture, are the ones who step out of the frame.*
>
> — **Darius Xerxes Cama, to Methwold**
> (Salman Rushdie, The Ground Beneath Her Feet)

In this thesis we have explored the contact-constrained locomotion problem, and proposed solvers to improve the efficiency and the quality of the solutions. This thesis centers around the equation (1.16), and contributes four main pieces of published works and one piece of software.

7.1 Contributions

7.1.1 Proxy constraint for kinematic feasibility

We deal with the CoM feasibility constraint inside the centroidal problem by learning the probability density, and maximizing the occupancy measure of the CoM probability as part of the centroidal problem cost function. As a result, the feasibility of the CoM with respect to the kinematic configuration can be handled easily inside an optimal control problem, and when we deal with the remaining centroidal constraints later in Chapter 5, this provides a much better centroidal guess to the whole-body solver.

7.1.2 Whole-body solver for contact-constrained dynamics

Since centroidal solvers cannot correctly anticipate and generate the AM required by the motion of the limbs, the whole-body solver needs to be able to fulfill this requirement. We proposed using DDP as the whole-body solver, and modify the dynamics with the KKT constraints in order to handle the contact equality constraint. We are able to use our solver to perform aggressive motions on the robot HRP-2,

and to the best of our knowledge, this was the first time that the motions generated by DDP has been transferred to a bipedal system.

We developed a numerical optimization solver named Crocoddyl, which is dedicated to the Markovian dynamics and the contact-constrained structure of our locomotion problem. We have shown that Crocoddyl, by virtue of its technical features, can perform real-time computations. These technical features include analytical derivatives of the dynamics (Sohl and Bobrow, 2001; Carpentier and Mansard, 2018a), multi-threaded parallel computation of these derivatives, efficient transcription that reduces memory allocation and so on.

7.1.3 Alternating to ensure feasibility and consensus

Finally, we (re)introduce to the robotics community ADMM (Alternating Direction Method of Multipliers), which is a form of penalty method with dual where each update of the dual variable tries to create a consensus between the centroidal and the whole-body subproblems. As a result, we are able to also take the centroidal dynamics into account when solving for the whole-body problem. We use this solver to create really dynamic motions like jumping and hopping.

7.2 Perspective on Multi-body Locomotion

While the chapters in this thesis connect towards generating trajectories for the locomotion problem, there are some issues and some features resulting from our approach, which need to be addressed. These offer potential areas where this approach can be even further improved.

7.2.1 What about contacts?

Contacts have been assumed to be pre-defined throughout this thesis. While it is possible to vary the contact timings in our approach, the contact placements and contact sequences have not been tampered with. This is one drawback of the decoupling approach: the contact placements cannot be used inside the whole-body optimization scheme. It would be worthwhile to investigate whether this can be improved through a similar alternation scheme. However, putting contact planning in loop with whole-body trajectory generation is not straightforward.

The whole-body motion is a continuous time problem, while contact placement is a mixed-integer problem. As a result, this discrepancy is difficult to model in a

way that we modeled the coupling constraints between centroidal and whole-body dynamics.

Thus, in order for the contact plan to be consistent with the whole-body trajectory optimization, without any feedback from the whole-body solver, the contact planner needs to consider two types of constraints:
1) The *feasibility constraints*, where the contact plan has to be feasible by the whole-body problem, and
2) the *global optimality constraint*, where the contact plan has to result in the globally optimal whole-body trajectory.

Note that the *global optimality constraint* is not the same as the optimality constraint (2.2d) that we defined for the reduced problem. The above two constraints can be seen this way: The first constraint says that there is at least one solution that lies on the manifold defined by the contact plan. The second constraint says that the optimal solution lies on the sequence of manifolds defined by the contact plan.

The approaches that currently compute contact plans can only model the feasibility of whole-body constraints in their constraints. And even this feasibility problem is difficult, and often intractable. Modeling the optimality of the whole-body trajectories as constraints for the contact planner is thus far off in the future. Some of the recent works, such as Deits and Tedrake, 2014, Mastalli et al., 2017 or Tonneau et al., 2019 are prime examples that demonstrate this. All of these approaches use relaxations in order to model the feasibility constraints in a continuous manner, and this makes the problem tractable enough for the solver. But none of these works can handle the global optimality constraint.

A possible way forward could be by using machine learning tools. While modeling these constraints analytically is intractable for the solvers, machine learning tools can provide an approximation that is simpler for the solver to deal with. These tools, including neural networks, could thus be used to encode whole-body optimality criterion that are useful when searching for contact plans. We saw something similar in Chapter 2, when we encoded the kinematic feasibility constraints. However, such use of proxies for encoding feasible constraints are one small step, and encoding optimality properties for the high-dimensional humanoid robots is still far off.

7.2.2 How should we use the duals?

The dual of a constrained optimization problem is an effective way to account for the behaviour of its constraints. In this thesis, we have seen how the duals can be

used to maintain consistency within the two subproblems. As a result of this consistency, we can let the centroidal solver take care of the stability criteria, and this simplifies the constraints for the whole-body solver. Thus while the centroidal solver takes care of the reference generation and the constraint satisfaction, the whole-body solver finds dynamic motions of the robot. Since the dual update keeps the solutions consistent with each other, there are possibilities to use the dual update as an outer loop for the whole-body references. This update of references would increase the search space for the whole-body solver, and promises an increase in the quality of solutions. Another avenue which hasn't been properly investigated is using machine learning to learn the dual parameters as a function of the problem and the solutions of the subproblems. Good initialization of the dual parameters would vastly improve the performance of the ADMM algorithm. Moreover, since the dual is in the centroidal space, the dimensions would always remain low. Thus, the learning would be low dimension, and would encode the nested optimality constraint that we saw in Eq 2.2d. In such a case, we would not need alternation in order to maintain consensus.

7.2.3 Could we go faster?

If the sparsity inside the problem structure is exploited properly, there are possibilities of increasing the efficiency of the solution without any limiting assumptions. With DDP, we have shown how existing methods can be used to inherently deal with the sparsity within a Markovian structure. There are still further ways in which the structure can be exploited, as the value functions, cost functions and their derivatives follow a specific pattern. Once these are properly accounted for, we should see further improvement in the efficiency of the whole-body solver. In our team at Gepetto, this has been the principle followed in the development of the Pinocchio dynamic library, and now it is one of the main principles that we followed during the development of the Crocoddyl solver.

Bibliography

Aceituno-Cabezas, Bernardo, Carlos Mastalli, Hongkai Dai, Michele Focchi, Andreea Radulescu, Darwin G. Caldwell, Jose Cappelletto, Juan Carlos Grieco, Gerardo Fernandez-Lopez, and Claudio Semini (2017). "Simultaneous Contact, Gait and Motion Planning for Robust Multi-Legged Locomotion via Mixed-Integer Convex Optimization". In: *IEEE Robotics and Automation Letters*, pp. 1–1. URL: http://ieeexplore.ieee.org/document/8141917/ (cit. on p. 11).

Anitescu, M. and F. A. Potra (1997). "Formulating Dynamic Multi-Rigid-Body Contact Problems with Friction as Solvable Linear Complementarity Problems". In: *Nonlinear Dynamics* 14.3, pp. 231–247. URL: http://link.springer.com/10.1023/A:1008292328909 (cit. on p. 4).

Anitescu, Mihai (2005). "On Using the Elastic Mode in Nonlinear Programming Approaches to Mathematical Programs with Complementarity Constraints". In: *SIAM Journal on Optimization* 15.4, pp. 1203–1236. URL: http://epubs.siam.org/doi/10.1137/S1052623402401221 (cit. on p. 4).

Asfour, T. and R. Dillmann (2003). "Human-like motion of a humanoid robot arm based on a closed-form solution of the inverse kinematics problem". In: *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003) (Cat. No.03CH37453)*. Vol. 2. IEEE, pp. 1407–1412. URL: http://ieeexplore.ieee.org/document/1248841/ (cit. on p. 11).

Atkeson, C. G., B. P. W. Babu, N. Banerjee, et al. (2015). "No falls, no resets: Reliable humanoid behavior in the DARPA robotics challenge". In: *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*. IEEE, pp. 623–630. URL: http://ieeexplore.ieee.org/document/7363436/ (cit. on p. 7).

Baudouin, Leo, Nicolas Perrin, Thomas Moulard, Florent Lamiraux, Olivier Stasse, and Eiichi Yoshida (2011). "Real-time replanning using 3D environment for humanoid robot". In: *2011 11th IEEE-RAS International Conference on Humanoid Robots*. IEEE, pp. 584–589. URL: http://ieeexplore.ieee.org/document/6100844/ (cit. on p. 7).

Benning, Martin, Florian Knoll, Carola-Bibiane Schönlieb, and Tuomo Valkonen (2016). "Preconditioned ADMM with Nonlinear Operator Constraint". In: *IFIP Conference on System Modeling and Optimization* 494, pp. 117–126. URL: http://link.springer.com/10.1007/978-3-319-55795-3{_}10 (cit. on p. 81).

Betts, John T. (2010). *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming*. Society for Industrial and Applied Mathematics. URL: http://epubs.siam.org/doi/book/10.1137/1.9780898718577 (cit. on pp. 4, 13).

Bishop, Christopher M (2006). *Pattern recognition and machine learning*. Springer, p. 738. URL: http://olin.tind.io/record/132308 (cit. on p. 34).

Boyd, Stephen, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein (2010). "Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers". In: *Foundations and Trends® in Machine Learning* 3.1, pp. 1–122. URL: http://www.nowpublishers.com/article/Details/MAL-016 (cit. on pp. 79–81, 83).

Boyd, Stephen P. and Lieven. Vandenberghe (2004). *Convex optimization*. Cambridge University Press, p. 716. URL: https://dl.acm.org/citation.cfm?id=993483 (cit. on p. 69).

Brasseur, Camille, Alexander Sherikov, Cyrille Collette, Dimitar Dimitrov, and Pierre-Brice Wieber (2015). "A robust linear MPC approach to online generation of 3D biped walking motion". In: *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*. IEEE, pp. 595–601. URL: http://ieeexplore.ieee.org/document/7363423/ (cit. on pp. 8, 9, 18).

Bretl, Timothy (2006). "Motion Planning of Multi-Limbed Robots Subject to Equilibrium Constraints: The Free-Climbing Robot Problem". In: *The International Journal of Robotics Research* 25.4, pp. 317–342. URL: http://journals.sagepub.com/doi/10.1177/0278364906063979 (cit. on p. 29).

Brockett, R W (1983). "Asymptotic stability and feedback stabilization". In: *Differential Geometric Control Theory* 27, pp. 181–191. URL: https://ci.nii.ac.jp/naid/10004576193/ (cit. on pp. 12, 42).

Brogliato, B. (2003). "Some perspectives on the analysis and control of complementarity systems". In: *IEEE Transactions on Automatic Control* 48.6, pp. 918–935. URL: http://ieeexplore.ieee.org/document/1205187/ (cit. on p. 4).

Budhiraja, Rohan, Justin Carpentier, Carlos Mastalli, and Nicolas Mansard (2018). "Differential Dynamic Programming for Multi-Phase Rigid Contact Dynamics". In: *2018 IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids)*. IEEE, pp. 1–9. URL: https://ieeexplore.ieee.org/document/8624925/ (cit. on pp. 16, 61, 64, 84).

Caron, Stephane, Quang-Cuong Pham, and Yoshihiko Nakamura (2017). "ZMP Support Areas for Multicontact Mobility Under Frictional Constraints". In: *IEEE Transactions on Robotics* 33.1, pp. 67–80. URL: http://ieeexplore.ieee.org/document/7782743/ (cit. on p. 9).

Carpentier, Justin and Nicolas Mansard (2018a). "Analytical Derivatives of Rigid Body Dynamics Algorithms". In: *Robotics: Science and Systems XIV*. Robotics: Science and Systems Foundation. URL: http://www.roboticsproceedings.org/rss14/p38.pdf (cit. on pp. 52, 59, 62, 66, 68, 102).

– (2018b). "Multicontact Locomotion of Legged Robots". In: *IEEE Transactions on Robotics* 34.6, pp. 1441–1460. URL: https://ieeexplore.ieee.org/document/8558661/https://hal.archives-ouvertes.fr/hal-01520248/ (cit. on pp. 17, 56, 77, 84).

Carpentier, Justin, Steve Tonneau, Maximilien Naveau, Olivier Stasse, and Nicolas Mansard (2016). "A versatile and efficient pattern generator for generalized legged locomotion". In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 3555–3561. URL: http://ieeexplore.ieee.org/document/7487538/ (cit. on pp. 9, 11, 17, 21, 27, 52, 56, 58).

Carpentier, Justin, Rohan Budhiraja, and Nicolas Mansard (2017a). "Learning Feasibility Constraints for Multicontact Locomotion of Legged Robots". In: *Robotics: Science and Systems XIII*. Robotics: Science and Systems Foundation. URL: http://www.roboticsproceedings.org/rss13/p31.pdf (cit. on pp. 17, 19, 44).

Carpentier, Justin, Andrea Del Prete, Steve Tonneau, Thomas Flayols, Florent Forget, Alexis Mifsud, Kevin Giraud, Dinesh Atchuthan, Pierre Fernbach, Rohan Budhiraja, Mathieu Geisert, Joan Solà, Olivier Stasse, and Nicolas Mansard (2017b). "Multi-contact Locomotion of Legged Robots in Complex Environments The Loco3D project". In: *RSS Workshop on Challenges in Dynamic Legged Locomotion*. Boston, United States, 3p. URL: https://hal.laas.fr/hal-01543060 (cit. on p. 20).

Carpentier, Justin, Guilhem Saurel, Gabriele Buondonno, Joseph Mirabel, Florent Lamiraux, Olivier Stasse, and Nicolas Mansard (2019). "The Pinocchio C++ library – A fast and flexible implementation of rigid body dynamics algorithms and their analytical derivatives". In: *SII 2019 - International Symposium on System Integrations*. Paris, France. URL: https://hal.laas.fr/hal-01866228 (cit. on pp. 52, 62, 66, 68, 75).

Dai, Hongkai and Russ Tedrake (2016). "Planning robust walking motion on uneven terrain via convex optimization". In: *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*. IEEE, pp. 579–586. URL: http://ieeexplore.ieee.org/document/7803333/ (cit. on pp. 9, 17, 30).

Dai, Hongkai, Andres Valenzuela, and Russ Tedrake (2014). "Whole-body motion planning with centroidal dynamics and full kinematics". In: *2014 IEEE-RAS International Conference on Humanoid Robots*. IEEE, pp. 295–302. URL: http://ieeexplore.ieee.org/document/7041375/ (cit. on pp. 9, 11, 12, 17, 18, 30, 79).

Deits, Robin and Russ Tedrake (2014). "Footstep planning on uneven terrain with mixed-integer convex optimization". In: *2014 IEEE-RAS International Conference on Humanoid Robots*. IEEE, pp. 279–286. URL: http://ieeexplore.ieee.org/document/7041373/ (cit. on pp. 7, 17, 27, 30, 103).

Deits, Robin, Twan Koolen, and Russ Tedrake (2019). "LVIS: Learning from Value Function Intervals for Contact-Aware Robot Controllers". In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 7762–7768. URL: https://ieeexplore.ieee.org/document/8794352/ (cit. on p. 4).

Del Prete, Andrea, Nicolas Mansard, Oscar E. Ramos, Olivier Stasse, and Francesco Nori (2016). "Implementing Torque Control with High-Ratio Gear Boxes and Without Joint-Torque Sensors". In: *International Journal of Humanoid Robotics* 13.01, p. 1550044. URL: https://www.worldscientific.com/doi/abs/10.1142/S0219843615500449 (cit. on p. 21).

Dempster, A. P., N. M. Laird, and D. B. Rubin (1977). *Maximum Likelihood from Incomplete Data via the EM Algorithm*. URL: https://www.jstor.org/stable/2984875 (cit. on p. 34).

Djoudi, D., C. Chevallereau, and Y. Aoustin (2005). "Optimal Reference Motions for Walking of a Biped Robot". In: *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*. IEEE, pp. 2002–2007. URL: http://ieeexplore.ieee.org/document/1570407/ (cit. on p. 13).

Dominici, Nadia, Yuri P Ivanenko, Germana Cappellini, Andrea D'Avella, Vito Mondì, Marika Cicchese, Adele Fabiano, Tiziana Silei, Ambrogio Di Paolo, Carlo Giannini, Richard E Poppele, and Francesco Lacquaniti (2011). "Locomotor primitives in newborn babies and their development." In: *Science (New York, N.Y.)* 334.6058, pp. 997–9. URL: http://www.ncbi.nlm.nih.gov/pubmed/22096202 (cit. on p. 6).

D'Souza, A., S. Vijayakumar, and S. Schaal (2001). "Learning inverse kinematics". In: *Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems. Expanding the Societal Role of Robotics in the the Next Millennium (Cat. No.01CH37180)*. Vol. 1. IEEE, pp. 298–303. URL: http://ieeexplore.ieee.org/document/973374/ (cit. on p. 10).

Dunn, J. C. and D. P. Bertsekas (1989). "Efficient dynamic programming implementations of Newton's method for unconstrained optimal control problems". In: *Journal of Optimization Theory and Applications* 63.1, pp. 23–38. URL: http://link.springer.com/10.1007/BF00940728 (cit. on p. 50).

Eckstein, Jonathan and Dimitri P. Bertsekas (1992). "On the DouglasRachford splitting method and the proximal point algorithm for maximal monotone operators". In: *Mathematical Programming* 55.1-3, pp. 293–318. URL: http://link.springer.com/10.1007/BF01581204 (cit. on p. 81).

Englsberger, Johannes, Christian Ott, and Alin Albu-Schaffer (2015). "Three-Dimensional Bipedal Walking Control Based on Divergent Component of Motion". In: *IEEE Transactions on Robotics* 31.2, pp. 355–368. URL: http://ieeexplore.ieee.org/document/7063218/ (cit. on p. 18).

Escande, Adrien, Abderrahmane Kheddar, and Sylvain Miossec (2006). "Planning support contact-points for humanoid robots and experiments on HRP-2". In: *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, pp. 2974–2979. URL: http://ieeexplore.ieee.org/document/4058848/ (cit. on p. 29).

Featherstone, Roy (2008). *Rigid Body Dynamics Algorithms*. Boston, MA: Springer US. URL: http://link.springer.com/10.1007/978-0-387-74315-8 (cit. on pp. 52, 62).

Feng, Siyuan, Eric Whitman, X. Xinjilefu, and Christopher G. Atkeson (2015). "Optimization-based Full Body Control for the DARPA Robotics Challenge". In: *Journal of Field Robotics* 32.2, pp. 293–312. URL: http://doi.wiley.com/10.1002/rob.21559 (cit. on p. 7).

Fernbach, Pierre, Steve Tonneau, and Michel Taïx (2018). "CROC: Convex Resolution Of Centroidal dynamics trajectories to provide a feasibility criterion for the multi contact planning problem". In: *International Conference on Intelligent Robots and Systems*. URL: https://hal.archives-ouvertes.fr/hal-01726155/ (cit. on pp. 7, 11, 17).

Fletcher, Roger, Sven Leyffer, Danny Ralph, and Stefan Scholtes (2006). "Local Convergence of SQP Methods for Mathematical Programs with Equilibrium Constraints". In: *SIAM Journal on Optimization* 17.1, pp. 259–286. URL: http://epubs.siam.org/doi/10.1137/S1052623402407382 (cit. on p. 4).

Frizot, Michel (2001). *Etienne-Jules Marey : chronophotographe*. Nathan. URL: http://www.openbibart.fr/item/display/10068/766347 (cit. on p. 42).

Full, R J and D E Koditschek (1999). "Templates and anchors: neuromechanical hypotheses of legged locomotion on land." In: *The Journal of experimental biology* 202.Pt 23, pp. 3325–32. URL: http://www.ncbi.nlm.nih.gov/pubmed/10562515 (cit. on p. 8).

Gill, Philip E., Walter Murray, and Michael A. Saunders (2005). "SNOPT: An SQP Algorithm for Large-Scale Constrained Optimization". In: *SIAM Review* 47.1, pp. 99–131. URL: http://epubs.siam.org/doi/10.1137/S0036144504446096 (cit. on p. 4).

Grandia, Ruben, Farbod Farshidian, Alexey Dosovitskiy, Rene Ranftl, and Marco Hutter (2019). "Frequency-Aware Model Predictive Control". In: *IEEE Robotics and Automation Letters* 4.2, pp. 1517–1524. URL: https://ieeexplore.ieee.org/document/8629284/ (cit. on p. 21).

Hardt and Michael William (1999). "Multibody dynamical algorithms, numerical optimal control, with detailed studies in the control of jet engine compressors and biped walking". PhD thesis. University of California, San Diego. URL: https://dl.acm.org/citation.cfm?id=929270 (cit. on p. 13).

Hart, Peter, Nils Nilsson, and Bertram Raphael (1968). "A Formal Basis for the Heuristic Determination of Minimum Cost Paths". In: *IEEE Transactions on Systems Science and Cybernetics* 4.2, pp. 100–107. URL: http://ieeexplore.ieee.org/document/4082128/ (cit. on p. 7).

Heess, Nicolas, Dhruva TB, Srinivasan Sriram, Jay Lemmon, Josh Merel, Greg Wayne, Yuval Tassa, Tom Erez, Ziyu Wang, S. M. Ali Eslami, Martin Riedmiller, and David Silver (2017). "Emergence of Locomotion Behaviours in Rich Environments". In: arXiv: 1707.02286. URL: http://arxiv.org/abs/1707.02286 (cit. on pp. v, vii, 5, 7).

Herdt, A, N Perrin, and Pierre-Brice Wieber (2010a). "Walking without thinking about it". In: *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, pp. 190–195. URL: http://ieeexplore.ieee.org/document/5654429/ (cit. on pp. 9, 12, 17, 30).

Herdt, Andrei, Holger Diedam, Pierre-Brice Wieber, Dimitar Dimitrov, Katja Mombaur, and Moritz Diehl (2010b). "Online Walking Motion Generation with Automatic Footstep Placement". In: *Advanced Robotics* 24.5-6, pp. 719–737. URL: https://www.tandfonline.com/doi/full/10.1163/016918610X493552 (cit. on p. 12).

Hereid, Ayonga and Aaron D. Ames (2017). "FROST: Fast robot optimization and simulation toolkit". In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 719–726. URL: http://ieeexplore.ieee.org/document/8202230/ (cit. on p. 13).

Hereid, Ayonga, Eric A. Cousineau, Christian M. Hubicki, and Aaron D. Ames (2016). "3D dynamic walking with underactuated humanoid robots: A direct collocation framework for optimizing hybrid zero dynamics". In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 1447–1454. URL: http://ieeexplore.ieee.org/document/7487279/ (cit. on p. 13).

Hershey, John R. and Peder A. Olsen (2007). "Approximating the Kullback Leibler Divergence Between Gaussian Mixture Models". In: *2007 IEEE International Conference on Acoustics, Speech and Signal Processing - ICASSP '07*. IEEE, pp. IV–317–IV–320. URL: http://ieeexplore.ieee.org/document/4218101/ (cit. on p. 34).

Herzog, Alexander, Ludovic Righetti, Felix Grimminger, Peter Pastor, and Stefan Schaal (2014). "Balancing experiments on a torque-controlled humanoid with hierarchical inverse dynamics". In: *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, pp. 981–988. URL: http://ieeexplore.ieee.org/document/6942678/ (cit. on p. 11).

Herzog, Alexander, Nicholas Rotella, Stefan Schaal, and Ludovic Righetti (2015). “Trajectory generation for multi-contact momentum control”. In: *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*. IEEE, pp. 874–880. URL: http://ieeexplore.ieee.org/document/7363464/ (cit. on pp. 9, 11, 27, 79).

Herzog, Alexander, Nicholas Rotella, Sean Mason, Felix Grimminger, Stefan Schaal, and Ludovic Righetti (2016a). “Momentum control with hierarchical inverse dynamics on a torque-controlled humanoid”. In: *Autonomous Robots* 40.3, pp. 473–491. URL: http://link.springer.com/10.1007/s10514-015-9476-6 (cit. on pp. v, vii, 11, 12, 18, 20, 22, 84).

Herzog, Alexander, Stefan Schaal, and Ludovic Righetti (2016b). “Structured contact force optimization for kino-dynamic motion generation”. In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 2703–2710. URL: http://ieeexplore.ieee.org/document/7759420/ (cit. on pp. 17, 18, 78, 79).

Hirukawa, H., S. Hattori, K. Harada, S. Kajita, K. Kaneko, F. Kanehiro, K. Fujiwara, and M. Morisawa (2006). “A universal stability criterion of the foot contact of legged robots - adios ZMP”. In: *IEEE International Conference on Robotics and Automation*. IEEE, pp. 1976–1983. URL: http://ieeexplore.ieee.org/document/1641995/ (cit. on p. 9).

Holmes, Philip, Robert J. Full, Dan Koditschek, and John Guckenheimer (2006). “The Dynamics of Legged Locomotion: Models, Analyses, and Challenges”. In: *SIAM Review* 48.2, pp. 207–304. URL: http://epubs.siam.org/doi/10.1137/S0036144504445133 (cit. on p. 6).

Hwangbo, Jemin, Joonho Lee, Alexey Dosovitskiy, Dario Bellicoso, Vassilios Tsounis, Vladlen Koltun, and Marco Hutter (2019). “Learning agile and dynamic motor skills for legged robots”. In: *Science Robotics* 4.26, eaau5872. URL: wwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwww (cit. on p. 5).

Jaeheung Park and O. Khatib (2006). “Contact consistent control framework for humanoid robots”. In: *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.* IEEE, pp. 1963–1969. URL: http://ieeexplore.ieee.org/document/1641993/ (cit. on p. 11).

Johnson, Matthew, Brandon Shrewsbury, Sylvain Bertrand, et al. (2015). “Team IHMC’s Lessons Learned from the DARPA Robotics Challenge Trials”. In: *Journal of Field Robotics* 32.2, pp. 192–208. URL: http://doi.wiley.com/10.1002/rob.21571 (cit. on p. 7).

Johnson, Matthew, Brandon Shrewsbury, Sylvain Bertrand, et al. (2017). “Team IHMC’s Lessons Learned from the DARPA Robotics Challenge: Finding Data in the Rubble”. In: *Journal of Field Robotics* 34.2, pp. 241–261. URL: http://doi.wiley.com/10.1002/rob.21674 (cit. on p. 7).

Kajita, S., F. Kanehiro, K. Kaneko, K. Yokoi, and H. Hirukawa (2001). “The 3D linear inverted pendulum mode: a simple modeling for a biped walking pattern generation”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. Vol. 1. IEEE, pp. 239–246. URL: http://ieeexplore.ieee.org/document/973365/ (cit. on pp. x, 8).

Kajita, S., F. Kanehiro, K. Kaneko, K. Fujiwara, K. Harada, K. Yokoi, and H. Hirukawa (2003). “Biped walking pattern generation by using preview control of zero-moment point”. In: *IEEE International Conference on Robotics and Automation*. IEEE, pp. 1620–1626. URL: http://ieeexplore.ieee.org/document/1241826/ (cit. on pp. v, vii, 8, 10, 12, 79).

Kalakrishnan, Mrinal, Sachin Chitta, Evangelos Theodorou, Peter Pastor, and Stefan Schaal (2011). "STOMP: Stochastic trajectory optimization for motion planning". In: *2011 IEEE International Conference on Robotics and Automation*. IEEE, pp. 4569–4574. URL: http://ieeexplore.ieee.org/document/5980280/ (cit. on pp. x, 14).

Kanehiro, Fumio, Hirohisa Hirukawa, and Shuuji Kajita (2004). "OpenHRP: Open Architecture Humanoid Robotics Platform". In: *The International Journal of Robotics Research* 23.2, pp. 155–165. URL: http://journals.sagepub.com/doi/10.1177/0278364904041324 (cit. on p. 56).

Khatib, O. (1987). "A unified approach for motion and force control of robot manipulators: The operational space formulation". In: *IEEE Journal on Robotics and Automation* 3.1, pp. 43–53. URL: http://ieeexplore.ieee.org/document/1087068/ (cit. on p. 11).

Khatib, O., L. Sentis, J. Park, and J. Warren (2004). "Whole-body dynamimc behavior and control of human-like robots". In: *International Journal of Humanoid Robotics* 01.01, pp. 29–43. URL: https://www.worldscientific.com/doi/abs/10.1142/S0219843604000058 (cit. on p. x).

Koch, Kai Henning, Katja Mombaur, and Philippe Soueres (2012). "Optimization-based walking generation for humanoid robot". In: *IFAC Proceedings Volumes* 45.22, pp. 498–504. URL: https://www.sciencedirect.com/science/article/pii/S147466701633659X (cit. on p. 8).

Koenemann, J., Andrea Del Prete, Y. Tassa, Emmanuel Todorov, O. Stasse, M. Bennewitz, and Nicolas Mansard (2015). "Whole-body model-predictive control applied to the HRP-2 humanoid". In: *IEEE International Conference on Intelligent Robots and Systems*. Vol. 2015-Decem. IEEE, pp. 3346–3351. URL: http://ieeexplore.ieee.org/document/7353843/ (cit. on pp. x, 14).

Komura, T., H. Leung, Shunsuke Kudoh, and J. Kuffner (2005). "A Feedback Controller for Biped Humanoids that Can Counteract Large Perturbations During Gait". In: *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*. IEEE, pp. 1989–1995. URL: http://ieeexplore.ieee.org/document/1570405/ (cit. on p. 12).

Koolen, Twan, Jesper Smith, Gray Thomas, et al. (2013). "Summary of Team IHMC's virtual robotics challenge entry". In: *2013 13th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*. IEEE, pp. 307–314. URL: http://ieeexplore.ieee.org/document/7029992/ (cit. on p. 11).

Koolen, Twan, Sylvain Bertrand, Gray Thomas, Tomas de Boer, Tingfan Wu, Jesper Smith, Johannes Englsberger, and Jerry Pratt (2016). "Design of a Momentum-Based Control Framework and Application to the Humanoid Robot Atlas". In: *International Journal of Humanoid Robotics* 13.01, p. 1650007. URL: https://www.worldscientific.com/doi/abs/10.1142/S0219843616500079 (cit. on p. x).

Koyanagi, K., H. Hirukawa, S. Hattori, M. Morisawa, S. Nakaoka, K. Harada, and S. Kajita (2008). "A pattern generator of humanoid robots walking on a rough terrain using a handrail". In: *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, pp. 2617–2622. URL: http://ieeexplore.ieee.org/document/4650686/ (cit. on p. 11).

Kudruss, M., M. Naveau, O. Stasse, N. Mansard, C. Kirches, P. Soueres, and K. Mombaur (2015). "Optimal control for whole-body motion generation using center-of-mass dynamics for predefined multi-contact configurations". In: *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*. IEEE, pp. 684–689. URL: http://ieeexplore.ieee.org/document/7363428/ (cit. on pp. 9, 27).

Kuffner, J.J. and S.M. LaValle (2000). "RRT-connect: An efficient approach to single-query path planning". In: *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*. Vol. 2. IEEE, pp. 995–1001. URL: http://ieeexplore.ieee.org/document/844730/ (cit. on p. 7).

Kuffner, J.J., K. Nishiwaki, S. Kagami, M. Inaba, and H. Inoue. "Footstep planning among obstacles for biped robots". In: *Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems. Expanding the Societal Role of Robotics in the the Next Millennium (Cat. No.01CH37180)*. Vol. 1. IEEE, pp. 500–505. URL: http://ieeexplore.ieee.org/document/973406/ (cit. on p. 7).

LaValle, Steven M. and James J. Kuffner (2001). "Randomized Kinodynamic Planning". In: *The International Journal of Robotics Research* 20.5, pp. 378–400. URL: http://journals.sagepub.com/doi/10.1177/02783640122067453 (cit. on p. 7).

Lee, Sung-Hee and Ambarish Goswami (2012). "A momentum-based balance controller for humanoid robots on non-level and non-stationary ground". In: *Autonomous Robots* 33.4, pp. 399–414. URL: http://link.springer.com/10.1007/s10514-012-9294-z (cit. on p. 11).

Leineweber, Daniel B., Irene Bauer, Hans Georg Bock, and Johannes P. Schlöder (2003). "An efficient multiple shooting based reduced SQP strategy for large-scale dynamic process optimization. Part 1: theoretical aspects". In: *Computers & Chemical Engineering* 27.2, pp. 157–166. URL: https://www.sciencedirect.com/science/article/pii/S0098135402001588 (cit. on p. 86).

Lengagne, Sébastien, Joris Vaillant, Eiichi Yoshida, and Abderrahmane Kheddar (2013). "Generation of whole-body optimal dynamic multi-contact motions". In: *The International Journal of Robotics Research* 32.9-10, pp. 1104–1119. URL: http://journals.sagepub.com/doi/10.1177/0278364913478990 (cit. on pp. 13, 45).

Mansard, N., A. DelPrete, M. Geisert, S. Tonneau, and O. Stasse (2018). "Using a Memory of Motion to Efficiently Warm-Start a Nonlinear Predictive Controller". In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 2986–2993. URL: https://ieeexplore.ieee.org/document/8463154/ (cit. on p. 5).

Mansard, Nicolas (2019). *Feasibility-prone Differential Dynamic Programming: Is DDP a Multiple Shooting Algorithm?* Tech. rep. Toulouse: LAAS. URL: https://github.com/loco-3d/crocoddyl/tree/master/doc/reports/fddp (cit. on pp. 51, 58, 59, 69, 70, 75).

Mastalli, Carlos, Michele Focchi, Ioannis Havoutis, Andreea Radulescu, Sylvain Calinon, Jonas Buchli, Darwin G. Caldwell, and Claudio Semini (2017). "Trajectory and foothold optimization using low-dimensional models for rough terrain locomotion". In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 1096–1103. URL: http://ieeexplore.ieee.org/document/7989131/ (cit. on pp. 11, 103).

Mastalli, Carlos, Ioannis Havoutis, Michele G Focchi, Darwin G Caldwell, and Claudio G Semini (2018). "Motion planning for quadrupedal locomotion: coupled planning, terrain mapping and whole-body control". URL: https://hal.laas.fr/hal-01673438 (cit. on p. 17).

Mastalli, Carlos, Rohan Budhiraja, Wolfgang Merkt, Guilhem Saurel, Bilal Hammoud, Maximilien Naveau, Justin Carpentier, Sethu Vijayakumar, and Nicolas Mansard (2020). "Crocoddyl: An Efficient and Versatile Framework for Multi-Contact Optimal Control". In: *ICRA 2020 - IEEE International Conference on Robotics and Automation*. arXiv: 1909.04947. URL: http://arxiv.org/abs/1909.04947 (cit. on pp. 51, 59).

Mayne, David Q. (1973). "Differential Dynamic ProgrammingA Unified Approach to the Optimization of Dynamic Systems". In: *Control and Dynamic Systems* 10, pp. 179–254. URL: https://www.sciencedirect.com/science/article/pii/B9780120127108500108 (cit. on pp. 44, 46).

McGeer, T. (1990). "Passive Dynamic Walking". In: *The International Journal of Robotics Research* 9.2, pp. 62–82. URL: http://ijr.sagepub.com/content/9/2/62.short (cit. on p. 6).

Michel, P., J. Chestnutt, J. Kuffner, and T. Kanade (2005). "Vision-guided humanoid footstep planning for dynamic environments". In: *5th IEEE-RAS International Conference on Humanoid Robots, 2005*. IEEE, pp. 13–18. URL: http://ieeexplore.ieee.org/document/1573538/ (cit. on p. 7).

Mistry, Michael, Jun Nakanishi, Gordon Cheng, and Stefan Schaal (2008). "Inverse kinematics with floating base and constraints for full body humanoid robot control". In: *Humanoids 2008 - 8th IEEE-RAS International Conference on Humanoid Robots*. IEEE, pp. 22–27. URL: http://ieeexplore.ieee.org/document/4755926/ (cit. on p. 11).

Mistry, Michael, Jonas Buchli, and Stefan Schaal (2010). "Inverse dynamics control of floating base systems using orthogonal decomposition". In: *2010 IEEE International Conference on Robotics and Automation*. IEEE, pp. 3406–3412. URL: http://ieeexplore.ieee.org/document/5509646/ (cit. on p. 11).

Mombaur, Katja D., Richard W. Longman, Hans Georg Bock, and Johannes P. Schlöder (2005a). "Open-loop stable running". In: *Robotica* 23.1, pp. 21–33. URL: https://www.cambridge.org/core/product/identifier/S026357470400058X/type/journal{_}article (cit. on p. x).

Mombaur, K.D., H.G. Bock, J.P. Schloder, and R.W. Longman (2005b). "Self-stabilizing somersaults". In: *IEEE Transactions on Robotics* 21.6, pp. 1148–1157. URL: http://ieeexplore.ieee.org/document/1549941/ (cit. on p. 5).

Mordatch, Igor, Zoran Popović, and Emanuel Todorov (2012a). "Contact-invariant Optimization for Hand Manipulation". In: *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. SCA '12. Goslar Germany, Germany: Eurographics Association, pp. 137–144. URL: http://dl.acm.org/citation.cfm?id=2422356.2422377 (cit. on p. 4).

Mordatch, Igor, Emanuel Todorov, and Zoran Popović (2012b). "Discovery of complex behaviors through contact-invariant optimization". In: *ACM Transactions on Graphics* 31.4, pp. 1–8. URL: http://dl.acm.org/citation.cfm?doid=2185520.2185539 (cit. on pp. 4, 9, 17, 27).

Mordatch, Igor, Igor Mordatch, Emanuel Emmanuel Emanuel Todorov, Igor Mordatch, and Emanuel Emmanuel Emanuel Todorov (2014). "Combining the benefits of function approximation and trajectory optimization". In: *Robotics: Science and Systems X*. Robotics: Science and Systems Foundation. URL: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.494.4434http://www.roboticsproceedings.org/rss10/p52.pdf (cit. on p. 79).

Moreau, J. J. (1988). "Unilateral Contact and Dry Friction in Finite Freedom Dynamics". In: *Nonsmooth Mechanics and Applications*. Vienna: Springer Vienna, pp. 1–82. URL: http://link.springer.com/10.1007/978-3-7091-2624-0{_}1 (cit. on p. 4).

Nakamura, Y. and R. Mukherjee (1990). "Nonholonomic path planning of space robots via bi-directional approach". In: *Proceedings., IEEE International Conference on Robotics and Automation*. IEEE Comput. Soc. Press, pp. 1764–1769. URL: http://ieeexplore.ieee.org/document/126264/ (cit. on p. 12).

NASA (1988). *Zero-G: "Movement in Microgravity: Skylab to Space Shuttle"1988 NASA Weightlessness Footage*. URL: https://www.dailymotion.com/video/x2qbus8 (cit. on p. 43).

Neunert, Michael, Markus Stauble, Markus Giftthaler, Carmine D. Bellicoso, Jan Carius, Christian Gehring, Marco Hutter, and Jonas Buchli (2018). "Whole-Body Nonlinear Model Predictive Control Through Contacts for Quadrupeds". In: *IEEE Robotics and Automation Letters* 3.3, pp. 1458–1465. URL: http://ieeexplore.ieee.org/document/8276298/ (cit. on pp. 14, 16, 76).

Orin, David E., Ambarish Goswami, and Sung-Hee Lee (2013). "Centroidal dynamics of a humanoid robot". In: *Autonomous Robots* 35.2-3, pp. 161–176. URL: http://link.springer.com/10.1007/s10514-013-9341-4 (cit. on pp. x, 9, 16, 65).

Orthey, Andreas and Olivier Stasse (2013). "Towards reactive whole-body motion planning in cluttered environments by precomputing feasible motion spaces". In: *2013 13th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*. IEEE, pp. 274–279. URL: http://ieeexplore.ieee.org/document/7029987/ (cit. on pp. 27, 30).

Papadopoulos, Evangelos G. (1993). "Nonholonomic Behavior in Free-floating Space Manipulators and its Utilization". In: *Nonholonomic Motion Planning*. Boston, MA: Springer US, pp. 423–445. URL: http://link.springer.com/10.1007/978-1-4615-3176-0{_}11 (cit. on p. 12).

Parzen, Emanuel (1962). *On Estimation of a Probability Density Function and Mode*. URL: https://www.jstor.org/stable/2237880 (cit. on p. 33).

Perrin, Nicolas, Olivier Stasse, Léo Baudouin, Florent Lamiraux, and Eiichi Yoshida (2012). "Fast Humanoid Robot Collision-Free Footstep Planning Using Swept Volume Approximations". In: *IEEE Transactions on Robotics* 28.2, pp. 427–439. URL: http://ieeexplore.ieee.org/document/6078435/ (cit. on pp. 27, 30).

Ponton, Brahayam, Alexander Herzog, Stefan Schaal, and Ludovic Righetti (2016). "A convex model of humanoid momentum dynamics for multi-contact motion generation". In: *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*. IEEE, pp. 842–849. URL: http://ieeexplore.ieee.org/document/7803371/ (cit. on p. 18).

Popovic, M., A. Hofmann, and H. Herr (2004). "Angular momentum regulation during human walking: biomechanics and control". In: *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004*. IEEE, 2405–2411 Vol.3. URL: http://ieeexplore.ieee.org/document/1307421/ (cit. on p. 12).

Posa, Michael and Russ Tedrake (2013). “Direct Trajectory Optimization of Rigid Body Dynamical Systems through Contact”. In: Springer, Berlin, Heidelberg, pp. 527–542. URL: http://link.springer.com/10.1007/978-3-642-36279-8{_}32 (cit. on p. 13).

Posa, Michael, Cecilia Cantu, and Russ Tedrake (2014). “A direct method for trajectory optimization of rigid bodies through contact”. In: *The International Journal of Robotics Research* 33.1, pp. 69–81. URL: http://journals.sagepub.com/doi/10.1177/0278364913506757 (cit. on pp. x, 4).

Pratt, Jerry, John Carff, Sergey Drakunov, and Ambarish Goswami (2006). “Capture Point: A Step toward Humanoid Push Recovery”. In: *2006 6th IEEE-RAS International Conference on Humanoid Robots*. IEEE, pp. 200–207. URL: http://ieeexplore.ieee.org/document/4115602/ (cit. on p. 9).

Pratt, Jerry, Twan Koolen, Tomas de Boer, John Rebula, Sebastien Cotton, John Carff, Matthew Johnson, and Peter Neuhaus (2012). “Capturability-based analysis and control of legged locomotion, Part 2: Application to M2V2, a lower-body humanoid”. In: *The International Journal of Robotics Research* 31.10, pp. 1117–1133. URL: http://journals.sagepub.com/doi/10.1177/0278364912452762 (cit. on p. 7).

Press, William H., Saul Teukolsky, William Vetterling, and Brian Flannery (2007). *Numerical recipes : the art of scientific computing*. 3rd ed. Cambridge University Press, p. 1235 (cit. on pp. 53, 64, 76).

Rajamaki, Joose, Kourosh Naderi, Ville Kyrki, and Perttu Hamalainen (2016). “Sampled differential dynamic programming”. In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 1402–1409. URL: http://ieeexplore.ieee.org/document/7759229/ (cit. on p. 14).

Rajeswaran, Aravind, Kendall Lowrey, Emanuel V. Todorov, and Sham M. Kakade (2017). “Towards Generalization and Simplicity in Continuous Control”. In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. NIPS’17. USA: Curran Associates Inc., pp. 6550–6561. URL: http://dl.acm.org/citation.cfm?id=3295222.3295401http://papers.nips.cc/paper/7233-towards-generalization-and-simplicity-in-continuous-control (cit. on pp. v, vii, 5).

Rostami, Mostafa and Guy Bessonnet (2001). “Sagittal gait of a biped robot during the single support phase. Part 2: optimal motion”. In: *Robotica* 19.3, pp. 241–253. URL: https://www.cambridge.org/core/product/identifier/S0263574700003039/type/journal{_}article (cit. on p. 13).

Roussel, L., C. Canudas-De-Wit, and A. Goswami (1998). “Generation of energy optimal complete gait cycles for biped robots”. In: *Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No.98CH36146)*. Vol. 3. IEEE, pp. 2036–2041. URL: http://ieeexplore.ieee.org/document/680615/ (cit. on p. 13).

Saab, Layale, Oscar E. Ramos, Francois Keith, Nicolas Mansard, Philippe Soueres, and Jean-Yves Fourquet (2013). “Dynamic Whole-Body Motion Generation Under Rigid Contacts and Other Unilateral Constraints”. In: *IEEE Transactions on Robotics* 29.2, pp. 346–362. URL: http://ieeexplore.ieee.org/document/6482266/ (cit. on pp. 11, 36, 40, 52, 56).

Schultz, G and K Mombaur (2010). “Modeling and Optimal Control of Human-Like Running”. In: *IEEE/ASME Transactions on Mechatronics* 15.5, pp. 783–792. URL: http://ieeexplore.ieee.org/document/5345770/ (cit. on pp. 5–7, 45).

Sentis, L. and O. Khatib (2005). “Control of Free-Floating Humanoid Robots Through Task Prioritization”. In: *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*. IEEE, pp. 1718–1723. URL: http://ieeexplore.ieee.org/document/1570361/ (cit. on p. 11).

Sherikov, Alexander, Dimitar Dimitrov, and Pierre-Brice Wieber (2014). “Whole body motion controller with long-term balance constraints”. In: *IEEE-RAS International Conference on Humanoid Robots*. IEEE, pp. 444–450. URL: http://ieeexplore.ieee.org/document/7041399/ (cit. on pp. 8, 12).

SmarterEveryDay (2012). *Slow Motion Flipping Cat Physics*. URL: https://www.youtube.com/watch?v=RtWbpyjJqrU (cit. on p. 43).

Sohl, Garett A. and James E. Bobrow (2001). “A Recursive Multibody Dynamics and Sensitivity Algorithm for Branched Kinematic Chains”. In: *Journal of Dynamic Systems, Measurement, and Control* 123.3, p. 391. URL: http://dynamicsystems.asmedigitalcollection.asme.org/article.aspx?articleid=1478039 (cit. on pp. 59, 68, 102).

Stephens, B J and C G Atkeson (2010). “Dynamic Balance Force Control for compliant humanoid robots”. In: *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, pp. 1248–1255. URL: http://ieeexplore.ieee.org/document/5648837/ (cit. on p. 11).

Stewart, D. and J.C. Trinkle (2000). “An implicit time-stepping scheme for rigid body dynamics with Coulomb friction”. In: *IEEE International Conference on Robotics and Automation*. Vol. 1. IEEE, pp. 162–169. URL: http://ieeexplore.ieee.org/document/844054/ (cit. on pp. x, 4).

Stewart, David E. (2000). “Rigid-Body Dynamics with Friction and Impact”. In: *SIAM Review* 42.1, pp. 3–39. URL: http://epubs.siam.org/doi/10.1137/S0036144599360110 (cit. on p. 4).

Talele, Nihar and Katie Byl (2018). “Toward Efficient and Robust Biped Walking Optimization”. In: arXiv: 1807.09905. URL: http://arxiv.org/abs/1807.09905 (cit. on p. 13).

Tassa, Yuval, Tom Erez, and Emanuel Todorov (2012). “Synthesis and stabilization of complex behaviors through online trajectory optimization”. In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, pp. 4906–4913. URL: http://ieeexplore.ieee.org/document/6386025/ (cit. on pp. 8, 44, 50).

Tassa, Yuval, Nicolas Mansard, and Emmanuel Emo Todorov (2014). “Control-limited differential dynamic programming”. In: *Proceedings - IEEE International Conference on Robotics and Automation*. IEEE, pp. 1168–1175. URL: http://ieeexplore.ieee.org/document/6907001/http://homes.cs.washington.edu/{~}todorov/papers/TassaICRA14.pdf (cit. on pp. 14, 69, 70, 75).

Tevatia, G. and S. Schaal (2000). “Inverse kinematics for humanoid robots”. In: *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*. Vol. 1. IEEE, pp. 294–299. URL: http://ieeexplore.ieee.org/document/844073/ (cit. on pp. x, 10).

Todorov, Emanuel (2004). “Optimality principles in sensorimotor control”. In: *Nature Neuroscience* 7.9, pp. 907–915. URL: http://www.nature.com/articles/nn1309 (cit. on p. 5).

– (2010). “Implicit nonlinear complementarity: A new approach to contact dynamics”. In: *2010 IEEE International Conference on Robotics and Automation*. IEEE, pp. 2322–2329. URL: http://ieeexplore.ieee.org/document/5509739/ (cit. on p. 5).

Todorov, Emanuel, Tom Erez, and Yuval Tassa (2012). “MuJoCo: A physics engine for model-based control”. In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, pp. 5026–5033. URL: http://ieeexplore.ieee.org/document/6386109/ (cit. on p. 5).

Tonneau, Steve, Pierre Fernbach, Andrea Del Prete, Julien Pettré, and Nicolas Mansard (2018a). “2PAC: Two-Point Attractors for Center Of Mass Trajectories in Multi-Contact Scenarios”. In: *ACM Transactions on Graphics* 37.5, pp. 1–14. URL: http://dl.acm.org/citation.cfm?doid=3278329.3213773 (cit. on p. 7).

Tonneau, Steve, Nicolas Mansard, C. Park, D. Manocha, F. Multon, and J. Pettré (2018b). “A Reachability-Based Planner for Sequences of Acyclic Contacts in Cluttered Environments”. In: Springer, Cham, pp. 287–303. URL: http://link.springer.com/10.1007/978-3-319-60916-4{_}17 (cit. on pp. 29, 56, 86).

Tonneau, Steve, Andrea Del Prete, Julien Pettre, Chonhyon Park, Dinesh Manocha, and Nicolas Mansard (2018c). “An Efficient Acyclic Contact Planner for Multiped Robots”. In: *IEEE Transactions on Robotics* 34.3, pp. 586–601. URL: https://ieeexplore.ieee.org/document/8341955/ (cit. on pp. 17, 20).

Tonneau, Steve, Daeun Song, Pierre Fernbach, Nicolas Mansard, Michel Taix, and Andrea Del Prete (2019). “SL1M: Sparse L1-norm Minimization for contact planning on uneven terrain”. In: arXiv: 1909.09044. URL: http://arxiv.org/abs/1909.09044 (cit. on p. 103).

Toussaint, Marc (2017). “A tutorial on Newton methods for constrained trajectory optimization and relations to SLAM, Gaussian Process smoothing, optimal control, and probabilistic inference”. In: *Geometric and Numerical Foundations of Movements, Springer*. Ed. by Jean-Paul Laumond. Springer (cit. on pp. 50, 63).

Trinkle, J. C., J.-S. Pang, S. Sudarsky, and G. Lo (1997). “On Dynamic Multi-Rigid-Body Contact Problems with Coulomb Friction”. In: *ZAMM - Journal of Applied Mathematics and Mechanics / Zeitschrift für Angewandte Mathematik und Mechanik* 77.4, pp. 267–279. URL: http://doi.wiley.com/10.1002/zamm.19970770411 (cit. on p. 4).

Udwadia, F. E. and R. E. Kalaba (1992). “A New Perspective on Constrained Motion”. In: *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 439.1906, pp. 407–410. URL: http://rspa.royalsocietypublishing.org/cgi/doi/10.1098/rspa.1992.0158 (cit. on p. 51).

Vaillant, Joris, Abderrahmane Kheddar, Hervé Audren, François Keith, Stanislas Brossette, Adrien Escande, Karim Bouyarmane, Kenji Kaneko, Mitsuharu Morisawa, Pierre Gergondet, Eiichi Yoshida, Suuji Kajita, and Fumio Kanehiro (2016). “Multi-contact vertical ladder climbing with an HRP-2 humanoid”. In: *Autonomous Robots* 40.3, pp. 561–580. URL: http://link.springer.com/10.1007/s10514-016-9546-4 (cit. on p. 11).

Wang, Yu, Wotao Yin, and Jinshan Zeng (2019). “Global Convergence of ADMM in Nonconvex Nonsmooth Optimization”. In: *Journal of Scientific Computing* 78.1, pp. 29–63. URL: http://link.springer.com/10.1007/s10915-018-0757-z (cit. on p. 81).

Westervelt, E.R. and J.W. Grizzle (2002). “Design of asymptotically stable walking for a 5-link planar biped walker via optimization”. In: *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292)*. Vol. 3. IEEE, pp. 3117–3122. URL: http://ieeexplore.ieee.org/document/1013706/ (cit. on p. 13).

Wieber, P.-B. (2008). “Viability and predictive control for safe locomotion”. In: *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, pp. 1103–1108. URL: http://ieeexplore.ieee.org/document/4651022/ (cit. on pp. 20, 21, 26).

– (2002). “On the stability of walking systems”. In: *Proceedings of the International Workshop on Humanoid and Human Friendly Robotics*. Tsukuba, Japan. URL: https://hal.inria.fr/inria-00390866 (cit. on p. 28).

Wieber, Pierre-Brice (2006a). “Holonomy and nonholonomy in the dynamics of articulated motion”. In: *Lecture Notes in Control and Information Sciences*. Vol. 340. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 411–425. URL: http://link.springer.com/10.1007/978-3-540-36119-0{_}20 (cit. on pp. 8, 15, 42, 44, 51, 57, 78).

Wieber, Pierre-brice (2006b). “Trajectory Free Linear Model Predictive Control for Stable Walking in the Presence of Strong Perturbations”. In: *2006 6th IEEE-RAS International Conference on Humanoid Robots*. IEEE, pp. 137–142. URL: http://ieeexplore.ieee.org/document/4115592/ (cit. on p. 10).

Winkler, Alexander W., Carlos Mastalli, Ioannis Havoutis, Michele Focchi, Darwin G. Caldwell, and Claudio Semini (2015). “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)*. IEEE, pp. 5148–5154. URL: http://ieeexplore.ieee.org/document/7139916/ (cit. on p. 11).

Winkler, Alexander W., C. Dario Bellicoso, Marco Hutter, and Jonas Buchli (2018). “Gait and Trajectory Optimization for Legged Systems Through Phase-Based End-Effector Parameterization”. In: *IEEE Robotics and Automation Letters* 3.3, pp. 1560–1567. URL: http://ieeexplore.ieee.org/document/8283570/ (cit. on p. 11).

Xie, Zhaoming, C. Karen Liu, and Kris Hauser (2017). “Differential dynamic programming with nonlinear constraints”. In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 695–702. URL: http://ieeexplore.ieee.org/document/7989086/ (cit. on p. 75).

Zaytsev, Petr (2015). “Using Controllability Of Simple Models To Generate Maximally Robust Walking-Robot Controllers”. PhD thesis. Cornell University. URL: https://ecommons.cornell.edu/handle/1813/39466 (cit. on pp. 27, 30).

Zucker, M., N. Ratliff, A. D. Dragan, M. Pivtoraiko, M. Klingensmith, C. M. Dellin, J. A. Bagnell, and S. S. Srinivasa (2013). “CHOMP: Covariant Hamiltonian optimization for motion planning”. In: *The International Journal of Robotics Research* 32.9-10, pp. 1164–1193. URL: http://ijr.sagepub.com/content/32/9-10/1164.short (cit. on p. 14).

List of Figures

1.1 **Multi-body Locomotion**. Examples of locomotion by making and breaking contacts. The gaits, kinematics, and the number of contacts differ, but the essential features of the movement remain the same. 2

1.2 Overview of our multi-stage locomotion framework Carpentier et al., 2017b. Given a requested path request between start and goal positions (left image), $\mathcal{P}_1$ is the problem of computing a guide path in the space of equilibrium feasible root configurations. We achieve this by defining a geometric condition, the reachability condition (abstracted with the transparent cylinders on the middle image). $\mathcal{P}_2$ is then the problem of extending the path into a discrete sequence of contact configurations. Finally, $\mathcal{P}_3$ creates a whole-body trajectory by solving for centroidal dynamics trajectories and using them as a reference. 20

2.1 Illustration of HRP-2 robot and TALOS robot making contacts with their environment. The green "ice-cream" cones are dispatched on the 4 vertices of the feet, symbolizing the friction cones with friction coefficient of value 0.3. 29

2.2 Illustration of the probability density distribution of the CoM w.r.t. the right foot frame of HRP-2. The PDF are projected along the three axis X,Y,Z and represented by the means of color map: the low values are closed to the blue colour while the high values tend to be more red. The first row corresponds to the ground truth distribution estimated through KDE. The KDE is composed of 20000 points. The second row is the colour map of the GMM used in the OCP and composed of 7 Gaussian kernels. 35

2.3 Evolution of the KL divergence between the KDE distribution and GMMs of different sizes for the four end-effectors of the HRP-2 robot. 36

2.4 Illustration of the probability density distribution of the CoM w.r.t. the right foot frame of TALOS robot. The PDF are projected along the three axis X,Y,Z and represented by the means of color map: the low values are closed to the blue colour while the high values tend to be more red. The first row corresponds to the ground truth distribution estimated through KDE. The KDE is composed of 20000 points. The second row is the colour map of the GMM used in the OCP and composed of 4 Gaussian kernels. The axes have the same scale as in Fig 2.2. 37

2.5 Projection of the CoM trajectory inside the right foot frame with and without taking into account the log-pdf term in the cost function. The level set corresponds to the GMM distribution used in our OCP. 38

2.6 Snapshots of the climbing up 10-cm high steps motion with the HRP-2 robot. 38

2.7 Snapshots of the climbing up 15-cm high steps motion with the HRP-2 using the guardrail. 38

2.8 Snapshots of the climbing 15-cm high steps motion with guardrail by the TALOS robot in simulation. 39

3.1 An astronaut rotating himself in the transverse plane. There is a mismatch in the inertia properties of the upper and the lower limbs. As a result, the astronaut can create different velocities for the upper and lower limbs, while maintaining the angular momentum conservation constraint. Eventually, this changes the orientation of his body. Video thanks to NASA, 1988 43

3.2 Caption for Cat 43

3.3 Evolution of the different cost functions (normalized) with respect to iterations. DDP reduces the applied torques by recalculating the CoM tracking. It improves the contact force by taking into account the whole-body angular momentum. The result is a continuous improvement in the performance as compared to IK. We stop after 100 iterations. ("FF SO3" refers to the free-flyer orientation cost; CoM refers to Center of Mass tracking cost, and so on...) 54

3.4 Comparison between the IK and DDP trajectory for 100 cm stride on the HRP-2 robot. *Top*: Knee torques generated in the left leg. *Bottom*: GRFs generated in the left foot. The DDP formulation allows us to utilize the angular momentum of the upper body, which reduces the requirement on the lower body to create a counterbalancing motion. This results in a lower torque in the lower body, as well as lower GRFs. Around $t = 14s$ we can see high peaks for the IK and DDP trajectories of 895 N and 755 N, respectively. 54

3.5 Joint torques for the astronaut maneuver. Our method plans a smart strategy by kick-starting the rotation, and then tries to maintain the velocity by small bang-bang control signals. Towards the end, it changes again the velocity of the lower legs in order to bring the system to a stop. 55

3.6 Snapshots of 100 cm stride on a flat terrain used to evaluate the performance of our whole-body trajectory optimization method. The DDP trajectory reduces significantly the normal forces peaks compared with classic whole-body IK. 55

3.7 Attitude adjustment maneuver conducted by the robot in gravity free space. DDP solver takes into account the non-holonomic angular momentum constraint and uses internal actuation to rotate 360° without the need for contact forces. 55

4.1 Illustrating the First Order Markov structure which guides our problem. The constraints and costs of a node are dependent only on the current state, and the current state is only dependent on the previous state. (Toussaint, 2017) 63

4.2 Creating a shooting problem using crocoddyl 67

4.3 Different legged gaits optimized by our FDDP algorithm. (top) from left to right, walking and trotting gaits on the ANYmal robot, respectively; (bottom) biped walking gait using the Talos' legs. You can run these results in our repository 71

4.4 Weight values for each cost function used in the generation of the legged gaits. The cost functions are (a) CoM: CoM tracking of the interpolated postures; (b) Swing: swing tracking of the reference foot trajectory; (c) footstep deviation from the predefined placement; (d) state regularization; (e) control regularization. Note that the values are in logarithmic scale. 71

4.5 Snapshots of highly-dynamic maneuvers in legged robots using the FDDP algorithm in Crocoddyl. (top) jumping obstacles in a humanoid robot; (middle) front-flip maneuver in a biped robot; (bottom) jumping obstacles in a quadruped robot. 72

4.6 Gaps contraction and convergence rates for different motions. (top) Gaps are closed in the first iteration for simpler motions such as biped walking and quadrupedal gaits. Instead, the FDDP solver chooses to keep the gaps open for the early iterations for highly-dynamic maneuvers. Here we use the L2-norm of the total gaps, i.e., gaps for all the nodes of the trajectory. (bottom) The required iterations increases mainly with the dynamics of the gait and numbers of nodes. For instance, we can see lower rate of improvement in the first nine iterations in the ANYmal (jump-4f) and ICub jumps (jump-2f). In case of the quadrupedal walking, we have very short durations in the four-feet support phases, making it a *dynamic* walk. 74

4.7 Computation time per iteration for different CPUs and level of parallelism. Note that the use of hyper-threading decreases the computation frequency for all tested CPUs. 74

4.8 Computation frequency per iteration for different motions for different PCs. PC1 has specifications typical for on-board computers found on robots, while PC3 uses high-performance CPU and RAM. The reported values use the optimal number of threads as identified in Fig. 4.7. . . . 75

5.1 Walking sequence generated for HRP-2 robot using the proposed ADMM solver. 85

5.2 Evolution of the total norm of the constraint residual along the iterations of the ADMM solver. 86

5.3 CoM Trajectory in the XY plane for the first three iterations of the ADMM solver. 87

5.4 Residue between the centroidal and whole-body trajectory of AM and CoM over the iterations. 88

6.1 *Problem Definition*: The robot is asked to make a jump forward. The current feet contacts, and the feet contacts on landing, are provided. The robot has to find the state and control trajectory of motion that satisfy its kinematic and dynamic constraints. 92

6.2 Solution of the Centroidal Optimization Problem (6.1). The ball shows the trajectory followed by the center of mass, when we obtain the solution of our optimization problem (6.1). Since we did not consider the feasibility constraints, the solution assumes really long robot legs. This trajectory can be improved by taking kinematic information in our optimization. 93

6.3 Center of Mass points with respect to left foot placement. Joint configurations are sampled, and for each non self-colliding joint configuration, the center of mass position is computed and learned with respect to the end-effectors. 95

6.4 Solution of the Centroidal Optimization Problem (6.1) with feasibility constraints, as discussed in Sec 2.3. The ball shows the trajectory followed by the center of mass. We see that the new solution takes into account the kinematic limits of the robot. Instead of simply ascending, it first descends in order to create space of motion. After landing, we see similar behaviour, where it descends in order to distribute the impact to a longer duration. 96
6.5 Comparison of the Angular Momentum trajectory, after the centroidal (red) and whole-body(blue) optimization. The angular momentum output from the centroidal optimization is close to zero. However, the angular momentum required by the whole-body optimization is high, because of the dynamic nature of the problem. As a result, tracking the centroidal references does not result in a good whole-body solution. . . 97
6.6 Comparison of the Center of Mass trajectory, after the centroidal (red) and whole-body(blue) optimizations. We took the CoM kinematic feasibility into account in our centroidal problem. However, as a result of the tracking error in angular momentum, the whole-body solver cannot properly track the center of mass trajectories. 97
6.7 Snapshots that show the final jumping motion as a result of ADMM iterations on our centroidal and whole-body optimization formulation. Our whole-body solver (DDP) does not take into account the constraints on the contact forces. On the other hand, our centroidal solver does not take into account the information about the dynamics of our multi-body robot. However, both these solvers work together and are able to arrive at the common solution which is both highly dynamic, and dynamically feasible. 99

Colophon

This thesis was typeset with LaTeX 2_ε. It uses the *Clean Thesis* style developed by Ricardo Langner. The design of the *Clean Thesis* style is inspired by user guide documents from Apple Inc.

Download the *Clean Thesis* style at `http://cleanthesis.der-ric.de/`.

Declaration

I declare the emperor penguins to be the true monarchs.

Toulouse, November 29, 2019

Rohan Budhiraja