DISSERTATION

submitted

to the

Combined Faculty of the Natural Sciences and Mathematics

of

HEIDELBERG UNIVERSITY, GERMANY

for the degree of

Doctor of Natural Sciences

Put forward by

Yue Hu

Born in:

Jinzhou (Liaoning), China

Oral examination:

The role of compliance in humans and humanoid robots locomotion

Advisors

Prof. Dr. Katja Mombaur

Dr. Francesco Nori

Zusammenfassung

Wir konstruieren Roboter, die wie Menschen arbeiten, wie Menschen aussehen oder allgemein vom Menschen inspiriert sind. Viele menschliche Eigenschaften haben wir jedoch noch nicht vollständig verstanden, da der Mensch ein sehr komplexes System darstellt. Eine grundlegende Eigenschaften ist die sogenannte *Konformität (Compliance)*. Wären unsere Körper vollkommen starr, so könnten wir nicht so einfach auf Bäume klettern oder Bergpfaden folgen. Obwohl humanoide Roboter vom Menschen inspiriert sein sollen, werden sie seit ihrem ersten Erscheinen mit starren Glieder konstruiert, die mittels starrer Gelenke verbunden sind. Erst in neuster Zeit wurden sie durch die Entwicklung von konformen Aktuatoren menschenähnlicher gestaltet.

Ziel dieser Arbeit ist es, die Rolle der Konformität bei menschlichen Bewegungen und der Bewegung humanoider Roboter zu analysieren. Wir modellieren sowohl den menschlichen Körper als auch die humanoiden Roboter als *starre Mehrkörpersysteme*. Beide Systeme sind stark redundant, aus diesem Grund ist Optimierung ein erforderliches Instrument zum Erreichen unserer Ziele. In diesem Fall sind es vornehmlich *Optimalsteuerungsmethoden*.
In den meisten aktuellen Fortbewegungsmechanismen wird Konformität auf der Ebene der Gelenkwinkel durch elastische Komponenten mit konstanter Steifigkeit eingeführt. Dies liegt darin begründet, dass das Verändern der Steifigkeit relativ kompliziert ist bzw. Aktuatoren mit variabler Steifigkeit vergleichsweise große Abmessungen haben. Biomechanische Studien zeigen hingegen, dass sich die Steifigkeit menschlicher Gelenke während der Bewegung ändert. Daher möchten wir folgender Frage nachgehen: Wie verändert sich die Steifigkeit beim menschlichen Gehen und welchen Einfluss haben diese Modulationen auf den Gang?
Um diese Fragen zu beantworten, verwendeten wir Bewegungsdaten von Menschen, die wir mit Hilfe von Motion-Capture-Systemen gewonnen haben, sowie ein dynamisches 2D Modell des menschlichen Körpers, bei dem die Gelenke in den Beinen als Torsionsfeder und bi-artikularer Kupplungsfeder mit variabler Steifigkeit modelliert sind. Wir errechneten so die Steifigkeitsprofile dieser Federn, was uns aufzeigte, wie sich die Stefigkeit während des Laufzyklus verändert und dass diese ebenfalls sehr große Werte annehmen kann; ganz im Gegensatz zu aktuellen Laufmechanismen humanoider Roboter. Des Weiteren untersuchten wir, wie sich der Gang verändert, wenn die Steifigkeitsmodulation reduziert wird. Der ursprüngliche Gang konnte in nicht-beschränkten Laufszenarien, wie dem Laufen auf ebenem Untergrund oder auf einer Steigung, näherungsweise reproduziert werden, in beschränkten Szenarien wie dem Treppensteigen hingegen nicht. Dieses Ergebnis zeigt die Bedeutung der Steifigkeitsmodulation während des Gehens und kann dem Design zukünftiger konformer Aktuatoren dienen.
Es gibt mehrere humanoide Roboter mit konformen Aktuatoren. iCub, ein weit verbreiteter, humanoider Forschungsroboter, ist einer von ihnen. Er erhielt erst vor kurzem Beine, die mit Serial-Elastic-Actuator (SEA) ausgestattet sind. Eine reduzierte Version des iCub, der HeiCub, wurde Ende 2014 an die Universität Heidelberg geliefert und ist der in dieser Arbeit verwendete Roboter.
Zunächst analysierten wir die Bewegung einer Kniebeuge, die als Optimalsteuerungproblem formuliert wurde. Dazu wurden nur die drei Pitch-Gelenke der Beine verwendet, wobei trotzdem die gesamte Dynamik des Körpers beachtet wird. Kniebeugen-Bewegungen wurden auf diese Weise mit unterschiedlichen Zielfunktionen erzeugt, jeweils mit und ohne SEA. In einem weiteren Schritt wurden auf die gleiche Weise Push-Recovery-Bewegungen erzeugt, ebenfalls unter Einbezug der SEA.
Aus Mangel an Literatur und Experimenten zu iCubs Geh-Fähigkeit entschieden wir uns, die Leistungsfähigkeit von HeiCub in diesem Bereich genau zu analysieren. Wir verwendeten das Tischwagenmodell, um Lauftrajektorien auf ebenem Boden, Steigungen und Treppen zu erzeugen, die bisher noch nie von einem anderen iCub Robotern bewältigt wurden. Auf diese Weise konnten wir bisher unbekannte Details über die Plattform gewinnen, die für zukünftige Formulierung von Optimalsteuerungsproblemen von grundlegender Bedeutung sind. Dank dieser Arbeit haben künftige Entwicklungen des Frameworks zur Steuerung der Gehbewegungen der iCub Roboterfamilie nun eine Referenz.

Abstract

We build robots that are meant to look and work like humans, with humans, inspired by humans. But many are the human characteristics that we have not yet understood, as humans are highly complex systems. One fundamental characteristic is *compliance*, which characterizes human movements. If our body was completely rigid, we would not be able to climb up trees or walk on mountainous paths as easily as we do. But despite being inspired to be a copy of human beings, humanoid robots had rigid links connected with rigid joints since their first appearance. It is only recently that they started to be more "human-like", with the development of compliant actuators.

In this thesis the objective is to analyze of the role of compliance in human walking and in humanoid robots motions. We model both the human body and humanoid robots as *rigid multi-body systems*. Both systems are highly redundant, reason for which optimization represents an essential tool to achieve our goals. In particular, we adopt *optimal control approaches*.

In many state of the art compliant walking mechanisms, compliance is introduced at joint level by means of elastic components with constant stiffness, due to the difficulty of varying stiffness and the considerable dimensions of currently available variable stiffness actuators. This is the reason for which many studies focused on finding constant joint stiffness during human walking. However, biomechanics studies have shown that stiffness changes in human joints during movements. The questions we want to address are therefore: how does stiffness modulate during human walking and what is the influence of such modulations on the gait?
To answer these questions, we used walking motions from motion capture data and a 2D dynamic model of the human body, where the actuation of the leg joints are modeled with torsional springs and bi-articular coupling springs with variable stiffness. We computed the stiffness profiles of these springs, which showed how stiffness changes over the walking cycle and can also assume big values, contrasting with many state of the art walking mechanisms. We proceeded by analyzing how walking gaits are modified if the stiffness modulation is reduced. This further step showed that the original walking gait could be approximated in unconstrained walking scenarios such as level ground and slopes but not in constraint ones as stairs. This result demonstrated the importance of stiffness modulation during walking and can serve for future compliant actuators design.

There are several existing humanoid robots with compliant actuators. Among these, the iCub is a widely spreaded advanced research humanoid that has recently acquired legs with Series Elastic Actuators (SEA). The reduced version of it, HeiCub, was delivered to Heidelberg University by the end of 2014 and is the robot used in this thesis.
We first analyzed the motion of squatting. The problem is formulated as an optimal control problem where only the three pitch joints of the legs are considered active and the whole-body dynamics of the robot is used. Squat motions for different objective functions are generated for the robot with and without the use of SEA. A step further is taken in using all the actuated degrees of freedom of the robot to generate push recovery motions with the same approach, also considering the SEA.
As there is a lack of literature and experiments of iCub walking, for this complex task we aimed at exploiting the capabilities of HeiCub by measuring its walking performances. We used the table cart model to generate walking trajectories on level ground, slope and stairs, which have never been achieved before by other iCub robots. In this way we could gain details of the platform that were unknown beforehand that are fundamental to be used in future optimal control fomulations. Thanks to this study, future developments of walking control frameworks for the iCub family robots have now a point of reference.

Acknowledgements

The work of this thesis has been conducted with the financial support of the *European Project KoroiBot (grant n. 611909)*, in collaboration between the *Optimization in Robotics and Biomechanics (ORB)* group of the *Interdisciplinary Center for Scientific Computing (IWR)*, *Ruprecht-Karls-Universität Heidelberg, Germany* and the *iCub Facility* department of *Fondazione Istituto Italiano di Tecnologia (IIT), Genoa, Italy*.

First of all, thanks to my supervisor *Prof. Dr. Katja Mombaur* who gave me the possibility of carrying out my work with high independence and at the same time illuminating and helping me with useful ideas and discussions, allowing me to build international experience with the participation to several conferences and workshops.
Thanks to my co-supervisor *Dr. Francesco Nori* from IIT, who gave me the opportunity to learn about and work with more iCubs and build collaborations with IIT researchers by granting me the access to the IIT infrastructures in Genoa.
Thanks to the *Simulation and Optimization* group (IWR - Ruprecht-Karls-Universität Heidelberg) of *Prof. Dr. Dr. h.c. mult. Hans Georg Bock* for giving me the possibility to work with the optimal control software package MUSCOD-II, which has been largely used to achieve the results of this thesis.

Thanks to my colleagues, both from Heidelberg and IIT, who supported me during the last three and half years in building up my projects. Especially to *Debora, Manuel* and *Kevin*, who shared the office with me, and to *Malin, Paul, Benjamin* and *Davide* who reviewed parts of this manuscript.

Thanks to all my friends, as we were able to keep our friendship despite being scattered around the world and many of us have dived into the path of PhD, sharing the bitter and sweet of it. To my dear friends *Claudia* and *Marina*, who shared with me the anxiety and stress of the writing period. In particular *Marina*, who carefully translated my abstract to German together with *Benjamin*.

Special thanks to my parents, as both have always supported me in chasing my own way and following my dreams, despite being far away from home. Especially to my dad, who also went down the path of the PhD and has always been so proud of my choice, I am sure he will be happy to have two Dr. Hu in the family.

Last but not least, and this might seem weird, I would like to thank my past self from five years ago, the 22 years old Yue Hu who chose without hesitation to follow her passion for robotics and left for France. It is thanks to that choice that now I have the opportunity to be here, getting closer to realising my dreams.

Contents

Introduction and organisation

Locomotion —

Movement or the ability to move from one place to another. (Oxford dictionary)

When referring to humans, the term locomotion is commonly used to indicate the motions of walking or running. As humanoid robots are machines built in the likeness of humans, the same terminology indicates the same type of motions for humanoid robots.
Locomotion is indeed one of the most characterizing motions of humans. Thanks to locomotion capabilities, humans are able to move in an agile way in a high variety of environments, ranging from everyday life offices and homes to complex factories and disaster sites.
Many research fields, such as neuroscience, biomechanics and robotics, have been trying to understand how do humans walk or run from different perspectives. In biomechanics and robotics the motions are mostly studied from the dynamics point of view, i.e. the human body is seen as a dynamic system and the motions can be explained with theories from mechanics. This requires to create dynamic models of the human body.
Given the complexity of the human body, it is common to make assumptions and create reduced models to understand certain characteristic of the human motions, e.g. simple mechanical systems and/or multi-body models. In robotics, these characteristic have inspired the design and control of a series of machines that can be grouped as walking machines, including humanoid robots, exoskeletons and prosthesis.

Compliance —

The property of a material of undergoing elastic deformation or (of a gas) change in volume when subjected to an applied force. It is equal to the reciprocal of stiffness. (Oxford dictionary)

Humans are able to perform highly dynamic locomotion in any kind of terrains, ranging from simple straight walking on flat floors to walking on a slackline, also thanks to the compliance of the body, which has been proven to be a fundamental feature of walking and running in humans [41, 36].
Robots, instead, have lacked the feature of being compliant for many years since their first appearance in the early 20^{th} century. As recently human-robot interaction is a concern of the robotics community due to the aim of using robots in collaboration with humans, many researchers have put effort in building compliant robots, by introducing compliant actuators. These actuators have a series of advantages over the classic rigid actuators, apart from safer human-robot interaction, they also allow for energy saving and shock absorbtion.
Compliant actuators and compliance behaviours are desirable in wearable walking robotics systems such as exoskletons, prosthesis and orthoses, as these devices have the main goal of aiding humans to move, possibly in the most human-like way possible.
Therefore, in biomechanics and robotics many researchers have focused on the study of compliance during human locomotion, where compliance in terms of joint stiffness can be useful in the design of compliant actuators for walking machines.
Despite some studies have proved that stiffness changes during human movements [47], the stiffness of the elastic components of compliant actuators used in walking machines, e.g.

springs, is often assumed to be constant due to the difficulty of varying the stiffness, reason for which many researchers have tried to individuate possible constant stiffness at joint level during walking motions.
Humanoid robots design has been based on the traditional design approach that emphasizes the use of rigid actuation systems, but in recent developments compliance has been desirable in bipedal humanoid robots not only for safety and dependability reasons but also for energy saving. However, humanoid robots locomotion is a very difficult task to achieve, where the control of the postural stability during the execution of motions represents the main issue. The introduction of compliant actuators in recent humanoid robots increases further the complexity of the problem, reason for which controlling humanoid robots walking with compliant actuators is a challenging open research problem and the advantages of using these actuators are still in debate.

Objectives

This thesis has been carried out as part of the European Project *KoroiBot - Improving humanoid walking capabilities by human-inspired mathematical models, optimization and learning* [7], which had as main objective the improvement of existing humanoid robots walking capabilities, by learning also from a large variety of human walking motions.

In this thesis the objective is to study compliance aspects of locomotion in both humans and humanoid robots by applying common methodologies: rigid multi-body dynamics and optimal control. Computational models are created for both the human and humanoid robot in order to be used in optimal control problems to achieve analysis on human movements and to generate robot motions.

In the case of humans, the aim of many studies in biomechanics and robotics is to look for a possible way to recreate human-like walking motions with simple mechanisms such as linear springs, which cannot fully reproduce the complex dynamics of human locomotion. In our case instead, we aim at understanding the importance of stiffness modulation, by analysing compliance during human walking in terms of joint and bi-articular coupling stiffness. This is done by mapping human walking motions recorded from experiments on a multi-body dynamic model of the human body, in which we introduced springs with variable stiffness in the actuation of the leg joints that serve to compute the stiffness profiles for specific walking motions. In particular, we use optimal control to compute the profiles for walking on level ground, slope and stairs and analyze the influence of stiffness modulation on the reproducility of the walking gait.

In humanoid robots the introduction of compliant actuators increases the complexity of both modeling and controlling the robot, where the problem of walking is highly challenging. In this thesis we use the humanoid robot HeiCub, which is a reduced version of the iCub humanoid robot [75] of the Fondazione Istituto Italiano di Tecnologia (IIT). This robot is equipped with the compliant actuator Series Elastic Actuators (SEA) [87]. The objective is to create a whole-body dynamic model of the robot also including the SEA and to study the effect of using the SEA in the execution of different motions generated by means of optimal control. Experiments on the robot serve for validation purposes as well as to gather further details to be added to the model of the robot. As iCub has rarely performed walking motions prior this thesis, the problem of walking is addressed with a simpler approach by means of the table cart model, where the objective is to carry a thourough evaluation of the walking capabilities of the robot.

Contributions

Analysis of compliance in human walking - We carried out a thourough analysis of compliance in human leg joints and also bi-articular couplings between hip and knee during walking in different environments. We verified by means of optimal control and multi-body dynamic models that joint and bi-articular coupling stiffness profiles are not constant through the walking cycle. When these profiles are constrained to have very small modulations over a single or the whole walking cycle, then the original gait is more difficult to reconstruct, mainly in the case of constrained environments such as stair climbing. This means that a certain amount of stiffness modulation should be desirable in walking mechanisms if human-like gaits are to be reproduced, mainly in wearable systems.

Whole-body motion generation of the iCub considering SEA - We used optimal control to generate whole-body motions by considering the whole-body dynamics of the robot including also the SEA. This is the first time that this approach is used for an iCub robot. Both models with and without SEA are considered and used to generate squat motions and push recovery motions. In the first case the problem is simpler as the robot is assumed to have only three active joints on the sagittal plane, while in the push recovery the motion is whole-body, therefore all the degrees of freedom were considered.

Walking motions generation and performance evaluation of the iCub - Despite being a research platform that is spread over more than 30 institutes and research centers in the world since 2006, the iCub has shown little walking capabilities and rare are the documented walking experiments. We generated walking motions for the HeiCub using reduced models and created a framework that is perfectly applicable also to the full body iCub. This framework allowed the robot to perform walking on level ground, and for the first time for an iCub robot, also up and down slopes and up stairs. These experiments allowed to exploit the capabilities as well as limitations of the robot which were unknown, giving to the iCub community as well as to the scientific community an overview of the walking capabilities of the iCub with a systematic documentation of performance indicators.

Organisation

The thesis is divided in three fundamental parts: state of the art and the theoretical background, the study of compliance in human motions and the study of humanoid robot control and motion generation.

Part I - Preliminaries

In this part we first review the state of the art of human biomechanics and humanoid robots in **chapter 1**, with more detail on the compliance aspects of both, then the fundamental theories and methods on which the whole work of the thesis is based is illustrated in **chapter 2**. In this chapter, the rigid multi-body dynamics with the notations as used in all next chapters is briefly described, including model assumptions on contacts and impacts. Then the optimal control theory is shown with particular attention to the formulation of multiphase optimal control problems, followed by an overview on the numerical methods for solving nonlinear optimal control problems, with a more detailed explanation of the one adopted in this thesis which is the direct multiple shooting method.

Part II - Study of compliance in human locomotion with computational methods

In this part the experiments carried out to measure human walking motions are described in **chapter 3**, where the computational model details, walking phases description and the process of mapping recorded motions to the described models are illustrated. In **chapter 4** specific walking data are used to compute stiffness profiles of human leg joints and bi-articular couplings using optimal control. Given that these profiles have high modulations, a further analysis on the influence of the modulation on the walking gait is carried out. In **chapter 5** a brief conclusion on the analysis of compliance in human motions is done.

Part III - Modeling and control of the compliant humanoid robot iCub

The focus of this part is on the humanoid robot iCub, mainly on the reduced version of it, HeiCub. In **chapter 6** the robot is described including its hardware, software architecture, kinematic and dynamic models, where a particular attention is given to the SEA. In the next chapters, optimal control is used to generate motions including the SEA. In **chapter 7** the squat motion is generated using a reduced model of the robot and a set of objective functions. In **chapter 8** the problem of recovering from external perturbations is formulated using optimal control. In **chapter 9** how the first walking motions were generated on the iCub using the table cart model is illustrated with an evaluation of the performances of the robot. A brief conclusion is made in **chapter 10**.

Part I

Preliminaries

1 Human and humanoid robot locomotion

The term *locomotion* refers to multiple way of *moving* around, but in the specific case of this thesis, we use it as synonym of *walking*. And with walking we refer to the motion where at least one of the feet has contact with the ground at each time instant, which differs from running where there is a certain amount of time during which there is no contact with the ground.
Walking is an ability that most of humans acquire when reaching one year of their life, after which it becomes a motion that is mastered to cope with many different situations. For this reason it is highly interesting to understand how do humans walk and how do they adapt to the several surroundings.
Humanoid robots, instead, are still far from have "learned to walk" like humans, despite they have been actively researched for more than 40 years already. But recent developments have brought significant improvements in humanoid robots locomotion.

In this chapter we first make an overview on how humans can be modeled for biomechanics gait analysis, then we show a brief history of their counterparts in robotics, followed by a brief description of how can they be modeled and controlled. A deeper insights on the compliance aspects of both is taken, which is the main topic of this thesis.

1.1 Human biomechanics

In biomechanics the analysis of human movements consists in several steps starting from the collection of data via measurements to the analysis of these data under certain model assumptions.
In the case of gait analysis, we want to understand the underlying kinematic and dynamic properties of the movements, where a model of the human body is necessary. There are different models existing in literature, ranging from simple mechanical models such as the pendulum with one or two legs, to complex articulated bodies and sophisticated muscle systems.
In the particular context of this thesis, we choose to use an articulated multi-body system to model the human body based on data from biomechanics, i.e. the model has to be defined in terms of segment lengths, position of the joints, segment masses, inertial parameters, etc. In this case the body parts are models as rigid bodies, therefore soft tissues such as skin, muscles, tendons, are not taking into account.

As humans are all different from each other, it is difficult to create a generalized model that can be used for any study. However, it is also clearly highly difficult to create a subject specific model as we are not able to measure the kinematic and dynamic properties of body parts of a person, unless this person can be cut in pieces and reassembled.
Therefore, it is common to use biomechanics data from literature where kinematic and dynamic descriptions of human bodies can be found and are based on measurements taken from a high number of subjects. The most notable and largely used biomechanics data in literature are the data of Winter [123] and the adjustments to Zatriorsky-Seluyanov's data of DeLeva [23], which target mainly healthy adult subjects, where there exist also recent data adjustments for elderly people [45].

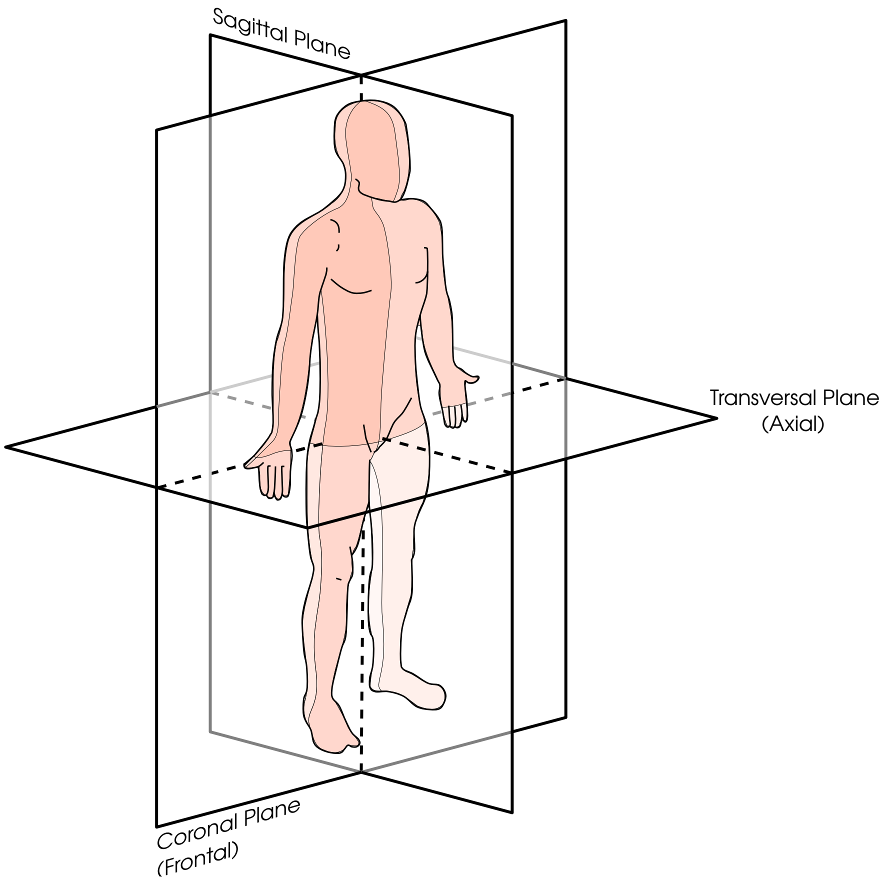


Figure 1.1: Definition of anatomical planes.

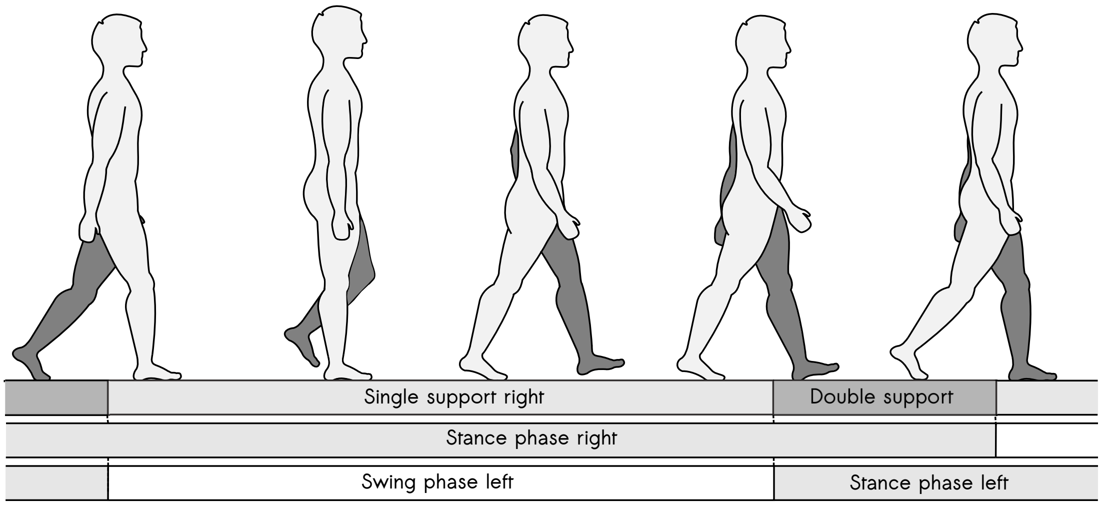


Figure 1.2: Main phases of a single walking step on flat floor.

To properly describe human movements, the *anatomical planes* are defined as in Fig. 1.1. The anatomical planes divide the human body in half from three directions perpendicular to each other. In particular, the *sagittal plane* is the one which separates the body into the left and right parts that are usually assumed to be symmetric. The sagittal plane is also the one that is the

most interesting for the analysis of walking motions.
Movements can be described in several ways, for example by segmenting a complex motion into simpler ones. In the particular case of locomotion, the sequence pf walking can be divided into *phases*, which are commonly identified with motions happening on the sagittal plane, consisting mainly in the double and single support phases according the alternation of swing and stance of the legs, as in Fig. 1.2. More detailed phases can be identified for motion analysis purposes, and can vary according to the type of model chosen to describe the motion. The detailed phases used in this thesis as per our model choices are explained in Section 3.5.

1.1.1 Compliant aspects

The body movements are generated by muscles, tendons and joints, it is therefore of high interest a proper model of these parts. However, there are hundres of muscles and tendons in just a single part of the human body and joints are usually very complex structures to be modeled with single joint systems, where another factor that contributes to the challenge is the compliance of these elements.
As humans are also composed of soft tissues, the movements are characterized by compliance. Studies in biomechanics, motor control and robotics have shown that compliance is a fundamental feature in locomotion, and the stiffness of the legs and joints modulate during the execution of the movements [47].

Compliance was demonstrated to be helpful in reproducing human-like locomotion on level ground and rough terrains by means of simple spring loaded inverted pendulum models (SLIP) [41, 71].
Compliance at joint level has been shown to play a central role in locomotion [65], where many researchers addressed the analysis of joint stiffness by studying the torque angle relationship of the leg joints [120, 121]. Some studies have carried out analysis on hip, knee and ankle dynamic joint stiffness both in walking in [102, 103, 101] and running in [42], where bi-articular muscles were also included as coupling between these joints [52, 77].

Despite many researchers tried to focus on the identification of possible constant stiffness to be used in real-life applications such as prosthesis, exoskeletons and walking robots, humans modulate the stiffness by co-contraction of agonist and antagonistic muscles acting on the joints during the execution of movements to cope with surrounding environments [36, 47].

Simple models such as the SLIP are very popular in biomechanics as they can help in understanding the general dynamic behaviour of locomotion in an easy way, but they do not allow to gather insights on detailed dynamics of the human body, e.g. interactions between the segments, torques required at joint level etc. Highly complex musculoskeletal system exist, but require considerable computational power to be effectively used to model whole-body motions such as walking.
A reasonable alternative consists in the articulated multi-body model as the one used in [34], with each segment of the human body modeled as rigid body, connected to each other by means of simple rotational joints. In this case the model was used to generate dynamically feasible motions, while a similar model was used to carry out the study of joint stiffness modulation during running motions in [77], by introducing spring-damper models. We also choose articulated rigid multi-body models to carry out an analysis on joint stiffness modulation during walking in different enviroments, as will be illustrated in Chapter 3.
In many works inverse dynamics is used to compute the joint torques in order to analyze joint stiffness [102, 103, 101] as data at joint level that can be measured from human movements

typically consists in joint angles, and velocities and accelerations can be obtained with differentiation. Instead, we choose to use forward dynamics with an optimal control formulation, where the dynamics of the system is evaluated at each time instant and joint angles, velocities, accelerations and torques are optimized at the same time, allowing also to model the hybrid nature of the walking dynamics, where the dynamics changes for different contacts between th system and the enviroment.
This method has been used to generate walking motions in [34] and to study human push recovery motions in [97]. In this thesis it is used for the first time to analyze the role of compliance in human walking considering the whole-body motions and dynamics.

1.2 State of the art humanoid robots

When people think about humanoid robots, the first ones they imagine are typically those from science fictions. These robots are depicted as being able to do any sort of things that humans can do (and even better) and can be disguised as humans. Therefore, when talking about *humanoid robots* it is difficult for many people to understand or even imagine how real state of the art humanoid robots work and look like, i.e. far behind what is depicted in science fictions.
The reasons of creating human shaped robots, i.e. humanoid robots, is often topic of discussion, ranging from whether it is an optimal mechanical design choice to ethical issues. The choice of building machines that look like humans are multiple. These robots are thought to be used in environments originally designed for and populate by humans, often in cooperation with humans. The main sites where they are supposed to work in are factories, public spaces like hospitals, homes etc. Therefore the idea is that it is easier to design robots that can work in environments shaped for humans rather than change the environments to accomodate robots, where the bipedal choice over designs like wheels target situations like stairs and similar surroundings. Humanoid robots are also seen as possible replacement for humans in disaster situations such as the Fukushima Daiichi nuclear disaster (Japan, 2011), which highly endangers human health.

An “ideal” humanoid robot should be an embodiment of different technologies and disciplines, as it is a mechanical structure that should move intelligently. This is however highly difficult to achieve with current resources, therefore many of the existing platforms have focused on particular tasks, e.g. some can walk in challenging and unpredictable environments, some have sophisticated manipulation skills, some have astonishing artificial intelligence (AI), but there’s no machine that is able to do all of this at once.

The first bipedal humanoid robots appeared in the early 70s in Japan, when in 1973 WABOT-I [59] was revealed. This robot was able to perform most of the tasks “expected” from a humanoid robot, including walking, manipulating, object and voice recognition, though at a very preliminary stage. However, it gave a start to the following generation of bipedal humanoid robots. In 1984 the WABOT-II was released [108] and in 1996 Honda revealed the robot P2, result of a confidential project which development led to one of the now most known humanoid robots, ASIMO [95]. As for 2016, the latest generation of ASIMO is able to interact with humans, handle objects, walk in different environments and run. In 1998 the Japan Ministry of Economy, Trade and Industries started the Humanoid Robotics Project (HRP) [44], which resulted in the production of the HRP series by Kawada Industries. The robot HRP-2 of the series is used in many research institutes to conduct studies on walking and dynamic motions [100, 116].

In the 80s Marc Raibert started working on hopping robots that are able to balance on a single leg by continuosly hopping, which later led to a bipedal 3D hopper [92]. These robot do not have any cognition ability and target purely the study of dynamic stability and locomotion. Raibert later started the company Boston Dynamics which built several legged robots that are able to perform locomotion in highly rough terrains. With the support of the Defense Advance Research Projects Agency (DARPA), Boston Dynamics developed the Atlas humanoid robot to be used in disaster sites. The Atlas was the main platform used in the DARPA Robotics Challenge (DRC) [4] in 2015, a competition focused on disaster and emergency response. The competition consisted in going through a disaster site parcour in which the most advanced humanoid robots participated. During the competition eight tasks were set and one hour time was given to complete them. However, only 3 out of the 24 teams who participated were able to finish the track in time. This competition served from one side to develop new hardware and control architectures for robotics, from the other side it showed how humanoid robots are still far from being able to perform what humans are able to, not to mention to what robots are able to in science fiction.

While the aforementioned robots are all fully actuated and controlled, a class of bipedal robots that have shown efficient walking are the passive walkers, which have none or little actuation. McGeer showed the first passive walker without any actuation in the late 80s, which was able to walk down a slope [74]. In 2004 Wisse presented a passive walker which has an upper body [124]. The advantage of these robots is that they are highly energy efficient, being without actuation, but real life applications are difficult given that they are able to walk only in regular environments like a slope or on level ground with additional hip actuation.

A more recent development in humanoid robots follows the advances of actuators design. As robots started to come out from the factories and enter human populated environments, researchers and engineers started to give more importance to the compliant aspects by designing compliant actuators, which are also inspired by human biomechanics and can ensure safer human-robot interaction.
Humanoid robots using compliant actuators started to appear, such as the Lucy robot using pneumatic artificial muscles [115], the Roboray using tendon driven actuators [62], and M2V2 [88], the COMAN [19] and WALK-MAN [113] using Series Elastic Actuators (SEA) [87]. In particular, the COMAN humanoid robot showed to be able to perform stable walking with SEA [70, 79, 21]. Besides from safer human-robot interaction, the introduction of compliant actuators has also the aim of absorbing impacts, generating more human-like movements and energy efficiency.
Another class of humanoid robots that target compliance aspects are the torque controlled robots. Classic robots use rigid actuators which are usually position controlled and have high position precision. Torque control allows to introduce a certain compliance in the motion that can cope with external perturbations and allow for safer interactions as well. One of the most famous torque controlled compliant robots is the KUKA Lightweight robotic arm [15], which design was also used on the torque controlled humanoid from the German Aerospace Center (DLR), ToRo [84]. The iCub humanoid robot [75], which has inherited the design of the legs from the COMAN in its latest version, has also proved to be able to perform whole-body dynamic balancing with torque control [83].

1.2.1 Stability criteria

Stability is one of the most important desired features in humanoid robots and represents the basic problem of bipedal robots. Humans are very effective in keeping stability, while robots need to be accurately controlled such that balance can be maintained during the execution of motions.

Passive walkers walk in a stable way with cyclic stability, i.e. using the limit cycle method which is based on analyzing eigenvalues of the Poincaré return maps [40]. This method, as the passive walkers, find little application in real life situations.

In 1972 Vukobratović and Stephanenko defined the Zero Moment Point (ZMP) [118] as an indicator of stability of bipedal walking, which definition given in [117] is: *ZMP is defined as that point on the ground at which the net moment of the inertial forces and the gravity forces has no component along the horizontal axes.*
For a robot to be dynamically stable, the ZMP has to exist, i.e. inside the support polygon of the robot, which is the smallest convex set including all contact points between the robot and the ground. The ZMP always lies inside the support polygon and in this case it corresponds to the center of pressure (CoP). It also corresponds to the Foot Rotation Indicator (FRI) when approaching the edge of the support polygon, which is when the robot starts to rotate on the edge of the foot.
Since its introduction, the ZMP has been the most popular stability criterion in bipedal locomotion, used in most of the existing motion generation algorithms and controls.

A more recent criterion is based on the Capture Point (CP) [90], also called the Divergent Component of Motion (DCM) [27], commonly used to perform push recoveries of humanoid robots. The capture point is defined as the point on the ground on which the robot has to step in order to be stable within one step, i.e. stability is ensured if the center of mass is located over the stance foot and the horizontal velocity is zero when the robot comes to a rest. The set of capture points form the capture region. When this region is outside the kinematically reachable area of the robot, then the robot is not able to recover stability within one step.

1.2.2 Motion generation methods

The generation of motions for robots is a problem that dates back to the first industrial manipulators. The general workflow consists in setting up a certain goal/task at a higher level and the generated motion happens at lower levels aiming at fulfilling this goal. With increasing complexity of robots and also requirements, many methods have been studied and can be categorised in different classes. Humanoid robot walking motions can be generated with motion remapping or with model based methods.

Motion remapping consists in mapping trajectories of human walking on robots. In this case the difference in the mechanics needs to be taken into account. A proper scaling of the motion has to be performed first, then the motion can be either divided into motion primitives to be concatenated to generate complex motions, or fully transferred to the robot. Robot specific constraints including physical constraints and stability criteria have to be taken into account. An example of this approach was applied to the humanoid robot HRP-2, where movement primitives learned from humans were transferred to the robot [80].

In the case of model based methods, the reduced models are popular for online control, where famous examples are the 3D linear inverted pendulum (3D-LIPM)[56], the table cart [57] and the spring loaded inverted pendulum (SLIP) [41, 71]. Stability criteria that are mostly used are typically the ZMP and more recently, the Capture Point. The whole body posture is adjusted to fulfill the chosen criterion.

With reduced models certain desired properties and targets are specified, e.g. center of mass (CoM) and feet positions, and the dynamics of the reduced models are used to compute stable trajectories for certain specified points. To obtain the whole-body robot motion these specific points are mapped onto points on the body of the robot, from which the whole-body motion is retrieved such that the robot can follow the trajectories of these points. The mismatch between the reduced model and the whole-body model dynamics are compensated with online adjustments.
When whole-body models are used, optimization represents an essential tool in the motion generation and control. It was proposed as a tool to generate robot motions [67] and optimal control has been used to succesfully generate stable fast dynamic motions of bipedal robots [78], challenging walking motions with template models [17], highly dynamic whole-body overstep motions with big obstacles for humanoid robots [63], and complex humanoid robot walking in combination with optimization generated motion primitives [18].

1.2.3 Control of compliant robots

Compliance has been seen as an *undesired* feature in robotics for many years, as it introduces disturbances in high precision control systems. However it has been deliberately introduced in recent robotic systems, under the form of *active control*, with the introduction of *passive elements* or a combination of the two.
In the case of active control, the flexibility is introduced at control level by controlling virtual springs [72], where with passive elements one or more elastic elements are introduced in the links or in the joints by means of compliant actuators.
Compliant actuators can have fixed or variable stiffness. In the former a spring like element is introduced, and the most famous example is the SEA [87] which has a spring in series with the actuator.
Variable stiffness actuators are still at a research level where many different designs have been proposed. Examples are the Mechanically Adjustable Compliance and Controllable Equilibrium Position Actuator (MACCEPA) [114], which changes the spring preload and was designed of a bipedal humanoid robot and later also used for orthosis with a change in the design [76], the Variable Stiffness Actuator (VSA) [112] based on antagonistic springs with antagonistic motors, the Actuator with Mechanically Adjustable Series Compliance (AMASC) [51], based on antagonistic springs with independent motors and designed to be used on a 2D bipedal running robot.

The control of compliant actuators has been studied and developed at actuation level, but when incorporating them into robots, in our interest mainly in bipedal robots, a control of the whole-body behaviour taking into account the joint compliance is necessary. There are only few existing walking humanoid robots using compliant actuators and most of them employ control schemes that generate reference torques with an inverse dynamics approach, then a low-level torque tracking control is implemented on the robot as independent joint control using classic PD/PID (Proportional, Integral and Derivative) schemes, i.e. each joint tries to follow as closely as possible the reference torque by using the tracking error as feedback in the PD/PID loop, this control is generally independent from the high-level control that generates the references.

The use of compliant actuators with passive elements and mainly the ones with variable stiffness is still very limited in bipedal machines, as the size of these actuators is still too big to be embedded in human-size robots and the control both at joint and whole-body level is way more complex than robots with classic actuators.

Among the compliant humanoid robots metioned before, the robot M2V2 uses the 3D-LIPM and and capture point combined with a state machine to generate reference trajectories for walking which are given as inputs to the inverse dynamics to generate for reference joint torques [91], as the SEA of the robot are force controlled. The humanoid robot Lucy uses Dynamic Balancing Force Control (DBFC) [107] combined with the Virtual Mode Control (VMC) [89] which takes as input desired contact forces and uses whole-body models to generate joint torques. On the humanoid robot COMAN different approaches were implemented, such as the feedback control on the desired center of mass trajectories [70] and the use of Kinematic Motion Primitives (KPMs) [79], which are a set of invariant waveforms that can be combined to describe complex motions.

In the latest version of the openly available humanoid robot iCub [75], SEA have been introduced in the knee and ankle pitch joints, following the recent compliant robots developments. The spring of the SEA of the iCub can be unmounted to use the robot also with classic rigid actuation, allowing to analyze motions with both types of actuators. The introduction of the SEA were proved to improve balance recovery of the robot while standing [25], however they have never been used to achieve other motions with the iCub.
HeiCub is an iCub without arms and head, and it was delivered to Heidelberg University by the end of 2014. It is the humanoid robot used in this thesis, where we are interested in analyzing the use of SEA in several motions. Unlike many approaches, we want to consider the whole-body dynamics of the robot, where the robot is modeled as a rigid multi-body system. We use forward dynamics and optimal control to generate optimal motions with and without SEA under certain assumptions and contact conditions as will be described in chapters 7 and 8.

2 Methods and theory

In the scope of this thesis we aim at using dynamic models in optimal control frameworks for both the cases of studying the role of compliance in humans and humanoid robots, in particular the humanoid robot HeiCub. It is therefore of primary importance the proper description of the model dynamics and the optimal control method with clear definition of the conventions used.
In this chapter, we first briefly illustrate the rigid body dynamics theory and then the optimal control theory with particular focus on the direct multiple shooting method which is the one used in the rest of the thesis.

2.1 Rigid multi-body dynamics

A generic multi-body system is as shown in Fig. 2.1, where a certain number of rigid bodies are connected with joints. Joints can be complex structures that connect more than one body togethere, but in this thesis we always consider the joint as a simple structure that represents either a rotation or a translation along one axis, in the particular case of human models and the HeiCub model, only rotational joints will be used.

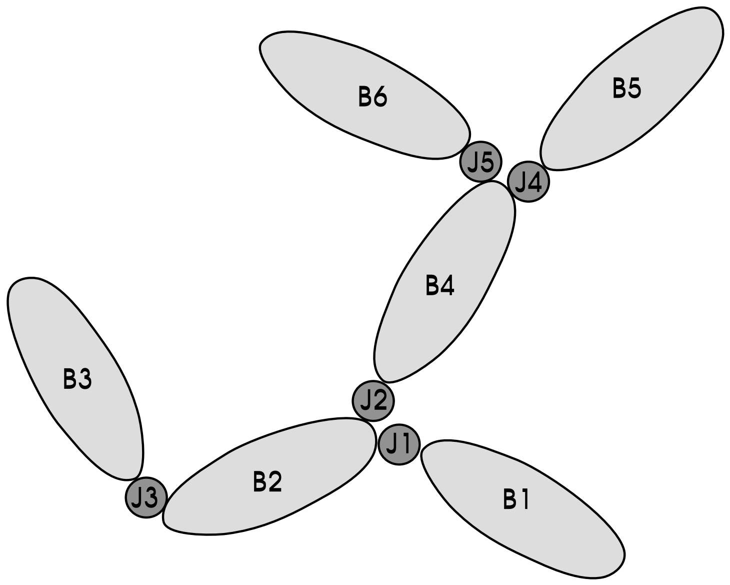


Figure 2.1: A generic multi-body model with 6 bodies connected by 5 joints.

A rigid multi-body system can be attached to a fixed base, like the industrial robots, or have a floating base, i.e. free to move in the space, like mobile robots, legged robots including bipedal humanoid robots.
Such a system can move with a certain number of *degrees of freedom* (DOF), which corresponds to the number of independent parameters that can describe the configuration of the system. These parameters are referred to as *generalized coordinates*. In the case of fixed base industrial manipulators, the number of degrees of freedom corresponds to the number of actuated

joints, but this is not the case of floating base systems such as humanoid robots, where additional degrees of freedom can be introduced to describe the configuration of the system in the space. In this case, the number of controllable degrees of freedom is lower than the actual ones, therefore the system is *redundant*.

In this thesis both human and humanoid robot are modeled as floating base rigid multi-body dynamic systems. Both are highly complex systems to model, mostly when interactions with external environments occur. As the main problem treated in this thesis concerns locomotion, these interactions need to be handled.
Such interactions are modeled with external contacts. When one or more points on the multi-body system collides with an external environment, which can be the floor, an object or another multi-body system, an *impact* occurs. The impact can be followed by a contact phase in which one or more points of the multi-body system keep the contact with the other object, we refer to these contacts as *external contacts*.

A certain number of assumptions are usually introduced to simplify the dynamics descriptions, mainly in the case of humans, but also in the case of humanoid robots.
In the rest of this section as well as of this thesis, the following assumptions are made:

- Both humans and robots are treated as rigid multi-body systems,
- External contacts are perfectly rigid, i.e. no penetration,
- Impacts are instantaneous and inelastic.

All the theory presented in this section are based on these assumptions and apply for both the human model described in chapter 3 and the robot model in chapter 6.

Rigid body dynamics literature is widely spreaded and well established, in this section we report only the essential parts to clarify the notations and assumptions used in this thesis. For the basic rigid body dynamics, it is possible to refer to textbooks such as [61] or [104].

The dynamics of a rigid multi-body system can be described using the generalized coordinates $\mathbf{q} \in \mathbb{R}^{n_{dof}}$, where n_{dof} is the number of degrees of freedom of the robot. The *inverse dynamics* form is defined as follows:

$$\boldsymbol{\tau} = ID(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}) \tag{2.1}$$

in which the joint torques $\boldsymbol{\tau}$ are computed with known joint positions $\mathbf{q}$, velocities $\dot{\mathbf{q}}$ and accelerations $\ddot{\mathbf{q}}$. This form is also often referred to as the *dynamic model* [61].
The problem of finding the joint accelerations $\ddot{\mathbf{q}}$ is addressed as *direct dynamic model* or *forward dynamics*:

$$\ddot{\mathbf{q}} = FD(\mathbf{q}, \dot{\mathbf{q}}, \boldsymbol{\tau}) \tag{2.2}$$

which is the form that will be used for the rest of this thesis in combination with optimal control.

For rigid multi-body systems, the dynamics can be described by means of the following equation of motion:

$$\mathbf{H}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) = \boldsymbol{\tau}. \tag{2.3}$$

In this equation, the matrix $\mathbf{H}(\mathbf{q})$ is a square matrix and represents the joint space inertia matrix and $\boldsymbol{\tau}$ are the torques applied to the joints.
The term $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$ is the vector of all nonlinear terms including Coriolis, centrifugal and gravitational forces:

$$\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) = \mathbf{A}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{Q}(\mathbf{q}) \tag{2.4}$$

where $\mathbf{A}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}}$ is the Coriolis and centrifugal forces and $\mathbf{Q}(\mathbf{q})$ the gravity forces.
For floating base systems it is common to introduce additional *virtual* joints in the system such that the formulation as per classic robotics as in Eq. (2.3) can still be used. The number of degrees of freedom of the floating base is usually 3 or 6, according to if the system is moving on a plane (2D) or in the space (3D). When moving on the plane the floating base consists of two translations and one rotation, while in the space of three translations and 3 rotations (e.g. represented with Euler Angles).
In either case, the joint torques can be generated only for actually *actuated* joints, and not for the virtual ones introduced for the floating based. Therefore, the vector of joint torques $\boldsymbol{\tau}$ can also be written as $\mathbf{S}^T\boldsymbol{\tau}$, where $\mathbf{S}$ is a matrix that takes into account the actuation.
For simplicity of notation, in this thesis we assume the following:

$$\boldsymbol{\tau} = \begin{bmatrix} \mathbf{0} \\ \boldsymbol{\tau}_A \end{bmatrix} \tag{2.5}$$

where the $\mathbf{0} \in \mathbb{R}^{n_{fb}}$, being n_{fb} the number of degrees of freedom of the floating base and $\boldsymbol{\tau}_A \in \mathbb{R}^{n_{actuated}}$, being $n_{actuated}$ the number of actuated degrees of freedom, therefore $n_{fb} + n_{actuated} = n_{dof}$ the total number of degrees of freedom.
Both inverse and forward dynamics can be solved using Eq. (2.3). However, long computation times might be required for complex systems, in particular in the case of forward dynamics, therefore recursive numerical methods are often used to solve the system dynamics.

Walking can be defined as a *hybrid dynamics system* as the dynamics switches according to contact conditions, i.e. hybrid and non-smooth dynamics. Additional constraints to Eq. (2.3) need to be taken into account, such as the presence of the floating base, the interaction of the different segments as well as contact and impact forces. In the following, the dynamics taking into account contacts and impacts is described.

2.1.1 Contacts

Under the stated assumptions, external contacts can be described as a set of position constraints with:

$$\mathbf{g}(\mathbf{q}) = \mathbf{0} \tag{2.6}$$

where $\mathbf{g} : \mathbb{R}^{n_{dof}} \rightarrow \mathbb{R}^m$, with m being the number of constraint equations. In order for the system to be solvable, m must be smaller than n_{dof}.
When external contacts are involved, Eq. (2.3) can be reformulated using Lagrange multipliers to take them into account:

$$\mathbf{H}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) = \boldsymbol{\tau} + \mathbf{G}^T(\mathbf{q})\boldsymbol{\lambda} \tag{2.7}$$

where $\mathbf{G}$ is the contact Jacobian resulting from $\mathbf{G} = (\partial\mathbf{g}/\partial\mathbf{q})$ and $\boldsymbol{\lambda}$ the vector of contact forces.
An additional set of equations for the contact constraints is therefore derived from Eq. (2.6),

i.e. by deriving the position constraints twice:

$$\mathbf{G}(\mathbf{q})\ddot{\mathbf{q}} + \dot{\mathbf{G}}(\mathbf{q})\dot{\mathbf{q}} = \mathbf{0}. \tag{2.8}$$

Combining Eq. (2.7) and (2.8), the dynamics equations can be written as linear system for unknowns $\ddot{\mathbf{q}}$ and $\boldsymbol{\lambda}$:

$$\begin{bmatrix} \mathbf{H}(\mathbf{q}) & \mathbf{G}^T(\mathbf{q}) \\ \mathbf{G}(\mathbf{q}) & \mathbf{0} \end{bmatrix} \begin{bmatrix} \ddot{\mathbf{q}} \\ -\boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \tau - \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) \\ -\gamma(\mathbf{q}, \dot{\mathbf{q}}) \end{bmatrix} \tag{2.9}$$

where $\gamma = \dot{\mathbf{G}}(\mathbf{q})\dot{\mathbf{q}}$ is the generalized acceleration independent part of the contact point accelerations.

2.1.2 Impacts

The assumption is that the transition from the collison of the system to the contact phase is instantaneous, i.e. an impact occurs where the generalized velocities before and after the impact changes. The equations describing impact dynamics are obtained by integrating over a time singleton Eq. (2.7) and (2.8) as per [122], resulting in:

$$\mathbf{H}(\mathbf{q})\dot{\mathbf{q}}^+ - \mathbf{H}(\mathbf{q})\dot{\mathbf{q}}^- = \mathbf{G}(\mathbf{q})^T \boldsymbol{\Lambda} \tag{2.10}$$

$$\mathbf{G}(\mathbf{q})\dot{\mathbf{q}}^+ + e\mathbf{G}(\mathbf{q})\dot{\mathbf{q}}^- = 0 \tag{2.11}$$

where $\dot{\mathbf{q}}^-$ and $\dot{\mathbf{q}}^+$ are the generalized velocities before and after the impact, $\boldsymbol{\Lambda}$ the impulses at each constraint and $e \in [0, 1]$ is the coefficient of restitution.
The above equations can be combined in a linear system for unknown $\dot{\mathbf{q}}^+$ and $\boldsymbol{\Lambda}$:

$$\begin{bmatrix} \mathbf{H}(\mathbf{q}) & \mathbf{G}(\mathbf{q})^T \\ \mathbf{G}(\mathbf{q}) & \mathbf{0} \end{bmatrix} \begin{bmatrix} \dot{\mathbf{q}}^+ \\ -\boldsymbol{\Lambda} \end{bmatrix} = \begin{bmatrix} \mathbf{H}(\mathbf{q})\dot{\mathbf{q}}^- \\ -e\mathbf{G}(\mathbf{q})\dot{\mathbf{q}}^- \end{bmatrix}. \tag{2.12}$$

As impacts are assumed to be perfectly inelastic, $e = 0$ is imposed in the rest of this thesis. This condition enforces also that the contact points eventually stick to the contact surfaces after impact, this might be not true or even undesirable for certain applications, however a proper way to define impulsive forces is still an open research problem, reason for which inelastic impacts are often assumed.

As we will see in the upcoming chapters, each phase of dynamic locomotion is described with different phases characterized by different constraint sets consisting in sets of contact points and/or impacts. Each of these phases is described with a different set of equations as in Eq. (2.9) and (2.12) described in this section.

The dynamics described in this section can be solved with many available algorithms, notable ones are the recursive algorithms described in R. Featherstone's book [28], which have been implemented in several libraries. In this thesis we use the C++ Implementation of Martin Felis, the Rigid Body Dynamics Library [33, 30]. Implementation details of the human and robot models using this library are described in the Appendix A.

2.2 Optimal Control Problem (OCP) formulation

Optimization has become an essential tool in many disciplines, from economics to engineering. A system is modeled with variables, objectives and constraints and the goal is to find the set of variables that optimize the specified objective within the constraints. Control design of a system is a trial and error process to find parameters for the system suitable for certain processes.
The combination of the two is the optimal control theory: optimal control problems (OCP) consist in solving optimization problems for time dependent processes where states, controls and parameters are simultaneously computed with respect to certain objective functions and set of constraints.
In this thesis, optimal control is used to analyze human movements as well as to generate motions for the humanoid robot HeiCub, where we treat both as dynamic systems that can be described with the rigid multi-body dynamics illustrated in the previous section. Due to the complexity of the dynamics, the optimal control problems are nonlinear and require numerical methods to be solved.

In this section we will first illustrate numerical methods to solve nonlinear optimal control problems, then in particular the direct multiple shooting method is illustrated with more details, with reference to how it was implemented in the software package MUSCOD-II [68], which is used in this thesis to carry out the computations.

2.2.1 Formulation and numerical solutions of optimal control problems

A generic nonlinear optimal control problem can be defined as follows:

$$\min_{\mathbf{x},\mathbf{u},\mathbf{p},T} \quad \int_0^T \Phi_L(t,\mathbf{x}(t),\mathbf{u}(t),\mathbf{p})dt + \Phi_M(T,\mathbf{x}(T),\mathbf{p}) \tag{2.13}$$

subject to:

$$\begin{array}{rcl} \dot{\mathbf{x}}(t) & = & \mathbf{f}(t,\mathbf{x}(t),\mathbf{u}(t),\mathbf{p}),\ t \in [0,T] \\ \mathbf{g}(t,\mathbf{x}(t),\mathbf{u}(t),\mathbf{p}) & \geqslant & \mathbf{0} \\ \mathbf{r}^{eq}(\mathbf{x}(t_0),..,\mathbf{x}(T),\mathbf{p}) & = & \mathbf{0} \\ \mathbf{r}^{ineq}(\mathbf{x}(t_0),..,\mathbf{x}(T),\mathbf{p}) & \geqslant & \mathbf{0} \end{array} . \tag{2.14}$$

In this formulation, the objective function can be of Lagrangian type $\Phi_L(\cdot)$ or a Mayer type $\Phi_M(\cdot)$, or a combination of the two, in this case it is commonly called Bolza type. The time horizon is represented by T, over which the lagrangian type objective function is integrated. The objective function is minimized with respect to the states $\mathbf{x}(t) \in \mathbb{R}^{n_{xd}}$, the controls $\mathbf{u}(t) \in \mathbb{R}^{n_u}$ and the model parameters $\mathbf{p} \in \mathbb{R}^{n_p}$ (if any).
The right hand side of $\dot{\mathbf{x}}(\cdot) = \mathbf{f}(\cdot)$ is a set of ordinary differential equations (ODE) describing the dynamics of the considered system, such as described in the previous section as in Eq. (2.9). The continuous path constraints are formulated with $\mathbf{g}(\cdot)$ and boundary and point constraints are formulated with $\mathbf{r}(\cdot)$, where $\mathbf{r}^{eq}$ are the equality constraints and $\mathbf{r}^{ineq}$ are the inequality constraints.

In this thesis we treat mainly the problem of walking, which is a multiphase problem with discontinuous dynamics which is described using rigid multi-body dynamics as in the previous section. The same applies to motion generation problems involving different contacts. Therefore multiphase optimal control problems need to be formulated for these classes of problems. Modifications are introduced to the single phase optimal control problem as illustrated above for the multiphase formulation:

$$\min_{\mathbf{x},\mathbf{u},\mathbf{p},s_0,s_1,\dots,s_{n_{ph}}} \sum_{j=0}^{n_{ph}-1} \int_{s_j}^{s_{j+1}} \Phi_L(t,\mathbf{x}(t),\mathbf{u}(t),\mathbf{p})dt + \Phi_M(s_{j+1},\mathbf{x}(s_{j+1}),\mathbf{p}) \tag{2.15}$$

subject to:

$$\begin{aligned} \dot{\mathbf{x}}(t) &= \mathbf{f}_j(t,\mathbf{x}(t),\mathbf{u}(t),\mathbf{p}), \\ & \quad t \in [s_j, s_{j+1}] \\ & \quad j = 0,\dots,n_{ph}-1, \\ & \quad s_0 = 0, s_{n_{ph}} = T \\ \mathbf{x}(s_j^+) &= \widetilde{\mathbf{J}}_j(\mathbf{x}(s_j^-),\mathbf{p}),\ j = 0,..,n_{ph}-1 \\ g_j(t,\mathbf{x}(t),\mathbf{u}(t),\mathbf{p}) &\geqslant \mathbf{0},\ t \in [s_j, s_{j+1}] \\ r^{eq}(\mathbf{x}(0),..,\mathbf{x}(T),\mathbf{p}) &= \mathbf{0} \\ r^{ineq}(\mathbf{x}(0),..,\mathbf{x}(T),\mathbf{p}) &\geqslant \mathbf{0} \end{aligned} \tag{2.16}$$

where the objective function, differential equations $\mathbf{f}_j(\cdot)$ and constraints can be defined for each phase $j = 0,...,n_{ph}-1$, with n_{ph} the total number of phases. The transition function between two subsequent phases is defined by $\widetilde{\mathbf{J}}_j(\mathbf{x}(s_j^-),\mathbf{p})$, where s_j is the time boundary of the phase j, s_j^+ and s_j^- are the time before and after the transition.

In the case of dynamic systems that will be studied in this thesis, the transition functions correspond to the impact dynamics and the continuous phases of different phases correspond to set of dynamic equations with different contact constraints.

Among the existing methods for solving nonlinear optimal control problems numerically, we can identify three main methods: the *dynamic programming*, *indirect* methods and *direct* methods.

Dynamic programming consists in solving the problem by finding the value function that satisfies the Hamilton-Jacobi-Bellman equation (HJB) which is a partial differential equation (PDE), by discretizing the state and control spaces [14]. The optimization process is based on the principle that the remainder of an optimal trajectory is also optimal, therefore the solution is found iteratively by minimizing the sum of the costs for the next time step and the cost to go from the sampling point reached in the next time step to the goal, where the state sequence is also optimized. The disadvantage of dynamic programming is that the complexity increases exponentially with the number of states and controls, which means that for complex dynamic systems as the ones we treat in this thesis this method is not applicable in a feasible way.

In the so called indirect methods the problem is first transformed into a boundary value problem by means of the Pontryagin's Maximum principle [86], then this boundary value problem is numerically solved, e.g. with Newton method. Indirect shooting methods were also proposed to solve problems under this formulation. This method provides highly accurate solutions but initialisation is non intuitive and the convergence domain is small, therefore it results to be easy to use with simple cases but difficult for complex systems.

In direct methods first the states and controls are discretized with respect to time ("first discretize, then optimize"), i.e. the infinite dimensional optimization problem is transformed into a finite dimensional one by means of appropriate function approximation. In this way the optimal control problem is ported to a Nonlinear Programming problem (NLP) which can then be solved using well established optimization techniques such as the Sequential Quadratic Programming (SQP) method. The disadvantage of direct methods is that solutions are local optima, however it has several advantages over the previous two methods, such as the larger convergence domain, a larger variety of problems can be treated and the computational complexity increases with low polynomial order with the complexity of the problem. There are different methods to perform discretization, the main ones are collocation, direct shooting and direct multiple shooting.

A direct method using multiple shooting is chosen for this thesis, which is explained in detail in the following.

2.2.2 Direct multiple shooting method

The software package MUSCOD (MUltiple Shooting CODe for optimization) of IWR[1], Heidelberg University, is based on the theory of direct multiple shooting described in [16] and the first version was released in 1981, implemented in Fortran. It was later improved and ported to C/C++ in the release of MUSCOD-II in [68].
In the direct multiple shooting method states and controls are discretized, the phase times d_j are divided into m_j intervals, over which simple functions (e.g. constant or linear) are chosen for the controls. In this way the continuous optimal control problems is reformulated as an NLP problem solved using the SQP algorithm. The differential equations are solved independetly on each of the multiple shooting intervals in parallel to the NLP. We illustrate in the following how the problem is discretized and solved in MUSCOD-II.

Discretization of controls

The controls of each phase j is divided into subintervals by parametrizing the control function with a piecewise function, e.g. piecewise constant. The time interval of each phase j is divided into a grid:

$$s_j = t_0^j < t_1^j < \dots < t_{m_j-1}^j < t_{m_j}^j = s_{j+1}, \text{ for } j = 0, \dots, n_{ph} - 1. \tag{2.17}$$

On this grid the controls $\mathbf{u}_j(\cdot)$ are discritized by means of the piecewise function $\varphi(\cdot)$ and parameter $\mathbf{a}_i^j$:

$$\mathbf{u}_j(t, \mathbf{a}_i^j) = \varphi_j(t, \mathbf{a}_i^j), \text{ for } i = 0, \dots, m_j - 1 \tag{2.18}$$

where $\varphi_j(\cdot)$ can be chosen to be the same function for all phases or different for each. In the current version of MUSCOD-II, the user can choose the implementation of $\varphi_j(\cdot)$ between piecewise constant, linear or cubic. Higher order can also be used but would increase the complexity of the problem.
In all the applications of this thesis, both for the human model and the HeiCub model, the chosen function is the piecewise constant, i.e. $\varphi_j(t, \mathbf{a}_i^j) = \mathbf{a}_i^j$.

Discretization of states

The discretization grid of the states can be the same as for the controls or different. For simplicity, we assume them to be the same, but we would like to remind that in principle they can be different.

[1]Interdisciplinary Center for Scientific Computing

The states are parametrized with the multiple shooting nodes, while the time interval between two nodes is called multiple shooting interval.
Given our assumption, with reference to the notations used for controls discretization, the number of multiple shooting intervals of a phase j corresponds to m_j, while the number of multiple shooting nodes is $m_j + 1$. This means that the total number of discretized points is $n_{mp} = \sum_{j=1}^{n_{ph}} m_j + 1$.
For each interval the states are parametrized as:

$$\dot{\mathbf{x}} = \mathbf{f}_j(t, \mathbf{x}(t), \varphi_j(t, \mathbf{a}_i^j)), \text{ for } i = 0, ..., m_j - 1 \tag{2.19}$$

where initial values are defined at each interval i:

$$\mathbf{x}(t_i^j) = \mathbf{X}_i^j, \text{ for } i = 0, ..., m_j - 1 \tag{2.20}$$

with $\mathbf{X}_i^j$ being the initial values of interval i and phase j, which have to be satified at the solution of the problem and not for every SQP iteration. In this way we are defining a set of initial value problems.
Continuity conditions are necessary to obtain continuous solutions, we define the condition as:

$$\mathbf{x}(t_{i+1}^j; \mathbf{X}_i^j, \mathbf{a}_i^j) - \mathbf{X}_{i+1}^j = \mathbf{0}, \text{ for } = 0, ..., m_j - 1 \tag{2.21}$$

which ensures that the end point of the multiple shooting interval coincides with the starting value of the following interval.

Discretized optimal control problem

The duration of each phase $d_j = [s_j, s_{j+1}]$, and consequently the whole duration T can be either fixed or it can be left free to the optimization problem. This is achieved by means of a time transformation to the unity interval $t \in [s_j, s_{j+1}] \longrightarrow h \in [0, 1]$ and the time is introduced as optimization variable. The phase duration of transition phases is zero, these phases are treated with a single shooting node.
Continuous path constraints are discretized by evaluating them on the multiple shooting nodes:

$$\mathbf{g}(\mathbf{X}_i^j, \mathbf{a}_i^j) \geqslant 0, \text{ for } i = 0, ..., m_j - 1, \tag{2.22}$$

We define the vector of all discretized variables **a**, **X**, parameters **p** and phase durations **d** as $\boldsymbol{\omega}$ to simplify the notations hereafter. All the equality constraints are grouped into the vector $\mathbf{G}(\boldsymbol{\omega})$ and the inequality constraints into the vector $\mathbf{R}(\boldsymbol{\omega})$.
From these notation simplifications, the discretized form the optimal control problem results to be:

$$\min_{\boldsymbol{\omega}} \quad \boldsymbol{\Phi}(\boldsymbol{\omega}) \tag{2.23}$$

subject to:

$$\begin{array}{rcl} \mathbf{G}(\boldsymbol{\omega}) & = & \mathbf{0} \\ \mathbf{R}(\boldsymbol{\omega}) & \geqslant & \mathbf{0} \end{array} \tag{2.24}$$

where $\boldsymbol{\Phi}(\boldsymbol{\omega})$ is the discretized objective function. In this form also continuous path constraint, boundary and interior point constraints are discretized and the parameters **p** are added as a

vector of free variables when free parameters are to be optimized.
The discretized form is a large and complex NLP for systems as the ones we want to treat in this thesis. Such problems are solved using SQP methods in MUSCOD-II. SQP solves the NLP as subproblems using quadratic optimization problems (QP). In unconstrained cases, SQP corresponds to using the Newton method, while under constraints it applies the Newton method on the Karush-Kuhn-Tucker (KKT) conditions for optimality of nonlinear constrained optimization problems, i.e. first order optimality conditions.
The SQP starts with an initial guess of $\boldsymbol{\omega}$ and then iterates at every SQP iteration k until convergence:

$$\boldsymbol{\omega}_{k+1} = \boldsymbol{\omega}_k + \alpha_k \Delta\boldsymbol{\omega}_k \tag{2.25}$$

where α_k is the step length and $\Delta\boldsymbol{\omega}_k$ is the step direction.
The QP subproblem to be solved is formulated as follows:

$$\min_{\Delta\boldsymbol{\omega}_k \in \Omega} \quad \nabla\boldsymbol{\Phi}(\boldsymbol{\omega}_k)^T \Delta\boldsymbol{\omega}_k + \tfrac{1}{2}\boldsymbol{\Delta}\omega_k^T \mathbf{H}_k \boldsymbol{\Delta}\omega_k \tag{2.26}$$

subject to:

$$\begin{array}{rcl} \mathbf{G}(\boldsymbol{\omega}_k) + \nabla_{\boldsymbol{\omega}}\mathbf{G}(\boldsymbol{\omega}_k)^T \Delta\boldsymbol{\omega}_k & = & \mathbf{0} \\ \mathbf{R}(\boldsymbol{\omega}_k) + \nabla_{\boldsymbol{\omega}}\mathbf{R}(\boldsymbol{\omega}_k)^T \Delta\boldsymbol{\omega}_k & = & \mathbf{0} \end{array} \tag{2.27}$$

in which $\nabla_{\boldsymbol{\omega}}\mathbf{G}(\boldsymbol{\omega}_k)$ and $\nabla_{\boldsymbol{\omega}}\mathbf{R}(\boldsymbol{\omega}_k)$ are the Jacobians and $\mathbf{H}_k$ is the approximated Hessian of the Lagrangian function, which can be chosen to be a diagonal matrix at the beginning and then refined during the SQP. The trust region Ω is used to have valid QPs as positive definitiveness of the Hessian is not guaranteed.
Due to the multiple shooting method applied, the Hessian and the Jacobians are highly sparsed and structured, therefore in MUSCOD-II before solving the QP a condensing is performed as described in [68], which removes the sparsities and a standard QP solver can be used.
The iterations stop when convergence is reached for the KKT conditions expressed as KKT tolerance which is set as desired value by the user.

2.2.3 Locomotion as multiphase optimal control problem

In this thesis, optimal control is mainly applied to the problem of locomotion.
In chapter 4 we formulate multiphase optimal control problems for human walking, where each walking phase is characterized by a set of different contacts, i.e. contact points. In chapter 8 a similar formulation is done for the problem of push recovery where the robot has to perform a step to recover from falling.
The different phases are modeled as continuous phases, which dynamics are described as in Eq. (2.9). The states of the optimal control problem are the generalized coordinates and velocities:

$$\mathbf{x}(t) = \begin{bmatrix} \mathbf{q}(t) \\ \dot{\mathbf{q}}(t) \end{bmatrix} \tag{2.28}$$

such that the lower part of the differential state $\dot{\mathbf{x}}(t) = \mathbf{f}_j(\cdot)$ as in Eq. (2.16) is the vector of generalized accelerations $\ddot{\mathbf{q}}$, which is computed with forward dynamics for each phase j:

$$\dot{\mathbf{x}}(t) = \begin{bmatrix} \dot{\mathbf{q}}(t) \\ \ddot{\mathbf{q}}(t) \end{bmatrix} = \begin{bmatrix} \dot{\mathbf{q}}(t) \\ FD(\mathbf{q}, \dot{\mathbf{q}}, \boldsymbol{\tau}) \end{bmatrix}. \tag{2.29}$$

When a discontinuity occurs, the transition function $x(s_j^+) = \widetilde{\mathbf{J}}_j(\cdot)$ between two phases is described with the impact dynamics as in Eq. 2.12.
When using forward dynamics formulations, controls $\mathbf{u}(t)$ are typically the joint torques $\boldsymbol{\tau}$ but can also be defined with different quantities according to the desired application.

In all the applications that will be shown in chapters 4, 7 and 8, the treated problems are always multiphase but we always specify the same objective function for all phases and therefore the expression of the objective function is as in Eq. (2.13) rather than Eq. (2.15).

Specific differences in the definition of states, controls, constraints and objective functions will be explained in detail in each chapter targetting each application.

Part II

Study of compliance in human locomotion with computational methods

3 From experimental data to computational models

To learn human walking characteristics, human walking motions are recorded with motion capture systems. Several types of commercially available motion capture systems exist and can be categorized mainly in marker-based and marker-less.
In the marker-based systems, markers are placed in startegic points of the human body, mainly corresponding to locations that can help to reconstruct joint trajectories, these markers are then tracked by cameras and the location of each marker is being recorded. Commercially available marker-based systems are for example Vicon[1] and Qualisys[2]. There are also several types of marker-less systems, most notable examples are the Inertial Measurement Units (IMU), which can be placed in different points of the body similarly to the markers, and force plates, which measure ground reaction forces.
Despite the motion capture system used, the common objective is to extract whole-body motion or information of the movements. To extract these information it is necessary to define a specific computational model that is used to recover the information under certain assumptions. These models are either kinematic and/or dynamic descriptions of the human body, and respect a set of imposed constraints.

In the context of this thesis, the data recorded using marker-based motion recording systems are used, in particular these data are part of the extensive KoroiBot Motion Databse [1]. These data are used to recover whole-body joint trajectories by means of subject specific kinematic models, then mapped onto dynamic models under dynamic constraints, as described in Fig. 3.1.
In the following sections, we first describe the motion capture system used in KoroiBot, then how the kinematic and dynamic fits are carried out with detailed descriptions of the used models.

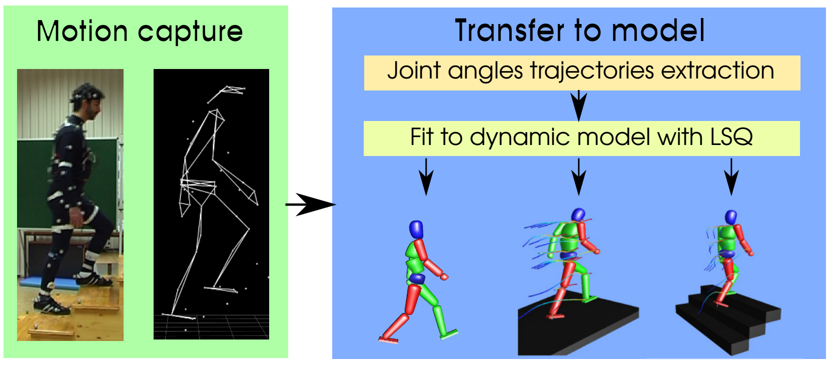

Motion capture
Transfer to model
Joint angles trajectories extraction
Fit to dynamic model with LSQ

Figure 3.1: Motion capture data are mapped to the human model considering whole body dynamics and feasibility constraints.

[1]VICON - https://www.vicon.com/
[2]Qualisys - http://www.qualisys.com/

3.1 Motion capture experiments

There exist different motion capturing systems. The one that the KoroiBot community chose is a marker-based motion capture system, in particular the commerically available Vicon system. Beside the marker-based motion capture system, devices which can record contact forces, such as force plates and force sensing shoes, were also used in some cases.
As in marker-based systems the markers can be placed freely mainly according to the convenience of the end-users, in the KoroiBot project a whole-body marker set[3] was defined, i.e. which markers have to be used and where they have to be placed. In this way a common post-process can be carried out, allowing for easier and less time consuming computations.
Alongside the marker set, an antropomorphic table[4] was also defined. This table consists in a precise definition of human body segment dimensions, such as lengths, widths and diameters, mainly corresponding to the segments defined with the KoroiBot marker set.
Each motion is recorded in a c3d file which describes the location of the markers at each time instant. In the KoroiBot Motion Database [1] also videos showing the actual experiment are available for certain trials.

3.1.1 Walking Experiments

The database collects data of many walking environments and type of motions, which were performed for many trials by several subjects.
The walking environments cover mainly the KoroiBot parcour, as in Fig. 3.2:

- Unconstrained level ground,
- Soft terrain, e.g. mattress, sand,
- Beam,
- Sea-saw,
- Stairs,
- Step-stones,

and many others that are not present in the scenario but are interesting walking scenarios, e.g. slopes of different inclinations.
Many subjects of both male and female gender, different ages, heights and weights participated to the motion recording experiments. In specific, until the end of the project, 149 subjects participated to the recordings for a total of 544 experiments and more than 5000 recorded trials.
Each subject was asked to walk in one or more scenarios, for which at least 5 trials were recorded. For each subject the segment dimensions were also measured and recorded in the anthropomorphic tables.

Given that the objective of KoroiBot is to study human locomotion in the different environments, in each recording the subjects were asked to walk for several steps in the most natural way possible as they usually walk. Slight changes in the walking style might be induced by several reasons, e.g. the motion capturing equipments that the subject is wearing, the awareness of being involved in a scientific experiment etc. However this problem has been negleted

[3]The marker set can be found at https://koroibot-motion-database.humanoids.kit.edu/marker_set/
[4]The antropomorphic table can be found at https://koroibot-motion-database.humanoids.kit.edu/anthropometric_table/

in the context of this thesis.

Common issues in marker-based systems are the occlusion of markers, i.e. when one or more markers are being hidden while performing some motions, the misplacement of the markers, as the markers can be placed freely in any point of the body, and the soft tissues that can slightly move the markers from their original position. These issues should be taken into account in the post process of the data, but they are still very challenging to be solved, therefore it is common to neglet or simplify the problem.

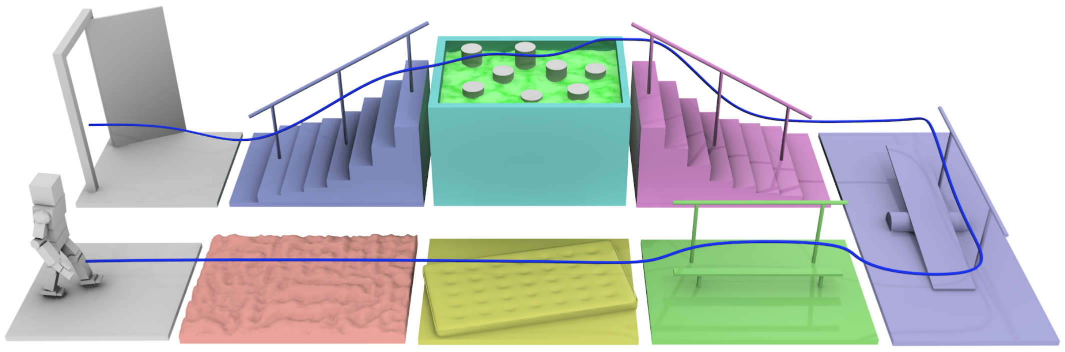


Figure 3.2: The KoroiBot parcour. Picture from [7], authored by M. Felis.

3.2 Model of the human body

The human body is extremely complex to model, as already stated in Section 2.1, several assumptions and simplifications are being made:

- The human body is a rigid body system composed of rigid links connected with joints.
- All contacts are rigid and impacts are instantaneous and inelastic.

Under these assumptions, the mechanics of the human body can be described with common rigid multi-body dynamics as presented in Section 2.1.
Despite the simplifications, the model is still very complex. It can be considered in the 3D space, where motions on all the anatomic planes can be analyzed, or in 2D space, where the motion on a single plane, commonly the sagittal plane in the case of walking, can be studied. Models in 3D space with 36 DOF were used to study emotional aspects of human locomotion [32] as well as to generate human walking motions based on human inspired criteria [31]. In this thesis, the interest is in the walking motion, and in particular in the complex feature of compliance, therefore the focus is posed on the sagittal plane.

The 2D model considers motions from the sagittal plane point of view, as it is the most relevant plane in walking motions. The model used in this thesis is the HeiMan model developed in [32]. It consists in a scalable rigid body model of 36 DOF in 3D space where segment dimensions and dynamic parameters such as masses and inertia are mostly based on DeLeva data [23] and guesses made upon observations of biomechanics measurements and motion recordings. A subset of DOF can be specified and the dynamic parameters are scaled by specifying height and weight of the subject.
In particular, the model used to carry out the studies of this thesis is as shown in Fig. 3.3:

- 14 segments, 2 for each arm, 3 for each leg, 3 in the torso and one for the head.

- 16 DOF, of which 3 for the floating base. The reference frame is located in the pelvis. The 3 DOF are two translations, along x and z directions and one rotation about the y axis. The axis pointing in the walking direction is x while the z axis is pointing up.
- Only motions in the sagittal plane are considered, which means all internal joints represent 1 DOF rotational joints with rotations about the y axis.
- Flat feet with two contact points, one on the toe and one on the heel. These two points allow to describe flat foot contact as well as heel only and toe only contact to represent the different walking phases.

  The points are respectively:
  - Right foot, $\mathbf{R}_{toe} = [R^x_{toe}, R^z_{toe}]^T$ and $\mathbf{R}_{heel} = [R^x_{heel}, R^z_{heel}]^T$.
  - Left foot, $\mathbf{L}_{toe} = [L^x_{toe}, L^z_{toe}]^T$ and $\mathbf{L}_{heel} = [L^x_{heel}, L^z_{heel}]^T$.
- Torsional springs in the left and right hip, knee and ankle joints. A damper is introduced in the ankle to avoid oscillations that can occur during the lift off of the foot from the ground.
- Coupling bi-articular springs between hip and knee joints of both legs.

The model is described with the generalized coordinates $\mathbf{q} \in \mathbb{R}^{n_{dof}}$, where $n_{dof} = 16$ is the total number of degrees of freedom, with actuated degrees of freedom $n_{actuated} = 13$. The stiffness of all the joints is set to be variable, including the bi-articular couplings, detailed modeling is described in Section 3.4. The rest positions of the springs and the value of the dampers at the ankles are fixed and can be set according to the specific subject and motion.

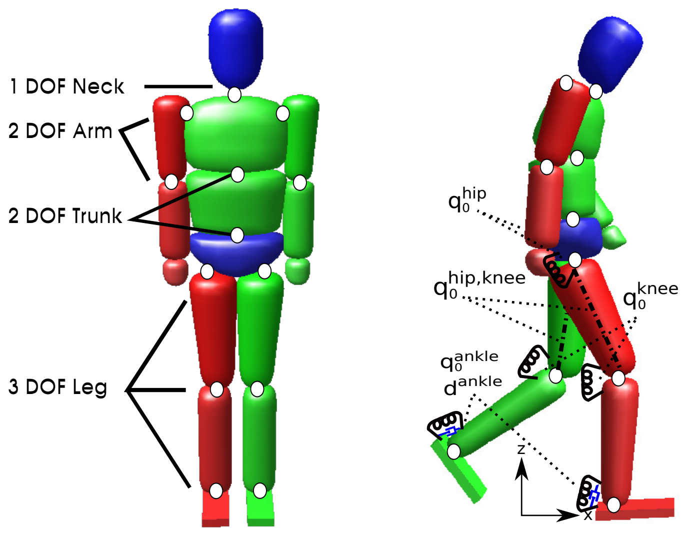


Figure 3.3: 2D human model with spring damper systems.

3.3 Joint angles extraction

The data to be extrated from the motion capture data are the joint angle trajectories. There are several commerically available tools to do this, such as the Vicon system. Free tools have also been developed, examples are the Puppeteer tool developed by Martin Felis [32] and the

Master Motor Map (MMM) framework [110] developed by Karlsruhe Institute of Technology (KIT). Both tools target the marker-based motion capture systems, in which an inverse kinematics procedure is carried out by mapping the recorded locations of the markers to markers placed virtually on an adjustable human model.

In this thesis the MMM framework is used. In MMM a subject speficic human kinematic model is defined. This model can have up to 63 DOF including fingers, as at the beginning MMM was thought mainly for manipulation motions. Any subset DOF of this model can be used. The segment dimensions of this model are based on the Winter biomechanics data [123] and can be scaled by specifying the subject height. Alternatively they can be set by the user in an arbitrary way. In the particular case of this thesis, they are set to those of specific subjects whose motions are recorded in the KoroiBot Motion Database [1].
A marker set is defined for the model, which has to correspond exactly to the one used in the recorded motions. In particular in KoroiBot, a common convention is chosen and all partners involved recorded motions with the same marker set, allowing for easy conversion of all motions with the MMM tools.
Once the model and the marker set are defined, the recorded positions of real markers are mapped onto those of the virtual markers with a least squares fit on each frame and the joint angles computed via inverse kinematics. The procedure is not able to take into account hidden markers. When a marker is occluted, it is being ignored by the fitting procedure, and therefore the resulting motion, which is computed based on the assumption that all markers are present, can be extremely different from the original one. In this case the occluted markers are excluded from the procedure by manually removing them to improve the fitting performance.

A kinematic fit of this kind highly relies on the correct estimation of the position and orientation of the pelvis frame as the hip, as well as the torso angles, are relative to the pelvis. This is a very difficult task to achieve, due to the small dimension of the pelvis and other problems such as the misplacement of the markers and the movements of the markers due to soft tissues. Also, this kind of kinematic fit do not take into account environmental constraints, contact constraints or any dynamic property of the human body. Therefore the resulting motion could violate any of the said constraints, reason for which a dynamic fit is performed in this thesis with optimal control methods, as will be explained in details in the next chapter.

The joint angle trajectories obtained from the MMM framework are stored in an MMM format file, which consists in an XML file. These trajectories are then transferred to the model described in the previous Section 3.2 via a dedicated script.

3.4 Modeling of compliance

Due to our model choice, there are mainly two possibilities of modeling the compliance at joint level. The first is to allow for *variable stiffness*, the second is to allow for *variable rest position* (or set point) of the spring.
The first possibility agrees with theories from biomechanics studies where it has been proven that humans vary their stiffness during movements due to the action of the agonist and antagonistic muscles on the joints [47]. While the second possibility agrees with the Threshold Control Theory (TCT) [29, 66] from motor control, which assumes that an equilibrium point exists for each movement and this equilibrium point is shifted when the movement changes.
However, in motor control it is assumed that motions are controlled at a neural level and is contrasted with biomechanics modeling choices, where the motion is studied from the system

dynamics point of view.
We choose to use the first possibility of varying stiffness, as with our methods and model choices we fall in the biomechanics branch of studies. Therefore the mathematical description should respect the most human-like model as per biomechanics, despite varying stiffness results to be more computational expensive for the optimal control than varying the rest positions, as stiffness will be treated as controls in the optimal control problem. However, the rest positions of our model are not fixed a priori, but are left free to the optimization to find the best suited value for specific motions, which also agrees to some extent with the TCT.

In Fig. 3.3 and in the description of the model, torsional springs are inserted in each of the joints of the legs, and a bi-articular coupling spring between the hip and the knee, which corresponds to the bi-articular muscles of humans, is also defined.
As per our choice, all the springs have variable stiffness and represent the actuation of the leg joints, given our ultimate objective of analysing the compliance at joint and bi-articular level. The torques are generated by these spring damper systems, and in particular, by changing the stiffness of the springs over time.
In particular, the hip and knee have the same type of actuation, as they are coupled by the bi-articular springs, while the ankle has independent actuation but has to take into account the effect of the damper.
Therefore the torque at the hip and knee joints is defined as:

$$\tau(t) = -k_{*\#}(t)(q_{*\#}(t) - q_0^{\#}) - k_{*hip,knee}(t)(q_{*hip}(t) + q_{*knee}(t) - q_0^{hip,knee}) \tag{3.1}$$

where # is either k for the knee or h for the hip, $*$ is either l for the left leg or r for the right leg. Therefore $k_{*\#}$ is the variable stiffness of the knee or hip joint, $k_{*hip,knee}(t)$ is the variable stiffness of the coupling between hip and knee joints, $q_{*\#}(t)$, $q_{*hip}(t)$ and $q_{*knee}(t)$ are the knee or hip joint angle trajectory, $q_0^{\#}$ and $q_0^{hip,knee}$ are the rest positions of the knee or hip and the rest position of the biarticular spring respectively.
While in the ankle the torque is defined as:

$$\tau(t) = -k_{*ankle}(t)(q_*(t) - q_0^{ankle}) - d^{ankle}\dot{q}_*(t) \tag{3.2}$$

where $k_{*ankle}(t)$ is the variable stiffness of the ankle joint, $q_*(t)$ and $\dot{q}_*(t)$ are the ankle joint angle trajectory and its derivative respectively, q_0^{ankle} is the ankle spring rest position and d^{ankle} is the damping factor of the ankle joint.
As walking is assumed as periodic motion, the rest positions of the springs as well as the damping of the ankle, are fixed (i.e. not time varying) and imposed to have the same values in both legs.

3.5 Walking phases

Walking can be split in three major stages:

- Initialisation, in which the walking starts from a resting position.
- Periodic gait, where the motion becomes cyclic and can be assumed to be periodic.
- Ending, in which the motion is brought to a resting position.

The major phases in which walking can be split are the single and double support phases, depending on if one or two feet are in contact with the ground. In biomechanics more detailed phases are identified for the full cycle of walking, i.e. for the initialisation steps, the periodic steps and also for the ending steps.

In this thesis only the periodic steps are considered, where the double and single support phases are further split into detailed phases according to the number of contacts of the feet with the ground. The phases depend therefore on the number of contacts defined for the foot, as well as on the foot model. Since the model as defined in Section 3.2 has a flat foot model with two contacts, the phases described in the following respect this rule.

From the motions recorded in the KoroiBot Motion Database [1], we observed that humans walk with different phases in different environments, therefore it is not possible to identify a set of phases that is representative of any walking environment.
In daily life we walk in many different environments, where the most common ones are level ground, up and down slopes of different inclinations, stairs of different sizes and different type of rough terrains. So it is in our interest to analyze walking in all these different scenarios. In particular in this thesis, level ground, slope and stairs are analyzed and the same phases are identified for walking on level ground and up a slope, while for walking up stairs a different set is defined. In all cases, one single step is considered, where the right leg is standing and left leg in swinging.

3.5.1 Level ground and slope walking phases

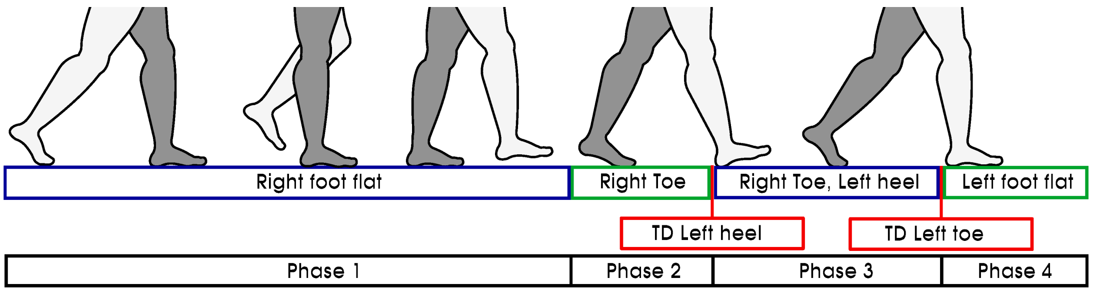


Figure 3.4: Walking cycle of a single step on level ground and up a slope.

The phases for level ground and slope walking are defined as in Fig. 3.4:

Phase 1 Right foot flat (toe and heel on the ground) and left leg swinging

Phase 2 Right toe on the ground and left leg swinging

Phase 3 Right toe and left heel on the ground

Phase 4 Right toe on the ground and left foot flat

In addition to the phases listed above, there are also two impacts due to the collision of the heel and the toe points on the ground. They are assumed to occur instantaneously, i.e. in zero time. These impacts are represented by the red blocks as in Fig. 3.4 and occur between phases 2 and 3 as well as between 3 and 4.
It should be noted that slope walking phases result to be the same as level ground walking only up to certain slope inclinations when walking up a slope. In fact, when the slope inclination is above certain thresholds, the phases can change as with highly inclined slopes, in order to compensate for the gravity actions, humans tend to not roll the foot from heel to toe after the touch down anymore, but rather to place the foot flat or even with the toe before the heel.

These are however extreme cases that are rare in daily life and therefore are not in the scope of this thesis anymore.

3.5.2 Stair climbing phases

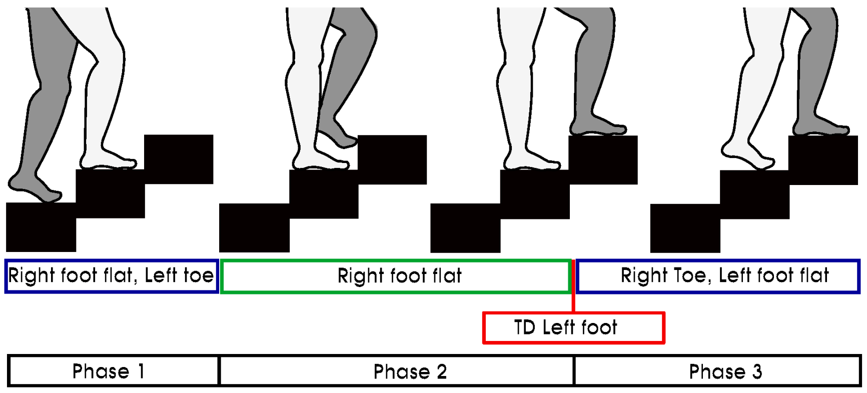


Figure 3.5: Walking cycle of a single step on a stairs.

While climbing stairs, humans change the typical unconstrained walking and therefore the walking phases necessary to describe the dynamics of the motion are different, resulting in three phases instead of four as in Fig. 3.5:

Phase 1 Right foot flat (toe and heel on the ground) and left toe on the ground

Phase 2 Right foot flat (toe and heel on the ground) and left leg swinging

Phase 3 Right toe on the ground and left foot flat

Instead of two impacts, there is only one between phases 1 and 2, when the left foot touches the ground. For simplicity it is assumed that the foot strikes the ground flat, thus with both points, toe and heel, simultaneously, as the average time between heel and toe touchdown when reaching the next step is almost zero. Given the lack of literature on systematic description of detailed walking phases in constrained environments such as stairs, this assumption is a result of observations made on several trials of different subjects recorded in the KoroiBot Motion Database [1].

3.6 Dynamic reconstruction with optimal control

With the kinematic mapping, the joint angle trajectories could be recovered from motion capture data, however they do not respect any dynamic constraint and we cannot gain dynamic information from these data. Therefore, a dynamic reconstruction (dynamic fit) as least square problem (LSQ) is carried out using the optimal control theory as described in Chapter 2.2 in combination with the 2D human model described in Section 3.2. It consists in the mapping of the joint angle trajectories obtained from motion capture data to the rigid multi-body dynamic model taking into account all dynamic properties and constraints, fulfilling a certain specified objective function or set of objective functions.
The objective function in this case consists usually in the minimization of the error between the measurements and the model, but it also depends on the goal of the analysis. Also the definiton of states, controls, parameters and constraints are application oriented and will be defined in the next chapter for the studies of this thesis.

However, in any motion reconstruction optimal control problem, the constraints on the feet contacts during the different phases can be generalized.

Referring to the two cases we have shown in Sections 3.5.1 and 3.5.2, the equality and inequality constraints are defined as follows.

Level ground and slope

In this case the number of phases is $n_{ph} = 4$, with two additional transition phases with zero time corresponding to the impacts.

- **Phase 1 - Right foot flat**

  At the beginning of the phase, for $t = s_0$, the foot has to be rigidly attached to the ground:

$$\begin{aligned} R^z_{toe} &= 0 \\ R^z_{heel} &= 0 \\ \dot{R}^z_{toe} &= 0 \\ \dot{R}^z_{heel} &= 0 \\ L^z_{toe} &= 0 \\ f(R^z_{toe}) &\geqslant 0 \\ f(R^z_{heel}) &\geqslant 0 \end{aligned} \tag{3.3}$$

  where $\dot{R}$ and $\dot{L}$ are the velocity of the indicated contact point of right or left foot, and f is a function that computes the ground reaction forces of a specified point.

  During the phase, for $t \in (s_0, s_1)$:

$$\begin{aligned} L^z_{toe} &\geqslant 0 \\ L^z_{heel} &\geqslant 0 \\ f(R^z_{toe}) &\geqslant 0 \\ f(R^z_{heel}) &\geqslant 0 \end{aligned} \tag{3.4}$$

  to ensure that the left foot does not go below the ground.

- **Phase 2 - Right toe** At the beginning of the phase, for $t = s_1$:

$$\begin{aligned} f(R^z_{heel}) &= 0 \\ L^z_{toe} &\geqslant 0 \\ L^z_{heel} &\geqslant 0 \\ \dot{R}^z_{heel} &\geqslant 0 \\ f(R^z_{toe}) &\geqslant 0 \end{aligned} \tag{3.5}$$

  to ensure the right toe lift off.

  During the phase, for $t \in (s_1, s_2)$:

$$\begin{aligned} L^z_{toe} &\geqslant 0 \\ L^z_{heel} &\geqslant 0 \\ R^z_{heel} &\geqslant 0 \\ f(R^z_{toe}) &\geqslant 0 \end{aligned} \tag{3.6}$$

- **Touchdown 1 - Right toe, left heel touchdown**

  The time of transition phases is zero, therefore $t = s_2$:

$$
\begin{array}{rcl}
L^z_{heel} & = & 0 \\
R^z_{heel} & \geqslant & 0 \\
L^z_{toe} & \geqslant & 0 \\
\dot{L}^z_{heel} & \leqslant & 0 \\
f(R^z_{toe}) & \geqslant & 0
\end{array}
\tag{3.7}
$$

  where the velocity of the touchdown point changes.

- **Phase 3 - Right toe, left heel**

  During the phase, for $t \in (s_2, s_3)$:

$$
\begin{array}{rcl}
R^z_{heel} & \geqslant & 0 \\
L^z_{toe} & \geqslant & 0 \\
f(R^z_{toe}) & \geqslant & 0 \\
f(L^z_{heel}) & \geqslant & 0
\end{array}
\tag{3.8}
$$

- **Touchdown 2 - Right toe, left heel, left toe touchdown**

  For time $t = t_3$:

$$
\begin{array}{rcl}
L^z_{toe} & = & 0 \\
R^z_{heel} & \geqslant & 0 \\
\dot{L}^z_{toe} & \leqslant & 0 \\
f(R^z_{toe}) & \geqslant & 0 \\
f(L^z_{heel}) & \geqslant & 0
\end{array}
\tag{3.9}
$$

- **Phase 4 - Left foot flat**

  For the whole duration of the phase, for $t \in [s_3, s_4]$:

$$
\begin{array}{rcl}
R^z_{heel} & \geqslant & 0 \\
f(R^z_{toe}) & \geqslant & 0 \\
f(L^z_{toe}) & \geqslant & 0 \\
f(L^z_{heel}) & \geqslant & 0
\end{array}
\tag{3.10}
$$

Stairs up

In the case of stairs walking, additional environmental constraints as inequality constraints are added to ensure that the feet do not walk into the stairs. The number of phases is $n_{ph} = 3$, with one transition phase with zero time corresponding to the single impact.

- **Phase 1 - Right foot flat, left toe**

  At the beginning of the phase, for $t = s_0$, the right foot is rigidly attached to the stairs step:

$$\begin{aligned}
R^z_{toe} - z_{stair} + h_{stair} &= 0\\
R^z_{heel} - z_{stair} + h_{stair} &= 0\\
L^z_{toe} - z_{stair} &= 0\\
\dot{R}^z_{toe} &= 0\\
\dot{R}^x_{heel} &= 0\\
\dot{R}^z_{heel} &= 0\\
\dot{L}^x_{toe} &= 0\\
\dot{L}^z_{toe} &= 0\\
L^z_{heel} &\geqslant 0\\
f(R^z_{toe}) &\geqslant 0\\
f(R^z_{heel}) &\geqslant 0\\
f(L^z_{toe}) &\geqslant 0
\end{aligned} \tag{3.11}$$

where z_{stair} is the current height of the stair on which the left foot is stepping on, as this could be different from dataset to dataset, and h_{stair} the height of the stairs step, which is fixed and known a priori from the environment setting.

During the phase, for $t \in (s_0, s_1)$:

$$\begin{aligned}
f(R^z_{toe}) &\geqslant 0\\
f(R^z_{heel}) &\geqslant 0\\
f(L^z_{toe}) &\geqslant 0
\end{aligned} \tag{3.12}$$

- **Phase 2 - Right foot flat**

  At the beginning of the phase, for $t = s_1$:

$$\begin{aligned}
R^z_{toe} - z_{stair} + h_{stair} &= 0\\
R^z_{heel} - z_{stair} + h_{stair} &= 0\\
\dot{L}^z_{toe} &\geqslant 0\\
f(R^z_{toe}) &\geqslant 0\\
f(R^z_{heel}) &\geqslant 0
\end{aligned} \tag{3.13}$$

  to ensure that right toe lifts off.

  During the phase, for $t \in (s_1, s_2)$:

$$\begin{aligned}
L^x_{toe} + x_{stair} &\geqslant 0\\
L^z_{toe} - z_{stair} &\geqslant 0\\
f(R^z_{toe}) &\geqslant 0\\
f(R^z_{heel}) &\geqslant 0
\end{aligned} \tag{3.14}$$

  where x_{stair} is the position of the stair step in which the left foot should not enter, this is to ensure that environmental constraints are respected.

- **Touchdown 1 - Right foot flat, Left foot touchdown**

  For time $t = s_2$:

$$\begin{array}{rcl} L_{toe}^z - z_{stair} + 2 * h_{stair} & \geqslant & 0 \\ L_{heel}^z - z_{stair} + 2 * h_{stair} & \geqslant & 0 \\ \dot{L}_{toe}^z & \leqslant & 0 \\ \dot{L}_{heel}^z & \leqslant & 0 \\ f(R_{toe}^z) & \geqslant & 0 \end{array} \quad (3.15)$$

  to ensure that the left foot is flatly positioned on the next step, therefore $2 * h_{stair}$.

- **Phase 3, Right toe, left foot flat**

  For the whole duration of the phase $t \in [s_2, s_3]$:

$$\begin{array}{rcl} f(R_{toe}^z) & \geqslant & 0 \\ f(L_{toe}^z) & \geqslant & 0 \\ f(L_{heel}^z) & \geqslant & 0 \end{array} \quad (3.16)$$

These constraints are used in the applications that will be presented in Chapter 4.

4 Analysis of the role of compliance in human walking

The objective is to apply optimal control to the analysis of human gait from the dynamics point of view. The mapping of the motions is carried out as described in Section 3.6. With this approach several aspects of human locomotion can be exploited, according to the formulation of the problem, e.g. in terms of how the states, controls, parameters, constraints and objective functions are being defined.
Here the feature of human walking being exploited is compliance, in terms of joint and bi-articular coupling stiffness. There is a large amount of literature on stiffness at joint level, which are mostly focused on level ground walking, with a much smaller amount of works on other walking scenarios. Some works focused on the analysis of kinematics and kinetics of slope walking [38, 105] and stair climbing [13, 12], but there is a lack of studies focused on joint stiffness.
In this chapter, our first goal is to undersand if and how compliance at joint and bi-articular level modulate during walking in different environments, then if and how this modulation influences the walking gait.
First the specific data used to carry out the analysis is being presented, then the two problems are shown in details with results and discussion.

4.1 Walking data

All the data used in this thesis are from the KoroiBot Motion Database [1]. To have more extensive comparable data and generalizable results, four male subjects walking in the exact same environments are chosen. The environments are:

- Unconstrained level ground.
- Unconstrained slope.
- Stairs.

Each of the subjects walked in the front forward direction and for at least 6 steps in each scenario, such that intermediate periodic steps can be isolated. For each of the environments, several trials are available. In this thesis a single trial is considered, but the rest of the trials are also used to characterize the walking of each subject in the different environments.
All trials are post processed with the MMM tools [110] as described in Section 3.3 to compute the joint angle trajectories, therefore, all computations hereafter are carried out using the obtained joint angle trajectories $\mathbf{q}_m(t)$.

4.1.1 Walking scenarios description

In the case of unconstrained level ground walking, the subjects walked for 6-7 steps in the front forward direction, including initialisation and ending steps. The steps occuring between the first two and the last two steps are assumed to be periodic.

Different slope inclinations are available in the KoroiBot Motion Database [1], the one chosen for this thesis is of 15° as in Fig. 4.1. It represents a reasonable inclination for which, from one side walking dynamics shows evident differences from the level ground case, therefore interesting insights can be gathered, and from the other side it still shares the same walking phases as level ground walking.

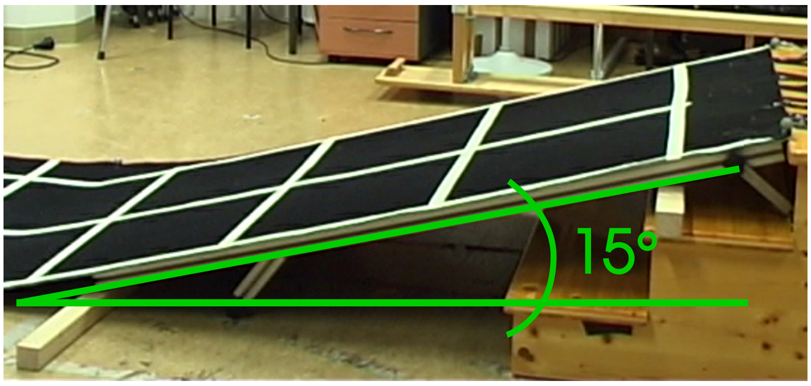


Figure 4.1: Slope of 15 ° performed by the four subjects.

Several stairs are also available in the database, however, as for the slope case, the chosen stairs are those performed by the four chosen subjects. Human size stairs usually have heights between 18 and 21 cm, and a depths between 25 and 32 cm. The ones chosen for this thesis have a height of 19 cm and a depth 28 cm as in Fig. 4.2.

The precise slope inclination and stair sizes could be retrieved also from the motion capture data, as markers were placed also on the objects and environments such that details could be included in the data analysis.

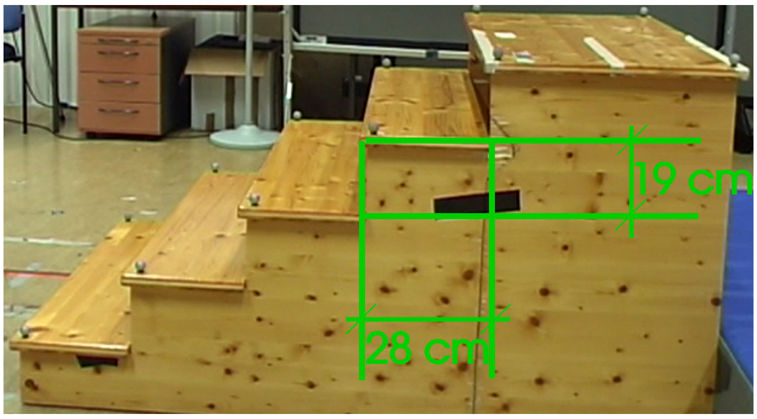


Figure 4.2: Stairs performed by the four subjects.

4.1.2 Single step data description

As walking is assumed to be periodic, the analysis is carried out on a single step and generalized for the walking cycle. The starting and ending steps are excluded from our considerations. For each subject and environment a single step from a single trial is extracted, according to the step phases described in Section 3.5.

Subject	**M1**	**M2**	**M3**	**M4**
Weight $[kg]$	85	80	87	85
Height $[m]$	1.81	1.88	1.77	1.81
Level ground				
Average duration $[s]$	0.61	0.58	0.54	0.58
Average length $[m]$	0.67	0.81	0.78	0.85
Average velocity $[m/s]$	1.1	1.38	1.44	1.44
Walking up slope				
Average duration $[s]$	0.51	0.41	0.65	0.60
Average length $[m]$	0.62	0.80	0.70	0.59
Average velocity $[m/s]$	1.24	1.90	0.95	0.97
Stair climbing				
Average duration $[s]$	0.58	0.73	0.59	0.49
Average forward velocity $[m/s]$	0.49	0.38	0.48	0.58
Average elevation velocity $[m/s]$	0.66	0.52	0.66	0.78

Table 4.1: Subject specific data, with average step duration, length and velocities computed over 3∼5 trials per subject and environment.

Subject	**M1**	**M2**	**M3**	**M4**
Level ground				
Duration $[s]$	0.69	0.63	0.55	0.60
Length $[m]$	0.67	0.86	0.80	0.85
Velocity $[m/s]$	0.97	1.36	1.45	1.42
Swing length $[m]$	1.36	1.65	1.61	1.75
Swing velocity $[m/s]$	1.97	2.61	2.93	2.92
Walking up slope				
Duration $[s]$	0.56	0.46	0.68	0.66
Length $[m]$	0.70	0.82	0.68	0.63
Velocity $[m/s]$	1.25	1.78	1.00	0.95
Swing length $[m]$	1.45	1.68	1.47	1.37
Swing velocity $[m/s]$	2.59	3.65	2.16	2.08
Stair climbing				
Duration $[s]$	0.61	0.71	0.64	0.59
Forward Velocity $[m/s]$	0.46	0.39	0.44	0.47
Elevation velocity $[m/s]$	0.62	0.53	0.59	0.64

Table 4.2: Specific data of the step chosen for each subject for the studies of this thesis.

For each subject and environment at least 3∼5 trials are available in which around 3∼4 periodic steps are recorded for the level ground case, 2∼3 while walking on the slope and 5 steps of the stairs. These trials are used to compute the average of:

- Step length, defined as the distance travelled by the pelvis in the forward direction.
- Duration, defined as the time used to perform the step from lift off to touch down of the swing foot.
- Velocity, defined as the step length divided by the duration. In the case of stairs, the velocity is defined as forward velocity and elevation velocity, the former being step length / step duration and the latter step height / step duration.

From each of the trials the middle steps, i.e. the periodic ones, are extracted and used to compute the average over all steps and trials per subejct. The resulting averages are as reported in Tab. 4.1.

The chosen steps as in Tab. 4.2 can be compared with the average periodic behaviour of the subject. In Tab.4.2, further information regarding the swing leg is included:

- Swing length, which is the distance travelled by the swing footfrom the lift up point of the toe to the touchdown point after the swing.
- Swing velocity, defined as swing length divided by step duration.

As it is possible to observe from the two tables, the data in Tab. 4.2 are close to those in Tab. 4.1, therefore the specific chosen steps are good representatives of the periodic gait of the subjects, and results obtained for these specific steps could be generalized for the walking cycle of the specific subjects.
Each obtained step is further split into phases as described in Section 3.5 and phase times are also identified.

4.2 Joints and bi-articular compliance profiles computation

To analyze compliance in human walking motions, we set up the optimal control problem in order to compute stiffness profiles of the joints and the bi-articular couplings, in particular we refer to the stiffness of the springs as defined in the model described in Section 3.2.
The objective is to perform a least squares optimization problem that reproduces the extracted steps in order to take into account also dynamic properties and therefore obtain dynamics information.
With the methods and formulations we adopt, unique torque profiles result from measured sets of joint angles and systems with given inertial properties. If the problem were to be investigated from a muscle prospective with several muscles - agonists and antagonists - contributing to the same joint torques, this redundancy would have to be resolved. However, we do not consider muscles, but springs at joint level as well as bi-articular springs. These stiffness profiles follow uniquely from the torques since any potential redundancy between simple and bi-articular springs is resolved by means of the regularizing term in the objective function.
In the following, all components of the optimal control problem are defined and the obtained results will be shown and discussed at the end of the section.

4.2.1 States, controls and parameters

As discussed in sections 2.2.3 and 3.6, walking can be formulated as multiphase optimal control problem. As shown in Section 2.1, the dynamics of human walking can be defined as ordinary differential equations. In the formulation of the optimization problem, the states are represented by the generalized positions and velocities of the rigid multi-body system:

$$\mathbf{x}(t) = \begin{bmatrix} \mathbf{q}(t) \\ \dot{\mathbf{q}}(t) \end{bmatrix}, \quad \dot{\mathbf{x}}(t) = \begin{bmatrix} \dot{\mathbf{q}}(t) \\ \ddot{\mathbf{q}}(t) \end{bmatrix} \tag{4.1}$$

with $\mathbf{q}, \dot{\mathbf{q}} \in \mathbb{R}^{n_{dof}}$, and $n_{dof} = 16$. In this case, the right hand side of $\dot{\mathbf{x}}(t)$ is obtained by solving equations as in Eq. (2.9). The transition functions $\widetilde{\mathbf{J}}_j(\cdot)$ are obtained by solving equations as in Eq. (2.12).
The controls are the torques for the upper body joints and the stiffness for the lower joints, including the stiffness of the bi-articular couplings:

$$\mathbf{u}(t) = \begin{bmatrix} \mathbf{u}_T(t) \\ \mathbf{u}_L(t) \end{bmatrix}, \quad \mathbf{u}_T(t) = \boldsymbol{\tau}(t), \quad \mathbf{u}_L(t) = \begin{bmatrix} k_{lhip} \\ k_{lknee} \\ k_{lankle} \\ k_{lhip,knee} \\ k_{rhip} \\ k_{rknee} \\ k_{rankle} \\ k_{rhip,knee} \end{bmatrix}. \tag{4.2}$$

In total the number of controls is therefore $n_u = 15$, including 7 for the upper body torques, 6 for the single joint stiffness and 2 for the bi-articular coupling stiffness. The torques of the lower body joints are computed as described in Section 3.4.
The rest positions of all the springs and the ankle damping factor are set as free parameters of the optimal control problem, i.e. they are not fixed a priori, but are left free to the optimization to find the best suited value for each analysed motion within imposed limits. Furthermore, by leaving the rest positions free, the errors introduced by the kinematic fit on the hip angle can be compensated, allowing to compare obtained stiffness profiles between the different subjects. The vector of parameters is therefore:

$$\mathbf{p} = \begin{bmatrix} q_0^{hip} \\ q_0^{knee} \\ q_0^{ankle} \\ q_0^{hip,knee} \\ d^{ankle} \end{bmatrix}. \tag{4.3}$$

As we want to reconstruct the original motion, the phase times are fixed to the ones obtained from the motion capture data.

4.2.2 Objective function

We perform a dynamic fit to compute the variable stiffness of joints and bi-articular springs. Reference joint angles $\mathbf{q}_m(t)$ are determined from motion capture data as explained in Section 3.3, then the best fitting trajectories $\mathbf{q}(t)$ of the model are computed with optimal control,

taking into account the whole body dynamics as described in Section 3.2.
To achieve the goal of reconstructing the original motion, the objective consists in minimizing the squares of the fitting error between the motion from motion capture data $\mathbf{q}_m(t)$ and the best fitting trajectories $\mathbf{q}(t)$:

$$\Phi_L(t, \mathbf{x}(t), \mathbf{u}(t), \mathbf{p}) = \|\mathbf{W}_q(\mathbf{q}_m(t_i) - \mathbf{q}(t_i))\|_2^2 + c_t\|\boldsymbol{\tau}(t)\|_2^2 \tag{4.4}$$

where a small torque minimization term as regularization is introduced in order to reduce the effect of measurement noise, therefore c_t is used to adjust the importance of the torque term and should be considerably smaller than the first term, with $c_t \leqslant 10^{-6}$. In this formulation, $\mathbf{W}_q$ is a weighting matrix that scales the contributions of different variables to the fit to take different dimensions and absolute values into account. The first term is evaluated at n_{mp} discrete points corresponding to the points of the joint angles obtained from the original motion capture data. In our specific case, n_{mp} corresponds to the sum of intervals in which each phase is divided into as described in Section 2.2.2.

4.2.3 Constraints

In this case the constraints are as the ones defined in Section 3.6. Boundary constraints are imposed on states, controls and parameters.
The stiffness are constrained to be always positive and a high range is allowed as we are interested in the maximum range it can actually reach. For this reason stiffness profiles are allowed to vary in the range [0,2000] [Nm/rad] for both the joint stiffness and bi-articular couplings. The upper body joint torques are allowed to vary in a range of [-100,100] [N].
The joint positions are also left to vary in a wide range of [-10,10] [rad] as we are also not interested in constraining the motion in any way but we want to exploit the motion fully. However it should be noted that since a fitting is performed, the joint angles are unlikely to deviate with extreme differences from the original ones.
Rest positions of all springs are allowed to vary in the range of [-10,10] [rad] and the damping factor [0,5] [Nm·s/rad].
The phase times are set to be fixed to the ones extracted from the original data, as we want to stay as close as possible to the original motion.

4.2.4 Results and analysis

With the optimal control formulation we computed the the variable stiffness of the torsional springs in the joints and the bi-articular coupling between the hip and knee for all 4 subjects. The average of these is computed for better analysis, which corresponds to the red lines in Fig. 4.3, where the grey area is the minimum and maximum.

The results in Fig. 4.3 are the plots of stiffness of the left and right leg joints (i.e. hip, knee, ankle) over a single step, where the right foot is always in contact with the ground and the left foot is partly in the air. The sequences of these phases for each scenario are as described in Section 3.5.

Since the phases times between subjects is different, a time normalisation is performed for each phase in order to compute the average and plot the results. Due to this difference, also the number of discretized points of the controls are different for each subject, therefore the data are interpolated to compute the average. Note that in the plot the scaling of the stiffness of springs in the joints and bi-articular springs are different, in fact they have maximum in

2000 [Nm/rad] and 150 [Nm/rad] respectively. Even if the maximum limit of the bi-articular springs was set to a much higher value in the optimization, in the final results they reached a maximum of 150 [Nm/rad].

Even if the four subjects have similar joint trajectories in the legs as can be observed from Fig. 4.4, the minimum obtained for each of their stiffness profiles is very different. In fact, from Tab. 4.1 we can see that step length, time and velocity can be quite different as well as the subjects weight and height. For this reason, there are several intervals where the minimum of the stiffness profiles of different subjects is about 0 and this is the reason for which the minimum in Fig. 4.3 is very small.

We computed the average fitting error for the leg joints, as they are the most interesting for walking. The error is computed as average of all the subjects and walking scenarios and is in an acceptable range around 0.1 [rad] for the hip, 0.13 [rad] for the knee and 0.14 [rad] for the ankle joint. Higher errors on ankle joints are understandable due to the flat foot model which does not include the human capability of rolling. It is also due to the fact that in the kinematic mapping from motion capture data to jont angles environment constraints such as the slope inclination and stairs size were not introduced, while they are included in the dynamic fit as described in Section 3.6. This is also a reason of the higher error on the knee joint and of possible fitting errors in the dynamic fit.

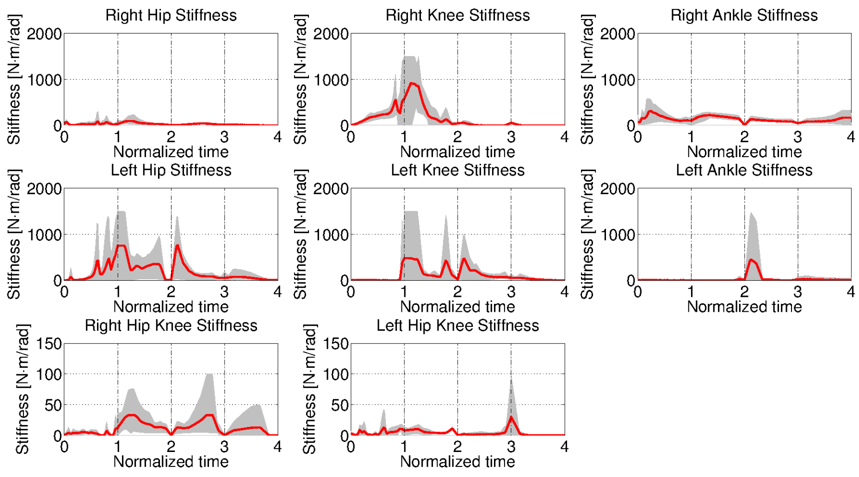
We can observe from Fig. 4.3a that when walking on level ground, the stiffness modulates in high ranges among the different walking phases. The bi-articular springs present higher stiffness in the right leg with respect to the left leg, while in the joint the right knee has a peak at the beginning of phase 2, corresponding to when the left leg is in swing phase and the right toe is the only contact with the ground. In the left leg the hip joint stiffness is overall high in the first two phases, which corresponds to the swing of the leg, while in phases 3 and 4 the left foot is already on the ground. The left ankle stiffness has a single high peak at the beginning of phase 3, corresponding to the impact of the left heel.

Unlike walking on level ground, we can see from Fig. 4.3b that when walking up a slope, the stiffness of the left leg joints is much smaller and almost constant among all the walking phases. The hip stiffness has higher values in the right leg, which is the stance leg, due to the higher work of the hip during slope walking [38]. In the left ankle a peak can still be obeserved at the beginning of phase 3, but much smaller than the level ground case. In this case the bi-articular spring stiffness profiles have much lower values than level ground, where in the last two phases the left leg has almost zero stiffness.

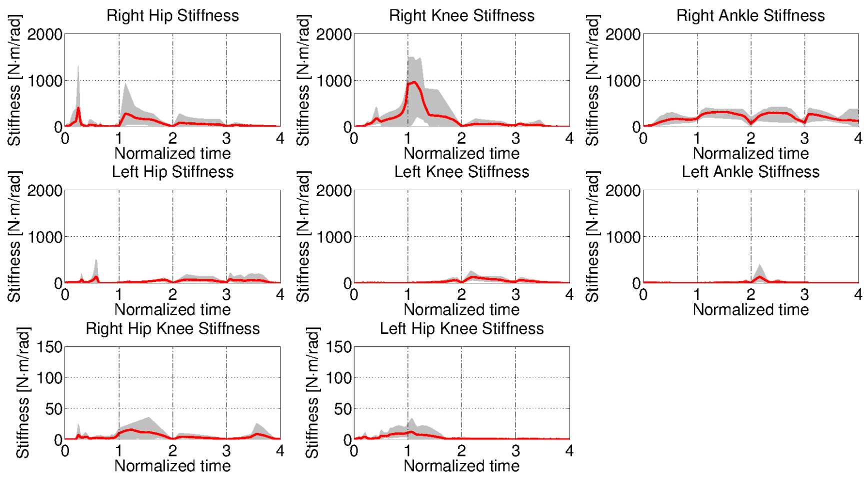
While climbing up stairs, the overall joint stiffness is lower than the level ground case and bi-articular coupling stiffness has higher values in both legs, as can be observed from Fig. 4.3c, but also in this case the right leg has higher values than the left leg. A peak can be observed in the right knee between phases 2 and 3 at the impact of the left foot with the ground, as after the impacts of the previous two cases. Here the left hip stiffness has higher values during phase 2, due to the swing of the leg from one step to the next one.

During stair and slope walking the ankle and knee angle trajectories, in particular in the left leg, vary less than in the level ground case, as can be observed from Fig. 4.4. This could also explain the stiffness of these joints having lower values than the level ground case. For the hip joint, instead, it is more bent during slope walking, which explains the higher stiffness of this joint in this scenario.

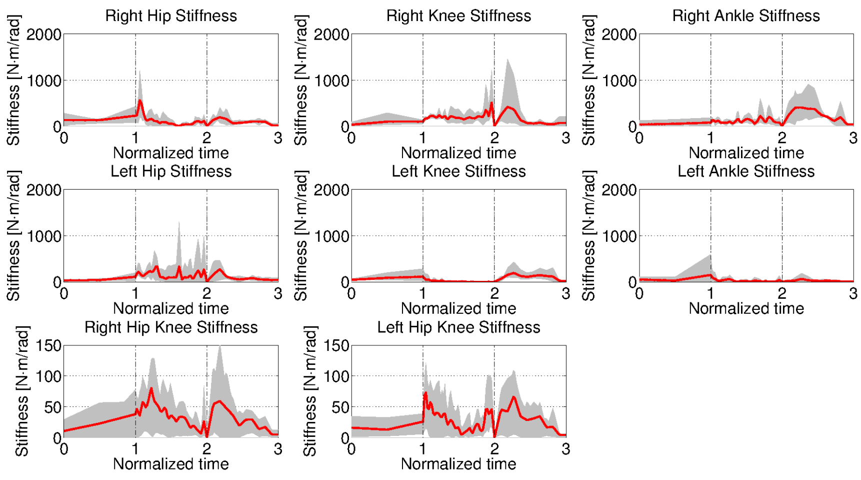
Summarizing, the stiffness profiles obtained for walking in the three considered environments have some common features but present overall very different properties among the walking



(a) Level ground

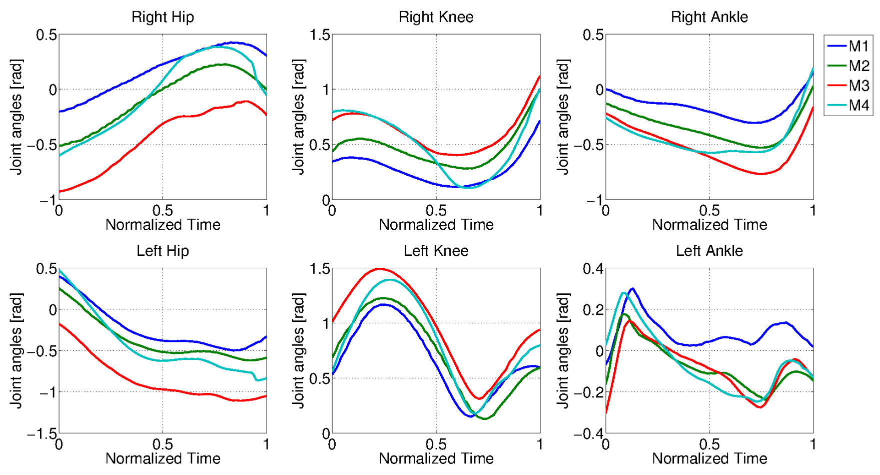


(b) Slope up

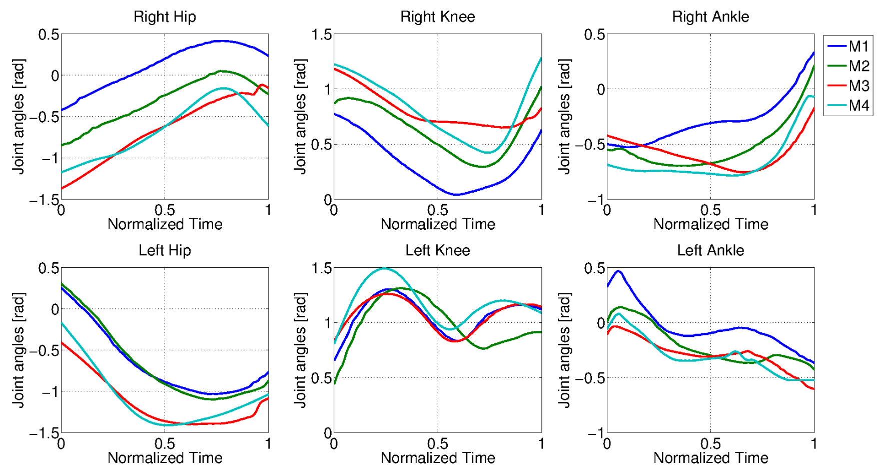


(c) Stairs climbing

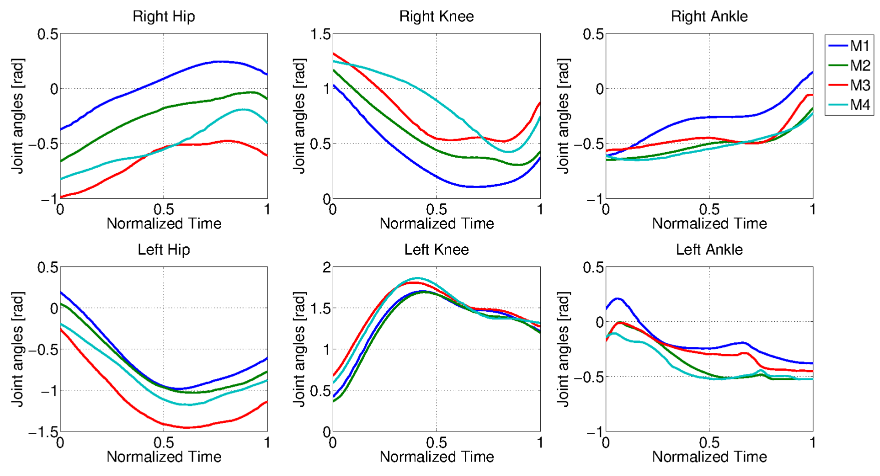
Figure 4.3: Variable stiffnesses of leg joints in the different walking scenarios. The red line is the average of all the 4 subjects, the area between the minimum and maximum among all subjects is represented in grey. The time on the x axis is normalized for each phase, which are divided by the vertical dotted line. Phase 1 is between 0 and 1, phase 2 between 1 and 2 and so on.



(a) Level ground



(b) Slope up



(c) Stairs climbing

Figure 4.4: Joint angles of the lower body joints converted from the motion capture data. Time is normalized to have comparable trajectories. Hip joint angles include different offsets for different subjects which however did not influence the stiffness profile of the joint, only the resulting rest position of the spring.

scenarios and can hardly be generalized for walking in any envinronment. In all the scenarios high modulations of the joint stiffness were observed in both stance and swing legs, in particular the knee joint stiffness of the stance leg during single support phase is higher than the other joints. The maximum stiffness reached among all subjects and environment is about 1500 [Nm/rad], while the bi-articular springs of 150 [Nm/rad].

In [49] a model with less DOF was used and bi-articular couplings were not included. In this case, in some phases the joint stiffness reached higher values than the ones obtained with the current model with bi-articular couplings. This means that the introduction of bi-aricular coupling could reduce stiffness, but the modulations are still high and can be hardly defined as constant through all the walking phases in all the three scenarios.

The obtained results show how stiffness modulation is an important feature in human gaits. It can have high modulation in small time intervals and a big range is also needed. In all scenarios, we also observed peaks in the stiffness profiles after impacts, which means that a jump in the stiffness level happens after a phase change due to impact.

4.3 Influence of compliance modulation on walking

Stiffness profiles obtained from Section 4.2 show how stiffness at joint and bi-articular level have high modulations, from which it is possible to deduce that stiffness modulation is characteristic of human walking. However, in many robotics applications constant stiffness is often used instead of variable stiffness. In particular in walking devices using physical elastic elements, constant stiffness is preferred, as state of the art variable stiffness actuators are still too big in size and heavy in weight to be conveniently used in real life applications.
Therefore, in this section we want to take a step further by analyzing the importance of the stiffness modulation and its effect on walking motions by computing the derivatives of the stiffness profiles. The objective is to understand if constant stiffness allows the generation of human like gaits and if this is not the case, how much modulation either in passive elements or active control would be needed for the walking machines to achieve such performances. From these considerations we can also obtain insights for the design of variable stiffness actuators, e.g. in terms of range of modulation and phases in which a change of stiffness is desired.

The main questions we want to address are:

- How much does the stiffness modulation at joint level influence walking motions?
- Is it possible to reproduce the same walking gait with constant stiffness over the whole walking cycle?

The problem is formulated in a similar way as in Section 4.2, by formulating an optimal control problem. The difference lies in the definition of the states, controls and parameters as well as the objective functions.

4.3.1 States, controls and parameters

We want to study the influence of joint stiffness modulation, hence we treat stiffness as part of the states vector, while the modulation, i.e. the stiffness derivatives, are the controls. This means that, with respect to the model as formulated in Section 3.2, the state vector is represented by the generalized coordinates and velocities and stiffness of the joints and the coupling springs:

$$\mathbf{x}(t) = \begin{bmatrix} \mathbf{q}(t) \\ \dot{\mathbf{q}}(t) \\ \mathbf{K}(t) \end{bmatrix}, \quad \dot{\mathbf{x}}(t) = \begin{bmatrix} \dot{\mathbf{q}}(t) \\ \ddot{\mathbf{q}}(t) \\ \mathbf{u}_L(t) \end{bmatrix} \tag{4.5}$$

where the stiffness profiles are:

$$\mathbf{K}(t) = \begin{bmatrix} k_{lhip} \\ k_{lknee} \\ k_{lankle} \\ k_{lhip,knee} \\ k_{rhip} \\ k_{rknee} \\ k_{rankle} \\ k_{rhip,kneek} \end{bmatrix} \tag{4.6}$$

which contribute to the actuation of the leg joint as per the description in Section 3.4, and $\mathbf{u}_L(t)$ are the controls of the lower body joints:

$$\mathbf{u}_L(t) = \dot{\mathbf{K}}(t) \tag{4.7}$$

being $\dot{\mathbf{K}}(t)$ the derivatives of the stiffness profiles.

The control inputs are composed similarly as in Section 4.2, from the upper body torques and in the lower body, instead of the stiffness, the derivatives. In total the number of controls is $n_u = 15$, including 6 for the single joint actuation stiffness profiles derivatives, 2 for the bi-articular stiffness profiles derivatives and 7 for the upper body torques.

As we have observed from Section 4.2 that the stiffness profiles have high jumps between phases, in particular when impacts occur, we treat these discontinuities by means of slack variables. This means that at phase change we introduce the slack variables by imposing:

$$\mathbf{K}^{+}(t) = \mathbf{K}^{-}(t) + \overline{\mathbf{p}}_{slack,*}, \tag{4.8}$$

where the subscript $*$ indicates the different discontinuity phases, i.e. impacts. This means that we have:

$$\mathbf{p}_{slack} = \begin{bmatrix} \overline{\mathbf{p}}_{slack,1} \\ \overline{\mathbf{p}}_{slack,2} \end{bmatrix}, \quad \overline{\mathbf{p}}_{slack,*} = \begin{bmatrix} p_{lhip,*} \\ p_{lknee,*} \\ p_{lankle,*} \\ p_{lhip,knee,*} \\ p_{rhip,*} \\ p_{rknee,*} \\ p_{rankle,*} \\ p_{rhip,knee,*} \end{bmatrix} \tag{4.9}$$

with subscript $*$ being either 1 or 2 in the case of level ground and slope walking, only 1 in the case of stair walking (as there is only one discontinuity).

The whole set of parameters is then represented by:

$$\mathbf{p} = \begin{bmatrix} \mathbf{p}_{springs} \\ \mathbf{p}_{slack} \end{bmatrix}. \tag{4.10}$$

where $\mathbf{p}_{springs}$ is defined as the same set as in Eq. (4.3).

4.3.2 Objective function

Similarly to Section 4.2, a motion fitting is performed. Alongside the motion fitting, an analysis on the influence of stiffness modulation over a single phase and over the whole walking cycle needs to be performed. So the following three objective functions are defined:

- Minimization of fitting error, different than Eq. 4.4 the regularization term of minimzing joint torques is removed, as regularization is introduced with the other objective functions.

$$\Phi_{fit} = ||W_q(\mathbf{q}_m(t_i) - \mathbf{q}(t_i))||_2^2, \tag{4.11}$$

- Minimization of the squares of the derivatives, which are the controls, to minimize the stiffness modulation.

$$\Phi_{der} = \mathbf{u}^T \mathbf{u} \tag{4.12}$$

- Minimization of slack variables, i.e. stiffness jumps after impacts, allowing to force stiffness to be continuous over the step cycle.

$$\Phi_{\Delta K} = ||\mathbf{p}_{slack}||_2^2 \tag{4.13}$$

The above defined three objective functions are combined into one with different weights:

$$\Phi_L = c_{fit}\Phi_{fit} + c_{der}\Phi_{der} + c_{\Delta K}\Phi_{\Delta K} \tag{4.14}$$

where the weights act also as scaling factors, taking into account that the three objectives involve quantities with different units and different scaling.

4.3.3 Constraints

As in Section 4.2, the constraints are the ones defined in Section 3.6. Boundary constraints imposed on states, controls and parameters are imposed similarly to the previous problem of computing stiffness profiles.
The difference is that the boundaries on the stiffness are not on the controls but on the states, while the controls represented by the stiffness derivatives $\mathbf{u}_L(t)$ are allowed to vary in the range of [-1000, 1000] [Nm/rad·s], as it has been shown to be enough when the objective functions on reducing the stiffness modulation is used.
We also computed an initial case in which the stiffness derivatives boundaries were imposed to [-100000, 100000] [Nm/rad·s] where the only objective function used is the fitting error minimization, in this case we verified that these limits have never been reached with any of the trials.
The slack variables are allowed to vary in the range [-2000, 2000] [Nm/rad] such that a full jump in either positive or negative directions is allowed.

4.3.4 Results and analysis

We obtained the stiffness profiles and their derivatives for the four male subjects and the three environments as described in Tab. 4.2, by using different combinations of the objective functions as described in Section 4.3.2 with sets of weighting factors as in Tab. 4.3, which are set in order to achieve our goal of analyzing the influence of stiffness modulation on walking motions.

Ref. number	c_{fit}	c_{der}	$c_{\Delta K}$
1	10	0	0
2	10	0	0
3	10	10^{-8}	0
4	10	10^{-8}	10^{-8}
5	10	10^{-8}	10^{-4}
6	10	10^{-8}	1
7	10	10^{-6}	0
8	10	10^{-6}	10^{-8}
9	10	10^{-6}	10^{-4}
10	10	10^{-6}	1
11	10	10^{-4}	0
12	10	10^{-4}	10^{-8}
13	10	10^{-4}	10^{-4}
14	10	10^{-4}	1
15	10	10^{-2}	0
16	10	10^{-2}	10^{-8}
17	10	10^{-2}	10^{-4}
18	10	10^{-2}	1

Table 4.3: Weights combinations used to obtain the results. The first two rows are the same, but differ in the limits used for the controls. In Ref. n. 1 the limits are left open, while in Ref. n. 2 is contrained in the range [-1000, 1000] [Nm/rad· s]. Please note that Φ_{fit} is of 10^3 order smaller than Φ_{der} and Φ_{slack}, reason for which the weights are not of the same order.

In particular, we performed a fit only first, where the objective function as in Eq. (4.11) alone was used, in order to compute the profiles under no constraints on the modulation, to obtain results with which we can compare the cases where modulation is punished.In the other cases, we used different sets of weights on objective functions of Eq. (4.12) and Eq. (4.13). Specifically we first minimize the stiffness derivatives with increasing weights to reduce the modulation of different amounts, in this way we obtained stiffness profiles that resulted to have smaller modulations until being almost constant in each phase, but still with big jumps across the phases. Then incremental weights are also used with objective function as in Eq. (4.13) to reduce the jumps between the phases, constraining the stiffness profiles to be almost constant over the whole walking cycle. All the sets of weights that are as in Tab. 4.3.

The fitting error is computed for all the considered sets of weights as average over the sampling points and all the leg joints:

$$e = \frac{\sqrt{\sum_{j=1}^{n_{joints}} \sum_{i=1}^{n_{mp}} (q_{m,j}(i) - q_j(i))^2}}{n_{mp} \cdot n_{joints}} \tag{4.15}$$

where $n_{joints} = 6$ in our case, n_{mp} is the same as in the optimal control problem formulation, $q_{m,j}(i)$ and $q_j(i)$ are the reference and fitted joint angle trajectories at sampling point i of joint j.

A subset of the trajectories and profiles obtained for one subject only (M3) is plotted in Fig. 4.6 and 4.5 here as example of the results for clarity reasons. The results of all cases and subjects is available in Appendix B.

We can observe from Fig. 4.5 that the obtained results for Ref. n. 1 are comparable with the ones we obtained in the Section 4.2. In particular the stiffness profiles present many spikes, mostly corresponding to when an impact occurs causing a phase change and thus a discontinuity in the dynamics. In this case, the maximum modulation that we obtained is in average, for all the subjects, of order 10^4 [Nm/rad·s], while in the case of combinations from Ref. n. 15 on, values smaller than 1 [Nm/rad·s].

The effect of weights affect the modulation of stiffness and jumps between phases can also be observed from the subset extracted from M3 in Fig. 4.5. Ref. n. 2 is the case when no punishment on modulation or jumps is imposed, but the stiffness derivatives were constrained to [-1000, 1000] [Nm/rad·s], and it is possible to observe how this already affects stiffness modulation. In Ref. n. 6 we can observe that no jumps occur between phases, differently than Ref. n. 1 and 2. In the cases of Ref. n. 15 and 18 a high weight is set on stiffness modulation, therefore we can see that stiffness profiles do not modulate within the single phases for Ref. n. 15 and are basically constant with Ref. n. 18. It is interesting to notice in the case of Ref. n. 15, how stiffness switches with the phases, assuming very different values after impacts occur.

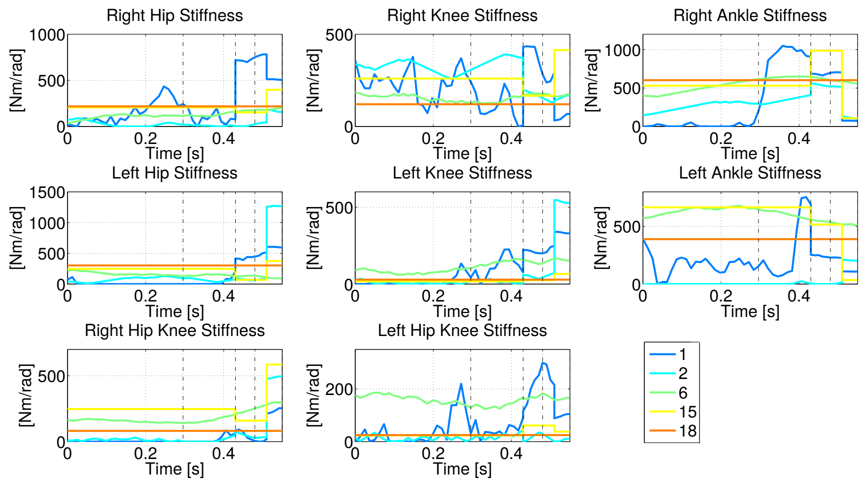
From Fig. 4.7 we can observe how the fitting error varies with the set of weights for each subject, showing how it increases with the increase of the weights on the objective functions on reducing stiffness derivatives and slack variables.

From Fig. 4.7 and Fig. 4.6 we can see that in the cases of level ground and walking up a slope, the fitting error is not increasing in a significant way with growing weights on modulation and jumps minimization, and thus the obtained trajectories do not have high deviations from the reference data. Taking again M3 as example, in the level ground case between Ref. n. 18 and Ref. n. 1 there is an increase of 75% in fitting error, while in slope case of 100%. In the case of stairs climbing the errors have a big increase when the weights are higher and thus the stiffness profiles almost constant. Leading to up to 17 [deg] average errors in M3, with an error increase of 240% between Ref. n. 18 and 1. From Fig. 4.6c we can see that the biggest error lies in the hip joint, where the angle becomes bigger with the increase of the weights, which means that when constant stiffness is imposed, the obtained optimal trajectories lead to lower upper body heights. The importance of the role of slack variables representing stiffness jumps could be observed from Fig. 4.7, where as the weight on stiffness jumps increases, in general there is also an increase in average fitting error.

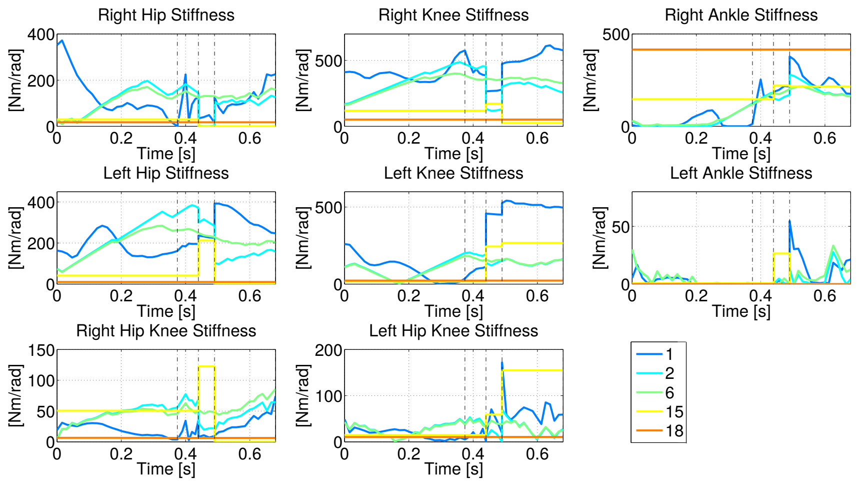
From these results we may conclude that in the cases of walking on level ground and small slopes, we could not observe a significant influence of the stiffness modulation on the walking gait. In the case of stairs climbing, when both the modulation and jumps are minimized with high weights, the error is much higher than the other two scenarios, which leads to the conclusion that in stair walking stiffness modulation plays a role that is more relevant than walking in unconstrained environments. However, despite the smaller errors in the cases of slope and level ground, from the results it is certain that the modulation improves the motion fitting. We can also observe that the overall behaviour of the fitting error is similar in the four subjects, however the range of errors is not the same, due to errors that are already caused by the kinematic fit that is performed to obtain the joint trajectories from motion capture data. This is the same reason for which a mean fitting error of the four subjects was not computed.

From the observations made on fitting errors and the obtained stiffness profiles, we can say that when active compliance control is used to generate motions, stiffness modulation should be taken into account in the controller design where continuous modulation should be a desired feature. For walking mechanisms using physical compliant systems, we can gain insights on possible designs of variable stiffness actuators. As continuos fast modulation of stiffness is highly difficult to achieve, we showed that in unconstrained environments it is possible to recreate walking gaits without big deviations from the original motion also with constant stiffness over the whole walking cycle, but with different stiffness values in the different joints. The gait can however be improved if the stiffness can be changed for different walking phases as shown in the case of Ref. n. 15, in particular when contrained environments are involved. Therefore, a trade-off between constant and highly modulated stiffness consists in the possibility of switching the stiffness value when transitioning into a different walking phase. The range of this switch is however high as we can see from Fig. 4.5, and the values obtained in this thesis for the different joints can be used as reference. Since stiffness as function of joint angles might be desired, we want to point out that despite stiffness is varying as a function over time $\mathbf{K}(t)$ as per our formulation, it is possible to recover $\mathbf{K}(\mathbf{q})$, as we compute also $\mathbf{q}(t)$ in the optimal control problem.

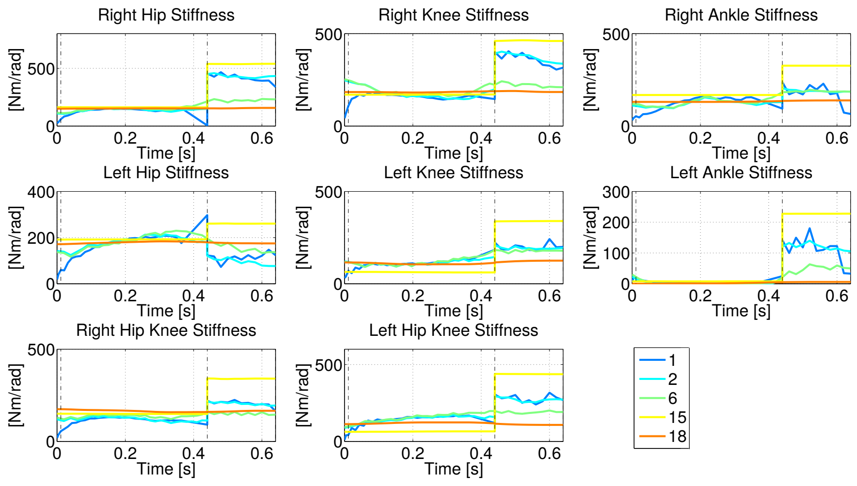
We want to remind that these observations are based on previous assumptions and simplifications of the model, which we cannot completely neglet in analyzing the outcomes of the optimal control problem. We assumed that the foot is flat with only two points contact, while it is well known that the foot has a typical rolling contact with the ground and it represents a critical feature for walking. Also, we considered that the motion is in 2D and thus did not take into account the effects on the frontal plane. Despite the simplifications, the results we obtained still give an important overview on how the stiffness modulation influences the walking gait from the sagittal plane point of view, which is the one that gives most insights about walking and the most interesting for walking mechanisms.



(a) Level ground

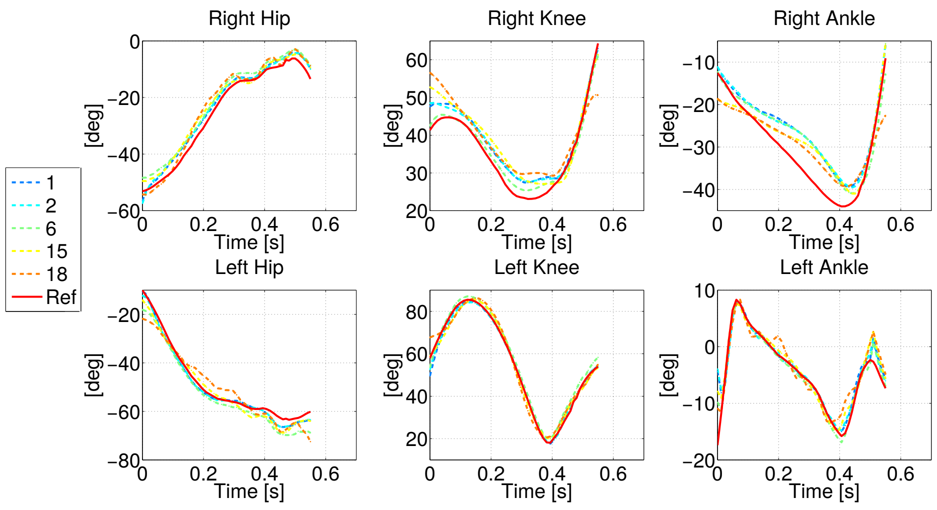


(b) Slope up

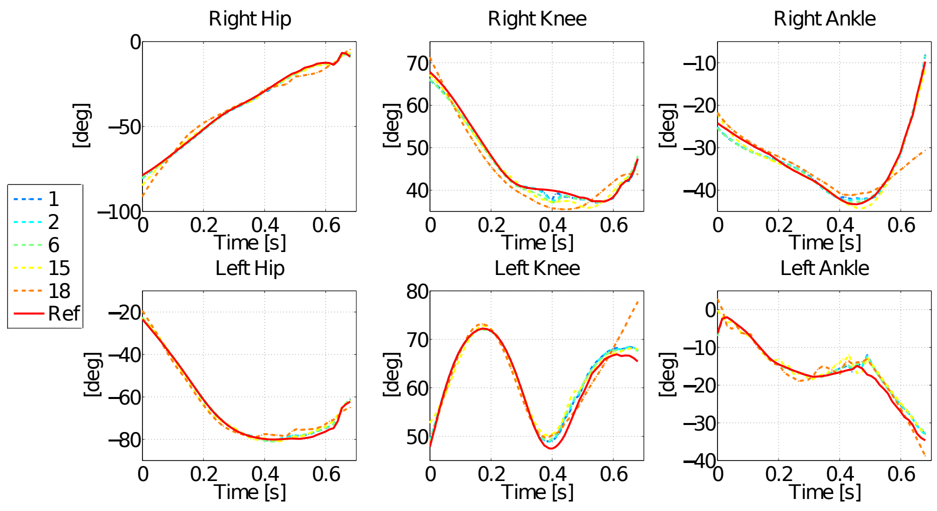


(c) Stairs climbing

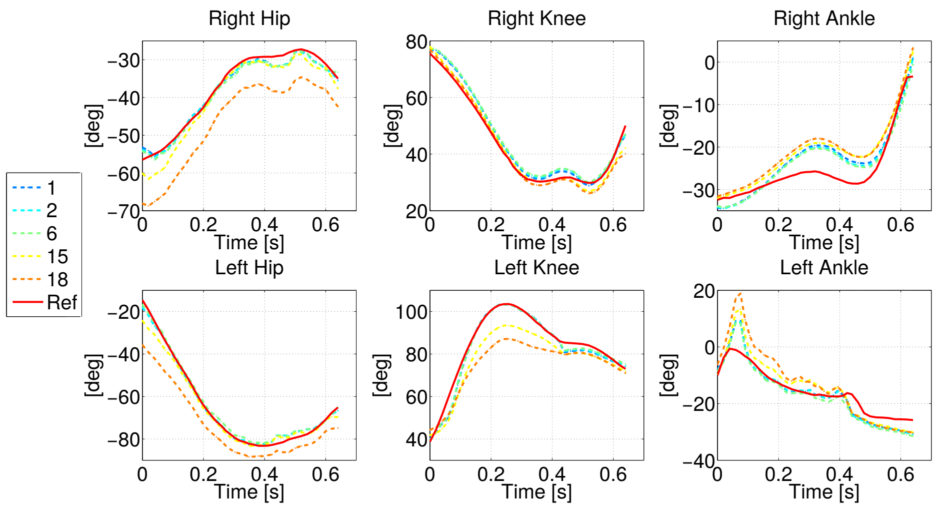
Figure 4.5: Representative subset of stiffness profiles of one subject (M3) obtained from cases as per Tab. 4.3. Please note that all the plots have different scaling.



(a) Level ground

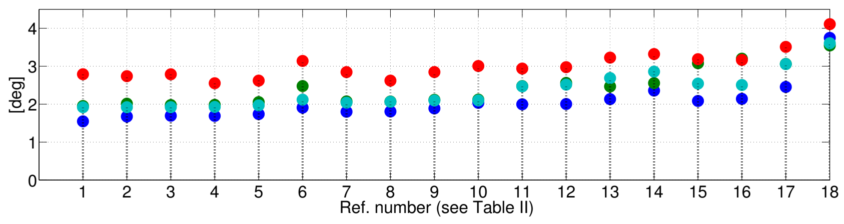


(b) Slope up

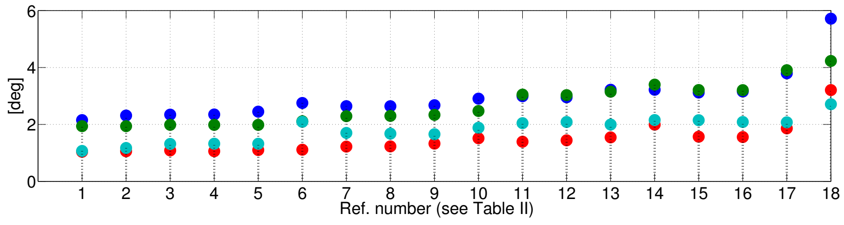


(c) Stairs climbing

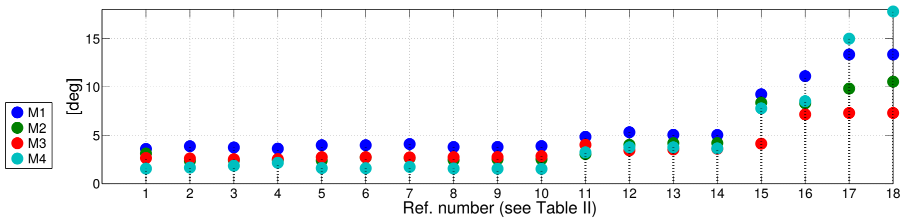
Figure 4.6: Representative subset of joint angles trajectories of one subject (M3) obtained from cases as per Tab. 4.3. Please note that all the plots have different scaling.



(a) Level ground



(b) Slope up



(c) Stairs climbing

Figure 4.7: Average standard deviation of the leg joint angle trajectories of the model from the original reference data of all 4 subjects in the three walking environments. Each point indicates a different set of weights combinations of the objective functions as per Tab. 4.3.

5 Conclusions

In this part we illustrated how human walking motions can be transferred from motion capture data to computational models, first with kinematic mapping to extract joint angle trajectories then with optimal control to perform dynamic fitting with rigid multi-body models under dynamic constraints.

The main interest was to analyze the modulation of joint and bi-articular coupling stiffness during walking motions. To achieve this goal, a 2D human model with 13 segments and 16 DOF was used, which dynamic properties such as mass and inertia are obtained from a scalable meta model. The joint trajectories are mapped on this model via an optimal control formulation considering dynamic and environmental constraints. This was carried out for three different environments: level ground, walking up a slope of 15 [deg] and climbing stairs.
Four male subjects walking on identical environments were considered and the stiffness profiles of each performing a periodic step on each of the environment were obtained. The obtained profiles present high modulation over the walking phases and can assume very high values. In particular joint stiffness of the stance leg are overall higher than the swing leg, and independently from the walking scenario and subject, the stiffness at joint level is much higher than the bi-articular couplings. High jumps and peaks occuring after impacts were also observed. However, comparing to the results obtained in a study on running [77], the values of the stiffness are much lower.

As many state of the art walking mechanisms use elastic elements with constant stiffness, we were interested in the actual influence of stiffness modulation on the walking gait, as according to our results stiffness is *not* constant but is characterized by rather high modulations.
To carry out this analysis, we used the same approach as the computation of stiffness profiles, but by introducing the derivatives of the stiffness as controls of the optimal control problem and formulating the objective as a combination of the minimization of fitting error, stiffness modulation in terms of stiffness profiles derivatives and of slack variables in terms of stiffness values jumps after impacts. By varying the weights of each objective, we obtained a set of stiffness profiles from which we could gain interesting insights on the effects of stiffness modulation on human walking in the three considered environments.
To answer the two questions posed at the beginning of Section 4.3, we observed that by reducing stiffness modulation to obtain almost constant stiffness, in the case of unconstrained walking (level ground and slope) the gait can be still vaguely approximated, but with bigger deviations from the original joint trajectories than the case with variable stiffness. In the case of stair climbing, the modulation has higher influence on the gait, as the deviations are much bigger. This means that the mechanisms that have compliant components with constant stiffness can help to reproduce more human-like walking gaits in unconstrained environments, but can have difficulty in more complex and constrained ones. In particular, these mechanisms should allow the possibility of switching stiffness values at least when changing walking phases, as continuos stiffness modulation is currently difficult to achieve.

As already pointed out in the results discussions, the analysis were performed basing on many assumptions and simplifications, such as the model in 2D, the flat foot with only two contact points and the structure of the springs introduced in the joints and the bi-articular coupling. Also, we neglected some factors that are being treated in other works, such as the gender and the age of the subjects. Gender related joint stiffness differences were highlited in [39] and relevant differences between yound and elderly men in ankle stiffness during stepping down motions was shown in [64].

With the model we used a converged solution could be found in around $400 \sim 500$ SQP iterations using MUSCOD-II for a running time of about $20 \sim 30$ minutes with a desktop computer with CPU Intel i7. Therefore with the simplified model we could obtain interesting results in reasonably short times.
The same setup can be used to conduct the same or similar studies on a high variety of subjects, and further model details can be improved, such as the introduction of rolling contacts instead of the flat foot contact might help in reducing the fitting error on the ankle joint. More complex models such as the 3D models can also be used, as in many optimal control problem the more detailed is the model the better are the obtained results. However, this also results in an increasing complexity of the problem to solve and consequently a consistent increase in computation time.

Part III

Modeling and control of the compliant humanoid robot iCub

6 The (He)iCub humanoid robot

The iCub humanoid robot is a research humanoid robot designed and produced by the iCub Facility department of Fondazione Istituto Italiano di Tecnologia (IIT), located in Genoa, Italy. As per end of 2016, it is the most distributed advanced humanoid robot, present in more than thirty institutions.
The iCub has the appearance of a 4 years old child and is the result of a six years European Project, RobotCub [96], which involved several institutions all over Europe and aimed at developing a platform dedicated to cognitive studies.
One of the objectives of RobotCub was to develop an openly available humanoid robot, therefore all the design details are openly available online, including mechanical design and software packages.

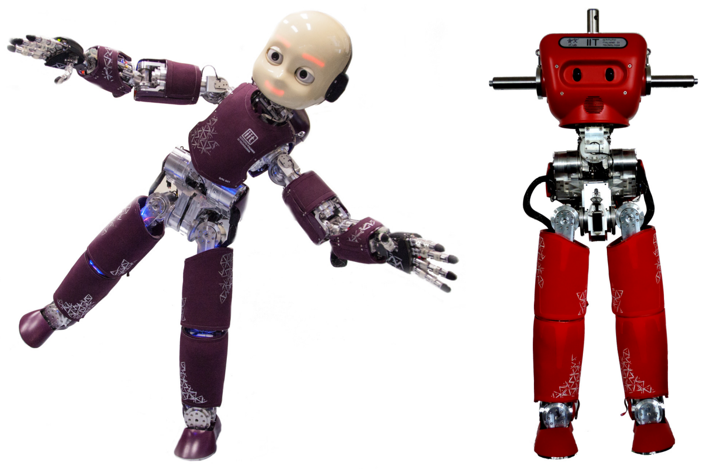


Figure 6.1: iCub and HeiCub robots, designed and built by iCub Facility department, IIT

The iCub version 1.0 had sophisticated mechanical design of the upper body, including eyes and head that are able to rotate as humans. The lower body, instead, was neither designed for standing nor walking tasks, as the feet were also of the size of a 4 years old child, which are too small to support the full weight of the robot while standing.

In order to be able to use the lower body of the robot as well, bigger feet were introduced, allowing the robot to perform the very first balancing tasks. In the latest hardware upgrade, the mechanical design of the legs has been changed [85] by removing the tendon based system which represented a weakness of the legs. The new design is derived from another humanoid robot of the same institute, the COMAN humanoid robot [19] of the Advanced Robotics department, which was designed with the aim of performing walking tasks and had proven to be able to carry out stable walking motions [70].
With the new design of the legs, the iCub was able to perform highly dynamic balancing with fast motions [83], but still very few walking experiments took place.

A reduced and customized version of the iCub was delivered to Heidelberg University (hereafter HeiCub - “Heidelberg iCub”) in the context of the European Project KoroiBot [7], where the aim was to improve walking capabilities of existing humanoid robots. HeiCub is an iCub without head and arms, with a custom torso shape and the latest version of legs hardware, it is the robot that is used to carry out all the studies and experiments of this thesis.

6.1 Description

Until end of 2016 there have been several hardware upgrades of iCub, which range from version 1.0 to version 2.5. The different versions can differ in the design of different parts. The HeiCub has version 2.5 legs. Despite the difference in hardware, all iCubs share the same software. A description of the hardware and the software of the iCub and HeiCub platforms is given in this section.

6.1.1 Hardware

iCub

The standard iCub platform is about 104 cm tall, and weights 32 [kg], the weight can differ among the different versions due to the difference in mechanical designs, in version 1.0 the legs were lighter and the weight was closer to that of a 4 years old child.
The standard iCub needs to be connected to external power source and network in order to operate. There exist also a version which has a battery pack that allows the robot to move without external cables and communications can happen via WiFi. In this case the battery pack adds an extra weight of circa 3 [kg].
It has a total of 53 DOF [75], of which:

- 6 in the head, of which 3 for the neck and 3 for the eyes,
- 16 in each arm, of which 7 for the arm and 9 for the hand,
- 3 in the torso,
- 6 in each leg.

The robot has an onboard PC104 which runs a Debian operating system booted from an external USB. It is equipped with brushless electric motors coupled with harmonic drive gears, with gear reduction of 1 : 100. The sensors include joint encoders in each joint, Inertial Measurement Unit (IMU) located in the head, two digital cameras in the eyes, several accelerometeres and gyroscopes distributed in the body and six custom made 6-axes force torque sensors, of which two in the arms, two in the upper leg and two in the feet. iCub is covered with skin sensors, distributed on its covers. These are pressure sensors that allow to measure the location and the intensity of forces exerted on the body of the robot. The joint positions of the robot are measured via encoders, while the motors positions are measured via motor encoders, which are located before the gear reduction.
Communications are transmitted via Ethernet boards in new versions of the iCub, while in older versions via CAN buses.

HeiCub

HeiCub has the latest standard iCub hardware (as per 2016), but without head and arms. Therefore it has 15 DOF, 3 in the torso and 6 in each leg as shown in Fig. 6.2. It weights 26.4 [kg], it is 0.97 [m] tall, the leg length (from the hip axis) is 0.51 [m], and feet are 0.2 [m] long and 0.1 [m] wide.

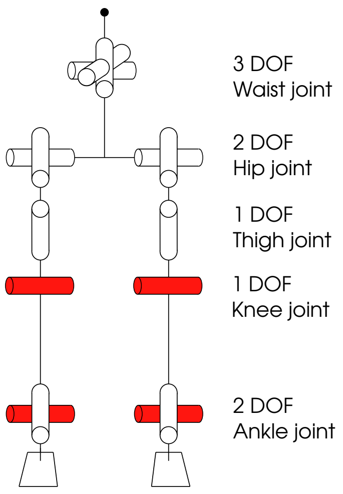


Figure 6.2: Model of the HeiCub humanoid robot, in red the joints with SEA.

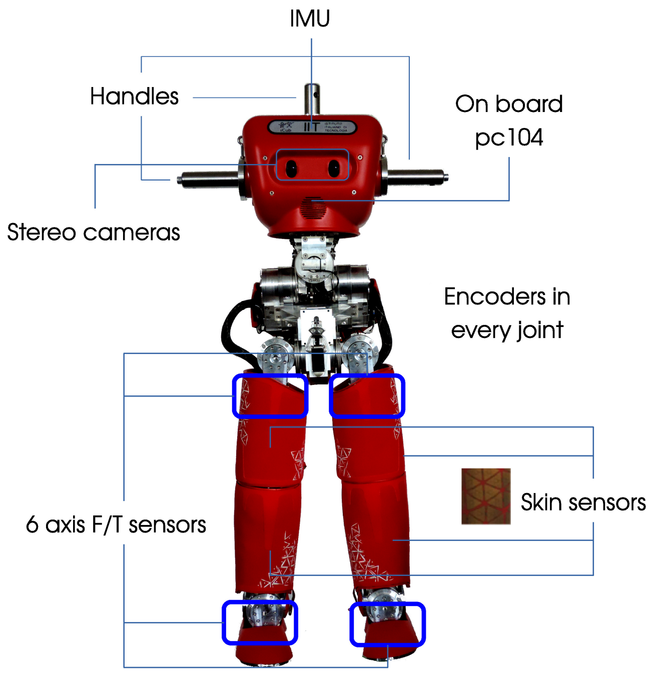


Figure 6.3: Location of all sensors of HeiCub.

The chest of HeiCub is specially designed to compensate for the missing upper body. The three handles as in Fig. 6.3 allow to add extra weight to the robot in a distributed way in order to reach the same weight of the full iCub.
The onboard PC104 is located inside the chest, together with the cameras of HeiCub. The PC104 has an Intel dual core CPU that runs at 2.16 Ghz. The operating system is real time

Debian and is booted from an external USB driver. The cameras are the same as the ones of the full iCub, but they are fixed. The IMU is also located in the chest, and instead of six force torque sensors, there are four, located in the upper leg and the feet. The skin sensors are distributed on the legs only. An overview of the locations of all the sensors is shown in Fig. 6.3.
Most of the joints are driven by a single motor, however the three torso joints and the hip pitch and roll joints are coupled. The three torso joints are driven by two motors and coupled through a tendon system, the same applies for the hip pitch and roll joints. These are related by a coupling matrix.
All joints have mechanical limits corresponding to the physical limits, i.e. over this limit a collision between two consecutive links would happen. The joint limits are as listed in Tab. 6.1.
The current limits are set via firmware for safety reasons, as allowing the user to use the real maximum current might damage the motors. This was initially set to 5 [A] and later extended to allow for dynamic walking motions, as will be described in Chapter 9.
Torque limits are also imposed via firmware, where the maximum is set to 40 [Nm] for all joints.

Table 6.1: HeiCub joint limits

Joint	**Limits** [deg]
l_hip_pitch, r_hip_pitch	[-33, 100]
l_hip_roll, r_hip_roll	[-19, 90]
l_hip_yaw, r_hip_yaw	[-75, 75]
l_knee, r_knee	[-100, 0]
l_ankle_pitch, r_ankle_pitch	[-36, 27]
l_ankle_roll, r_ankle_roll	[-24, 24]
torso_pitch	[-20, 60]
torso_roll	[-26, 26]
torso_yaw	[-50, 50]

Series Elastic Actuator

Both the iCub and HeiCub legs are equipped with four Series Elastic Actuators (SEA), located at the knee and ankle pitch joints of each leg. These actuators are a special compact design [85] of the original Series Elastic Actuators of Pratt et al [87].
The spring of the actuator is a C-shaped spring with a constant stiffness of 350 [Nm/rad] [85].
A peculiarity of the SEA of iCub is that the springs can be unmounted to be replaced with a rigid placeholder, transforming the SEA into a common high execution precision rigid actuator, as in Fig. 6.4.
This feature allows to exploit different capabilities of the robot with the two types of actuators, as well as to compare the behaviour between the two cases. Ideally, the stiffness of the spring could also be changed by using a different spring with desired stiffness.
The deflection of the spring can be measured by measuring the position of the joint by means of the joint encoder and the position of the motor with the motor encoder. In this way the measured deflection consists of not only the one of the spring but includes also the flexibility introduced by the transmission system, but the effect is being negleted for the time being, i.e. the transmission system is assumed to be perfectly rigid, as the proper estimation of additional

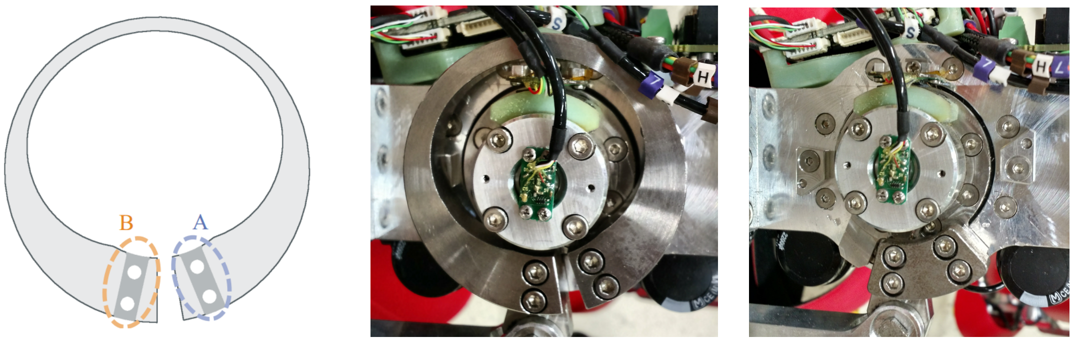


Figure 6.4: From left to right: the C-shaped spring (picture from [85]), the C-shaped spring mounted on the actuator, the rigid placeholder mounted on the actuator.

parameters that contributes to the flexibility of the transmission system is not in the purpose of this thesis.

6.1.2 Software

The software architecture of the iCub family robots is based on the middleware YARP (Yet Another Robotic Platform) [37]. YARP allows abstraction of hardware interfaces and independence from operating system and development environment. As hardware changes often occur, YARP provides functions to interface with the robot a layer above the low level control of the robot, i.e. the firmware. In this way, the developed programs do not need to change function calls at every firmware update or hardware upgrade of the robot.
The YARP protocol allows inter-process communications via *ports*, which can be used to deliver and receive messages of any type and size via the network, giving great advantage in decoupling required functionalities in different programs that can run on independent devices.
Both the middleware and the iCub software packages are openly available, as well as most of the software developed by projects involving the iCub, allowing the abilities of the robot to grow thanks to the contributions of the scientific community.

An example is the software developed within the European project CoDyCo [2], which is now also basic software of the iCub robot and has also been used in this thesis. The project had the aim of developing control frameworks for the iCub to cope with whole-body dynamic interactions with external contacts.
CoDyCo had a strong emphasis on whole-body control, which is also one of the main focuses of recent humanoid robotics developments, as well as of this thesis, as optimization and optimal control are one of the preferred tools to cope with whole-body motion generation and control problems.
To better integrate software and algorithms as well as easy transfer of these also among different robots, the WholeBodyInterface (WBI) [9] was designed as a mean to achieve this goal. It consists in an abstract class of functions divided in sub-interface, each of which targets different aspects of whole-body control:

- wholeBodyStates, which contains functions that allow to access estimated states of the robot.
- wholeBodyActuators, which interfaces with the robot actuators.
- wholeBodySensors, which allows to access all the sensors of the robot.

- wholeBodyModel, which contains functions to access the model of the robot including kinematics and dynamics.

The interface can be implemented for specific robots, e.g. in the case of iCub with YARP functions, and the same software can run on different platforms by adjusting certain entries of an external initialisation file.
The interface was implemented in MATLAB and Simulink (The MathWorks, Inc.) with the WholeBody Toolbox (WBT) [94], which gives the opportunity of using Matlab and Simulink tools to fast prototype new control frameworks before porting it to C++ or other programming languages.

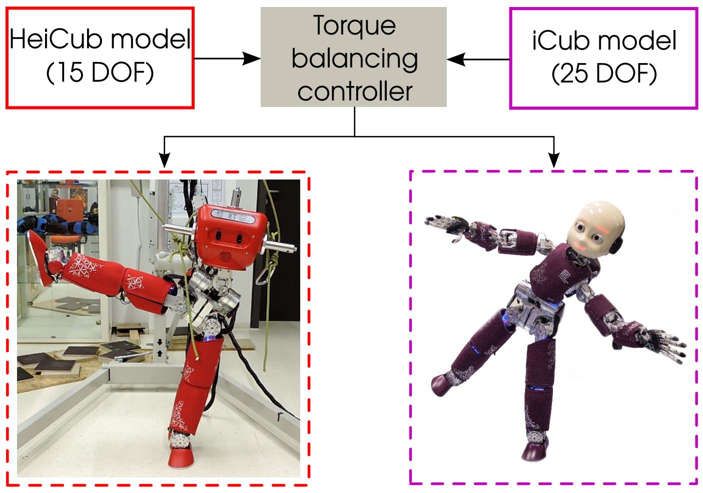


Figure 6.5: The torque balancing controller on the iCub and HeiCub.

For example, the whole-body torque balancing controller [83], which latest version is developed using the WBT, has been used to achieve similar results also on the HeiCub robot without any modification of the code, as shown in Fig. 6.5. The only difference between the two is the model, which is loaded from an external Unified Robot Description File (URDF) [8]. The same was applied also on the much bigger and very different humanoid robot WALK-MAN [81], which is also torque controlled.

6.1.3 Control modes

The iCub has several *control modes* at joint level implemented, which can be used to move the robot. All the currently available control modes are as summarized in Tab. 6.2.
For position and velocity controls there is the possibility of choosing between the *stiff mode* and the *compliant mode*. When using the stiff mode, the desired position and /or velocity of the joint is precisely executed and the external forces are being ignored, while in compliant mode the joint impedance is set with the stiffness σ and the damping factor μ.
There are two position control modes available, the difference is that *position control* uses a minimum jerk trajectory generator to reach the target position, while *position direct* executes the position without any interpolation. Therefore, when a trajectory is precomputed or continuously computed, this should be executed on the robot using position direct control mode.
In the case of *torque control*, since the robot does not have any joint torque sensor, the torques are estimated from the 6-axes force torque sensors using the *wholeBodyDynamicsTree* module [3], based on the dynamic model of the robot.

In all the above mentioned control modes, a PID (Proportional, Integral, Derivative) control

Control Mode	Accepted motor commands	Interaction mode	Additional parameters
Position control	q	Stiff mode Compliant mode	- σ, μ
Position direct	q	Stiff mode Compliant mode	- σ, μ
Velocity control	$\dot{q}$	Stiff mode Compliant mode	- σ, μ
Mixed velocity and position	$q, \dot{q}$	Stiff mode Compliant mode	- σ, μ
Torque control	τ	-	-
Openloop control	φ	-	-
Idle	-	-	-
Hardware fault	-	-	-

Table 6.2: The different control modes of iCub and the corresponding motor commands, where q is the desired joint angle, $\dot{q}$ the desired joint velocity, τ the desired torque and φ the desired PWM. In *idle* mode the robot is not being controlled, the same applies for the hardware fault. The difference is that hardware fault is a mode in which the joint enters for protection when something wrong happened and can be restored to other control modes only by switching the control mode first to idle mode. The user cannot choose to enter in hardware fault mode.

scheme is used to compute the PWM (Power Width Modulation) and correctly track the given commands using feedback errors. The *openloop* control mode instead gives the possibility of skipping the classic PID scheme and directly interface with the motor by sending PWM. The openloop control mode was used to control the SEA.

As in the standard position direct mode the PID loop corrects errors on the joint side, i.e. using the encoder data:

$$PWM = k_P(q_i^d - q_i) + k_I \int (q_i^d - q_i)dt + k_D(\dot{q}_i^d - \dot{q}_i) \tag{6.1}$$

where q_i^d is the desired joint position of joint i and k_P, k_I, k_D are the PID gains, which are tuned to ensure small tracking errors.

When using SEA, with the same gains the control would lead to high oscillations and therefore instability of the system due to the oscillation errors introduced by the spring. Therefore a new control mode was implemented in this thesis to exploit the SEA:

$$PWM = k_P(\theta_i^d - \theta_i) + k_I \int (\theta_i^d - \theta_i)dt + k_D(\dot{\theta}_i^d - \dot{\theta}_i) \tag{6.2}$$

where θ_i^d is the desired motor position of joint i considered after the gear reduction and possible couplings. The PID gains are ideally the same as for position direct mode. We will refer to this control mode as *motor position direct*, it is not present in Tab. 6.2 as this is not one of the standard control modes of iCub robots that is implemented in the firmware of the robot but is used only in the context of this thesis.

6.1.4 Simulation environment

The iCub simulation environment was first a custom developed simulator based on the Open Dynamics Engine (ODE)[1] to which it is possible to interface via YARP in the same way as with the real robot.
In more recent developments, the Gazebo simulator [5] started to become popular in robotics simulation [53]. Gazebo gives the user the possibility to choose between different physics engines, including ODE, Bullet[2] and Simbody[3]. The models are described with files using SD-Format[4], which is similar to the URDF format. Standard URDF can also be used directly for the simulation with appropriate modifications.
In the case of iCub robots, it is possible to interface with the simulator in the exact way a with the robot thanks to the Gazebo-YARP plugins [6, 46], which implements YARP drivers such that the same code tested on the simulator can be transfered to YARP-based robots with no changes or very few changes.

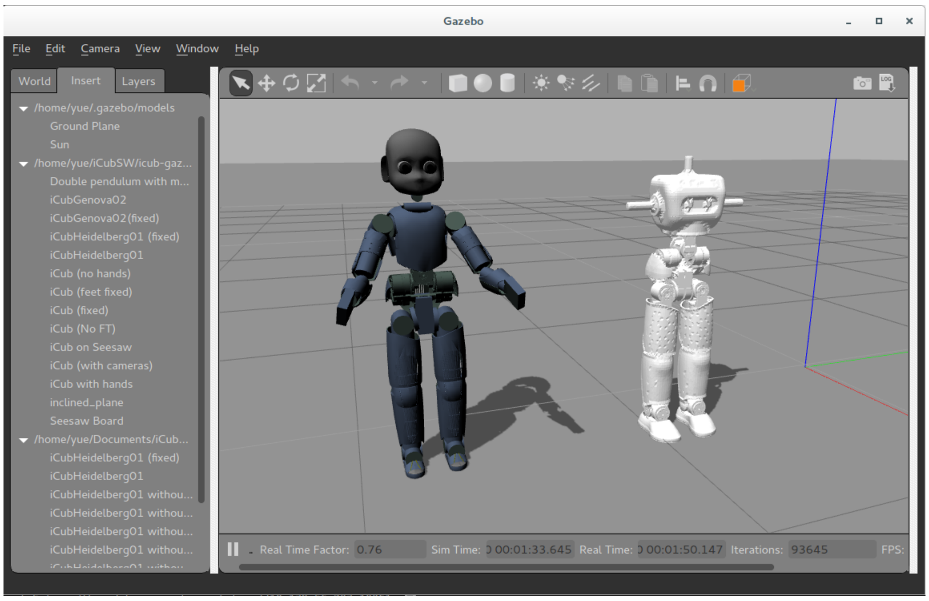


Figure 6.6: iCub and HeiCub models in the Gazebo simulator.

6.2 Kinematic model

The kinematic structure of HeiCub as well as the full iCub is stored inside Unified Robot Description Files (URDF) [8] extracted directly from the CAD model of the robot, which ensures high model precision.
The forward kinematics of a robot describes the transformation from joint space to task space. Given the joints configurations $\mathbf{q}_k$ of the robot, the velocity of the end effector i can be computed as:

$$ {}^{0}\dot{\mathbf{X}}_i = \mathbf{J}_i \dot{\mathbf{q}}_k \tag{6.3} $$

[1]Open Dynamics Engine (ODE) - http://www.ode.org/
[2]Bullet Physics - http://bulletphysics.org/
[3]Simbody - Multibody Physics API - https://simtk.org/projects/simbody/
[4]SDF - Describe your world - http://sdformat.org/

where ${}^{0}\dot{\mathbf{X}}_i \in \mathbb{R}^6$ the linear and angular velocities of the end effector i in the world reference frame and $\mathbf{J}_i$ is the Jacobian of end effector i.

The inverse problem consists in computing the joint angles from known end effector pose, where multiple end effectors can be taken into account. In the case of HeiCub, these are the feet and the center of mass.

The problem of inverse kinematics for humanoid robots is a redundant problem given that multiple solutions can be found. It is possible to solve it analytically [111] or numerically by using recursive methods [109]. The analytical solution is robot specific and lies on certain assumptions that might not apply to all robots, e.g. in humanoid robots the hip joint axes have to meet in the same point. Numerical solutions are more reliable but require longer computation times, therefore if a fast computation has the highest priority, one might choose to compute the analytical solution of the specific robot. In the following the numerical solution is illustrated and adopted in achieving the results of the thesis.

We can formulate the problem by defining first the residuals as in Sugihara's work [109]:

$$\mathbf{e}_j(\mathbf{q}) = \begin{cases} {}^{d}\mathbf{p}_j - \mathbf{p}_i(\mathbf{q}) \\ \mathbf{R}_i^T(\mathbf{q}) \cdot a({}^{d}\mathbf{R}_i \mathbf{R}_i^T(\mathbf{q})) \end{cases} \tag{6.4}$$

where ${}^{d}\mathbf{p}_j$ and $\mathbf{p}_j$ are the desired and current positions of the end effectors j and $a(\cdot)$ is a function that computes angular velocities from the current rotation matrix [5] $\mathbf{R}_i$ and the desired orientation expressed as rotation matrix ${}^{d}\mathbf{R}_j$.

The error is used to update the vector of joint positions, where the update step $k+1$ is performed as per Levenberg-Marquardt [69, 73] method as follows:

$$\mathbf{q}_{k+1} = \mathbf{q}_k + (\mathbf{J}_k^T \mathbf{W}_E \mathbf{J}_k + \mathbf{W}_N^k)^{-1} \mathbf{J}_k^T \mathbf{W}_E \mathbf{e}_k. \tag{6.5}$$

In order to solve for multiple end effectors at the same time, the Jacobian is computed for each specified end effector and stacked into one.

In Eq. (6.5) $\mathbf{e}_k$ is a stack of the errors of all the end effectors:

$$\mathbf{e}_k = \begin{bmatrix} \mathbf{e}_0 \\ \vdots \\ \mathbf{e}_N \end{bmatrix} \tag{6.6}$$

where N is the number of end effectors.

The matrix $\mathbf{J}_k = \mathbf{J}(\mathbf{q}_k)$ is the Jacobian of all the end effectors stacked

$$\mathbf{J}_k = \begin{bmatrix} \mathbf{J}_0 \\ \vdots \\ \mathbf{J}_N \end{bmatrix}. \tag{6.7}$$

The matrices $\mathbf{W}_E$ and $\mathbf{W}_N^k$ are respectively a priority weighting matrix and the damping factor matrix. In this thesis $\mathbf{W}_E = \mathbf{1}$ is applied for simplicity. While $\mathbf{W}_N^k$ is defined according to the

[5] It is also worth noting that the expression of the orientation error depends on the convention adopted for the rotation matrices, the original formulation from Sugihara [109] has been changed to a different convention adopted in the Rigid Body Dynamics Library (RBDL) [33], which is used to implement the numerical solution of inverse kinematics.

modification of the Levenberg-Marquardt method by Sugihara as:

$$\mathbf{W}_N^k = \operatorname{diag} \frac{1}{2} \tilde{\mathbf{e}}_k^2 + \widetilde{\mathbf{W}}_N^k \tag{6.8}$$

where

$$\tilde{\mathbf{e}}_k = \mathbf{J}_k^T \mathbf{W}_E \mathbf{e}_k \tag{6.9}$$

and $\widetilde{\mathbf{W}}_N^k = \operatorname{diag} \tilde{w}_{n,j}$, in this thesis we assume $\widetilde{\mathbf{W}}_N^k = \operatorname{diag} 0.001$, as it was demonstrated to be the best choice in [106].
The solution so obtained does not take into account possible constraints such as joint limits, which is fundamental to obtain feasible solutions for real systems. In the following the problem is being formulated as an optimization problem to be solved for desired objective functions and constraints.

6.2.1 Constrained inverse kinematics

The problem of inverse kinematics subjects to joint limits as constraints can be formulated as an optimization problem:

$$\min_{\mathbf{q}} \quad \tfrac{1}{2} \nabla \|\mathbf{e}(\mathbf{q})\|^2 \tag{6.10}$$

subject to:

$$\begin{aligned} q_i &\geqslant q_i^{min} \quad \text{for } i = 1, ..., n_{dof} \\ q_i &\leqslant q_i^{max} \quad \text{for } i = 1, ..., n_{dof} \end{aligned} \tag{6.11}$$

Where $\mathbf{e}$ is the residuals of stacked position and orientation errors of all end effectors as in Eq. (6.4) and q_i^{min} and q_i^{max} are the minimum and maximum limits of joint i, being n_{dof} the total number of DOF.
The following quantities are defined in order to solve the optimization problem:

- Gradient of the objective:

  $$\nabla \mathbf{e} = \mathbf{J}^T \mathbf{e} \tag{6.12}$$

  where $\mathbf{J}$ is the the Jacobian of all the end effectors stacked $\mathbf{J} = [\mathbf{J}_0, ..., \mathbf{J}_N]^T$.

- Hessian of the Lagrangian function, for which we use a Gauss-Newton approximation due to the complexity of computing the actual Hessian with our chosen library:

  $$\mathbf{H} \approx \mathbf{J}^T \mathbf{J} \tag{6.13}$$

  where $\mathbf{J}$ is the same Jacobian as in the gradient.

In common optimization problems the Jacobian of the constraints would also be required, but since in our case the constraints consist only of the box constraints represented by the joint limits and given that the task of matching the end effector positions are formulated in the objective function, we do not need to formulate a constraint Jacobian.
The openly available software package IPOPT [119] is used to solve the constrained inverse kinematics problem.

Both unconstrained and constrained versions of the inverse kinematics problem have been implemented in the Rigid Body Dynamics Library (RBDL) [33] in C++ programming language. The implementation was carried out as part of the master thesis of Kevin Stein [106], in which the methods have been thouroughly exploited with extensive tests on different models. The algorithms have been benchmarked with analysis of the performances in terms of accuracy and computation time, and it has been observed that the IPOPT [119] based inverse kinematics has computation times that can be 5∼10 times slower than the Sugihara modification if the same accuracy in the result is requested. Therefore a combination of the two methods was proposed, in which the IPOPT version is used to reach results for a lower accuracy and then the Sugihara version is used to refine the solution until the desired accuracy is reached.

6.3 Dynamic model

The dynamic model of the robot can be described with rigid multi-body dynamics as in Section 2.1, where quantities such as inertia, mass etc are stored in the URDF which also contains the kinematic structure of the robot. Special care needs to be taken in the dynamic model when elasticity is considered. In the following, the dynamics when considering the Series Elastic Actuators is described.

6.3.1 Series Elastic Actuator (SEA) modeling

Flexibility of robots can occur in their links and/or in their joints. In any real robotic system there can be flexibility in any of the links and/or joints, however, when rigid actuators are used, it is common to assume that the links and joints are perfectly rigid, as this simplifies the dynamics of the system. In certain cases it might be useful to model the flexibility at joint level also when using rigid actuators, as flexibility might be introduced by the transmission systems, by modeling the joint before and after the transmission system.

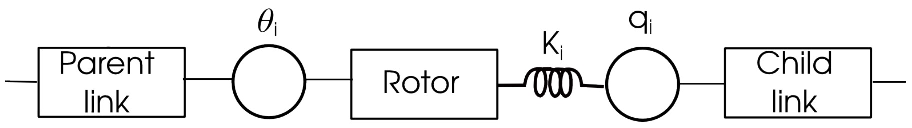


Figure 6.7: Model of a single joint with SEA.

This kind of formulation can be applied when explicit elasticity is introduced, as in the case of SEA. As in all modeling cases, assumptions are introduced to simplify the system. In particular, we model the SEA as per [22]:

$$\mathbf{H}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) + \mathbf{K}(\mathbf{q} - \boldsymbol{\theta}) = \mathbf{0} \tag{6.14}$$

$$\mathbf{B}\ddot{\boldsymbol{\theta}} + \mathbf{K}(\boldsymbol{\theta} - \mathbf{q}) = \boldsymbol{\tau} \tag{6.15}$$

where $\boldsymbol{\theta}$ is the vector of motor positions, **K** the stiffness matrix and **B** a diagonal matrix with the rotor inertia. In this case we assume the stiffness matrix to be also a diagonal matrix containing the stiffness of the springs of the SEA. We neglet the influence of damping and friction, which are parameters that are unknown and would require an identification process to be correctly introduced in the model.
In the rest of the thesis, the values of the SEA are set according to the physical properties of the actuator of the HeiCub. The entries of **K** corresponding to the SEA are set to the stiffness

of the spring of 350 [Nm/rad]. The rotor inertia is of 20.943 [kg·mm^2] for all joints. As the gear is placed before the joint deflection and has ratio $n = 100$, the entries of the matrix **B** are $\frac{\text{rotor inertia}}{\text{gear ratio}}$.
Taking into account contacts, we can rewrite Eq. (6.14) into:

$$\mathbf{H}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q},\dot{\mathbf{q}}) + \mathbf{K}(\mathbf{q}-\boldsymbol{\theta}) = \mathbf{G}^T(\mathbf{q})\boldsymbol{\lambda} \tag{6.16}$$

in which $\mathbf{G}(\cdot)$ and $\boldsymbol{\lambda}$ are the same quantities as in Eq. (2.7), i.e. the contact Jacobian and external forces computed for the set of contact points between the robot and the environment. As we need to compute the unknown motor accelerations, we reformulate the dynamics as in [50], in order to compute $\ddot{q}$ with standard forward dynamics algorithms by using as input torques:

$$\boldsymbol{\tau}_L = \mathbf{K}(\boldsymbol{\theta} - \mathbf{q}). \tag{6.17}$$

Transforming Eq. (6.16) into:

$$\mathbf{H}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q},\dot{\mathbf{q}}) = \boldsymbol{\tau}_L + \mathbf{G}^T\boldsymbol{\lambda}. \tag{6.18}$$

This means that from eq (6.18) the computation of $\ddot{\boldsymbol{\theta}}$ is carried out as:

$$\ddot{\boldsymbol{\theta}} = \mathbf{B}^{-1}(\boldsymbol{\tau} - \boldsymbol{\tau}_L). \tag{6.19}$$

The system taking into account contacts can be formulated as linear system as in Eq. (2.9) for unknown $\ddot{\mathbf{q}}$, $\ddot{\boldsymbol{\theta}}$ and $\boldsymbol{\lambda}$:

$$\begin{bmatrix} \mathbf{H}(\mathbf{q}) & \mathbf{0} & \mathbf{G}(\mathbf{q})^T \\ \mathbf{0} & \mathbf{B} & \mathbf{0} \\ \mathbf{G}(\mathbf{q}) & \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \ddot{\mathbf{q}} \\ \ddot{\boldsymbol{\theta}} \\ -\boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\tau}_L - \mathbf{C}(\mathbf{q},\dot{\mathbf{q}}) \\ \boldsymbol{\tau} - \boldsymbol{\tau}_L \\ -\boldsymbol{\gamma} \end{bmatrix}. \tag{6.20}$$

As for Eq. (2.10), by integrating Eq. (6.15) over a time singleton as in [125], we obtain:

$$\mathbf{B}(\dot{\boldsymbol{\theta}}^+ - \dot{\boldsymbol{\theta}}^-) = \mathbf{0} \tag{6.21}$$

which implies $\dot{\boldsymbol{\theta}}^+ = \dot{\boldsymbol{\theta}}^-$.
Therefore the motor velocities do not change during impact and the impact dynamics remain the same as in Eq. (2.12), which can be rewritten to take into account the dynamics of the elasticity as:

$$\begin{bmatrix} \mathbf{H}(\mathbf{q}) & \mathbf{0} & \mathbf{G}(\mathbf{q})^T \\ \mathbf{0} & \mathbf{I} & \mathbf{0} \\ \mathbf{G}(\mathbf{q}) & \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \dot{\mathbf{q}}^+ \\ \dot{\boldsymbol{\theta}}^+ \\ -\boldsymbol{\Lambda} \end{bmatrix} = \begin{bmatrix} \mathbf{H}(\mathbf{q})\dot{\mathbf{q}}^- \\ \dot{\boldsymbol{\theta}}^- \\ -e\mathbf{G}(\mathbf{q})\dot{\mathbf{q}}^- \end{bmatrix} \tag{6.22}$$

where we also make the assumption of inelastic impact, therefore the coefficient of restitution $e = 0$ is imposed.

The dynamics describing the robot when considering elasticity is implemented by adding further functions to the Rigid Body Dynamics Library [33], details can be found in Appendix A.

7 Squat motion generation

Beside the advantages introduced by the compliant actuators in humanoid robots, the complexity of motion generation and control of the robots also increases significantly. Humanoid robots are well known underactuated and redundant systems where the number of control inputs is much lower than the degrees of freedom.
In the case of physical elastic elements in the system, such as the case of the SEA of the iCub robot, the system can be modeled with generalized coordinates describing both the joint side and the motor side, i.e. after and before the elasticity introduced by the spring of the SEA. This means that the system is modeled with additional generalized coordinates with respect to the case of rigid actuators only, increasing the complexity of planning and control. Furthermore, systems with elastic elements are characterized by non-minimum phase zeros, which require planning of the control.

As the iCub is equipped with the SEA, it is in our interest to exploit the use of the SEA and the effects on the system. We use the squat motion generation to perform this analysis.
The squat motion is a simple motion to perform, but for a system to be able to perform it in a fast and stable way there are many issues to be taken into account. We set up the problem as an optimal control problem. As it has been done with the human model in the first part of this thesis, a similar problem formulation is carried out also for the HeiCub robot. The motion is generated using a set of different objective functions for both the model of the robot with and without SEA.
In this chapter, the model used to generate the squat motion is described, then the optimal control problem defined and the numerical results as well as example implementations on the actual platform are discussed.

7.1 Reduced model description

The full model of the robot as described in Chapter 6 has a total of 15 actuated DOF and 6 DOF for the floating base. For the problem of squatting, only the sagittal plane is considered, as squat is a motion performed in the sagittal plane only.
In a preliminary study [48] we used the model reduced to a 2 DOF manipulator, i.e. the foot was considered to be fixed on the floor and only half of the body of the robot used, as it is assumed that the robot is perfecly symmetric. In this case the knee and ankle pitch joints were used while the hip pitch was constrained to be fixed to a certain angle. The choice of the model was due to the simplicity and also to the fact that the SEA are located in the knee and ankle pitch joints. However, such a model does not allow to exploit the full range of motions that the robot could actually achieve within the joint limits, as with 2 DOF it is difficult to ensure stability when the motion is larger.

Therefore the model used in the rest of this chapter includes the additional hip pitch joint, resulting in a model of 3 DOF. This model allows for larger stable motions that can exploit the full limits of the joints of the robot.
Despite the simplification, the full body dynamic properties of the robot are used in the model,

as in optimal control modeling precision is very important. In fact, the model of the robot consists in a simplification of the full body model by fixing all the joints except for the three leg pitch joints. In this way, the generated motions are fully compatible with the full body model and can be transferred to the real platform without modifications.

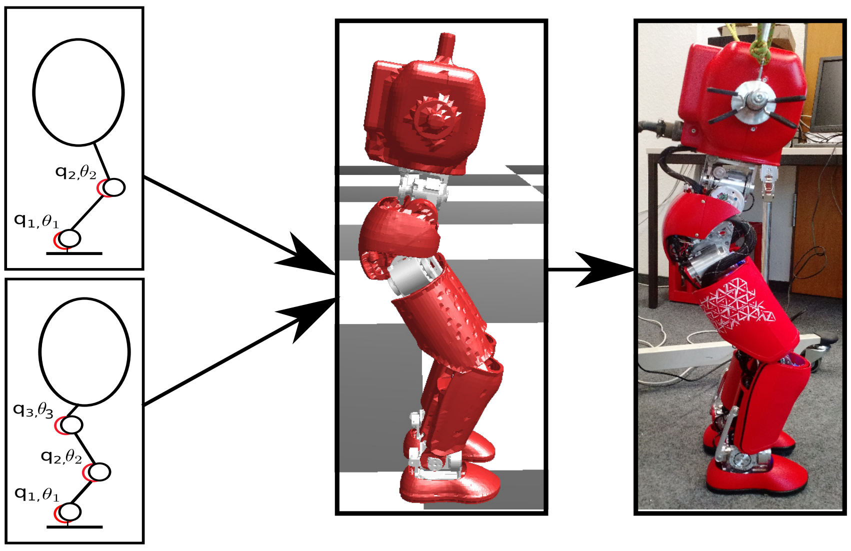


Figure 7.1: Simplified 2 DOF and 3 DOF models with springs are used to generate reference motions.

The dynamics of the robot is described as for fixed base rigid multibody systems without external contacts in Section 2.1 with rigid actuators and Section 6.3.1 for SEA, where the vector of generalized coordinates are:

- For the rigid actuators case: $\mathbf{q} = [q_1, q_2, q_3]^T \in \mathbb{R}^3$.
- In addition for the SEA case: $\boldsymbol{\theta} = [\theta_1, \theta_2, \theta_3]^T \in \mathbb{R}^3$.

Which means that in the SEA case, the number of generalized coordinates is exactly double with respect to the rigid actuators case.
In the elastic case, all the joints are modeled as elastic, including the hip pitch joint which does not have a SEA in the real robot. The difference lies in the entries of matrices **K** and **B**, where the stiffness of the hip pitch joint is set to a very high value. In particular, the stiffness and rotor inertia are set according to the physical data of the robot. Dynamics of the system is computed using the rigid dynamics formulation including elastic joints as in Section 6.3.1, with the implementation of the model as in Appendix A. In this case external contacts are not considered and impacts do not occur as the base is assumed to be fixed and interactions with external environments are not considered.

7.2 Optimal control problem

In the squat motion the robot goes down and then goes back upright, hence the squat problem can be formulated as a two continuous phases optimal control problem, therefore $n_{ph} = 2$. Precisely, the robot starts the motion from a fully stretched upright position and the two phases motion consists of:

Phase 1 Robot goes down to the minimum achievable height.

Phase 2 Robot goes back to the upright position.

The motion is defined as a periodic motion which starts and ends in the same conditions, this means that the obtained motion is ideally repeatable an infinite number of times consecutively. The robot does not follow any predefined trajectory, the whole-body trajectories are left free to the optimal control to find for the best ones given the objective function and constraints.
The optimal control problem is solved using the multiple shooting method as described in Section 2.2.

7.2.1 States, controls and parameters

The states of the optimal control problem are the generalized coordinates and a distinction is made between the two cases with and without SEA:

- with rigid actuators $\mathbf{x}(t) \in \mathbb{R}^6$

$$\mathbf{x}(t) = \begin{bmatrix} \mathbf{q}(t) \\ \dot{\mathbf{q}}(t) \end{bmatrix} \tag{7.1}$$

- with SEA $\mathbf{x}(t) \in \mathbb{R}^{12}$

$$\dot{\mathbf{x}}(t) = \begin{bmatrix} \mathbf{q}(t) \\ \boldsymbol{\theta}(t) \\ \dot{\mathbf{q}}(t) \\ \dot{\boldsymbol{\theta}}(t) \end{bmatrix}. \tag{7.2}$$

The control inputs in both cases are the joint torques:

$$\mathbf{u}(t) = \tau(t) \tag{7.3}$$

where $\tau(t) \in \mathbb{R}^3$.
From the definition of the states it is already clear that the problem is harder to solve when considering SEA, as the number of states is double respect to the case of rigid actuators, but the number of controls remain the same.
The squat problem has a single parameter which consists in the range of the squat $p = p_{range}$, defined as:

$$p_{range} = Z_e - Z_i \tag{7.4}$$

being Z_i and Z_e the height of the root of the robot at the initial position and at the end of the first phase (lowest point of the squat).

7.2.2 Objective functions

Several objective functions are defined to generate different squat motions. As at first the maximum achievable range is unknown, it is computed by using as objective the maximization of the squat range parameter:

$$\Phi_{M,\max p}(\cdot) = c_r \cdot p_{range} \tag{7.5}$$

which is a Mayer type objective where c_r is a weighting factor. The maximum squat range obtained using this function is then kept fixed with the other objective functions defined hereafter.

In total four objective functions are defined:

1. Joint torques minimization

$$\Phi_{L,\min\tau}(\cdot) = \sum_{i=1}^{n_{mp}} c_{\tau} \|\boldsymbol{\tau}(t_i)\|_2^2 \tag{7.6}$$

   where n_{mp} is the number corresponding to the sum of intervals m_{ph} in which each phase is divided into, as explained in Section 2.2. And c_{τ} is a weighting factor.

2. Joint accelerations minimization, with weighting factor $c_{\ddot{q}}$

$$\Phi_{L,\min\ddot{q}}(\cdot) = \sum_{i=1}^{n_{mp}} c_{\ddot{q}} \|\ddot{\mathbf{q}}(t_i)\|_2^2 \tag{7.7}$$

3. Motor accelerations minimization (only in the SEA case), with weighting factor $c_{\ddot{\theta}}$

$$\Phi_{L,\min\ddot{\theta}}(\cdot) = \sum_{i=1}^{n_{mp}} c_{\ddot{\theta}} \|\ddot{\boldsymbol{\theta}}(t_i)\|_2^2 \tag{7.8}$$

4. Absolute mechanical work minimization, with weighting factor c_W

$$\Phi_{L,\min W}(\cdot) = \sum_{i=1}^{n_{mp}} c_W \|\mathbf{u}(t_i) \cdot \dot{\mathbf{q}}(t_i)\| \tag{7.9}$$

which are combined into one:

$$\Phi_L = \Phi_{L,\min\tau} + \Phi_{L,\min\ddot{q}} + \Phi_{L,\min\ddot{\theta}} + \Phi_{L,\min W} \tag{7.10}$$

to be evaluated at the same time.
It is possible to impose different weights also on each of the joints/motors, however we do not as in a squat motion the three joints should have equal contributions and the values related to each of them are also in comparable ranges. But, as we will see later in the experiments, we imposed a higher gain on the hip joint due to the difference of this joint with the other two in the real HeiCub robot.

7.2.3 Constraints

As we are treating a system that has physical limitations, these limits are set as boundary constraints. For the HeiCub model constraints on the following quantities are imposed:

- Joint angles range $q_i \in [q_i^{min}, q_i^{max}]$,
- Motor angles range $\theta_i \in [\theta_i^{min}, \theta_i^{max}]$,
- Joint velocities $\dot{q}_i \in [\dot{q}_i^{min}, \dot{q}_i^{max}]$,
- Motor velocities $\dot{\theta}_i \in [\dot{\theta}_i^{min}, \dot{\theta}_i^{max}]$,
- Torques $\tau_i \in [\tau_i^{min}, \tau_i^{max}]$,

where $i = 1, ..., n_{dof}$, for $n_{dof} = 3$.
These limits ensure that the solutions found can be transferred to the physical robot. The right hand side of $\dot{\mathbf{x}}(t)$ in Eq. (2.16) is obtained by solving Eq. (2.9) for the case of rigid actuators and Eq. (6.20) when considering SEA, but without considering external contacts nor impacts. Equality and inequality constraints are imposed to generate stable squat motion:

- Vertical velocity of the robot expressed as velocity along the z axis of the root
  - Phase 1, $\dot{Z} \leqslant 0$, to bring the robot to the final squat position.
  - Phase 2, $\dot{Z} \geqslant 0$, to bring the robot back to the upright pose.
- ZMP is always in the support polygon, corresponding to the foot area of the robot, expressed as inequality constraints, to ensure the stability of the motion.
- Initial position equal to all zeros $\mathbf{q}(t_0) = \mathbf{0}$.
- Initial and final velocities equal to zeros $\dot{\mathbf{q}}(t_0) = \mathbf{0}$, $\dot{\boldsymbol{\theta}}(t_0) = \mathbf{0}$, $\dot{\mathbf{q}}(T) = \mathbf{0}$, $\dot{\boldsymbol{\theta}}(T) = \mathbf{0}$ where T is the final time at the end of the second phase.
- Periodicity constraints on states and controls, i.e. $\mathbf{x}(t_0) = \mathbf{x}(T)$, $\mathbf{u}(t_0) = \mathbf{u}(T)$, which forces the final position to be equal to the initial one and allows the motion to be repeatable consecutively.

In the above constraints, when referring to the motors, these are taken into account only in the case of the model with SEA.
Constraints on phase times will be discussed in the next section as it is strictly related to the obtained results and experiments performed.

7.3 Results and implementation

7.3.1 Numerical results

The number of shooting nodes n_{mp} is chosen to have the discretized intervals of MUSCOD-II to coincide with the thread rate of the controller that we use to send the commands to the real robot, i.e. 10 [ms].
We used first the objective as in Eq. (7.5) to compute the maximum achievable squat range. In the case of 2 DOF the hip pitch joint was kept fixed at 30 degrees, and the maximum obtained squat range with this setup is of 3.3 [cm], which is extremely small even considering the small size of HeiCub. In the case of 3 DOF instead, the full achievable range is of 18.1 [cm], which corresponds to almost half the length of the legs of HeiCub.

The results we illustrate in this chapter are of the 3 DOF case only. In all the obtained results we keep the squat range to be fixed at the maximum achievable one and vary instead the combination of objective functions. The phases time is first kept fixed and identical for both phases for all the combinations in order to obtain comparable results.
At first the maximum joint velocities of the robot were unknown and therefore the phases times were at first fixed to 0.5 [sec] each to perform a fast squat motion. However, with the first experiments, we gathered new information on the velocities and updated the constraints, resulting in a maximum time of 1.5 [sec] for each phase, as the actual velocities are much lower than the first guess.

The decision of keeping the time fixed is due to two main reasons:

- Keeping the integration interval between the shooting nodes the same as the desired control thread rate of the robot as stated at the beginning of this section,
- Obtaining comparable results with the different objective functions.

For the SEA case however we hcomputed also a case leaving the phase times free to the optimization to find the best ones, as it is interesting to exploit the spring nature of the SEA. In this case the first and second phase might have different times, differently than the fixed times case.

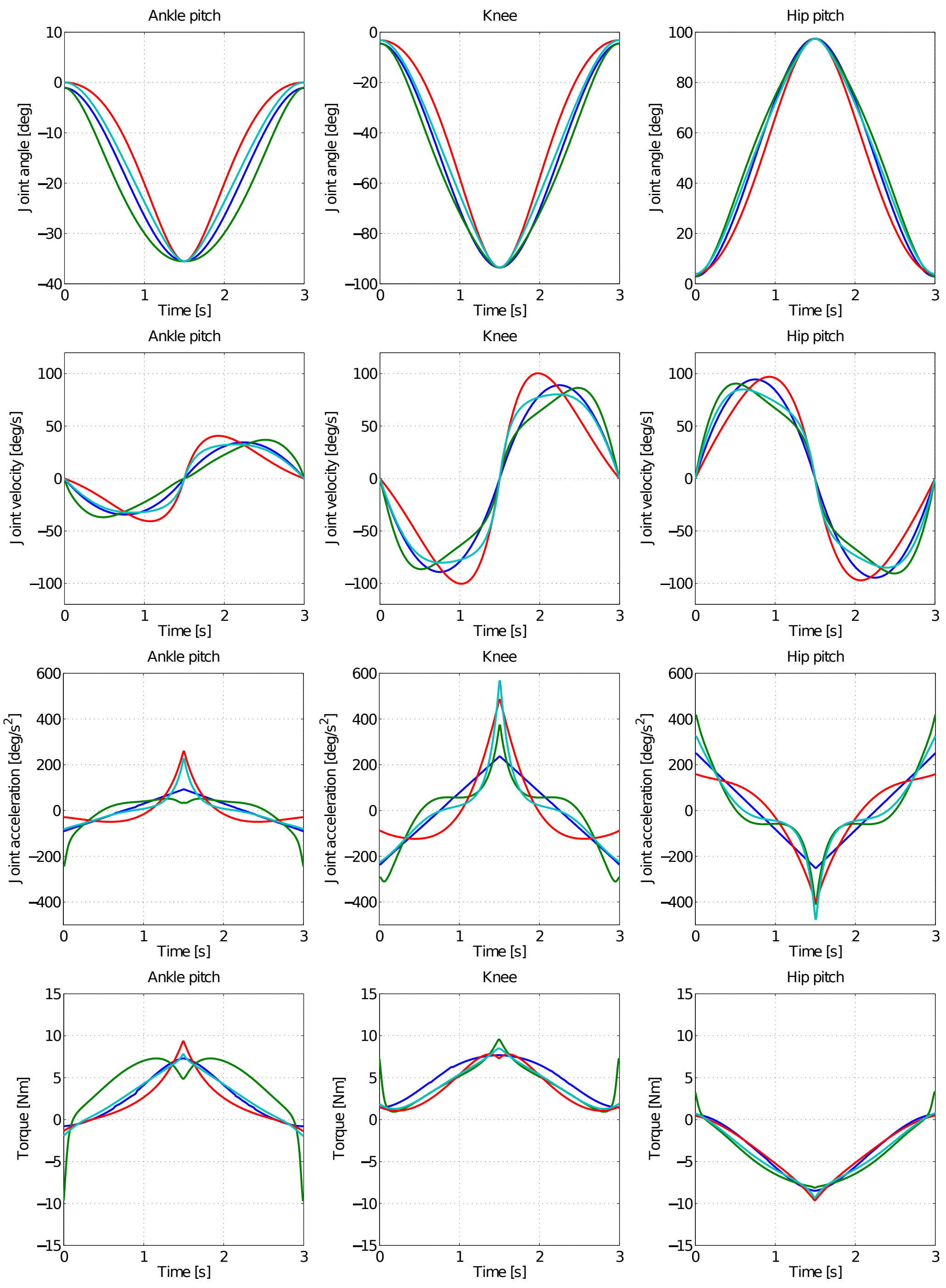
In the following we discuss the numerical results obtained for the case with phase time 1.5 [sec] and different combination of the objective functions listed in Section 7.2.2. In all cases the weights of the objective functions are set either to 1 or to 0 in order to take into account or exclude the contribution of an objective function. The results for the rigid actuators are shown in Fig. 7.2, while the ones with SEAs are in Fig. 7.3.

Rigid actuators

With the model with rigid actuators combinations of the objective functions 1,2 and 4 are used. When torque minimization is not considered, the torque has high variations in all the joints, as can be observed from Fig. 7.2. When only the joint acceleration minimization is considered instead, the joint accelerations are almost linear, while when it is used in combination with other objectives, it is subject to a sudden change when the phase changes from squat down to lift up. This is due to the higher average torque and mechanical work required to perform motions at low speed, reason for which the optimal control brings the robot down and up faster. By imposing a higher weight on the joint acceleration minimization the effect could be smoothed but is has of course a negative influence on the torque and mechanical work minimization.

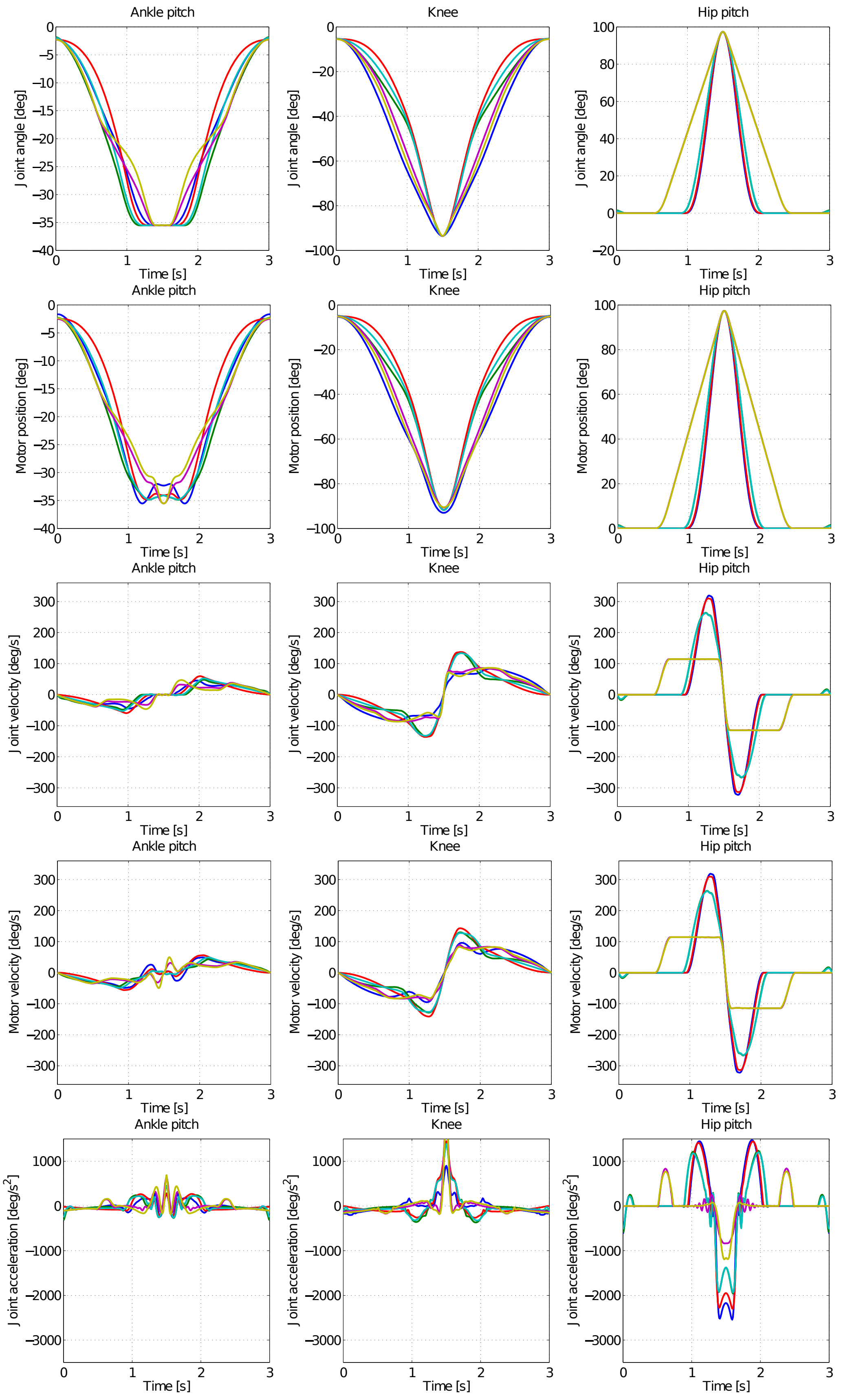
SEA

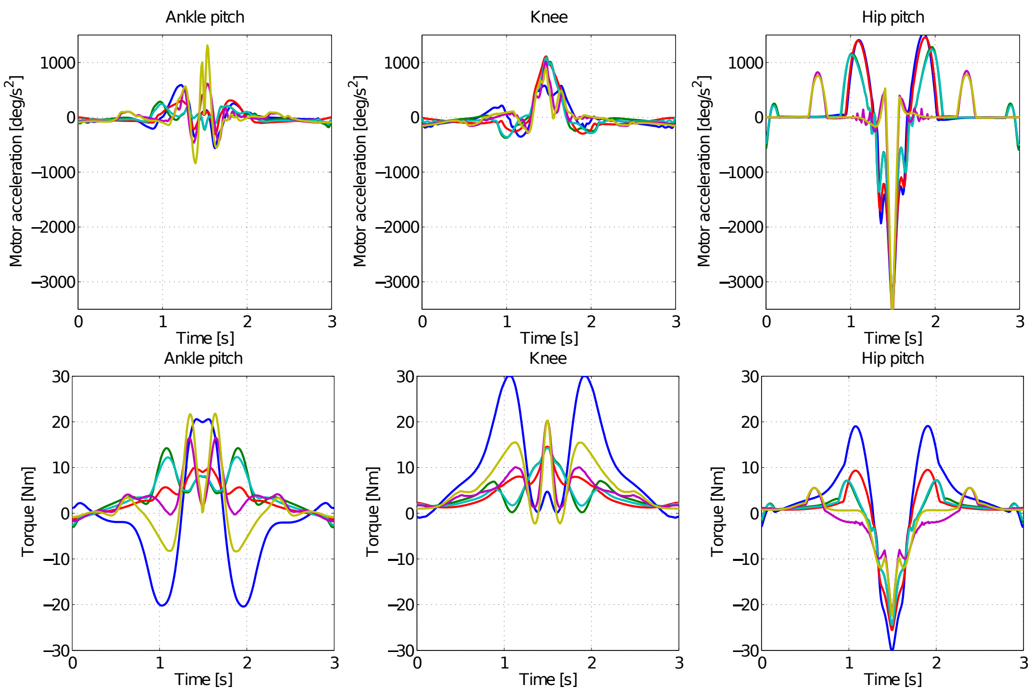
In the case of SEA we show first the results obtained with phase time 1.5 [sec], but we gained some insights from the 0.5 [sec] case that are taken into consideration in the 1.5 [sec] case.
With the shorter phase time of 0.5 [sec] the obtained trajectories are overall similar to the ones obtained with the model with rigid actuators, but with higher velocities and accelerations due to the shorter time. We have seen that when the motor accelerations are not being minimized, the the oscillations in the velocities of knee and ankle pitch motors are much higher. In fact, in the results with phase time 1.5 [sec] in Fig. 7.3 the motor accelerations are always included in the objective function to avoid such oscillations.
Due to the spring effect, at acceleration level we can observe that the acceleration variations are attenuated in the joint side. This can be noticed also in the hip joint, mainly in the transition between the two phases, where accelerations of the motors are much higher. Even if a very high stiffness was assigned to the hip pitch to emulate a rigid actuator, the effect of flexibility is still present, which is the reason for which in the next chapter this joint is modeled as perfectly rigid ($q_{hip} = \theta_{hip}$).
From Fig. 7.3 we could observe that the behaviour is overall different than the rigid actuators case. The elastic joints, i.e. knee and ankle pitch, are moved first and the rigid joint, i.e. hip pitch, is the last to move. Due to this delay in moving the hip joint, a high velocity is required for the hip pitch joint. This is due to the higher average effort required to move the hip pitch joint at lower speed and to the presence of the springs in the lower joints, since in the hip joint the stiffness is much higher, a higher average effort is required to move the joint at lower



Obj. function	$\Phi_{\min \tau}$	$\Phi_{\min \ddot{q}}$	$\Phi_{\min W}$
🟦	-	X	-
🟩	-	X	X
🟥	X	X	-
🩵	X	X	X

Figure 7.2: Results obtained for the case with rigid actuators with different objective functions and squat time 1.5 [sec].


Ankle pitch
Knee
Hip pitch
Joint angle [deg]
Motor position [deg]
Joint velocity [deg/s]
Motor velocity [deg/s]
Joint acceleration [deg/s^2]
Time [s]



Obj. function	$\Phi_{\min \tau}$	$\Phi_{\min \ddot{q}}$	$\Phi_{\min \ddot{\theta}}$	$\Phi_{\min W}$	High hip gain
🟦	-	X	X	-	-
🟩	-	X	X	X	-
🟥	X	X	X	-	-
🟦	X	X	X	X	-
🟪	X	X	X	X	X
🟨	X	10	10	X	-

Figure 7.3: Results obtained for the case with elastic actuators with different objective functions and squat time 1.5 [sec]. In the table 10 means that the weight was set to 10 instead of 1.

speeds. This further motivated our model choice in the next chapter in which the assumption of rigid actuators being SEA with high stiffness is dropped.

Given that the hip pitch joint of HeiCub is a coupled joint with tendons, the joint is able to reach maximum velocities that are lower than independent motor driven joints as the knee and ankle pitch, therefore a smaller limit on the hip pitch joint velocity was imposed as well as a higher weighting factor in the acceleration minimization of the same joint in order to obtain feasible motions for the robot.

With the phase times left free we used the combination with all objective functions and a higher weight on hip pitch accelerations. Interestingly, despite the two phases were not constrained to have the same phase time, they ended up with very close values. The total final time obtained is of 3.8 [sec]. We can observe that there is no significant difference between the cases with fixed and free times, as the hip pitch joint velocity still saturates. The main difference lies on the time which is almost 1 [sec] longer than the fixed time case. The result could also be due to the initial values set in the optimal control problem, the initial values are all zeros but the phase time has to be set to some initial guess, a different starting initial phase time might lead to a different final result. Also, due to the free phase time, in this case it is not possible to keep the relationship shooting interval = thread time. Therefore, for the implementation on the robot the trajectories had to be interpolated and sampled at the desired rate.

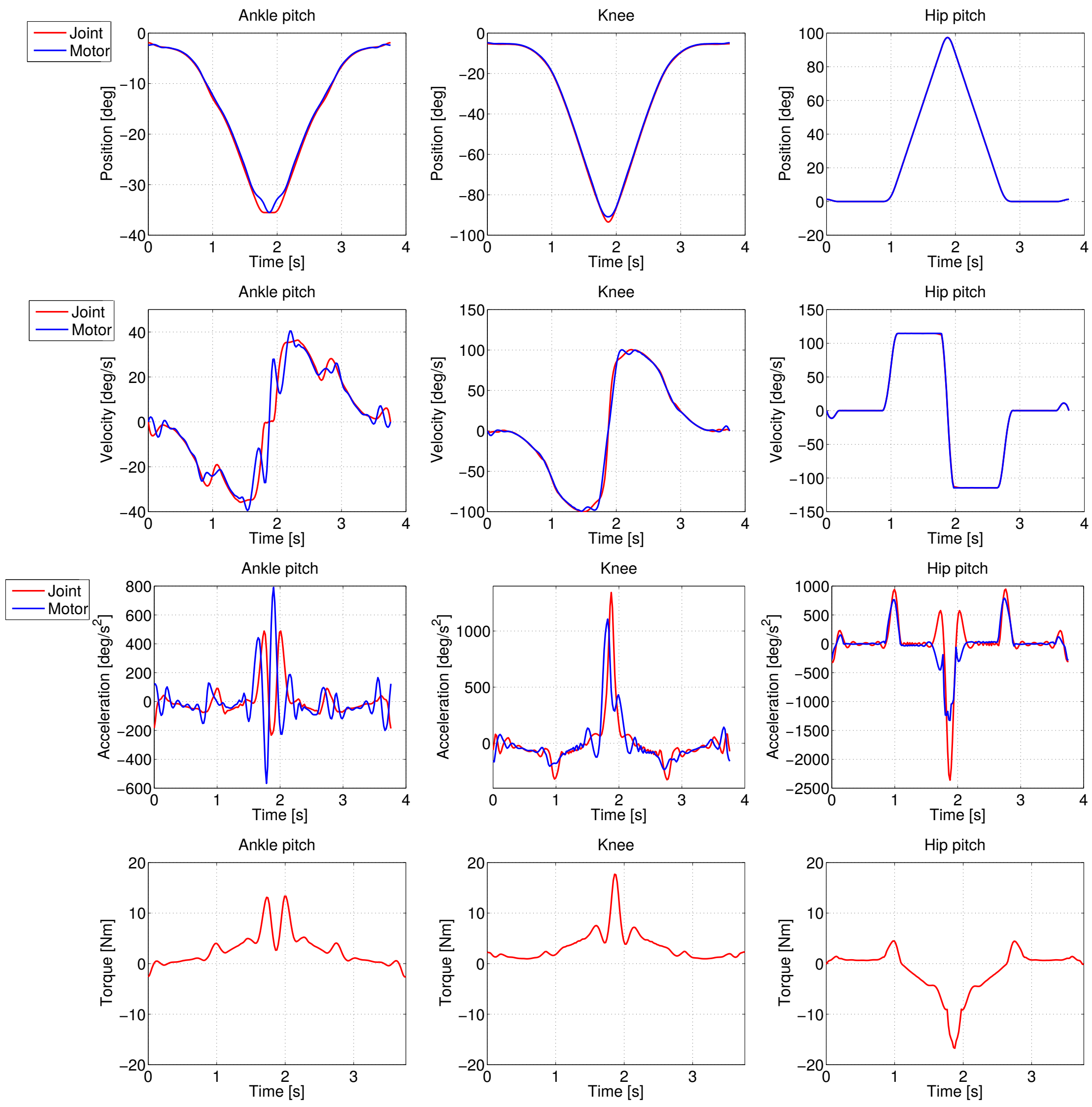


Figure 7.4: Results obtained for the case with rigid actuators with squat time left free to the optimization. Different than the other plots, here the motor side and joint side data are plotted in the same plot.

7.3.2 Experimental validation

Three out of all the motions obtained are selectd to be tested on HeiCub:

- Rigid actuators case: all objective functions combined with same weights.
- SEA case: all objective functions combined with same weights and higher weights on joint and motor accelerations, with smaller velocity limit on hip pitch joint and motor.
- SEA case: with the same combination of objectives as above, with free phase time.

For the testing, in order to test also the valdity of the periodicity constraints, the motion is concatenated into 5 consecutive motions. In the case of rigid actuators the motion is tested with position direct control mode at a thread rate of 10 [ms] in which only the hip pitch, knee and ankle pitch joints of both left and right legs are used while the rest are kept to the zero

position. The exact same trajectories are executed on both legs, as they are in theory perfectly symmetric.
In the following we present the results achieved on the left leg, as the performances are similar between the two legs.
In the case of rigid actuators the positions are tracked accurately in all the joints, as shown in Fig. 7.6. As expected, motor and joint positions coincide in the three joints. A higher error can be observed on the hip pitch joint, where a small delay is present. The achieved joint angle trajectory does not match as closely to the computed one as in the other two joints. This is due to the aforementioned issue of the hip pitch joint being a coupled joint, and therefore cannot reach the same maximum velocity as the independent driven joints.

With SEA the experiment is carried out with two different control modes, respectively the position direct and the motor position direct.
We can observe from Fig. 7.7 and Fig. 7.9 that with position direct the motor and joint positions do not match in the joints with SEA (knee and ankle pitch) as in the rigid actuators case, as expected. However, we can also observe that the motor positions do not coincide with the computed ones. This is due to two main reasons, the first that the position direct is not the proper control mode to be used with SEA. As stated in Section 6.1.3, the PID tries to correct the position errors on the joint side. The second reason is that the stiffness of the spring on the actual robot is not the same as the one used in the optimal control problem. This is possible as in the real robot between the motor and the joint there is not only the spring but also the gearbox and other mechanical components which can contribute to the flexibility of the joint. Furthermore despite the lower limit on the hip pitch joint velocity, the joint was not able to reach the desired position and has a clear delay in the execution of the motion. This can also be explained with the aforementioned mechanism issue.
With the motor position direct control mode, we can see from Fig. 7.8 and 7.10 that the measured joint and motor trajectories are both closer to the desired motor trajectories, when the measured joint trajectory should be closer to the desired joint trajectory. This is more evident in the ankle pitch joint. From these measurements it seems that the stiffness of the joint could be actually higher than in the model, as the measured joint and motor trajectories are quite close to each other.
The errors observed from the experiments might be also due to many other reasons than the ones already listed, such as stiction, damping etc. As in optimal control model precision is very important, these unknown parameters can only be identified on the actual robot via repeated experiments to further adjust the model.

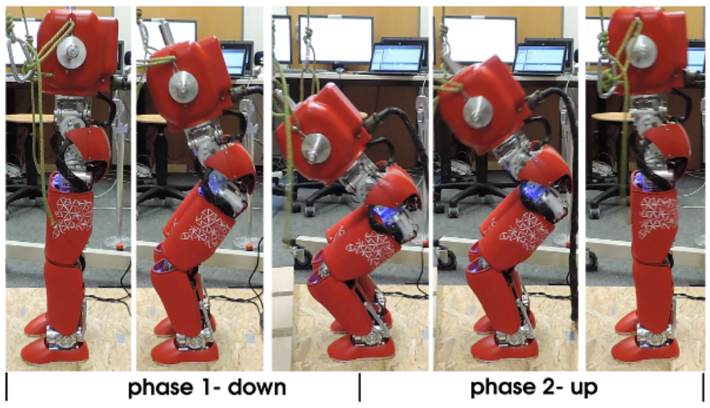


Figure 7.5: HeiCub performing squat in the two phases of optimal control.

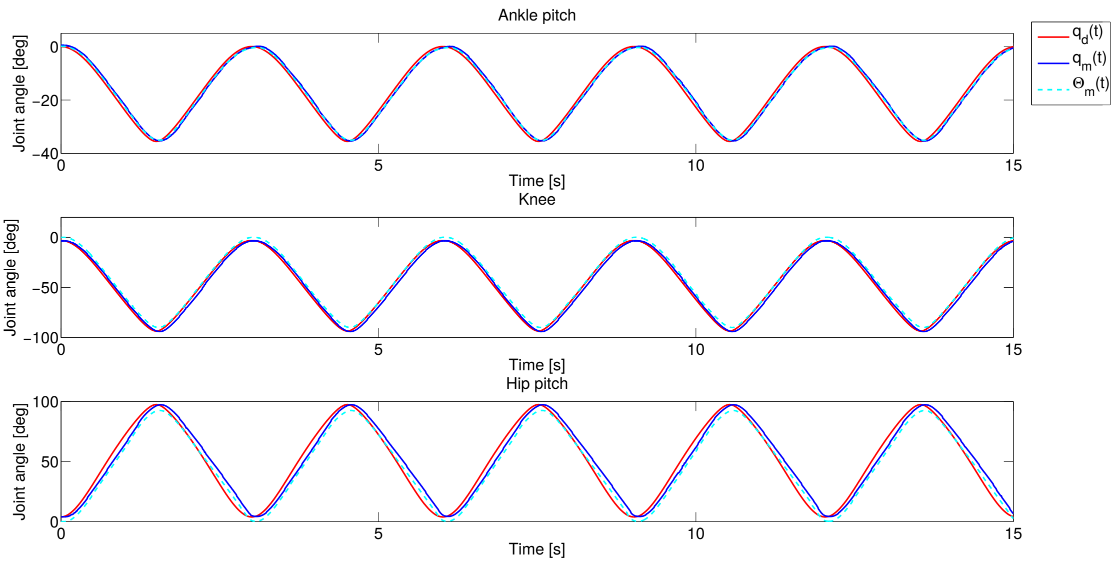


Figure 7.6: Experimental results with rigid actuators. In this case motor and joint encoders coincide and follow the desired joint trajectories.

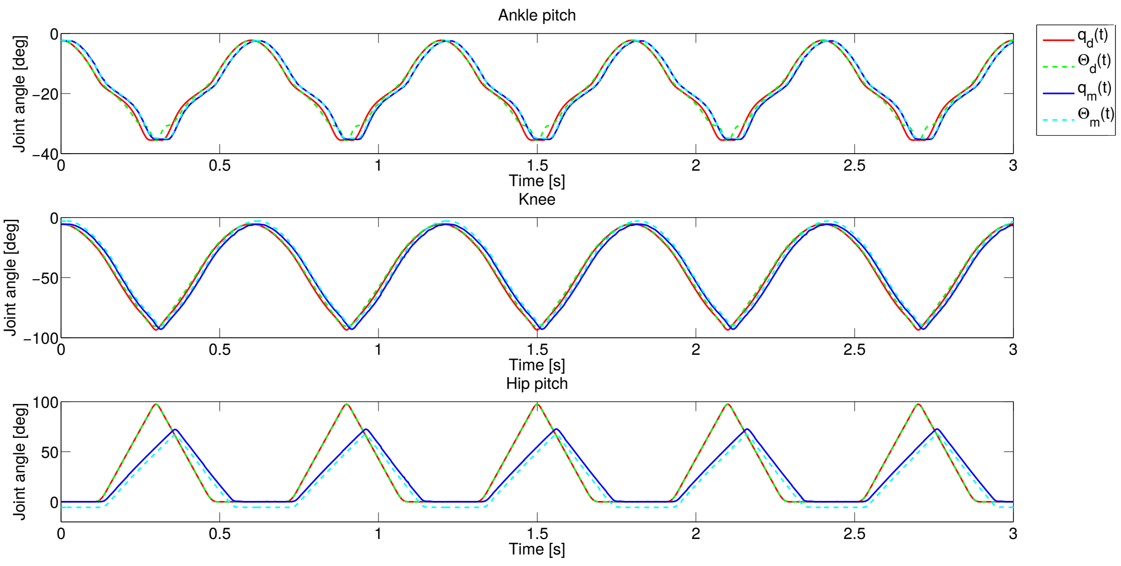


Figure 7.7: Experimental results with SEAs. In this case the difference between motor and joint encoders of knee and ankle pitch are due to the deflection of the spring, while in the hip the two encoder readings coincide.

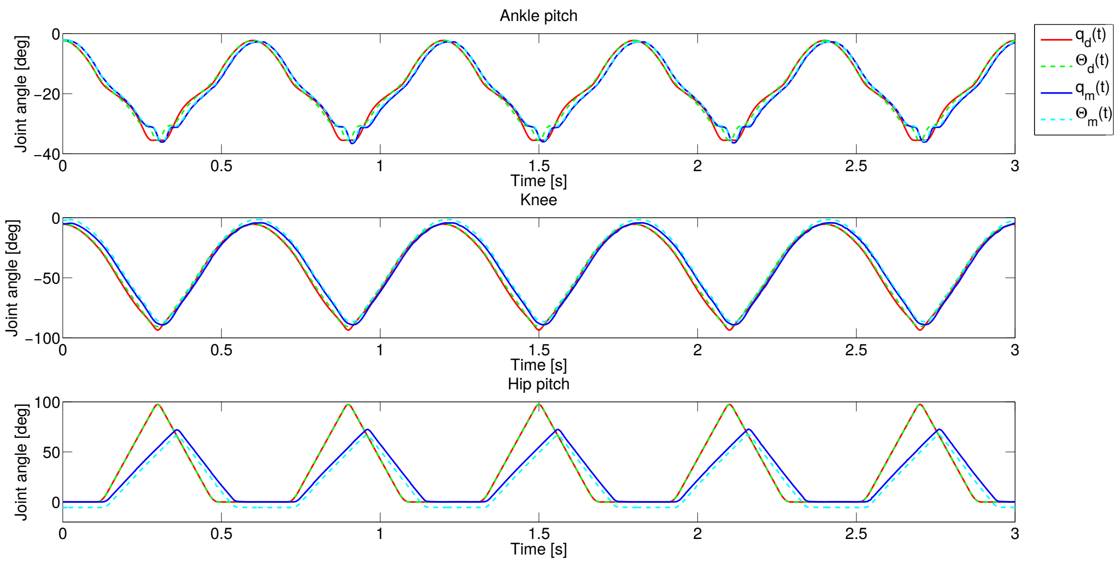


Figure 7.8: Experimental results with SEAs using motor position direct control mode.

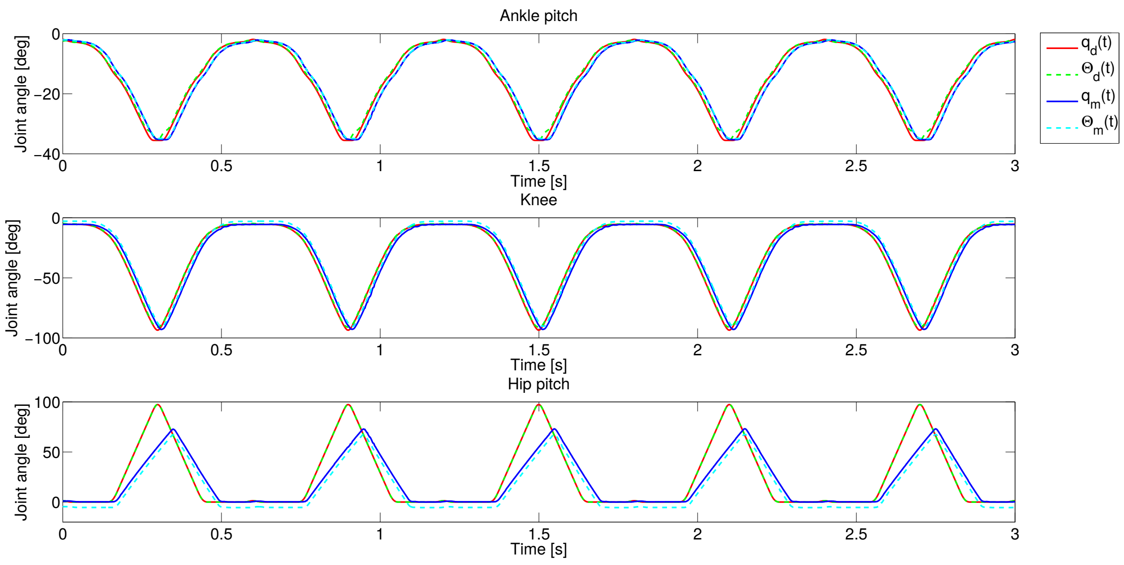


Figure 7.9: Experimental results with SEAs with free time.

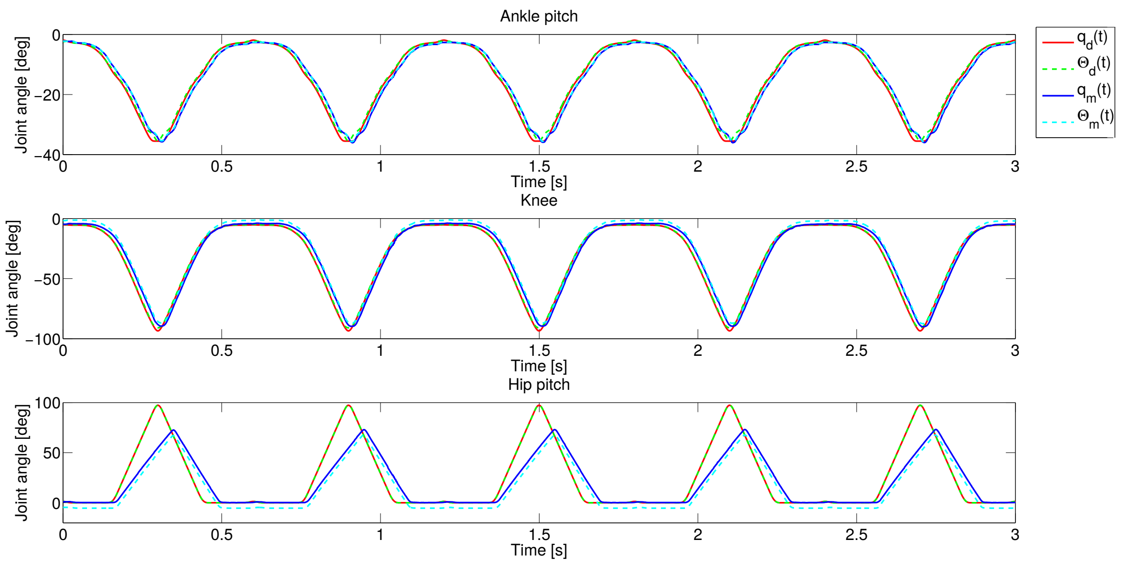


Figure 7.10: Experimental results with SEAs with free time using motor position direct control mode.

8 Push recovery

In everyday life we face the possibility of falling due to several reasons, ranging from tipping over or being perturbed by strong pushes, as shown in an experiment in Fig. 8.1. Humans have fast reactions that can avoid falling in certain cases, allowing to recover from the disturbance. Studies have tried to understand how humans can recover from falling using motion capture experiments and dynamic models [97].

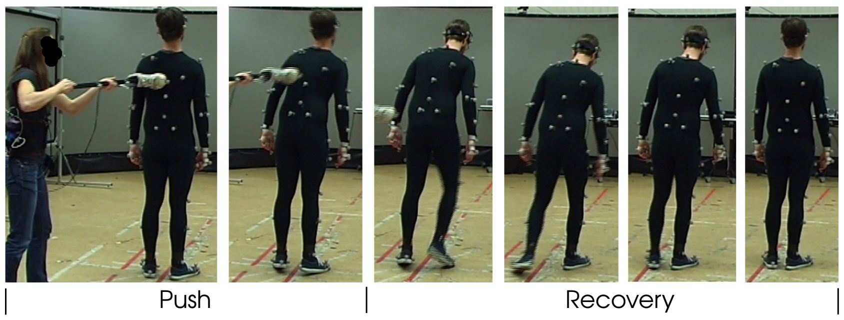


Figure 8.1: A human push recovery experiment.

The skill of recovering from falling is one of the most desired in legged robots as well, mostly in humanoid robots. As humanoid robots are thought to be mainly used in human populated enviroments, they can face the same daily challenges as humans. More over, falling for a robot also means possible costly damage to the hardware, and this should be avoided as much as possible. In literature there are several attempts of solving the problem by means of reduced models and concepts such as the capture point [90, 26]. In particular the capture point based method has been recently tested on the iCub humanoid robot [20] by means of a momentum based balancing controller [83].

We want to investigate the issue of push recovery by means of model based optimal control, using the whole-body dynamic model of the robot including also the SEA model.
Optimal control has been used to generate one step push-recovery for human models, in which the push is taken into account in the system dynamics as external force [98]. Here we apply the method to the HeiCub model, by considering robot specific constraints, such as the flat foot contact, the joint position, velocity and torque limits. The considered model consits of 15 DOF, as we assume the left foot of the robot to be always on the floor and the recovery is performed with the right foot within one step of variable step size, determined by the optimization process.
In this chapter we first illustrate the model of the robot, then we describe the optimal control problem with the states, controls, parameters, constraints and objective functions, then the results of the optimization are presented for both cases with and without SEA.

8.1 Model description

For the problem of push recovery we consider the model to have the left foot fixed on the ground, therefore the model consists of the internal 15 DOF only. The right foot is left to step freely within the imposed limits and constraints in order to recover from the push. The push recovery is performed for both the model with and withou the SEA.

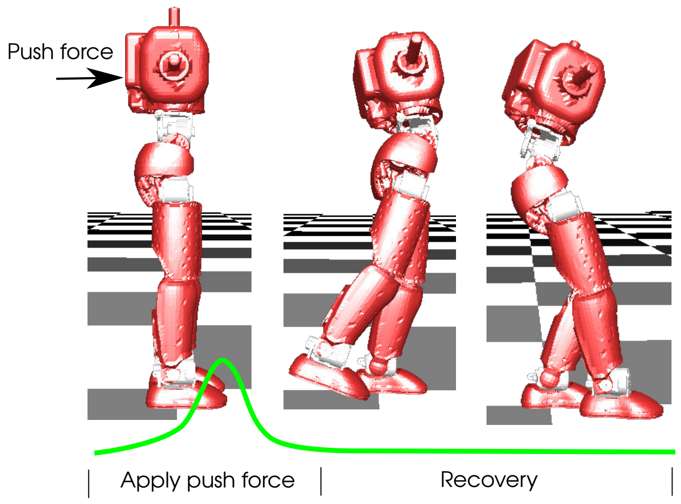


Figure 8.2: Push recovery for the HeiCub, the push force is modeled as a continuous function in time and the recovery is performed within one step.

The generalized coordinates describing the robot are:

- For the rigid actuators case: $\mathbf{q}(t) \in \mathbb{R}^{15}$
- In addition for the SEA case: $\boldsymbol{\theta}(t) \in \mathbb{R}^{4}$

Differently than the squat case, in this model only the joints with SEA are modeled as elastic. Therefore instead of imposing high stiffness on the rigid joints ending up with double the number of generalized coodrinates, we model only the SEA joints as elastic, assuming the non elastic ones to be perfectly rigid. In this way the problem complexity is much lower as the number of states of the optimal control problem is also lower.

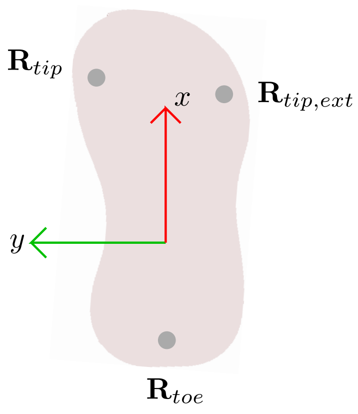


Figure 8.3: Contact points on the right foot, view from above.

As the right foot is free to be placed, three contacts points are defined for the right foot as in Fig. 8.3:

- $\mathbf{R}_{tip} = [R^x_{tip}, R^y_{tip}, R^z_{tip}]^T$
- $\mathbf{R}_{tip,ext} = [R^x_{tip,ext}, R^y_{tip,ext}, R^z_{tip,ext}]^T$

- $\mathbf{R}_{toe} = [R^x_{toe}, R^y_{toe}, R^z_{toe}]^T$

These points are used in the optimal control problem to ensure that the contact between the foot and the ground is properly modeled.

The push is modeled as an external force in the reconstruction and analysis of push recovery on humans with optimal control [98]. Here the same type of formulation is applied to the HeiCub robot, where the computation of the joint torques take into account the external perturbation:

$$\boldsymbol{\tau}_{push}(t) = \boldsymbol{\tau}(t) + \mathbf{G}_{push}(\mathbf{q})^T \boldsymbol{\lambda}_{push}(t) \tag{8.1}$$

with $\boldsymbol{\lambda}_{push}$ being the vector of all external forces exerted on the system and $\mathbf{G}_{push}(\mathbf{q})$ is the contact Jacobian computed for the point of application. This means that multiple points could be used, however here the force is supposed to be applied on a single point, therefore $\boldsymbol{\lambda}_{push}(t) \in \mathbb{R}^3$.

Commonly, the push force is seen as an impulse, therefore an instantaneous force. In our case the push force $\boldsymbol{\lambda}_{push}(t)$ is a smooth function over a specified time interval T_{push}, which has maximum corresponding to the hald of the interval $0.5 * T_{push}$.

8.2 Optimal control problem

Given the assumption of the left foot fixed on the ground, we model the recovery motion as a single step performed with the right leg, as shown in Fig. 8.2. At first, the robot is standing upright with all joints at zero positions. Then an external force is exerted on the back of the chest of the robot and the system is perturbed. We model the push recovery with two strategies:

1. Recovery with a single phase in which the robot comes to a stable position when touching the ground with the right foot.

2. Recovery in two phases with impact, in which the robot stabilizes the motion after the impact.

In particular, the second strategy consists in the following phases:

Phase 1 Left foot on the ground and right foot swinging to perform the recovery

Phase 2 Right foot on the ground to stabilize the motion after impact

In the first strategy, $n_{ph} = 1$, while in the second strategy $n_{ph} = 2$, including two continuous phases and one discontinuity corresponding to the impact, which is a phase with zero time.

The push force, duration, and application point are given as fixed parameters to the optimal control problem, therefore we assume a priori known push forces. The push consists of an horizontal push on the back of the chest of the robot and the amount is such that the robot is able to recover, as forces that are too big clearly cannot be compensated within the physical limitations of the robot. The optimal recovery whole-body trajectories are computed using optimal control for a set of specified objective functions.

8.2.1 States, controls and parameters

The states of the optimal control problem are the generalized positions and velocties of the robot in the case of rigid actuation:

$$\mathbf{x}(t) = \begin{bmatrix} \mathbf{q}(t) \\ \dot{\mathbf{q}}(t) \end{bmatrix} \tag{8.2}$$

where $\mathbf{q}, \dot{\mathbf{q}} \in \mathbb{R}^{15}$, so $\mathbf{x}(t) \in \mathbb{R}^{30}$. While in the case where SEA are included, it consists of:

$$\mathbf{x}(t) = \begin{bmatrix} \mathbf{q}(t) \\ \boldsymbol{\theta}(t) \\ \dot{\mathbf{q}}(t) \\ \dot{\boldsymbol{\theta}}(t) \end{bmatrix} \tag{8.3}$$

where $\boldsymbol{\theta}, \dot{\boldsymbol{\theta}} \in \mathbb{R}^4$, as we model only the knee and ankle pitch joints as elastic joints, given that these correspond to the physical SEA. This means $\mathbf{x}(t) \in \mathbb{R}^{38}$.
The controls are represented by the joint torques (excluding the contribution of the push force)

$$\mathbf{u}(t) = \boldsymbol{\tau}(t) \in \mathbb{R}^{15} \tag{8.4}$$

The set of parameters include the push force expressed in [N], the time interval of the push expressed in [sec] and the length of the recovery step

$$\mathbf{p} = \begin{bmatrix} p_{force} \\ p_{pushtime} \\ p_{steplength} \end{bmatrix}. \tag{8.5}$$

The push force and time are fixed parameters, while the step length is left free to the optimization to find the best value to allow the free placement of the foot. With step length here we mean the distance travelled in the forward or backward directions along the sagittal plane, so the lateral displacement as for the time being not considered as we are considering only horizontal push forces.
The push application point is given as external input to the optimal control at initialisation, therefore the push point can be easily changed and the dynamics computed accordingly.

8.2.2 Objective functions

A set of objective functions have to be defined to generate the recovery motion. In particular, two objective functions are set for the case of model with rigid actuators:

- Minimization of effort in terms of torques squares over the step length:

$$\Phi_{eff}(\cdot) = \frac{1}{p_{steplength}}(\mathbf{u}^T\mathbf{u}) \tag{8.6}$$

- Minimization of joint accelerations in order to obtain smooth velocity trajectories:

$$\Phi_{\min \ddot{q}}(\cdot) = \ddot{\mathbf{q}}^T\ddot{\mathbf{q}} \tag{8.7}$$

In the case of SEA we introduce a further criterion on minimization of motor accelerations to avoid high oscillations on the motor side:

$$\Phi_{\min\ddot{\theta}}(\cdot) = \ddot{\boldsymbol{\theta}}^T \ddot{\boldsymbol{\theta}} \tag{8.8}$$

The objective functions are combined with weights, for the case without SEA

$$\Phi_{rigid}(\cdot) = c_{eff}\Phi_{eff} + c_{\ddot{q}}\Phi_{\min\ddot{q}} \tag{8.9}$$

and the case with SEA

$$\Phi_{elast}(\cdot) = c_{eff}\Phi_{eff} + c_{\ddot{q}}\Phi_{\min\ddot{q}} + c_{\ddot{\theta}}\Phi_{\min\ddot{\theta}} \tag{8.10}$$

All objective functions are evaluated over the whole time interval.

8.2.3 Constraints

As in the case of squat motion generaton, boundary constraints are set according to the physical limits of the robot on:

- Joint angles range $q_i \in [q_i^{min}, q_i^{max}]$,
- Motor angles range $\theta_j \in [\theta_j^{min}, \theta_j^{max}]$,
- Joint velocities $\dot{q}_i \in [\dot{q}_i^{min}, \dot{q}_i^{max}]$,
- Motor velocities $\dot{\theta}_j \in [\dot{\theta}_j^{min}, \dot{\theta}_j^{max}]$,
- Torques $\tau_i \in [\tau_i^{min}, \tau_i^{max}]$,

where $i = 1, ..., n_{dof}$, for $n_{dof} = 15$ and $j = 1, ..., n_{dof,sea}$, for $n_{dof,sea} = 4$.
The right hand side of $\dot{\mathbf{x}}(t)$ in Eq. (2.16), is obtained by solving Eq. (2.9) in the case of rigid actuators and Eq. (6.20) when considering SEA, i.e. computing forward dynamics considering contacts.
In the case of the second strategy, the transition functions $\widetilde{\mathbf{J}}(\cdot)$ associated with state variable discontinuities consist in the impact dynamics equations as described in Eq. (2.12) for rigid actuators and Eq. (6.22) for SEA.

The equality and inequality constraints are set for the two strategies as follows.
For strategy 1:

- The initial state $t = t_0$ corresponds to the robot at a resting upright position $\mathbf{q}(t_0) = \mathbf{0}$, $\dot{\mathbf{q}}(t_0) = \mathbf{0}$.
- During the motion, i.e. for $t \in (t_0, T)$, the right foot is in the air and therefore the vertical component of the contact points is imposed to be bigger than zero

$$\begin{aligned} R^z_{tip} &\geqslant 0 \\ R^z_{tip,ext} &\geqslant 0 \\ R^z_{toe} &\geqslant 0 \end{aligned} \tag{8.11}$$

- At the end of the phase, for $t = T$, the right foot has to be placed flat on the ground

$$\begin{aligned} R^z_{tip} &= 0 \\ R^z_{tip,ext} &= 0 \\ R^z_{toe} &= 0 \end{aligned} \tag{8.12}$$

  and the joint and motor velocities have to be zero $\dot{\mathbf{q}}(T) = \mathbf{0}$, $\dot{\boldsymbol{\theta}}(T) = \mathbf{0}$ to ensure a complete stop.

For strategy 2:

- The initial state $t = s_0$ corresponds to the robot at a resting upright position $\mathbf{q}(s_0) = \mathbf{0}$, $\dot{\mathbf{q}}(s_0) = \mathbf{0}$.

- During the motion, i.e. the first phase for $t \in (s_0, s_1)$, the constraints are as in Eq. (8.11).

- At the impact, i.e. at the discontinuity $t = s_0^+$, the constraints are as in Eq. (8.12).

- In the stabilisation phase, i.e. for $t \in (s_1, s_2)$, the right foot has to keep the flat contact with the ground, therefore:

$$\begin{aligned} f(R^z_{tip}) &\geqslant 0 \\ f(R^z_{tip,ext}) &\geqslant 0 \\ f(R^z_{toe}) &\geqslant 0 \end{aligned} \tag{8.13}$$

  where f is the function that computes for the contact forces of a certain contact point.

- At the end of the stabilisation phase, i.e. $t = T$, the joint and motor velocities have to be zero $\dot{\mathbf{q}}(T) = \mathbf{0}$, $\dot{\boldsymbol{\theta}}(T) = \mathbf{0}$ to ensure a complete stop.

In both strategies the ZMP is constrained to be always in the support polygon by means of inequality constraints in order to ensure the stability of the generated motion.
Clearly, in the above constraints, when referring to the motors, these are included only in the case with SEA.
Collision avoidance is introduced as constraints by means of geometric capsules under the form of rounded cap cylinders [60]. The cylinders approximate the limits of the different links of the legs, including the upper leg, the lower leg and the foot. The distance d between the capsules is used as inequality constraint to avoid collisions between links $d \geqslant 0$.

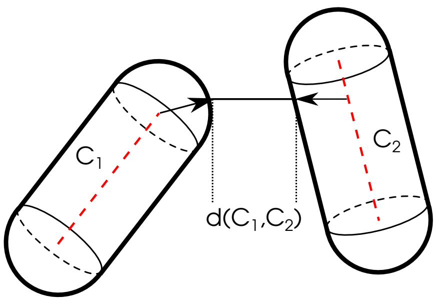


Figure 8.4: Collision detection with rounded cap cylinders.

8.3 Numerical results

We performed the optimization for push recovery with the two models and the two startegies as described in Section 8.1.
In all cases the push force is set to 30 [N], which is enough to perturb the robot given that the size of the robot is small and its total mass is less than 30 [kg], the push time interval T_{push} is of 0.5 [sec] and it is a horizontal force exerted on center of the back of the HeiCub chest. All the objective functions are used in combination, i.e. the minimization of effort, minimization of joint acceleration and minimization of motor accelerations when considering SEA.
The center of mass plots shown in Fig. 8.5,8.7,8.9,8.11 are referred to the world reference frame located at the center of the left foot, x is the frontal direction, y is the lateral direction and z the vertical direction.

Rigid actuators

In the case of rigid actuators, we obtained a recovery with a step lenth of $p_{steplength} = 14.9$ [cm] with the one phase strategy and $p_{steplength} = 13.3$ [cm] in the case of two phases strategy. The big step is due to the minimization of effort, which maximizes the step length while reducing the joint torques.
We can see from Fig. 8.5 that the center of mass has bigger variations with respect to the two phases case as in Fig. 8.7, and the time required by the two phases strategy is much longer than the single one. From Fig. 8.7 we can also observe a small jump on the center of mass trajectory in correspondence to the impact. From the torque profiles we can see that there was no saturation in any of the joints, and the highest torques are those of the left leg as it is the stance leg. In the two phases case the torques have high peaks due to the impact. This can be mostly noted in the right leg, which is the swing leg that impacts with the ground to recover from the falling. In particular, the torques of the ankle joints have spikes of the order of 5 [Nm] while in the single phase case they have very small values.

Elastic actuators

When SEA are introduced in the model the single phase recovery is performed with $p_{steplength} = 14.9$ [cm], and in the two phases strategy case with $p_{steplength} = 2$ [cm], which is significantly lower than all the other cases. In both cases however, as we can see from Fig. 8.9 and 8.11 the center of mass had smaller variations compared with the rigid actuators case, mainly in the vertical direction.
Due to the spring of the SEA, from Fig. 8.10 we can observe small oscillations in the knee joint. The effect is more evident in the two phases strategy case as in Fig. 8.12, where evident spikes and following oscillations are present in the both the left and right knee and ankle pitch joints. However, from the time point of view, with the two phases strategy the recovery times differ by only few milliseconds between the cases with rigid actuators and SEA. The major difference lies in the joints with SEA, where in the SEA case the torques spikes and oscillations are of about 5 [Nm], while in the rigid actuators case the spikes are of this magnitude only in the right ankle pitch. In order to reduce these oscillations, a higher weight can be imposed on the accelerations minimization after the impact, or the torque derivatives can be minimized to obtain smoother torque profiles, the latter can be beneficial for both cases with and without SEA.

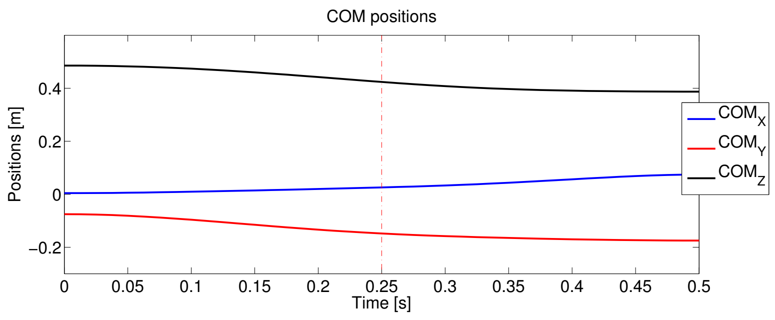


Figure 8.5: Center of mass trajectories of recovery with rigid actuators in a single phase. Red dotted line corresponds to when the push has maximum force.

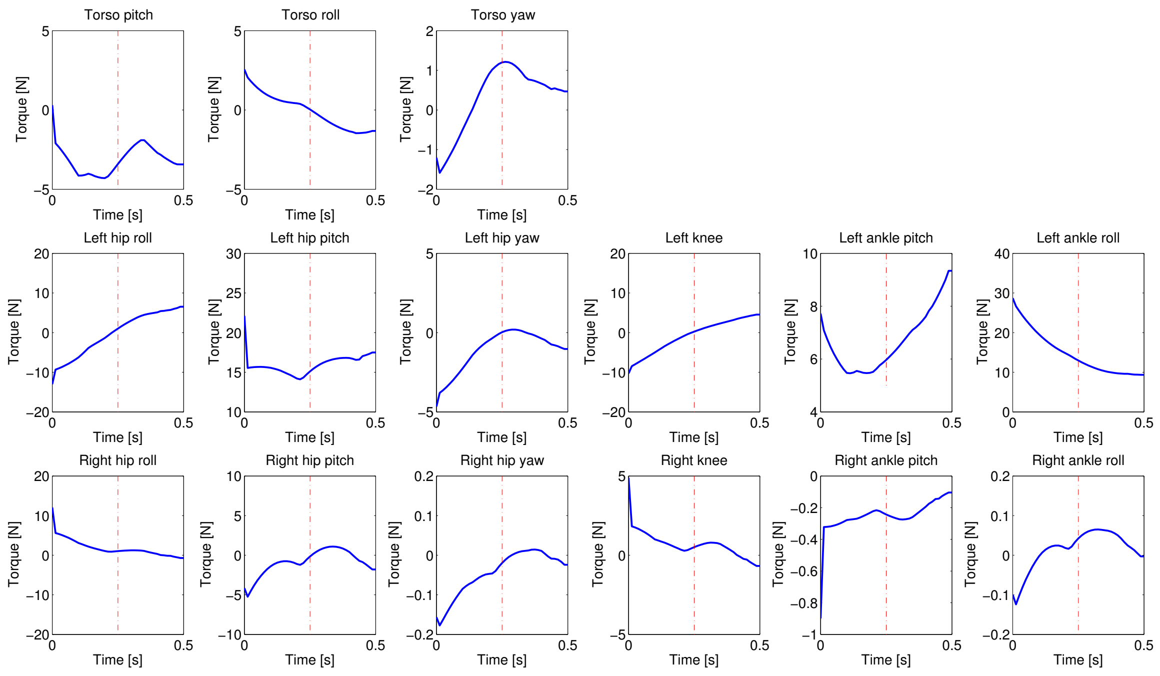


Figure 8.6: Joint torques of recovery with rigid actuators in a single phase. Red dotted line corresponds to when the push has maximum force.

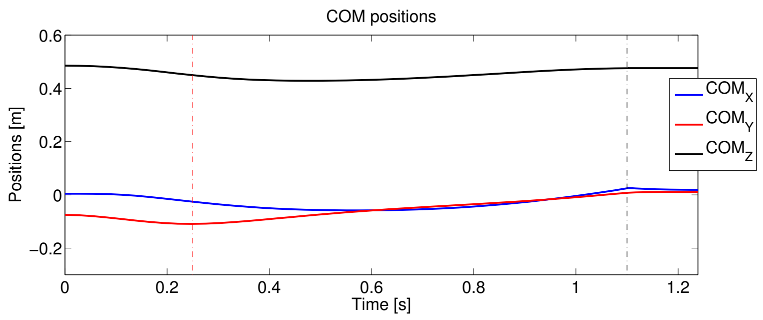


Figure 8.7: Center of mass trajectories of recovery with rigid actuators in a two phases. Red dotted line corresponds to when the push has maximum force. Black dotted line corresponds to the impact.

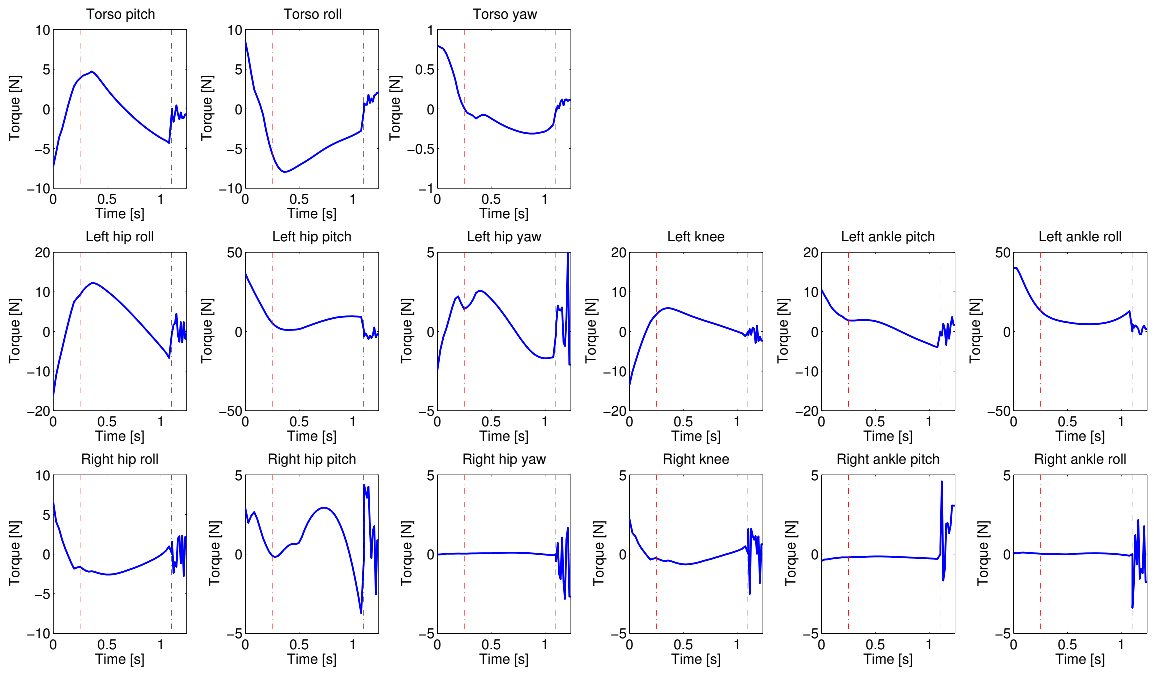


Figure 8.8: Joint torques of recovery with rigid actuators in a single phase. Red dotted line corresponds to when the push has maximum force. Black dotted line corresponds to the impact.

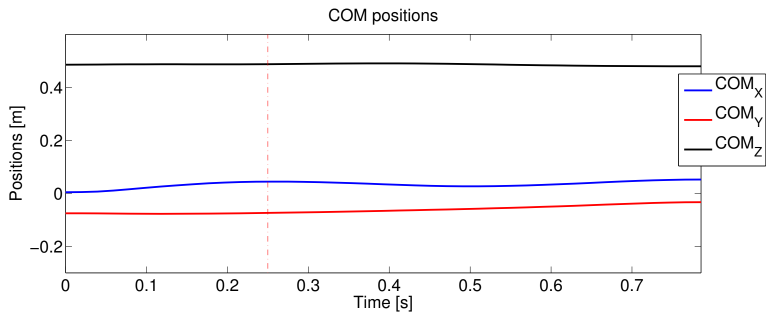


Figure 8.9: Center of mass trajectories of recovery with SEA in a single phase. Red dotted line corresponds to when the push has maximum force.

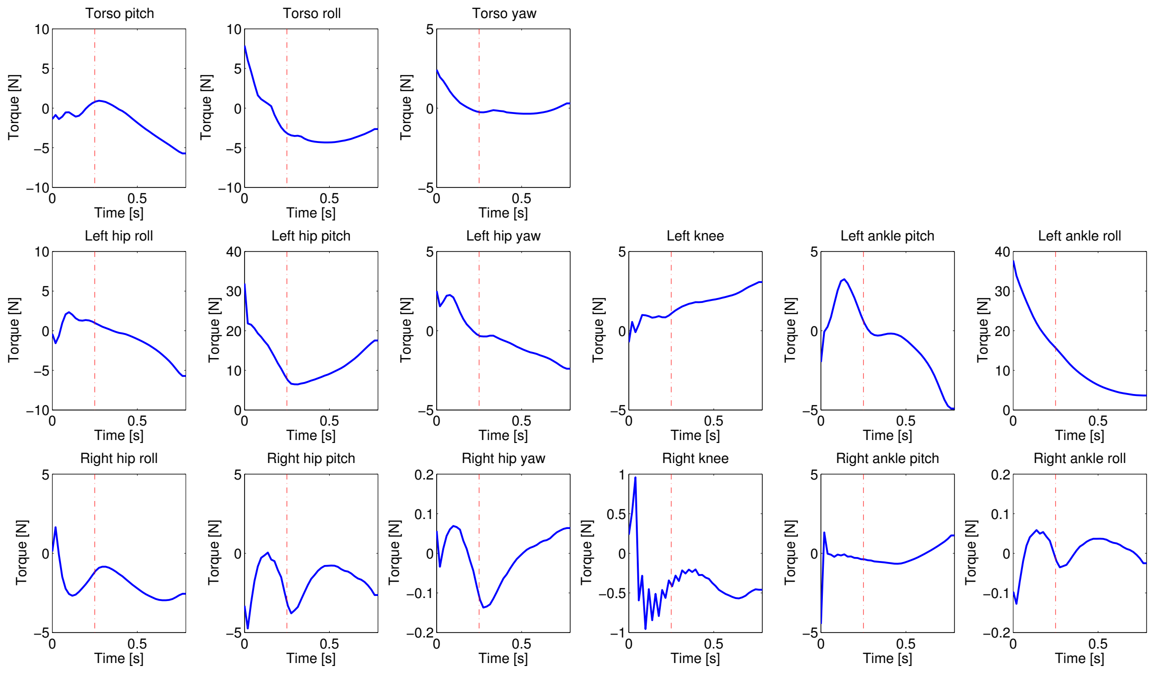


Figure 8.10: Joint torques of recovery with SEA in a single phase. Red dotted line corresponds to when the push has maximum force.

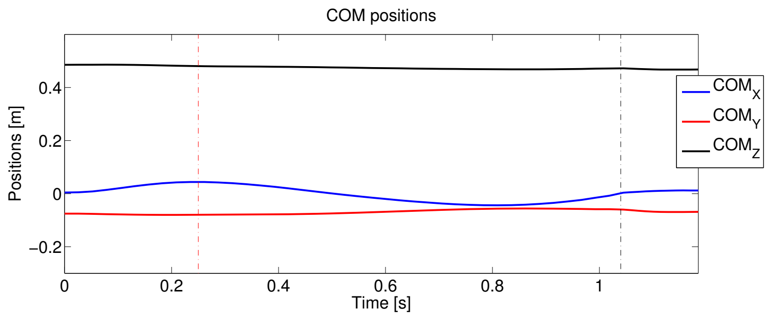


Figure 8.11: Center of mass trajectories of recovery with SEA in a two phases. Red dotted line corresponds to when the push has maximum force. Black dotted line corresponds to the impact.

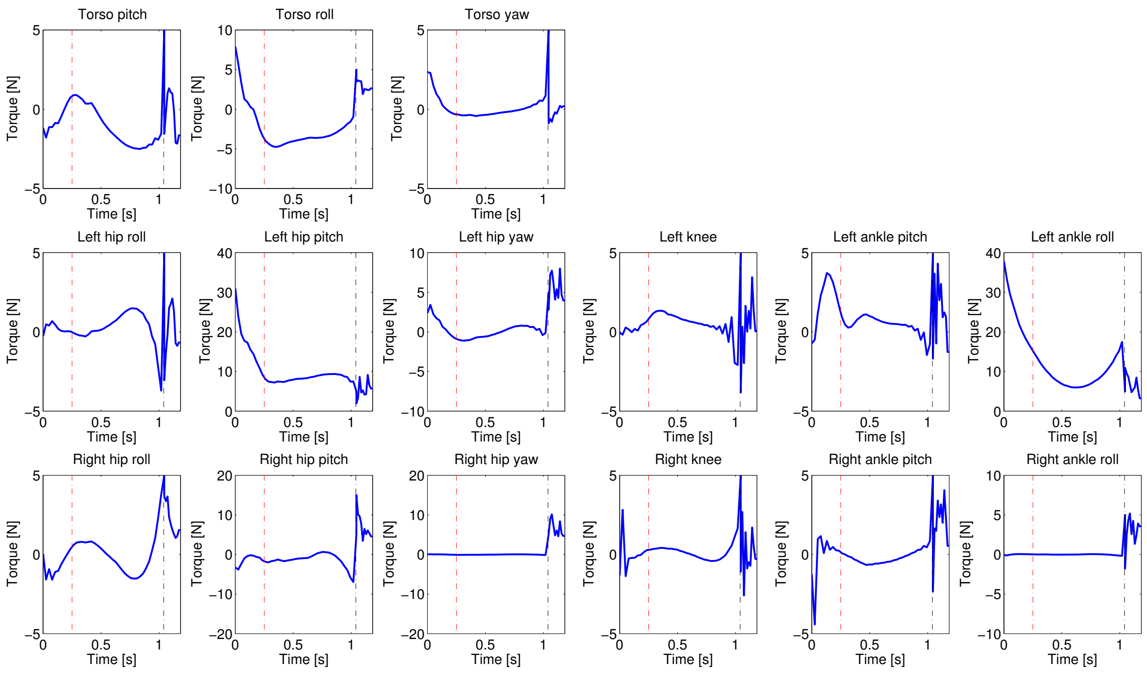


Figure 8.12: Joint torques of recovery with SEA in a two phases. Red dotted line corresponds to when the push has maximum force. Black dotted line corresponds to the impact.

9 Walking motion generation

Walking is a highly challenging task for humanoid robots due to the redundancy of the system and the instability of the bipedal posture. The biggest issue still to be completely solved is probably stability, as whole-body motions have to be carefully adjusted to compensate for the continuous instabiliy.
Two types of walking can be identified for bipedal robots [58]:

- Static walking, in which the projection of the center of mass (CoM) on the floor plane never leaves the support area.
- Dynamic walking, in which the projection of the CoM could leave the support area for certain periods.

Static walking is very stable, but it is a very conservative walking style employed by the first existing humanoid robots. The motions are slow and requires very big feet to achieve big support areas and the motions, and it is definitely not what is being performed by animals or humans. Humans walk in a dynamic way, as they have small feet, they do not need to keep the CoM within their feet, therefore new methodologies all target dynamic walking.

Many different walking control frameworks exist in the literature, ranging from simple assumptions on reduced models to complex whole-body dynamic models. In either case the common goal is to plan where to put the foot at the next step in order to fulfill certain goals and/or constraints and keep the stability of the robot. The planning of the next foot placement location could be achieved with:

- *A priori* knowledge, i.e. the user specifies where to put the foot in the space and then the full body controller tries to reach the target within a predefined set of constraints.
- *Reactive* control, i.e. the controller computes where to put the foot according to current constraints (e.g. environment, obstacles) while fulfilling to certain goals (e.g. desired walking velocity).

The second case is more difficult to achieve, as a big amount of online sensory data collection and analysis is needed and fast reactions need to be achieved.
As the iCub humanoid robot has very few documented walking experiments, and among the existing ones only level ground walking has been achieved, the objective of this chapter is to implement walking motions on the reduced version of the iCub, HeiCub, in order to extensively document and evaluate the walking capabilities of the robot not only on level ground, but also on different environments, which is a first for the iCub robot. The performances are measured and documented with Key Performance Indicators (KPIs) defined in KoroiBot, details will be explained in Section 9.4.1.
The presented method falls into the first case. It uses a so called pattern generator, which generates desired targets in terms of feet and/or CoM positions, and combines it with the generation of whole-body motions based on the given targets.
In the following, the pattern generator based on the 3D inverted pendulum is explained. This work was originally developed in collaboration with Jorhabib Eljaik to achieve level ground

walking motions also on the full body iCub [24] in IIT, in this chapter we describe the improved and augmented version and show the application on the HeiCub robot with experimental results of walking in different environments.

9.1 The 3D inverted pendulum model

A well established and commonly-used method to generate walking motions is based on a coarse-graining of the model of robot. The complex dynamics of humanoid robots does not allow for fast generation of motions with current methodologies and computational powers when taking into account the whole body dynamics. Therefore simplified models are often preferred. One of the most used simplified models is the inverted pendulum, in which the whole mass of the robot is assumed to be in one point, i.e. the CoM, attached to a massless leg of variable length.

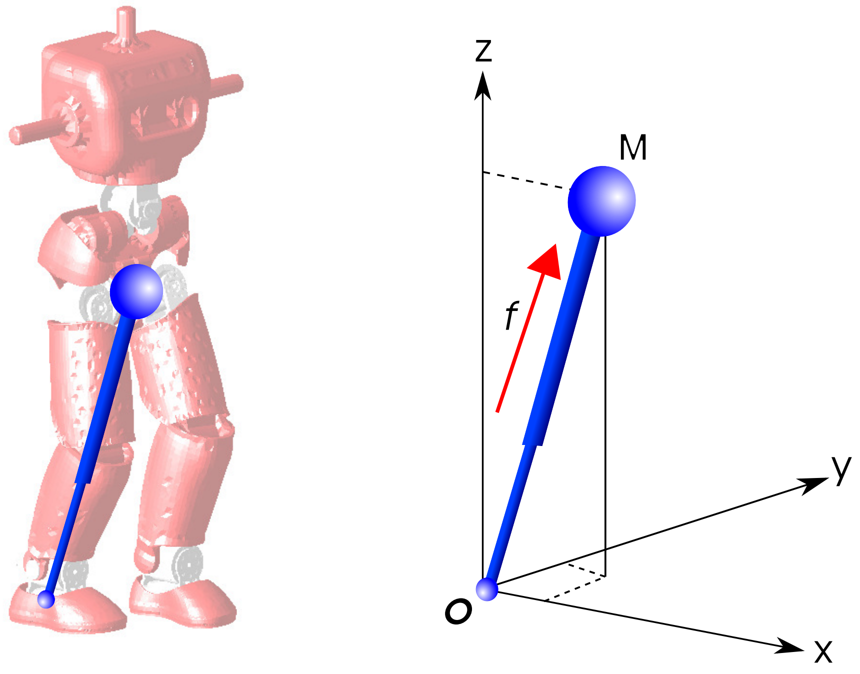


Figure 9.1: Coarse-graining the robot to a 3D inverted pendulum.

The model was introduced first by Kajita [55] to generate walking in 2D, i.e. in the sagittal plane, extended later to 3D. The pendulum equations are simple linear equations, in contrast to the complex nonlinear nature of the equations of motion describing whole body humanoid robots. From the pendulum model as in Fig. 9.1, the equation of motion of the CoM is as follows:

$$
\begin{aligned}
M\ddot{x} &= \frac{x}{l} f \\
M\ddot{y} &= \frac{y}{l} f \\
M\ddot{z} &= \frac{z}{l} f - Mg
\end{aligned}
\tag{9.1}
$$

where l is the length of the pendulum leg, M is the mass of the point mass, f is the kick force and g is the gravity constant, assumed to be -9.81 $[\mathrm{m/s^2}]$ unless specified otherwise.

To linearize the pendulum equations, Kajita introduced a constraint plane:

$$z = k_x x + k_y y + z_c \tag{9.2}$$

having the constants k_x and k_y defining the slope of the plane and z_c the height. Which means that the height of the CoM is constrained to be constant. With the introduction of such a plane, the CoM equations can be written as linear equations:

$$\begin{aligned} \ddot{x} &= \frac{g}{z_c} x \\ \ddot{y} &= \frac{g}{z_c} y \end{aligned} \tag{9.3}$$

which is an easy to solve system with two inputs x and y and two outputs $\ddot{x}$ and $\ddot{y}$.

However, this constraint is very conservative and therefore implies many restrictions on the possible generated motions, such as the humanoid knees being always bent, whereas humans obviously walk with stretched knees. This is because during human walking, the CoM height is variable and not constant.

To keep the system easy to solve for only two unknowns, the variation of the CoM height is introduced as predefined instead of unknown. Which means that, instead of the equations as in Eq. (9.3), the followings are considered:

$$\begin{aligned} \ddot{x} &= \frac{\ddot{z} + g}{z} x \\ \ddot{y} &= \frac{\ddot{z} + g}{z} y \end{aligned} \tag{9.4}$$

where z and $\ddot{z}$ are assumed to be known a priori.

The equations as in Eq. (9.4) are used to generate trajectories for the CoM that respect the stability criteria as per the Zero Moment Point (ZMP) [117], as we will see in the next section.

9.2 Pattern generator

A pattern generator generates reference end effector trajectories (e.g. feet, CoM positions, velocities etc) for a humanoid robot such that stable walking motions can be achieved. It is commonly based on reduced models, e.g. inverted pendulum or table cart. In this section we will describe the pattern generator based on the table-cart model combined with the stability criterion of ZMP [117]. This pattern generator was first proposed by Kajita et. al [56, 57], and allows arbitrary foot placement.

The table cart model is based on the same principle as the inverted pendulum model, and as a matter of fact, they share the same set of equations. The two models fundamentally differ in the relationship between input and output. When using the inverted pendulum model, the unknowns are the ZMP coordinates, therefore the model is used to compute the ZMP coordinates given desired CoM trajectories. In the table-cart model the input and output roles are reversed, i.e. the CoM trajectories are unknown and are computed from given ZMP coordinates.

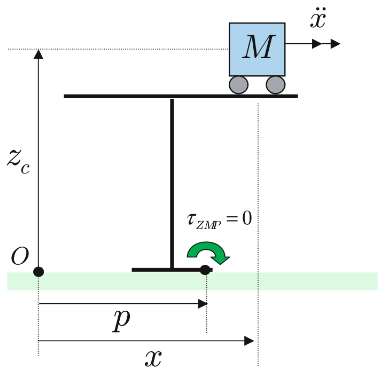


Figure 9.2: Table cart model. Picture from [57].

From Fig. 9.2 and Eq. (9.3) the center of pressure p (ZMP) can be computed as:

$$\mathbf{p} = \mathbf{x} - \frac{z_c}{g}\ddot{\mathbf{x}} \tag{9.5}$$

where $\mathbf{p} = [p_x, p_y]$ represents the ZMP coordinates, $\mathbf{x} = [x, y]^T$ the CoM coordinates. These equations are referred to as the *ZMP equations* and are the ones originally used in [57]. As in Eq. (9.4), the assumption of z_c constant is dropped, therefore the *ZMP equations* result to be:

$$\mathbf{p} = \mathbf{x} - \frac{z\ddot{\mathbf{x}}}{\ddot{z} + g}. \tag{9.6}$$

The above illustrated theory is applicale for both offline and online pattern generation. In the following the offline version is explained in detail. The method was first proposed by Kagami and Nishiwaki in [54], where the CoM acceleration is approximated with the second order central differentiation:

$$\begin{aligned} \ddot{x}_i &= \frac{x_{i-1} - 2x_i + x_{i+1}}{\Delta t^2} \\ \ddot{y}_i &= \frac{y_{i-1} - 2y_i + y_{i+1}}{\Delta t^2} \end{aligned} \tag{9.7}$$

where $i = 1...N$ is the index of the sample with N being the total number of samples.
Given the approximation, the ZMP equations as in (9.6) take the following discretized form:

$$\mathbf{p}_i = a\mathbf{x}_{i-1} + b\mathbf{x}_i + c\mathbf{x}_{i+1} \tag{9.8}$$

where the terms a,b and c are as follows:

$$\begin{aligned} a_i &= -\frac{z}{(\ddot{z} + g)\Delta t^2} \\ b_i &= \frac{2z}{(\ddot{z} + g)\Delta t^2} + 1 \\ c_i &= a_i \end{aligned} \tag{9.9}$$

which take into account the variation of the CoM height, contrasting to the original equations

proposed in [54].
Eq. (9.8) can be written in the form $\mathbf{p} = \mathbf{A}\mathbf{x}$, considering all discretized points $i = 1...N$:

$$\begin{bmatrix} p_1 \\ p_2 \\ \vdots \\ p_{N-1} \\ p_N \end{bmatrix} = \begin{bmatrix} a_1 + b_1 & c_1 & 0 & & \\ a_2 & b_2 & c_2 & & \\ & \ddots & \ddots & \ddots & \\ & a_{N-1} & b_{N-1} & c_{N-1} & \\ & 0 & a_N & b_N + c_N & \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{N-1} \\ x_N \end{bmatrix}. \tag{9.10}$$

The system is then solved for the CoM coordinates $\mathbf{x}$:

$$\mathbf{x} = \mathbf{A}^{-1}\mathbf{p} \tag{9.11}$$

where the matrix $\mathbf{A}$ has dimension $N \times N$.
Clearly, the higher the number of discretized points, e.g. due to smaller time itnervals or longer walking distances and/or times, the bigger the matrix **A** becomes and the more computational expensive becomes its inversion. Given that the computations are being performed offline and the computation times are reasonably low with a standard laptop with Intel i7 CPU with 4 cores (in average less than a minute for 5000∼10000 points), this does not represent a problem at this stage of the work.

9.3 Software module: concept and implementation

As we have seen from the previous section, the pattern generator takes the ZMP trajectory as input and computes for the CoM trajectory which ensures the table cart to move along the desired ZMP. The primary objective is to use the pattern generator with a bipedal robot. Therefore we need to compute reference trajectories for the end effectors of the robot (in this case, feet and CoM) that can be used to retrieve whole-body motions such that resulting motion is stable.
To achieve this goal, the software modules that we implemented consists of different blocks, as pictured in Fig. 9.3. Every single block represents an independent software module. In this case, the blocks *Pattern generator* and *Trajectory generator* are strictly tied together and depend on user defined parameters given in Tab. 9.1. They were implemented as separate modules as the *Pattern generator* could also be used independently or with a different *Trajectory generator*.

Trajectory generator and pattern generator

The trajectory generator represents the first step of the working flow. It is implemented such that a set of user defined feet holds pattern and CoM heights given at desired times can be specified as input, or it can automatically generate these patterns according to the parameters "n_strides", "z_c_offset" and "type". In the automatic case, a set of feet pattern that has equidistant step width and length is being generated. The automatically generated CoM height variation pattern depends highly on the "type" parameter. In the case of level ground walking we compose the pattern by introducing a height offset corresponding to half time of the single support time, such that the robot stretches the stance leg during this phase and then bends it to go towards the double support phase. In the case of stairs the height offset is introduced in correspondence to when the foot has to step over the stair, such that the robot can clear the edge of the stair by stretching the stance leg and impose less force on the knee joint of the stance leg.
Currently automatic pattern generation has been implemented for the following walking cases:

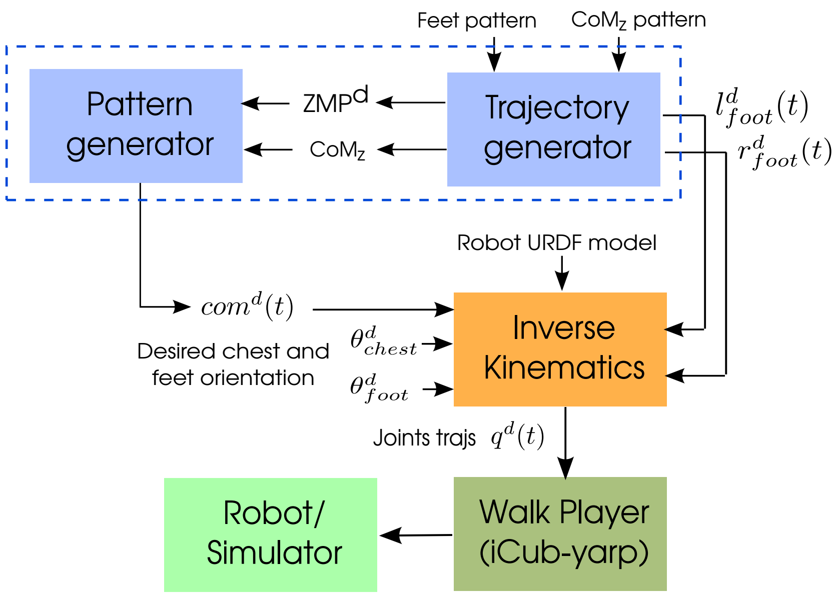


Figure 9.3: Workflow from pattern generator to the robot. The blue blocks in the dotted lines are implemented in MATLAB and generate the reference trajectories. Then the inverse kinematics block computes the whole-body joint trajectories using a specified URDF model, then the joint angle trajectories are streamed to the robot using the Walk Player. Code relative to the dotted blue box is available at https://bitbucket.org/yue_hu/pattern_generator, iCub/HeiCub specific implementation of the inverse kinematics box is available at https://bitbucket.org/yue_hu/icub_walk_pg_ik, the Walk Player is available in [3].

Name	Description
ts	sampling time
z_c	CoM height
z_c_offset	height variation of CoM
n_strides	number of strides
T_stride	time to perform 1 stride (2 steps)
T_switch	double support time
step_width	distance between two feet
step_length	distance traversed during 1 step
theta	inclination during slope walking
stair_length	length of the stair
stair_height	height of the stair
right_step_first	by default the first to step is the left foot
type	specifies the walking environment

Table 9.1: Pattern generator parameters

- Level ground straight walking,
- Slope walking up/down,
- Stairs climbing up and down (going down backwards and not forward).

The possibility of using arbitrary user-defined patterns gives the possibility of walking in irregular scenarios, e.g. step stones, variable step length/width.
In either case, the feet patterns are used to generate:

- ZMP trajectory, generated by connecting all the feet patterns and to guarantee that smooth transition from one foot to the other is performed, as the ZMP has to lie inside the support polygon to ensure stability. Specifically, the parameter "T_switch" corresponds to the double support time in which the ZMP shifts from one foot to the other, this time is included in the stride time. Therefore, the single support time, in which the leg swings from one foot hold to the next one, corresponds to (T_stride—T_switch)/2. This trajectory is to be used in the pattern generator.
- Desired CoM height variation CoM_z and the acceleration of this trajectory are generated using spline interpolation such that smooth height variation is guaranteed. These trajectories are used in the pattern generator as a priori known variation of the CoM height.
- Feet trajectories, combined with the step height parameters: these trajectories are generated with spline interpolation, such that the feet are lifted correctly to achieve the proper motions during single support phases.

The generated ZMP and CoM_z trajectories are then fed to the pattern generator, which implements the table-cart model as described in the previous section, from which the CoM trajectory on the walking plane (x, y) is computed.
Both the trajectory and pattern generators are implemented using MATLAB (The MathWorks, Inc.).

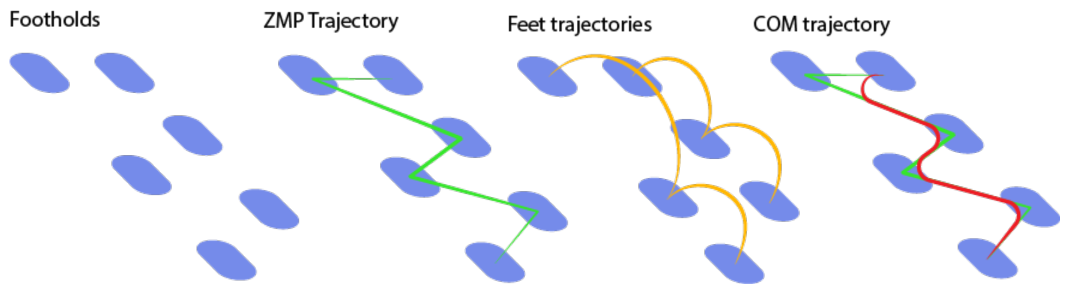


Figure 9.4: The ZMP and feet trajectories are generated from the feet patterns, then the CoM is obtained from the pattern generator.

Inverse kinematics

The obtained CoM trajectory $\mathbf{com}_d(t)$ is given as input to the inverse kinematics module, combined with the desired feet trajectories and desired chest and feet orientations, to compute the desired whole body joint trajectories.
The desired chest orientation serves to keep the chest of the robot upright during walking, while the feet orientation keeps the feet flat on the walking surface, e.g. on level ground the orientation is zero, while on a slope it corresponds to the slope inclination.

The inverse kinematics module retrieves the kinematics model of the robot from an external URDF file, therefore there is no a priori knowledge of the model and the model of any bipedal robot could be used.
The target end effectors used are:

- Left and right foot, “l_sole” and “r_sole” for the HeiCub model, specifically the centre of these two frames are used as target position.
- Chest, “chest” for the HeiCub model, which is the last link of the robot upper body kinematic tree. In this case the CoM is attached to the chest local frame, in order to use also the torso joints to reach the desired target. The local CoM is updated to the real CoM at every sample in order to match the correct CoM and have higher stability.

The inverse kinematics module used in this thesis is the IPOPT [119] and RBDL [33] based implementation as described in Section 6.2.

Walk Player

The joint trajectories obtained from the inverse kinematics module are now ready to be used on the robot. The YARP [37] based C++ module *Walk Player* is used for this purpose.
The *Walk Player* was originally developed by IIT and used in [24] to achieve walking on the full body iCub robot, which used the very first version of the workflow described in this thesis.
The module is part of the software package *codyco-modules* [3], which collects the software developed for the European Project CoDyCo [2].
The module consists in a thread that runs at specified thread rate and uses the YARP [37] interfaces to communicate with the robot. The module reads the files containing the joint trajectories, then first brings the robot to the initial positon with a minimum jerk position controller, then streams the joint positions using the position direct mode at the desired thread rate. This thread rate has to correspond to the sampling time of the pattern generator. The player received commands via YARP RPC interfaces, which allow to stop, restart or reset the motion at any moment.
The original version of the code has been modified to accomodate also data recording from the robot while executing the motion, such that sensory data such as joint encoders and force torque sensors can be collected in a synchronised way with the streaming of the motion.

9.4 Performance evaluation

As stated in the introduction of this chapter, we aim at using the implemented software module to perform extensive experiments on the HeiCub robot and document these tests in a systematic way. This means to analyze the performance of the HeiCub during walking tasks and create benchmarks that can be further used to compare different methods and quantitatively measure possible improvements. Such an analysis carried out on the HeiCub is perfectly comparable with any other iCub equipped with the same generation of hardware, given that, despite being a reduced version, it is still a standard iCub robot.
Benchmarking has always been a highly discussed topic in engineering. Unfortunately, in humanoid robotics it is difficult to define standards due to the variety of existing design, control and purpose of humanoid robots. As the KoroiBot project aims at improving existing humanoid robots walking capabilities, a precise definition of a way to measure the capabilities of the robots was necessary. Therefore, the project has defined the Key Performance Indicators (KPIs) [99].

The KPIs do not include only indicators for robot capabilities, but are split into different categories, such as *human likeness*, *computational indicators* and *technical indicators*. These indicators are not meant to compare different robots, as it is highly difficult to compare the capabilities of a robot which is only 30 [cm] high and walks in 2D with a full size humanoid of 1.7 [m]. The indicators are therefore meant to compare the achievements obtained on the same robot with different methods.
In the following a set of technical KPIs are defined for the iCub robot, which are then measured during the experiments performed on the HeiCub walking on level ground, slopes and stairs.

9.4.1 Key Performance Indicators (KPI)

In KoroiBot, the KPIs are defined for each robot according to the user experiences and the equipment (in terms of sensors and hardware) and capabilities of the robot itself.
The indicators defined for the iCub/HeiCub are based on measurable quantities that can be obtained from the sensors of the robot itself or quantities about the environment external to the robot, e.g. slope inclination.
As the HeiCub was a brand new robot at the beginning of this thesis, we were not aware of any capability of the robot, the KPIs have been defined with the implementation of the first walking motions and refined with the experiments. At every new scenario the robot was able to walk in, new KPIs were added for the specific scenario.

Most of the indicators defined below are used for all the environments, however there are also environment specific indicators:

- Cost of transport:

$$\mathbf{E}_{CT} = \frac{\sum_{m=1}^{M} \int_{t_0}^{t_f} I_m(t)V_m(t)dt}{m_{robot} \cdot g \cdot d} \tag{9.12}$$

  Where M is the total number of motors, in this case $M = 15$, I_m and V_m are the current and voltage measurements of the motor m, m_{robot} is the mass of the robot and d is the travelled distance.

- Froude number, which is a dimensionless number used in fluid mechanics to characterize the resistance of an object moving through water. Alexander used it to characterize animal locomotion, given the that also legged locomotion is a dynamic motion in gravity [11]. Here we use it for level ground only:

$$Fr = \frac{v_{max}}{\sqrt{g \cdot h}}, \text{ for } h = l_{leg} \tag{9.13}$$

- Maximum walking velocity v_{max}, only for level ground and slope.
- Maximum slope inclination α_{max}, for slope only.
- Maximum stair height h_{max}, for stair only.
- Precision of task execution: defined as a set of tracking errors[1] of the CoM, ZMP and

[1]Please note that quantities such as CoM and ZMP depend on the floating base estimation, which is now implemented in a module that performs the estimation using an odometry based method and the kinematic model of the robot in [3], which is not highly accurate. For this reason the CoM and ZMP errors are not very precise and might be smaller than they result in this thesis. If floating base estimation is to be used for online corrections, an improvement of the current version is necessary.

joint angles:

- CoM_e
- ZMP_e
- q_e

which are computed as RMSE (Root Mean Squared Error). Of these the most important is the ZMP error, as it represents the stability criterion, a certain margin is however allowed as long as the ZMP remains inside the support polygon.

The KPIs were measured for each walking scenario and the tables are reported in the corresponding environment in the next section.

9.4.2 Experimental results

.
With the presented framework, we could achieve walking on both HeiCub and full iCubs located in IIT facilities, by using the proper URDF model of the robot.

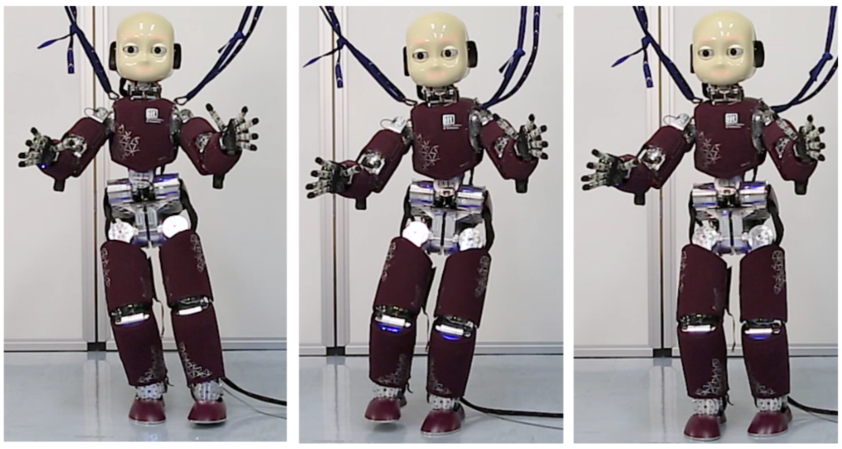


Figure 9.5: Full iCub with battery pack walking on level ground.

On the full iCub we performed walking on level ground as in Fig. 9.5, while the experiments carried out on the HeiCub robot comprise walking on: level ground, up and down a slope, up stairs as in Fig. 9.6.
In the following we illustrate the experiments done on the HeiCub.
In every environment we started the tests with conservative velocities and quantities such as inclination and stair height, then increased to push the limits that are achievable with the adopted method. Every case is tested for 5 times to ensure reproducibility of the motion and success rates were measured, as shown in Tab. 9.2.
The cases shown in the plots and reported in the KPIs tables are referred to the best achieved case in terms of maximum velocity, slope inclination and stair height.
It should be noted that walking down stairs was not performed due to restricted joint limits of the ankle pitch joints as in Fig. 9.7. When walking down the stairs, the ankle needs a very wide angle even in humans, where it reaches up to -40∼-45 [deg], as observed from the data from the KoroiBot Motion Database [1]. This limit needs to be higher in humanoid robots with flat feet such as the iCub, as there is no possibility of performing tip-toe rolling motions.

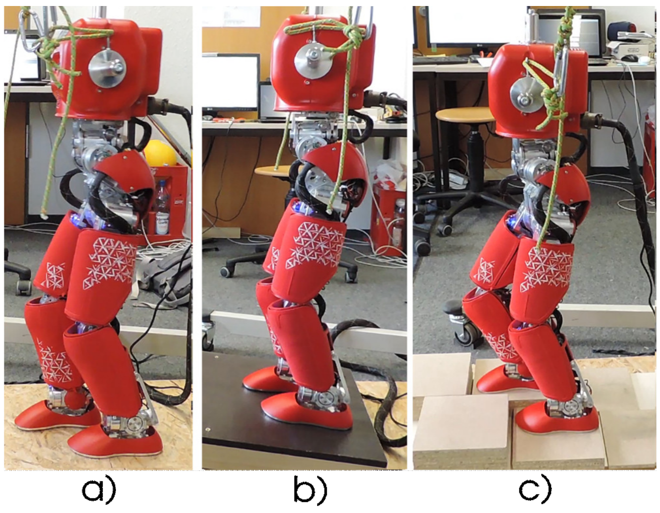


Figure 9.6: Tested environments for HeiCub: a) level ground, b) slope (up and down), c) stairs.

Level ground	T_stride [s]	Step length [m]	Success rate
	8	0.10	100 %
	6	0.10	100 %
	5	0.10	100 %
	4	0.10	100 %
	3	0.10	0 %
Slope up	T_stride [s]	Slope inclination [deg]	Success rate
	8	4.5	100 %
	8	7	100 %
	6	7	100 %
	5	7	40 %
Slope down	T_stride [s]	Slope inclination [deg]	Success rate
	8	4.5	100 %
	8	7	100 %
	6	7	20 %
	5	7	0 %
Stair up	T_stride [s]	Stair height [m]	Success rate
	10	0.01	100 %
	10	0.02	100 %
	8	0.02	100 %

Table 9.2: Walking environments and success rates over 5 trials for the HeiCub robot

Level ground

As shown in Tab. 9.2, level ground walk was tested with different walking velocities where the step length was kept constant at 0.1 [m], corresponding to the distance between the two feet when they are both on the ground, this means that the actual distance travelled by the foot in swing phase is twice the step length, as we can see also in Fig 9.8a.

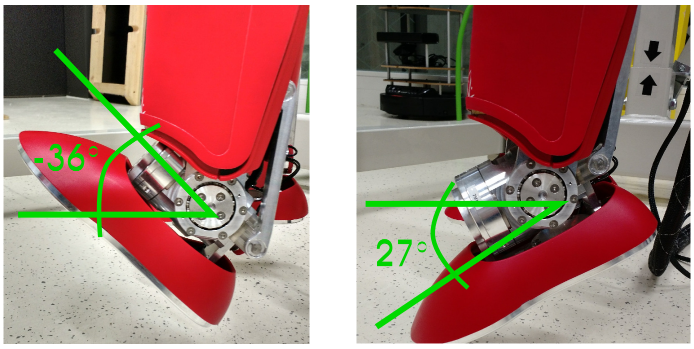


Figure 9.7: Limits of the ankle joint of HeiCub (and all iCubs with legs version 2.5).

Table 9.3: KPIs in level ground walk

Cost of transport E_{CT}	4.27
Maximum velocity v_{max}	0.037 [m/s]
Froude number Fr	0.0165
Execution precision	$\text{CoM}_e = 0.038$ [m] $\text{ZMP}_e = 0.09$ [m] $q_e = 2.11$ [deg]

Measurements are referred to the case as in Fig. 9.8.

Here we report only the fastest case achieved, as shown in Fig. 9.8, which corresponds to the case of 4 [sec] per stride. In this case double support time is set to 1 [sec] and single support time to 1.5 [sec].
The CoM was allowed to vary in height for 0.015 [m], this was extremely useful in achieving bigger step lengths without reaching joint limits as the robot stretches the stance leg while the other one is moving forward. This walking style is also much closer to what is done during human walking.
With 4 [sec] the CoM can be quite closely tracked and the ZMP is inside the boundaries of the support polygon as shown in Fig 9.8a.
The current limits were not reached with slower motions, however they had to be extended to 5.5 [A] as we increased the walking speed. The consumption in this case is higher on the hip roll and knee motors, as they are the most demanding ones during swing phases where the whole weight of the robot is on a single leg.
Further decrease of the stride time the walking could not be achieved anymore due to high destabilization of the motion. We computed the CoM, ZMP and joint angles tracking errors, and observed that there is an increase in the errors with the increase of walking velocity. We recorded the data also for the failed cases in order to analyze the reason of the failure.
With stride time bigger than 4 [sec], i.e. stable walking cases, the average CoM tracking error is in the range of $0.03 \sim 0.04$ [m], and average ZMP tracking error of $0.08 \sim 0.09$ [m]. The joint trajectories tracking errors increase with the reduction of the stride time, ranging from 0.57 [deg] in the 8 [sec] case up to 2 [deg] in the 4 [sec] case. When stride time is reduced, e.g. for walking with stride time of 3 [sec] the errors are much bigger, where the joint tracking error is in average of 6 [deg]. This led to very unstable motions that brought the CoM far from the desired one and consequently the ZMP out of the support area, from which the robot could not recover.

Slope

Table 9.4: KPIs in slope walk up

Cost of transport E_{CT}	4.27
Maximum velocity $v_{max,}$	0.03 [m/s]
Maximum angle α_{max}	7 [deg]
Execution precision	$\text{CoM}_e = 0.053$ [m] $\text{ZMP}_e = 0.12$ [m] $q_e = 1.03$ [deg]

Measurements are referred to the case as in Fig. 9.9.

Table 9.5: KPIs in slope walk down

Cost of transport E_{CT}	5.61
Maximum velocity $v_{max,}$	0.026 [m/s]
Maximum angle α_{max}	-7 [deg]
Execution precision	$\text{CoM}_e = 0.048$ [m] $\text{ZMP}_e = 0.09$ [m] $q_e = 0.88$ [deg]

Measurements are referred to the case as in Fig. 9.10.

Walking up and down slopes experiments are performed with two inclinations, respectively of 4 and 7 [deg]. The step length was initially kept at 0.05 [m] then increased to 0.1 [m], seeing that the motions achieved with smaller step length were very stable.
The fastest motion succesfully achieved up a slope is with inclination of 7 [deg] and stride time 5 [sec]. As we can see from Fig 9.9a the CoM and the ZMP could be tracked but not as closely as in the level ground case. In particular the ZMP is shifted in the x direction due to the robot slipping slightly down the slope. These are the main reasons for which the success rate is of 40% only, against the more stable case of 6 [sec].
The tracking errors are in average higher than the level ground case, where the CoM tracking error is in the range of $0.04 \sim 0.05$ [m], the ZMP tracking error of $0.11 \sim 0.13$ [m] and joint angles tracking error of 0.8 [deg]$\sim$ 1 [deg].
In this case the height variation of the CoM helped to achieve the higher slope inclination. The knee motor requires less current here due to the inclination of the slope that allows the knee to have smaller motion range.
In the downslope case we could achieve stable motions with the same inclination of 7 [deg], but with a minimum stride time of 6 [sec] and a lower success rate of only 20%. As shown in Fig. 9.10a, the ZMP is slightly shifted forward, for the same reason as in the upslope case. As the motion is slower than walking up slope, the tracking errors are smaller. The CoM tracking error is in the range of $0.4 \sim 0.5$ [m], the ZMP tracking error of $0.08 \sim 0.1$ [m] and the joint tracking error of 0.8 [deg]$\sim$ 1 [deg]. Stable motions with smaller stride time could not be achieved successfully because downslope situations are more difficult to handle given that the robot shifts the weight forward and the gravity and slippage effects destabilize the motion.
As the robot needs to bend the knee more to keep the weight on the stance leg during single

support phase, in which the swing leg is shifting downwards, the knee motor of the stance leg demands a higher current consumption. The same happens in humans, where walking down slopes is more demanding for the knee joint.
With a slope inclination of 7 [deg] the ankle pitch joint is very close to its limits. With proper control of the robot a slightly higher inclination might be achieved, however mechanical limitations do not allow stable motions in higher inclinations.

Stairs

Table 9.6: KPIs in stair climbing

Cost of transport E_{CT}	5.06
Maximum step height h_{max}	0.02 [m]
Execution precision	$\mathrm{CoM}_e = 0.09$ [m] $\mathrm{ZMP}_e = 0.14$ [m] $q_e = 0.9$ [deg]

Measurements are referred to the case as in Fig. 9.11.

During stair climbing, a higher effort is required in the knee joint also for humans, and we expect the same to apply for humanoid robots. Therefore, in the case of stair walking we tested first for a height of 0.01 [m], where we had experienced current limits on the knee motor which led the knee joint to switch to hardware fault mode. This limit had to be increased to 5.5 [A] in order to succesfully achieve the motion. For higher steps, the knee current limit was increased up to 6.5 [A], as we can observe from Fig. 9.11c. To avoid damaging the robot, this limit was not further increased and higher steps and walking velocities have not been tested.
As it is the most challenging motion among the ones tested, tracking errors in stair walking are the highest one. While the average joint angles tracking error is of 0.9 [deg], the CoM tracking error reached up to 0.09 [m] and the ZMP tracking error up to 0.14 [m].
As most of the walking humanoid robots, iCub also has legs that are short with respect to its feet sizes. This means that in stair walking, it is kinematically possible to perform a stair per step as humans do, but it would be dynamically demanding resulting a in a very high current required for the knee motor. In order to avoid this, the robot should climb one stair per time, i.e. put both feet on the stair at every second step. In our case, in order to achieve a more "human-like" stair walk, we created a scenario in which the stairs are shifted, as we can see from Fig. 9.6 and 9.11a. This means that except for the first and last steps, where the feet of the robot are aligned on the same level, the actual height performed is double of the step height (0.02 or 0.04 [m]).
Despite not being able to perform walking downstairs in a human-like way (walking down in the frontal direction), the HeiCub is able to walk downstairs backwards, in the exact same way it performs walking upstairs but with reversed reference trajectories.

Summary

The motions that were tested on the HeiCub are stable (within the ZMP criterion) within limitations of the robot and the applied methodology. From these experiments we were able to gather interesting information about the walking capabilities of iCub and measured the KPIs that can serve as reference for future improvements and development of new control frameworks.

From all the walking scenarios we analyzed, it is possible to summarize that the joint limits of the iCub are conservative for walking in environments different from flat ground.The biggest limitation is caused by the ankle pitch. Given that the feet of the robot are one single block with rigid flat surface, without the possibility of tip toe flexibility, it is necessary to increase this limit. The weakness of the motors represent a challenging issue of iCub in performing challenging walking motions. A replacement of the motors with more powerful ones is necessary to walk in more complex environments, however it is also difficult to use powerful motors in such a small robot, as powerful motors usually also have bigger sizes.

9.5 Other walking control frameworks

Based on the successful achievement of walking motions with the iCub robot, other walking control frameworks have been or are in development for the very same robot.

In particular, an online pattern generator has been implemented and extensively tested on the HeiCub, the work is carried out in [106] and consists of using a pattern generator which is also based on the 3D inverted pendulum. The difference with the work described in this chapter is that the target positions are no longer precomputed but are generated online according to specific given parameters.

Another work is the one tested in [24], which uses in combination an early version of the work described in this thesis with a momentum based torque controller [83].

In the following we perform a short description and discussion of the two methods.

Online pattern generator

The online pattern generator follows an idea developed initially by Herdt of "Walking without thinking about it" [43], where the robot walks with a given reference velocity and used model predictive control. The idea has been taken further by Naveau and Kudruss [82] who implemented a nonlinear version of the work to be used on the humanoid robot HRP-2 located in LAAS-CNRS, Toulouse, France. This version uses the nonlinear model of the 3D inverted pendulum and model predictive control, therefore it is also referred to as the NMPC (Nonlinear Model Predictive Control) Pattern Generator, where the optimization problem is solved in real time with qpOASES[35]. The NMPC Pattern Generator generates the reference feet placement and center of mass trajectories according to desired center of mass velocity and single and double support times.

The work has been ported to the HeiCub robot in [106], in which the workflow is similar to the one as described in Fig. 9.3. The main difference is that the references are generated online and therefore to bring the motion on the full body robot, the inverse kinematics module is run in real time concurrently with the pattern generator.

The control loop is also closed on the tracking of the position of the center of mass, therefore the current center of mass position is computed and used to correct the next references accordingly.

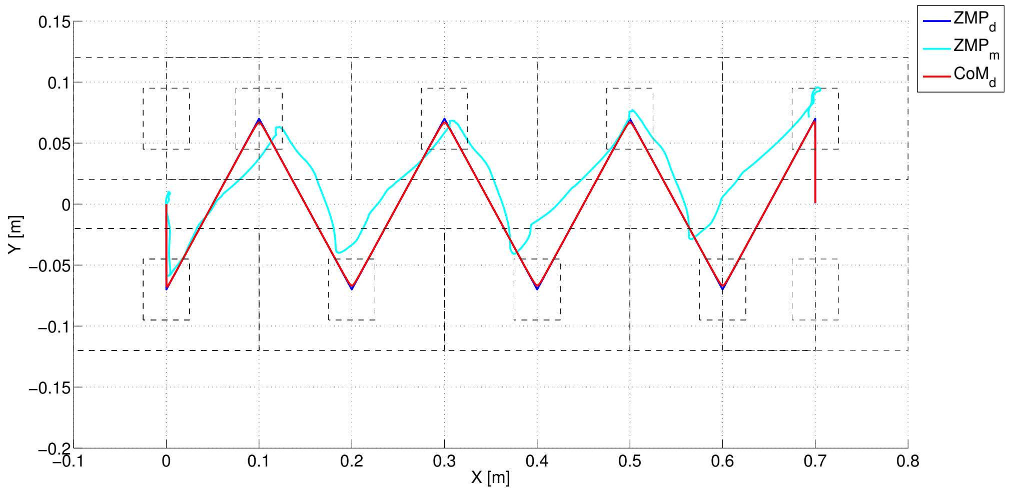
A modified version of the "Walk player" is used to send commands to the robot and to retrieve real time information for the feedback.

With the NMPC Pattern Generator the HeiCub robot successfully performed walking in different directions on level ground and the obtained motion proved to be faster and more stable than the ones obtained with the offline pattern generator. In particular the KPIs were measured again and compared with the ones presented in this chapter, details are listed in [106].

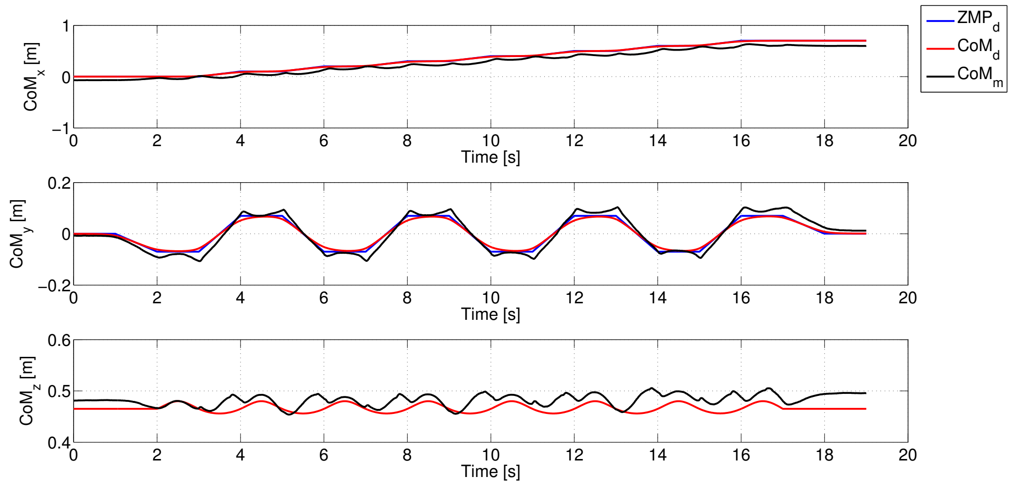
Torque control

The torque controlled walk is the only other walking method that has been tested on the iCub so far. It combines the early version of the pattern generator described in this chapter, when the only supported walking environment was level ground, with a momentum based torque controller described in [83].
The torque balancing controller has been used previously to achieve highly dynamic balancing motions with also contacts switching, i.e. from double to single support and vice versa. The controller allows to follow desired center of mass trajectories as well as whole body postures, while compensating for possible external perturbations by means of the torque balancing control.
The pattern generator framework was used to generate references for the torque balancing, which treats the walking as a state machine, where the states change according to contacts information. The highest priority of the controller is the adjustment of the linear and angular momentum, and only as low priority the tracking of the posture.
An offline pattern generator is however not the ideal type of framework to be used with the torque balancing controller, as the center of mass trajectory as well as the whole-body joint trajectories should cope with the actual state of the robot that is adjusted according to the momentum.

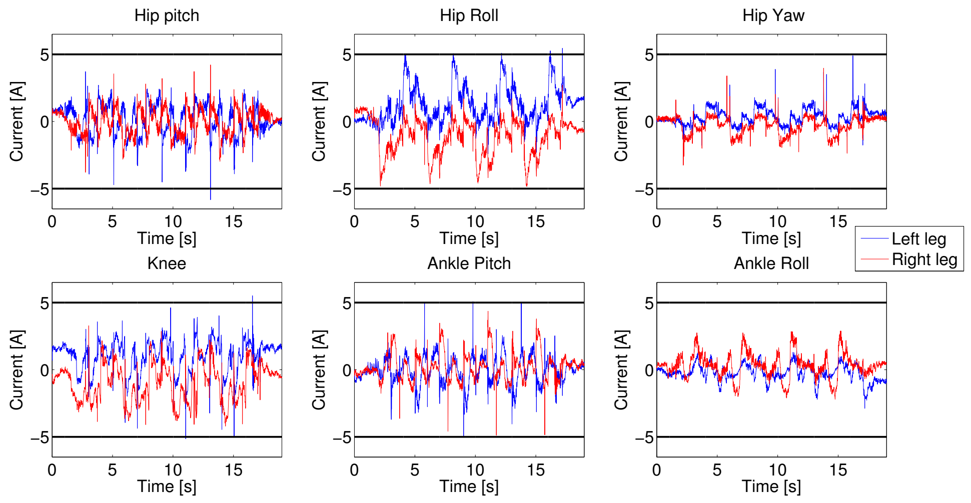
The only environment on which the robot could be tested in both the online pattern generator and torque control cases was level ground. A variation in the environment such as the introduction of slopes and stairs on which the robot has to step on is still a challenging open issue in bipedal robot locomotion, as many assumptions that are easily made on flat floor are not valid anymore, e.g. constant feet orientation. From this point of view, the offline pattern generator, despite being a pure offline generation, could still give interesting insights on the walking capabilities and limitations of the iCub robot that have never been exploited and quantified before.



(a) Feet pattern with ZMP and CoM

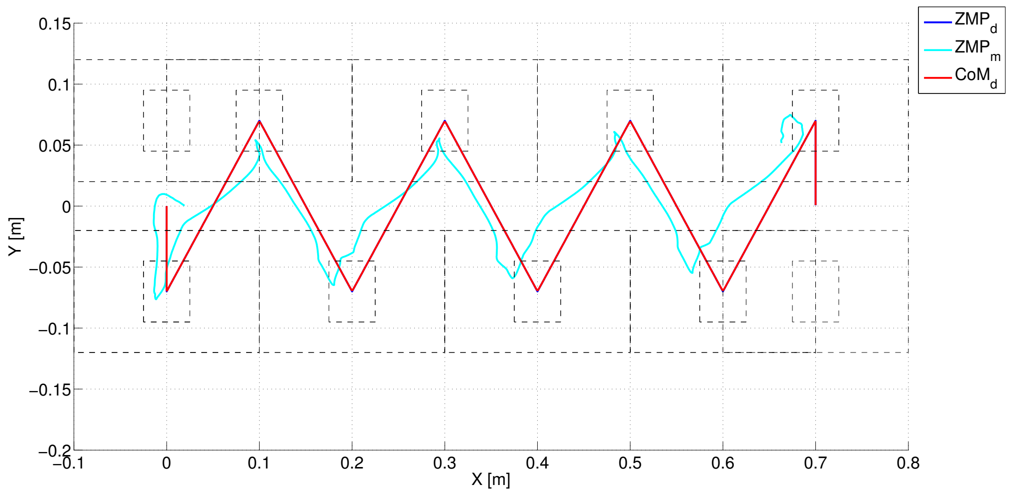


(b) ZMP and CoM trajectory

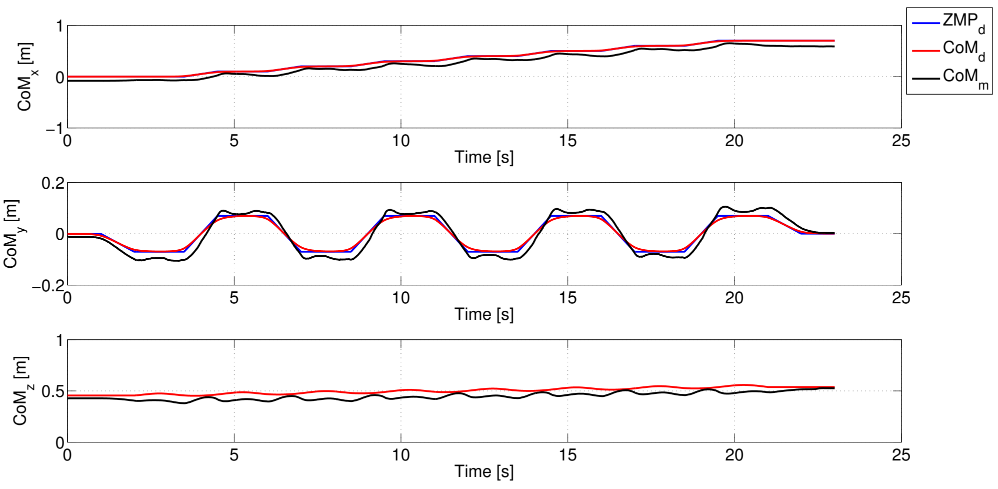


(c) Currents of leg motors, black lines are the default current limits of +/-5 A.

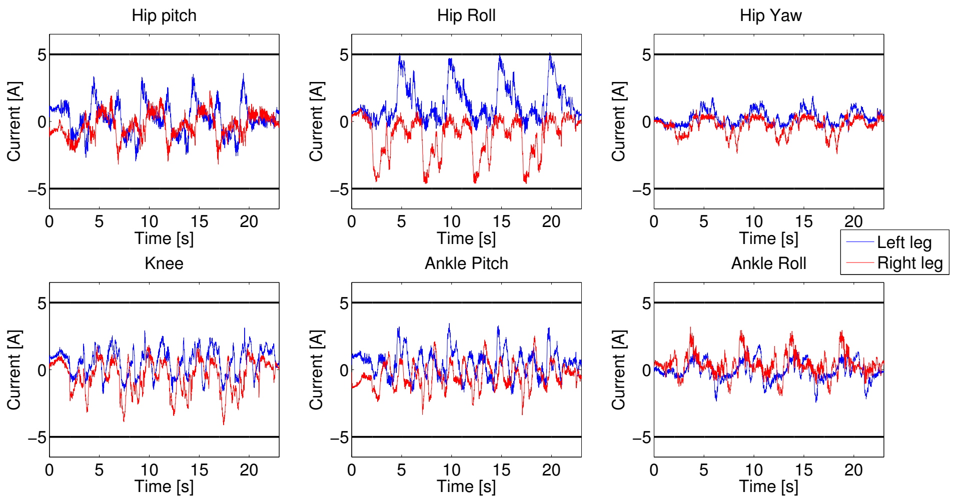
Figure 9.8: Walking on flat ground with T_stride 4 [s] and step_length 0.1 [m]. Center of mass has a variation of 0.015 [m]. Please note that CoM$_d$ and ZMP$_d$ are overlapped on the plane in 9.8a.



(a) Feet pattern with ZMP and CoM

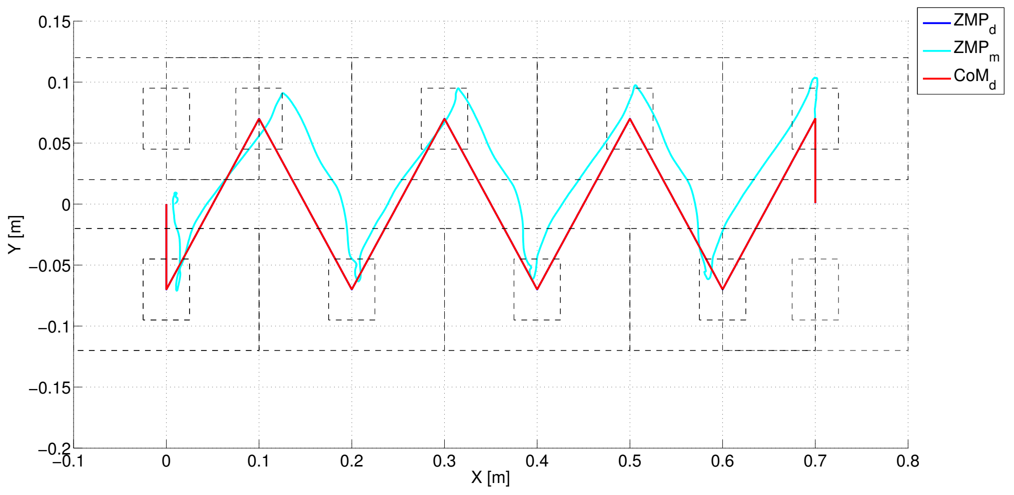


(b) ZMP and CoM trajectory

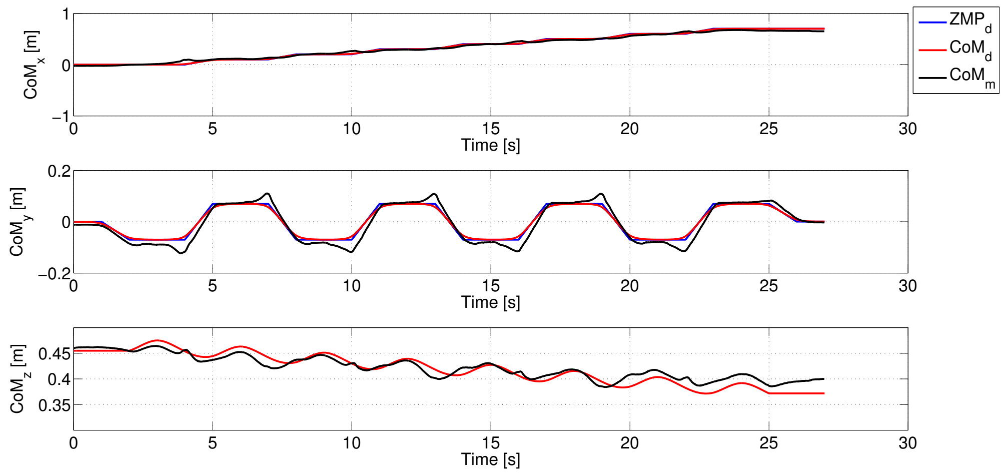


(c) Currents of leg motors, black lines are the default current limits of +/-5 A.

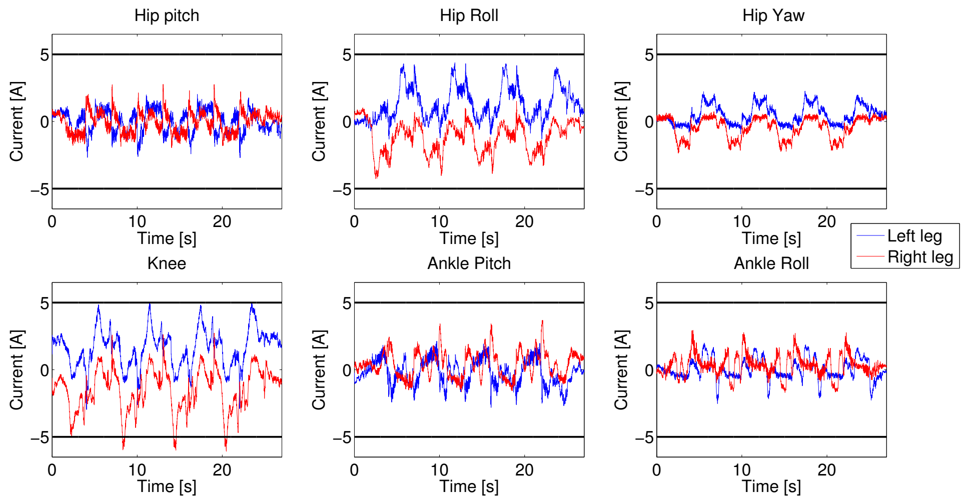
Figure 9.9: Walking up slope with T_stride 5 [s], step_length 0.1 [m] and slope inclination of 7 [deg]. Center of mass has a variation of 0.02 [m]. Please note that CoM$_d$ and ZMP$_d$ are overlapped on the plane in 9.9a.



(a) Feet pattern with ZMP and CoM

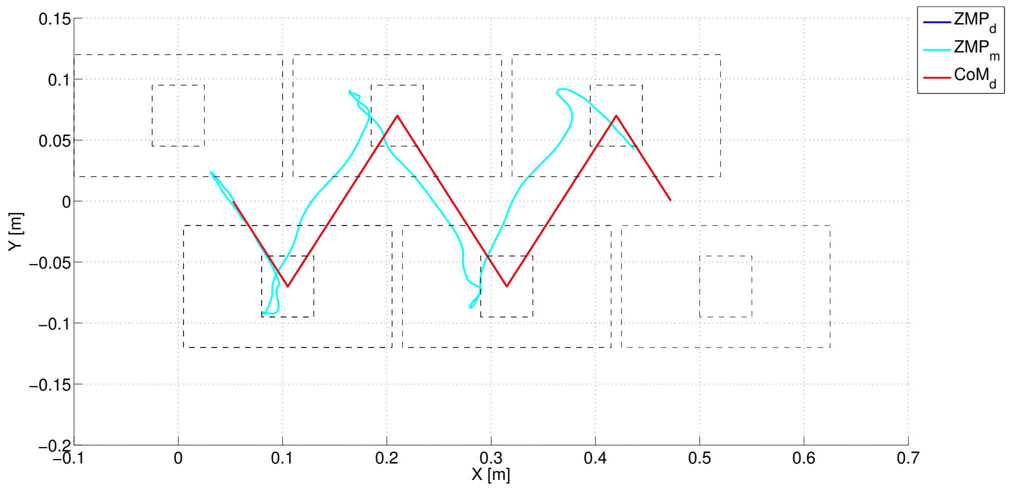


(b) ZMP and CoM trajectory

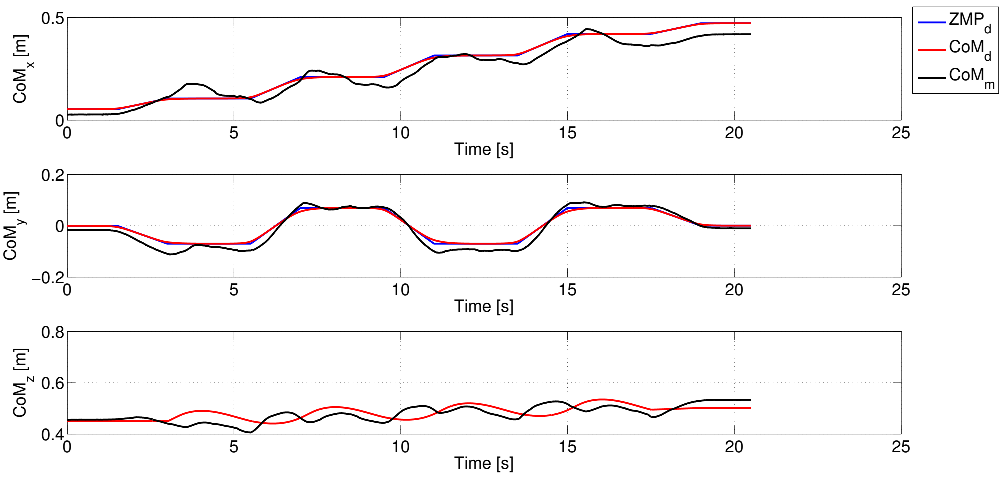


(c) Currents of leg motors, black lines are the default current limits of +/-5 A.

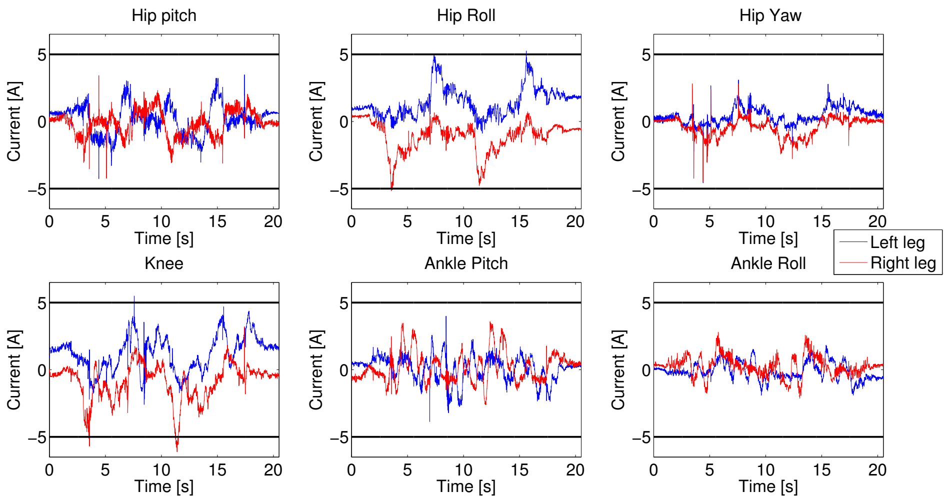
Figure 9.10: Walking down slope with T_stride 6 [s], step_length 0.1 [m] and slope inclination of -7 [deg]. Center of mass has a variation of 0.02 [m]. Please note that $\text{CoM}_d d$ and ZMP_d are overlapped on the plane in 9.10a.



(a) Feet pattern with ZMP and CoM



(b) ZMP and CoM trajectory



(c) Currents of leg motors, black lines are the default current limits of +/-5 A.

Figure 9.11: Walking up stairs with T_stride 8 [s], stair_height 0.02 [m] and stair_length 0.21 [m]. Center of mass has a variation of 0.025 [m]. Please note that CoM$_d$ and ZMP$_d$ are overlapped on the plane in 9.11a.

10 Conclusions

The reduced version of the largely distributed humanoid robot iCub, HeiCub, was presented in this part. HeiCub, despite having only the legs and torso unlike the full iCub, shares the same advantages and disadvantages of any other iCub.
The main abilities of HeiCub derive from collaborations between its birthplace IIT and Heidelberg University where it is located. Little documentation exist about iCub walking, as well as the use of the SEA in any task, therefore we had as objective the analysis the use of SEA in different motions and the implementation of walking motions. As for the human model, also in this case we chose to use optimal control as tool to generate motions for HeiCub.

Before walking, we treated the problem of generating squat motions in order to analyze also motions generated with SEA. In this case the full body model was reduced to a 3 DOF manipulator by fixing the appropriate joints. A series of motions were generated with combinations of a set of objective functions for both the model with rigid actuators and SEA. In both cases, one set of the obtained results was implemented on the actual robot to prove the feasibility of the computed trajectories. The objective was not to highlight the advantages of one with respect to the other, but more to identify suitable models, parameters and constraints for more complex problems. From the numerical results we have understood that the rigid joints should be treated as such rather than as elastic with high stiffness values in optimal control, as this might lead to stiff problems. From the experiments we have understood that the actual stiffness of the joints with SEA is most probably different from the one set in the model, as in the real system many other factors also play important roles.

The problem of push recovery represents the next step of using optimal control to generate whole-body motions. While in literature bipedal robot push recoveries are performed commonly with the capture point method, we formulated the problem as an optimal control problem. In this case all the 15 DOF of the robot were considered, where the left foot of the robot is assumed to be fixed on the floor and the recovery is performed within one step with the right leg.
The push was modeled as external force and taken into account in the model dynamics, it was kept at a constant value and exerted on a predefined fixed point. Optimal control allowed to generate whole-body recovery motions under the robot physical constraints and stablity criteria. From the obtained results we could observe that in average the recovery with two phases need longer time to recover due to the need of stabilizing the motion after impact. The single phase strategy allows for faster computetion, however the two phases strategy is likely a more realistic case, as we expect the system to require a stabilization period after impact, which is more important in the case of SEA, where the impact introduces oscillations.
We could not perform the test of the motion on the real platform, as a proper control of the SEA is needed for this kind of complex whole-body motions. In the case of implementation, the amount of force to be applied and the point of application should be as close as possible to the ones used to compute the recovery motion.

As the iCub robot has shown little walking capabilities in its ten years life, we decided to build a walking control framework based on existing methods to generate walking motions

for HeiCub in order to understand and evaluate the capabilities and performances of the robot in walking instead of formulating an optimal control problem with whole-body models as in the case of squat and push recovery, as the problem of walking is much more challenging and before proceeding to the complex optimal control problem, many unknown details and characteristics of the robot should be learned with simpler approaches.
The chosen method is the table cart combined with the ZMP criterion. However the original method was based on the assumption that the CoM height is constant, reason for which the typical "robot walk" is with bent knees and look very unnatural, furthermore in many robots the CoM is assumed to be located at a certain fixed point instead of using the real position. We took a step further by introducing the variation of the height of the CoM, allowing the robot to walk with stretched knees as well as to perform walking in more challenging situations such as slope and stairs. The CoM is not considered to be fixed but the actual position is used by continuosly updating its position in the computation of the whole body trajectories.
With the developed framework, HeiCub could walk on level ground, up and down slopes and up stairs. From the experiments we understood that the robot has several limitations:

- The limits of the ankle joints are too restricted to perform walking in challenging scenarios, e.g. it is not able to walk down stairs in a human-like way (it is able to if the motion is performed backwards).
- The motors are too weak to sustain the motion of walking up high stairs, as with a step height of 3 [cm] it is already reaching the current limits, for which the robot goes in hardware fault.
- The sole of the robot is too rigid and the surface too slippery which introduce additional problems in the execution of walking motions such as high impacts which can generate high joint tracking errors at the ankle joint which can be propagated to the rest of the joints. This could be solved by using an additional slightly compliant sole.

These considerations can be of use for future improvements in the design of the iCub. Together with the walking performance of the robot measured as KPIs, they can serve a reference for the many iCub users around the world. The developed framework can also be used on any iCub provided the model in URDF.
In particular, the framework was also used in combination with a torque controller as reference generator and with a control framwrok in which the iCub has to place the foot on sloped terrains with optimal ankle impedance [93].

The data acquired during the experiments with the framework shown in this thesis, together with many others found with different methods such as the online pattern generator and the torque control are fundamental for future setups of the motion generation using whole-body models in optimal control problem formulations.

Discussion

The core topic of this thesis is compliance during locomotion, both in humans and humanoid robots. Compliance alone might refer to many aspects of both, and stating that we have fully understood the role of compliance in both “systems” is very pretentious. But with this thesis we did analyze and obtain results for interesting and non-trivial aspects that were still unknown and/or unclear.
The approach we followed share a common methodology, treating both the human body and the humanoid robot as rigid multi-body systems and applying optimal control theories. In the former to understand compliance at joint and bi-articular level in human walking and in the latter to analyze motions generated for the HeiCub with and without compliant actuators.
In both cases a computer model has been created and used to achieve our goals.

While there exist extremely complex models of the human body that need super computers to be effectively used as computational models, we focused on a single characteristics which is the compliance at joint and bi-articular level and conducted the study by means of a human model in 2D which could grant useful and interesting results despite its simplicity, using a standard desktop computer in reasonable computation time.
We could compute **stiffness profiles** for the leg joints and the bi-articular coupling between the hip and knee with the variable stiffness springs inserted in the model which acted as actuation of the leg joints. The choice of varying stiffness was lead by the similarity to human biomechanics. The profiles obtained from 4 different subjects showed that **stiffness modulate in a high range** in all the leg joints during walking in level ground, up a slope and climbing up stairs. The range can reach up to **1500 [Nm/rad]** in the joints and 150 [Nm/rad] in the bi-articular couplings. Notably, stiffness values are higher in the stance leg during single support phase and fast increase of the stiffness values occur after impacts.
The results are contrasted to the common assumption of constant stiffness in the joints of existing walking machines, reason for which we went further with the study in analysing the **influence of these modulations on the walking gait**. By minimizing the stiffness modulation and jumps after phase change, we observed that in unconstrained environments such as level ground and slope, the original gait could be still reproduced but with increasing fitting errors. In the case of stair walking instead the error was big enough to come to the conclusion that the original gait could not be anymore reproduced when stiffness has very small modulations and stiffness jumps are not allowed.
Given that many mechanisms such as exosletons, prosthesis and orthosis are built to help humans walk as healthy subjects do and in environments that are the same as the healthy subjects, we came to the conclusion that **the modulation of stiffness is an important factor** that should be taken into consideration in the design of these mechanisms and the range should be at least as large as the ones obtained in this thesis. Where continuous stiffness modulation is not achievable, the switching of stiffness levels at phase change also contributes to recreate human-like gaits.

Despite our results on stiffness modulation in humans, the humanoid robot HeiCub has compliant actuators with physical springs that have constant stiffness, as other state of the art humanoid robots with compliant actuation, given that variable stiffness actuators are still too

big in size to be used in human size (or smaller) robots. HeiCub has been designed and built for the KoroiBot project by IIT, therefore the studies conducted on HeiCub were done in parallel to the analysis of compliance in human walking.
The iCub robot is one of the most spreaded advanced humanoid robots in the world, but it has almost never been used for locomotion purposes prior this thesis. One of the reasons is that the first versions of the robot had weak motors in the legs and small feet to allow for walking or even standing. In the latest version however, the leg design has been improved and SEA were also introduced in the knee and ankle pitch joints.
The objective of KoroiBot was to teach existing humanoid robots walking. As the iCub had little walking abilties and the use of SEA was also not largely exploited, we started with the simpler motion of **squatting**.
The problem was formulated as a motion generation problem with optimal control, where the approach is similar to the one used for human models but with different dynamic models and objectives. In this case the model of the robot was simplified to the 3 pitch joints of the leg, hip, knee and ankle, by fixing the rest of the joints of the robot that are unlikely to be involved in the squatting motion. The motion were generated for **the model with and without SEA** and in both cases different trajectories were obtained with different combinations of objective functions. These results were implemented on the robot and showed some model inaccuracies that should be taken into consideration in future works.
With the same approach we analyzed the problem of **push recovery** within one step. We did not use the capture point criterion as in literature, but let the optimization to find the best recovery trajectories by taking into account stability and physical constraints of the robot. In this case all the actuated DOF of the robot were used and the left foot was assumed to be fixed, while the right foot is left free to perform the recovery step. Also here results were obtained for both the models with and without SEA. Two startegies were used, where the first assumes recovery within one phase, while the second one undergoes a stabilisation phase after impact. In the case of SEA, the stablisation phase is more important as SEA introduces oscillations.
Interesting results were obtained using optimal control with whole-body dynamic models of the robot, however such problems are still at a computationally expensive level, reason for which, to **exploit the walking capabilities** of the HeiCub, we used **simplified models** and control methods. With the modified table cart model combined with the ZMP criterion we implemented a walking control framework for the HeiCub which is suitable also for any other iCub. The framework generates reference trajectories offline and then the joint trajectories are executed on the robot in an openloop fashion. The simplicity of the method allowed us to test **walking on different environments**, which have never been achieved on iCub robots before. The robot has proved to be able to walk in level ground with different speeds, up and down slopes of different inclinations and up stairs of different heights. During these experiments many limitations of the robot have been discovered which need to be taken into account in future designs. The data collected during the experiments serve also to the iCub community as reference for future improvements and comparisons of walking control frameworks.

Future developments

Studies on human locomotion

The human model in 2D could be used to obtain results on waking gait that have most of the characterizing motions on the sagittal plane. It can be used to achieve the same kind of study on extensive sets of data such as different slope inclinations, different stair sizes, different class of subjects, e.g. women, children, elderly. However, improvements in the model could help to achieve better results, such as a rolling foot contact instead of the two points flat contact would increase the accuracy of the ankle trajectory fitting. Subject specific segment length would also increase the precision of the obtained results, as at the moment the lengths are obtained from a scalable model based on the total height.
For a broader class of motions it is necessary to increase the complexity of the model, i.e. using a 3D model. As motions happening in other planes such as the frontal planes, play non-negligible roles. Example motions of this type are walking on step stones, walking on a tight beam, where the subjects oscillate left and right. In these cases a study on the sagittal plane only would lead to very similar results to straight walking, omitting the importance of the motions happening on the other moton planes.
In the case of 3D models, a revision of the spring model of the joints as well as the bi-articular couplings is also necessary, as in this case more couplings between joints need to be taken into consideration in the actuation.
The same problem setup can still be used, where an extension of the implementation of the model as in Appendix A is necessary as well as additional constraints to be taken into account mainly regarding external contacts, but the overall structure can be maintained.

Studies on robot locomotion

Based on the whole-body motions generated using optimal control taking into account whole-body dynamics and implemented on the HeiCub, we could learn more details of the robot that are unknown to the model. It is known that real robots are not perfect as they are affected by issues such as friction, unknown or imprecise inertia parameters. Many modeling details can be added to compensate these issues, as well as model identification can be performed and the obtained parameters can be added to the model to obtain a more precise one. As in optimal control model precision in fundamental, in the generation of more challenging motions such as walking, additional model details need to be taken into account to obtain motions that are feasible and stable for the robot.
The push recovery motions are hard to be used on the real robot with single computations. With a set of trajectories obtained for different push forces and application points, the results could be used in an online fashion by combining them as motion primitives to be selected. In this case an online module which detects the push force location and intensity need to be implemented, where the motion execution can be done with a modification of the modules used to execute motions on the robot of this thesis.
The walking control framework developed in this thesis is simple but could serve already for many purposes that would have required more additional work to the iCub users who need walking motions. For example as reference generator to test walking with the torque controller or to perform floating base estimation using vision (i.e. the cameras of the robot) [10].
But an offline generator and openloop trajectories execution is not the ideal for external perturbances and unexpected situations. Therefore online feedback is necessary, which was im-

plemented in [106], also based on reduced models.
The use of reduced models have many advantages but rely on simple dynamics which need to be corrected to match the one of the whole-body model and conservative stability criteria such as ZMP.
Whole-body models are computationally expensive but allow to exploit the whole-body dynamics of the system. As we did for squatting and push-recovery, the same model as implemented in Appendix A can be used to formulate an optimal control problem for walking, by taking into account also the SEA.
To implement motions on the actual robot, however, a proper control of the SEA also need to be implemented. In this case a torque controller is most probably the desirable one.
From the walking experiments on HeiCub we observed also the limitations of the robot, which mainly lie in the ankle joint limits and motor power. A possible way to cope with the current situation without mechanically modifying the robot is to build soles of different shapes for the robot. For instance, in the case of walking down stairs or up slope, the issue lies in the limited angle of the foot while bending towards the lower leg, while the joint does not need to bend in the other direction. In this case it is possible to build a slope shaped sole (a wedge) for the robot, such that the zero position of the ankle pitch is shifted, increasing the motion range. This might introduce further problems, as the height of the robot increases and the range of the ankle pitch is reduced in the opposite direction. Furthermore this method cannot be used in walking down a slope, where the opposite problem exists, and further investigations are necessary to verify if with such external sole the robot is still capable of level ground walking. Some custom made adjustments such as the one proposed could be achieved in an easy way without modifying the internal structure of the robot, but the weakness of the motors represent the more challenging issue of iCub. More powerful motors are necessary to perform walking motions in more complex environments. The use of the SEA, which were not adopted in these experiments, could help to cope with this limitation to some extent and it is worth to be investigated in further works.

A Models implementation details

To perform computations for dynamics systems it is necessary to have a proper computer model. In this appendix we illustrate how the model of the human body as described in Section 3.2 and the robot model as described in Chapter 6 are implemented to achieve the different goals of this thesis such as inverse kinematics, optimal control etc.
In both cases we have a description of the kinematics and dynamics of the rigid body model stored in an external file, then a proper C++ wrapper based on the Rigid Body Dynamics Library (RBDL) [33] is implemented to allow better access to the model details and functions to perform useful computations.
In the following only the header files are reported.

A.1 Human model

The C++ wrapper implemented for the human model has the aim of creating an easy way to interface with the optimal control software package MUSCOD-II [68] in order to retrieve useful information. The implementation is a modified version of the one used in [32] to better cope with the problems studied in this thesis.
The model kinematic and dynamic information are store in an external .lua file which contains also the contact points and constraint sets information. A constraint set is a set of contact points that are in contact, therefore when a certain contact set is active, the dynamics is computed taking into account the external forces acting on the points included in the constraint set.
The model is defined as a struct as follows:

```
struct WalkerModel{
  WalkerModel();

  unsigned int activeConstraintSet; // the current active set of constraints.
  unsigned int nDof; // number of DOF.
  unsigned int nActuatedDof; // number of active DOF.

  // Flags.
  bool dynamicsComputed;
  bool kinematicsUpdated;

  // Generalised coordinates, velocities, accelerations and torques.
  RigidBodyDynamics::Math::VectorNd q;
  RigidBodyDynamics::Math::VectorNd qdot;
  RigidBodyDynamics::Math::VectorNd qddot;
  RigidBodyDynamics::Math::VectorNd tau;

  // Parameters.
  RigidBodyDynamics::Math::VectorNd p;

  // The model as for RBDL convention.
  RigidBodyDynamics::Model model;

  //  Information on contact points.
  Point pointInfos[PointNameLast];
  // Information of the constraint sets.
  ConstraintSetInfo constraintSetInfos[ConstraintSetNameLast];

  // RDBL constraint sets that are used during forward dynamics and collision computations.
  std::vector<RigidBodyDynamics::ConstraintSet> constraintSets;

  // Copies state information from MUSCOD to the model and switches to the given constraint set.
  void updateState (const double  sd, const double  u, const double  p, const ConstraintSetName cs_name);

  // Updates the kinematics of the RDBL model.
  void updateKinematics();

  // Computes the forward dynamics for the model and the current active constraint set.
  void calcForwardDynamicsRhs (double  res, double  u);
  // Computes the change in velocities due to a collision with the current active constraint set.
  void calcCollisionImpactRhs (double  res, double  u, double  xd);

  // Returns the 3-D position, velocity, acceleration or force of a point
  RigidBodyDynamics::Math::Vector3d getPointPosition(const PointName &point_name);
  RigidBodyDynamics::Math::Vector3d getPointVelocity(const PointName &point_name);
  RigidBodyDynamics::Math::Vector3d getPointAcceleration(const PointName &point_name);
  RigidBodyDynamics::Math::Vector3d getPointForce(const PointName &point_name);

  // Load the model from external Lua file.
  bool loadFromFile(const char  filename, bool verbose=false);
  // Load the file containing contact points (could be the same as the model file).
  bool loadPoints(const char  filename, bool verbose=false);
  // Load the constraint sets from external file (could be the same as the model file).
  bool loadConstraintSets(const char  filename, bool verbose=false);
};

// Struct of a contact point, containing all necessary information.
struct Point{
  Point():
    name ("unknown"),
    body_id (-1),
    body_name (""),
    point_local (std::numeric_limits<double>::signaling_NaN(),
     std::numeric_limits<double>::signaling_NaN(),
     std::numeric_limits<double>::signaling_NaN()){}
  std::string name;
  unsigned int body_id;
  std::string body_name;
  RigidBodyDynamics::Math::Vector3d point_local;
};

// Struct of a single constraint information.
struct ConstraintInfo{
  ConstraintInfo() :
    point_id (PointNameLast),
    point_name (""),
    normal (std::numeric_limits<double>::signaling_NaN(),
      std::numeric_limits<double>::signaling_NaN(),
      std::numeric_limits<double>::signaling_NaN()){}
  unsigned int point_id;
  std::string point_name;
  RigidBodyDynamics::Math::Vector3d normal;
};

// Struct to store the information of constraints part of the same ConstraintSet.
struct ConstraintSetInfo {
  ConstraintSetInfo() :
    name ("undefined") {
  }
  std::vector<ConstraintInfo> constraints;
  std::string name;
};
```

In this struct, the right hand side of the optimal control problem is computed with the functions *calcForwardDynamicsRhs* for continuous phases and *calcCollisionImpactRhs* for impacts, the first computes forward dynamics taking into account the contacts according to the current constraint set and the latter computes the impact dynamics as per model assumption in chapter 2.1.

The states, controls, parameters, constraint sets, contact points as well as optimal control phases are defined with enumerates for easy access. They are different according to the problem treated.
Enumerates defined for the model walking on level ground and slope:

```
enum StageName {
  StageRightFlat = 0,
  StageRightHalx,
  StageRightHalxTouchdownLeftHeel,
  StageRightHalxLeftHeel,
  StageRightHalxLeftHeelTouchdownLeftHalx,
  StageRightHalxLeftFlat,
  StageNameLast
};

enum PointName {
  PointRightHeel,
  PointRightHalx,
  PointLeftHeel,
  PointLeftHalx,
  PointNameLast
};

enum ConstraintSetName {
  CSRightFlat = 0,
  CSRightHalx,
  CSRightHalxLeftHeel,
  CSRightHalxLeftFlat,
  ConstraintSetNameLast
};

enum ParamName {
  ParamRestLengthHip = 0,
  ParamRestLengthKnee,
  ParamRestLengthAnkle,
  ParamRestLengthHipKnee,
  ParamDamperAnkle,
  ParamNameLast
};
```

In the case of stair walking, the set of phases and constraints are defined as:

```
enum StageName {
  StageRightFlatLeftHalx = 0,
  StageRightFlat,
  StageRightFlatLeftFlatTD,
  StageRightHalxLeftFlat,
  StageNameLast
};

enum ConstraintSetName {
  CSRightFlatLeftHalx = 0,
  CSRightFlat,
  CSRightHalxLeftFlat,
  ConstraintSetNameLast
};
```

In the case of study of influence of compliance on human walking, the parameters set for level ground and slope becomes:

```
enum ParamName {
  ParamRestLengthHip = 0,
  ParamRestLengthKnee,
  ParamRestLengthAnkle,
  ParamRestLengthHipKnee,
  ParamDamperAnkle,
  ParamSlackRHipP1,
  ParamSlackRKneeP1,
  ParamSlackRAnkleP1,
  ParamSlackLHipP1,
  ParamSlackLKneeP1,
  ParamSlackLAnkleP1,
  ParamSlackRHipKneeP1,
  ParamSlackLHipKneeP1,
  ParamSlackRHipP2,
  ParamSlackRKneeP2,
  ParamSlackRAnkleP2,
  ParamSlackLHipP2,
  ParamSlackLKneeP2,
  ParamSlackLAnkleP2,
  ParamSlackRHipKneeP2,
  ParamSlackLHipKneeP2,
  ParamNameLast
};
```

And for stairs:

```
enum ParamName {
  ParamRestLengthHip = 0,
  ParamRestLengthKnee,
  ParamRestLengthAnkle,
  ParamRestLengthHipKnee,
  ParamDamperAnkle,
  ParamSlackRHipP1,
  ParamSlackRKneeP1,
  ParamSlackRAnkleP1,
  ParamSlackLHipP1,
  ParamSlackLKneeP1,
  ParamSlackLAnkleP1,
  ParamSlackRHipKneeP1,
  ParamSlackLHipKneeP1,
  ParamNameLast
};
```

States and controls were not listed here to avoid very long listings, they are defined exactly as they have been defined in the optimal control problem in sections 4.2 and 4.3.

A.2 HeiCub model

The HeiCub model has been used for multiple applications, including the inverse kinematics and optimal control problems.

Model builder

Given the necessity of loading the model from the URDF in a user-defined way, a class to load the model with multiple user specific needs was created as *Model_builder*. This class takes as input an external .lua file with a set of parameters according to which the model is built using RBDL functions. The class is also thought to be used with MUSCOD-II, therefore there are some MUSCO-II specific settings.

```
class Model_builder
{
  public:
    // The constructor reads in the configuration file
    Model_builder(std::string model_file);
    ~Model_builder();

    ////////// Configuration functions /////////////////////
    // Set the list of joints that should be active, if not aspecified, all joints as in the URDF file are active.
    void setActiveJoints(std::vector<std::string> user_joints);
    // Gives the possibility of fixing the specified joints at the specified angles.
    void setFixedAngles(std::vector<std::string> joints, std::vector<double> angles);
    // Set the gravity vector if the default (0,0,-9.81) is not desired.
    void setGravity(RigidBodyDynamics::Math::Vector3d g);
    // Set the floating base as planar, i.e. a 3D vector, otherwise by default it is a 6D vector.
    void setPlanarBase();
    // This function actually builds the model into a RBDL convention model.
    void build_model(RigidBodyDynamics::Model  built_model, bool verbose);

  private:
    // Reads in the URDF file
    bool ReadFromURDF(const char  filename, RigidBodyDynamics::Model  rbdl_model, bool verbose = false);
    // Build the model by parsing the URDF file and the takes into account the user specified parameters.
    bool construct_model(RigidBodyDynamics::Model  rbdl_model, ModelPtr urdf_model, bool verbose);

    std::string filename;
    std::vector<std::string> active_joints;
    std::vector<std::string> joints_fixed_angle;
    std::vector<double> fixed_angles;
    bool active_set;
    bool fixed_set;
    bool gravity_set;
    bool set_planar_base;
    RigidBodyDynamics::Math::Vector3d gravity_custom;
};
```

The configuration file is a .lua file which is parsed by the code:

```
iCubHeidelberg01 =
{
  urdf_file = "mymodel.urdf",
  contact_points_file = "my_contact_points.lua",
  constraint_set_file = "my_constraint_sets.lua",
  active_joints = {"j1","j2",..},
  fixed_joint = {"j3","j4",..},
  fixed_angles = {ang3,ang4,..},
  floating_base = true, --default false
  planar_base = true/false, -- default false
  gravity = {v1,v2,v3}, -- default 0,0,-9.81
  verbose = true/false, -- default false
--  for MUSCOD problems only --
  states = {"s1","s2",...},
  controls = {"c1","c2",...},
  parameters = {"p1","p2",...},
}
```

Even if the class *Model_builder* was initially implemented to be used with the HeiCub model, it is actually a generic class that can be used with any model which description is contained in an URDF. It was used also with the full model of the iCub to implement walking motions.

SEA dynamics

To take into account the elasticity in the dynamic model, additional functions that implement the theory as in Section 6.3.1 are used in the model wrapper illustrated in the next section. This implementation follows the same style of RBDL and allows to create an RBDL model with additional set of parameters regarding SEA as per our assumptions.

```
namespace ElasticJointsDynamics{

  // The model of a robot with elastic joints
  struct ModelElasticDynamics{
    ModelElasticDynamics():nDof(-1),nActuatedDof(-1),nPassiveDof(-1),
  partial_elastic(false){}

    //// Parameters to set
    // Total number of dof
    int nDof;
    // Number of actuated DOF
    int nActuatedDof;
    // Number of passive dofs, default is 0
    int nPassiveDof;
    // The stiffness matrix is treated as vector given it is assumed to be
  diagonal
    RigidBodyDynamics::Math::VectorNd stiffnessMatrix;
    // The motor inertia matrix is treated as vector given it is a diagonal
  matrix
    RigidBodyDynamics::Math::VectorNd motorInertia;

    // If set to true, only the joints specified in the list elasticJoints will
  be used with the elastic joint dynamics
    bool partial_elastic;

    // The list of indeces of joins with elasticity
    std::vector<int> elasticJoints;

    //// Functions
    // Set the parameters of the model
    void setElasticJointParameters(
  RigidBodyDynamics::Model &model, // standard RBDL model, "the rigid model"
  const int passiveDof, // number of non actuated DOF
  const RigidBodyDynamics::Math::VectorNd &K, // the stiffness matrix
  const RigidBodyDynamics::Math::VectorNd &B, // the motor inertia matrix
  std::vector<int>& eJ // the list of indeces of joins with elasticity
  );

    // Set stiffness of a certain joint
    void setStiffness(const int idx, const int k);
  };

  // Computes forward dynamics without external contacts
  void ForwardDynamicsElasticJoints (
    RigidBodyDynamics::Model &model,
    ModelElasticDynamics &model_elastic,
    const RigidBodyDynamics::Math::VectorNd &Q,
    const RigidBodyDynamics::Math::VectorNd &QDot,
    const RigidBodyDynamics::Math::VectorNd &Theta,
    const RigidBodyDynamics::Math::VectorNd &ThetaDot,
    const RigidBodyDynamics::Math::VectorNd &Tau,
    RigidBodyDynamics::Math::VectorNd &QDDot,
    RigidBodyDynamics::Math::VectorNd &ThetaDDot,
    std::vector<RigidBodyDynamics::Math::SpatialVector>  f_ext = NULL
    );

  // Computes inverse dynamics
  void InverseDynamicsElasticJoints (
    RigidBodyDynamics::Model &model,
    ModelElasticDynamics &model_elastic,
    const RigidBodyDynamics::Math::VectorNd &Q,
    const RigidBodyDynamics::Math::VectorNd &QDot,
    const RigidBodyDynamics::Math::VectorNd &QDDot,
    const RigidBodyDynamics::Math::VectorNd &Theta,
    const RigidBodyDynamics::Math::VectorNd &ThetaDot,
    const RigidBodyDynamics::Math::VectorNd &ThetaDDot,
    RigidBodyDynamics::Math::VectorNd &Tau,
    std::vector<RigidBodyDynamics::Math::SpatialVector>  f_ext = NULL
    );

  // Computes forward dynamics with external contacts
  void ForwardDynamicsContactsElasticJoints (
    RigidBodyDynamics::Model &model,
    ModelElasticDynamics &model_elastic,
    const RigidBodyDynamics::Math::VectorNd &Q,
    const RigidBodyDynamics::Math::VectorNd &QDot,
    const RigidBodyDynamics::Math::VectorNd &Theta,
    const RigidBodyDynamics::Math::VectorNd &ThetaDot,
    const RigidBodyDynamics::Math::VectorNd &Tau,
    RigidBodyDynamics::ConstraintSet &CS,
    RigidBodyDynamics::Math::VectorNd &QDDot,
    RigidBodyDynamics::Math::VectorNd &ThetaDDot
    );
}
```

HeiCub model wrapper

For the optimal control problems defined in MUSCOD-II a model wrapper was created as for the human model. This wrapper is slightly different than the human one and consists in a class rather than struct, and instead of enumerates to define the states, controls parameters etc, it

used maps, which can be defined in an easier and more modular way. This model was thought to be used not only for MUSCOD-II problems but also for general computation problems with the HeiCub model, therefore some MUSCOD-II specific functions and parameters are activated only if *MUSCOD_PROBLEM* is defined.
It shares the same *ConstraintInfo, ConstraintSetInfo* and *PointInfo* structs as in the human model wrapper and uses the *Model_builder* class to create the RBDL model. The class can be used with either the model with rigid actuators or SEA. When SEA are used, it is necessary to activate *MODEL_ELASTIC*.

```
class iCubHeidelberg01_model {
public:
  iCubHeidelberg01_model();
  iCubHeidelberg01_model(std::string conf_file);
  ~iCubHeidelberg01_model();

  RigidBodyDynamics::Model model;
#ifdef MODEL_ELASTIC
  ElasticJointsDynamics::ModelElasticDynamics model_elastic;
#endif

#ifdef MUSCOD_PROBLEM
  // parameters
  RigidBodyDynamics::Math::VectorNd p;
#endif

  // Joint variables
  RigidBodyDynamics::Math::VectorNd q;
  RigidBodyDynamics::Math::VectorNd qDot;
  RigidBodyDynamics::Math::VectorNd qDDot;
  RigidBodyDynamics::Math::VectorNd tau;

#ifdef MODEL_ELASTIC
  // Motor variables
  RigidBodyDynamics::Math::VectorNd theta;
  RigidBodyDynamics::Math::VectorNd thetaDot;
  RigidBodyDynamics::Math::VectorNd thetaDDot;
#endif

  std::map<std::string,int> constraint_sets_map;
  std::map<std::string,int> points_map;
  std::map<std::string,int> joints_map;

#ifdef MUSCOD_PROBLEM
  std::map<std::string,int> states_map;
  std::map<std::string,int> controls_map;
  std::map<std::string,int> params_map;
#endif

/////////// Kinetics utility functions ///////////////////

  /// Computes the CoM according to the actual state of the robot
  RigidBodyDynamics::Math::Vector3d getRobotCoM ();
  RigidBodyDynamics::Math::Vector3d getRobotCoMLocal (const int body_id);
  /// Compute ZMP using simplified model, i.e. 3D inverted pendulum
  RigidBodyDynamics::Math::Vector3d getSimplifiedZMP();

  /// Returns 3D vectors containing position, velocity or force of a point
    attached to a body, the point has to be one specified in the points list
  RigidBodyDynamics::Math::Vector3d getPointPosition (std::string point_name);
  RigidBodyDynamics::Math::Vector3d getPointVelocity (std::string point_name);
  RigidBodyDynamics::Math::Vector3d getPointForce (std::string point_name);
  RigidBodyDynamics::Math::Vector3d calcForceVector (unsigned int  body_id, const
    RigidBodyDynamics::Math::Vector3d &force);

  void performInverseKinematics(RigidBodyDynamics::Math::VectorNd &qres, const std
    ::vector<unsigned int> & body_id, const std::vector<RigidBodyDynamics::Math::
    Vector3d>& body_point, const std::vector<RigidBodyDynamics::Math::Vector3d >&
    target_pos);

//////////// Update functions ///////////////////////////////
#ifdef MODEL_ELASTIC
  void updatePos(const double  qi, const double  thetai);
  void updateVel(const double  qdoti,const double  thetadoti);
  void updateAcc(const double  qddoti, const double  thetaddoti);
  void updateRobot(const double  qi, const double  thetai, const double  qdoti,
    const double  thetadoti, const double  qddoti, const double  thetaddoti, const
     double  taui, std::string dyn);
#else
  void updatePos(const double  qi);
  void updateVel(const double  qdoti);
  void updateAcc(const double  qddoti);
  void updateRobot(const double  qi, const double  qdoti, const double  qddoti,
    const double  taui, std::string dyn);
#endif
  void updateTorque(const double  taui);
  void updateConstraint(std::string cs_name);
  void updateRobot(std::string dyn);

//////////// MUSCOD utility functions /////////////////////
#ifdef MUSCOD_PROBLEM
  /// Copies state information from MUSCOD to the model and switches to the given
    constraint set.
  void updateState (const double  sd, const double  u, const double  p_in, std::
    string cs_name);
  /// Computes the forward dynamics for the model and active constraint set
  void calcForwardDynamicsRhs (double  res);
  /// Computes forward dynamics of the model affected by push force
  void calcForwardDynamicsRhsPush (double  res, std::string point_name, const
    RigidBodyDynamics::Math::Vector3d &push_force);
  /// Computes the change in velocities due to a collision
  void calcCollisionImpactRhs (double  res);
#endif

/////////////////////// Points, constraints and collision
    /////////////////////////////
  /// List of points of the model
  std::vector<PointStruct> pointInfos;
  /// RDBL constraint sets that are used during forward dynamics and collision
    computations
  std::vector<RigidBodyDynamics::ConstraintSet> constraintSets;
  /// Information of the constraint sets (mostly used when parsing Lua file)
  std::vector<ConstraintSetInfo> constraintSetInfos;
  /// collision avoidance using capsules
  std::vector<Capsule> capsules;
  void initializeCapsules();
  /// Computed the distance between two capsules, if positive the two capsules are
     not touching each other
  double calcCapsulesDistance (int c1, int c2);

//////////          Private members          //////////////
private:
  void init_conf(std::string conf_file);
  void init(bool verbose = false, bool write_out = false);

  // files names
  std::string urdf_file;
  std::string constraint_set_file;
  std::string contact_points_file;
  std::string lua_file;
  std::string elastic_params_file;

  // Reduced model data
  std::vector<std::string> active_joints; // if this list is set, then ALL the
    joints that are NOT in the active joints will be set to FIXED
  std::vector<std::string> joints_fixed_angle;
  std::vector<double> fixed_angles;
  std::vector<std::string> half_bodies; // hack
  bool planarBase;
  int floating_base_dof;
#ifdef MODEL_ELASTIC
  bool partial_elastic;
  std::vector<int> ej;
#endif

  std::map<std::string,int> joints_map_full;
#ifdef MUSCOD_PROBLEM
  std::map<std::string,int> states_map_full;
  std::map<std::string,int> controls_map_full;
#endif

  int DOF_NAME_LAST;
  int JNT_NAME_LAST;

  unsigned int activeConstraintSet;

  bool forwardDynamicsComputed;
  bool inverseDynamicsComputed;
  bool kinematicsUpdated;

  /// Updates the kinematics of the RDBL model
  void updateKinematics();
  /// Update the dynamics by computing forward dynamics, this should be called
    after updateState in order to compute the correct dynamics
  void updateForwardDynamics();
  void updateInverseDynamics();

////////////// Helper functions to load everything from file
  bool loadModelFromFile (std::string filename, bool verbose=false, bool write_out
    =false);
#ifdef MODEL_ELASTIC
  bool loadElasticParametersFromFile(std::string filename, bool verbose=false);
#endif
  bool loadPointsFromFile (std::string filename, bool verbose=false);
  bool loadConstraintSetsFromFile (std::string filename, bool verbose=false);
};
```

In this code, the functions *calcForwardDynamicsRhs* and *calcCollisionImpactRhs* have the exact same role as in the human models, i.e. compute the right hand side for continuous phases and impacts. The difference is that in the implementation it is necessary to take into account the elasticity when $MODEL\_ELASTIC$ is active, in this case the dynamics computation are done with the class of the previous section.

The function *calcForwardDynamicsRhsPush* is used for the push recovery problem, in which the external push is taken into account as in section 8.1.
The collisions are handled for the leg links, which are identified by enumerates:

```
enum Capsules{
  CapsuleRightFoot = 0,
  CapsuleRightLowerLeg ,
  CapsuleRightUpperLeg ,
  CapsuleLeftFoot ,
  CapsuleLeftLowerLeg ,
  CapsuleLeftUpperLeg ,
  CapsuleTot
};
```

B Joint angle trajectories and stiffness profiles

In this appendix we put all the plots of leg joint angles and stiffness profiles of all the subjects (as in Tab. 4.2) and objective functions combinations as described in section 4.3. It is possible to observe the differences in the different subjects, where stiffness can assume different values for different subjects with the same combination of weights, this is also due to the difference in joint angle trajectories.

We remind that the objective function consists in the following combination:

$$\Phi_L = c_{fit}\Phi_{fit} + c_{der}\Phi_{der} + c_{\Delta K}\Phi_{\Delta K} \tag{B.1}$$

where Φ_{fit} is the minimization of the fitting error, Φ_{der} is the minimization of stiffness derivatives and $\Phi_{\Delta K}$ is the minimization of the slack variables representing the stiffness jumps when impacts occur.

The same table as Tab. 4.3 is reported here in Tab. B.1 to allow easier consultation.

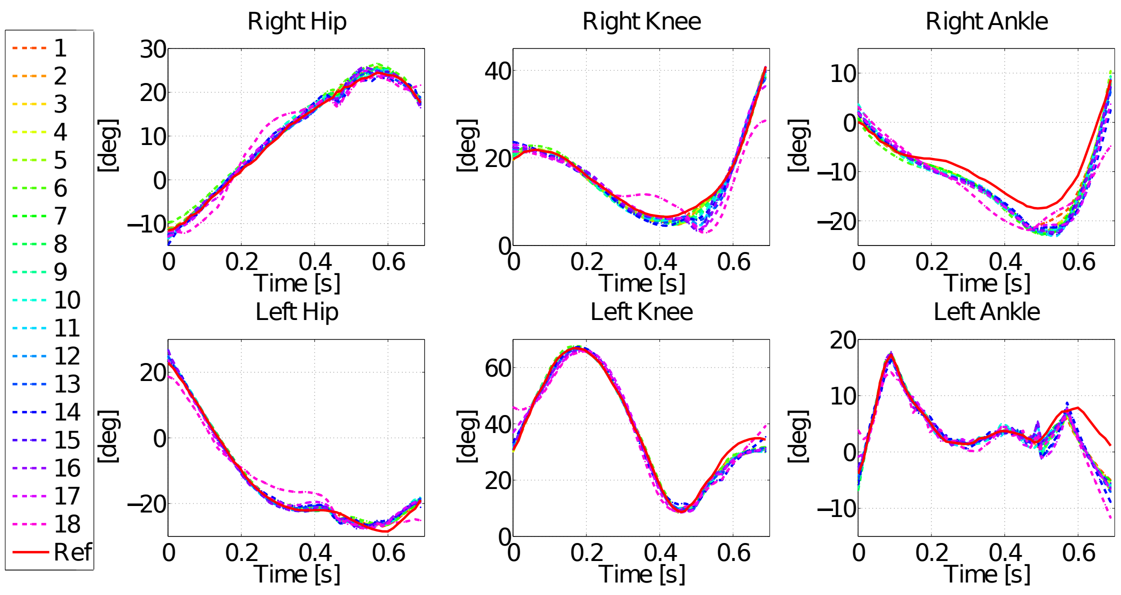
Ref. number	c_{fit}	c_{der}	$c_{\Delta K}$
1	10	0	0
2	10	0	0
3	10	10^{-8}	0
4	10	10^{-8}	10^{-8}
5	10	10^{-8}	10^{-4}
6	10	10^{-8}	1
7	10	10^{-6}	0
8	10	10^{-6}	10^{-8}
9	10	10^{-6}	10^{-4}
10	10	10^{-6}	1
11	10	10^{-4}	0
12	10	10^{-4}	10^{-8}
13	10	10^{-4}	10^{-4}
14	10	10^{-4}	1
15	10	10^{-2}	0
16	10	10^{-2}	10^{-8}
17	10	10^{-2}	10^{-4}
18	10	10^{-2}	1

Table B.1: Weights combinations used to obtain the results. The first two rows are the same, but differ in the limits used for the controls. In Ref. n. 1 the limits are left open, while in Ref. n. 2 is contrained in the range [-1000, 1000] [Nm/rad· s]. Please note that Φ_{fit} is of 10^3 order smaller than Φ_{der} and Φ_{slack}, reason for which the weights are not of the same order.

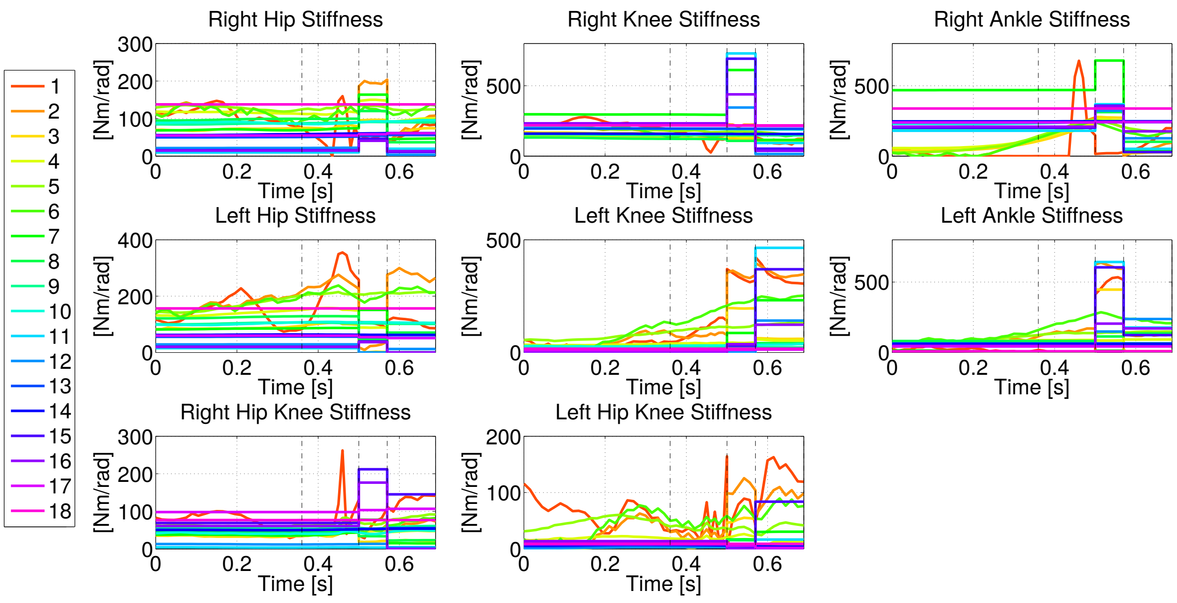
We remind that the scaling in all plots is different due to the difference in values not only among the subjects but also among the joints.

B.1 Level ground

Subject M1

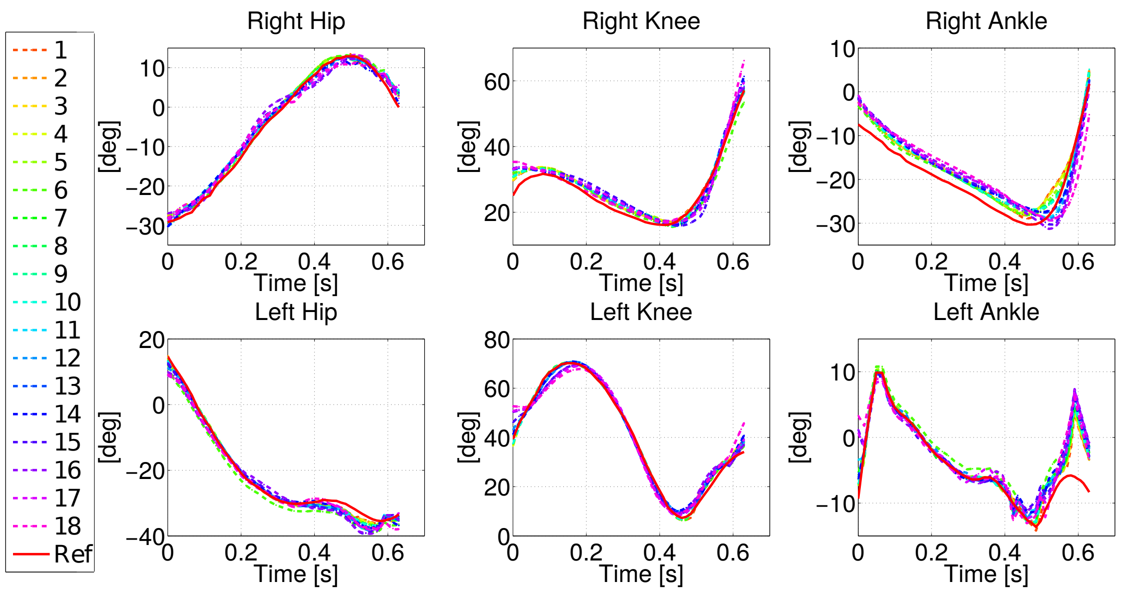


(a) Joint angles

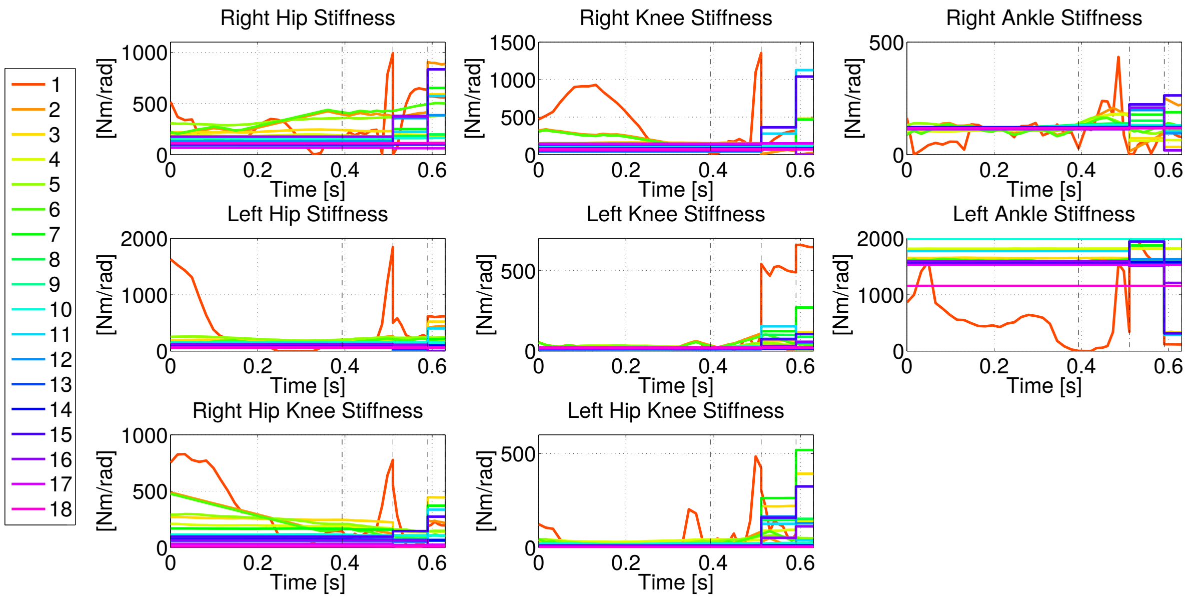


(b) Stiffness profiles

Subject M2

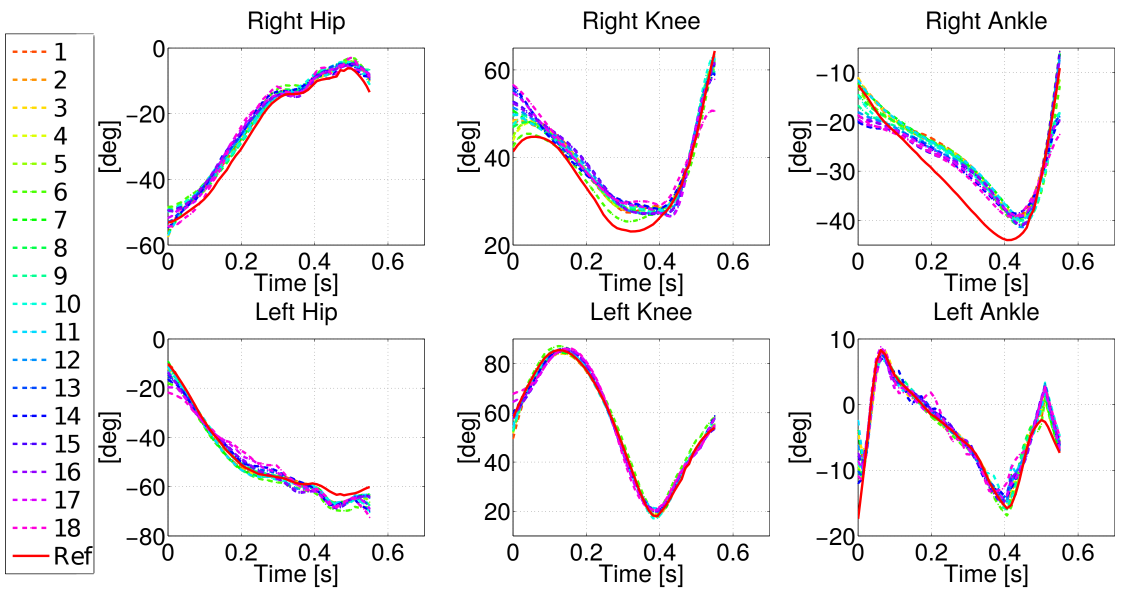


(a) Joint angles

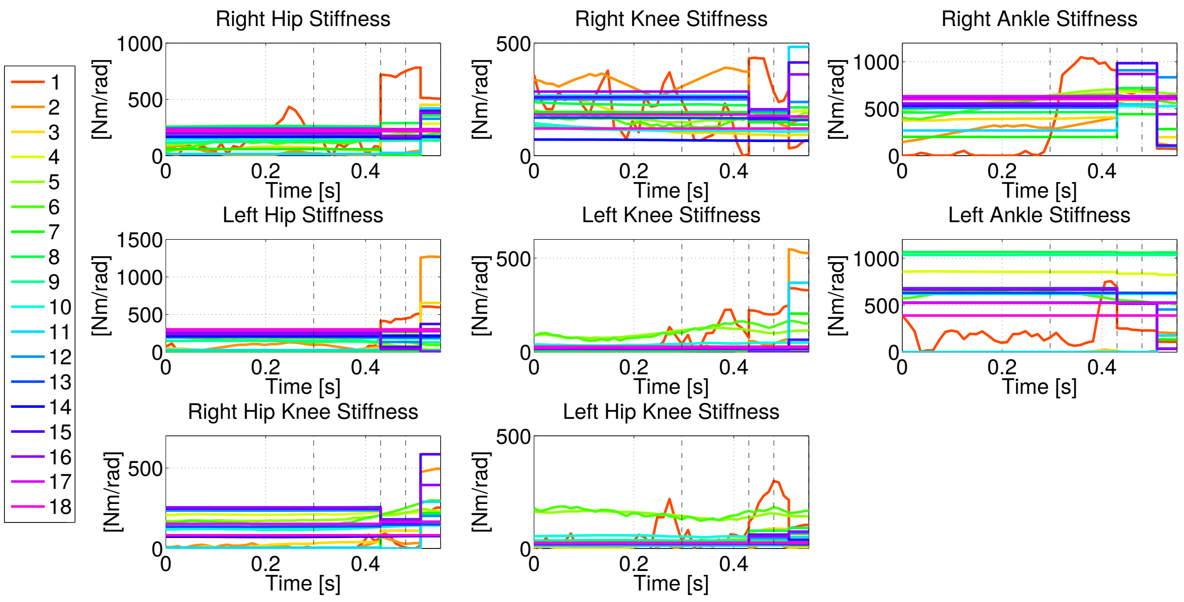


(b) Stiffness profiles

Subject M3

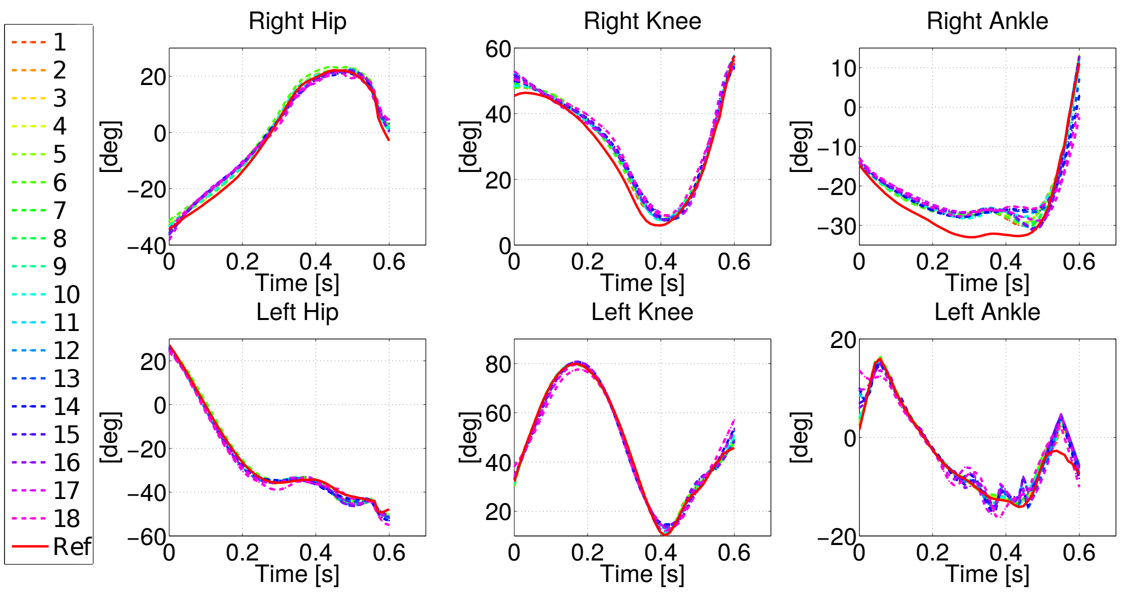


(a) Joint angles

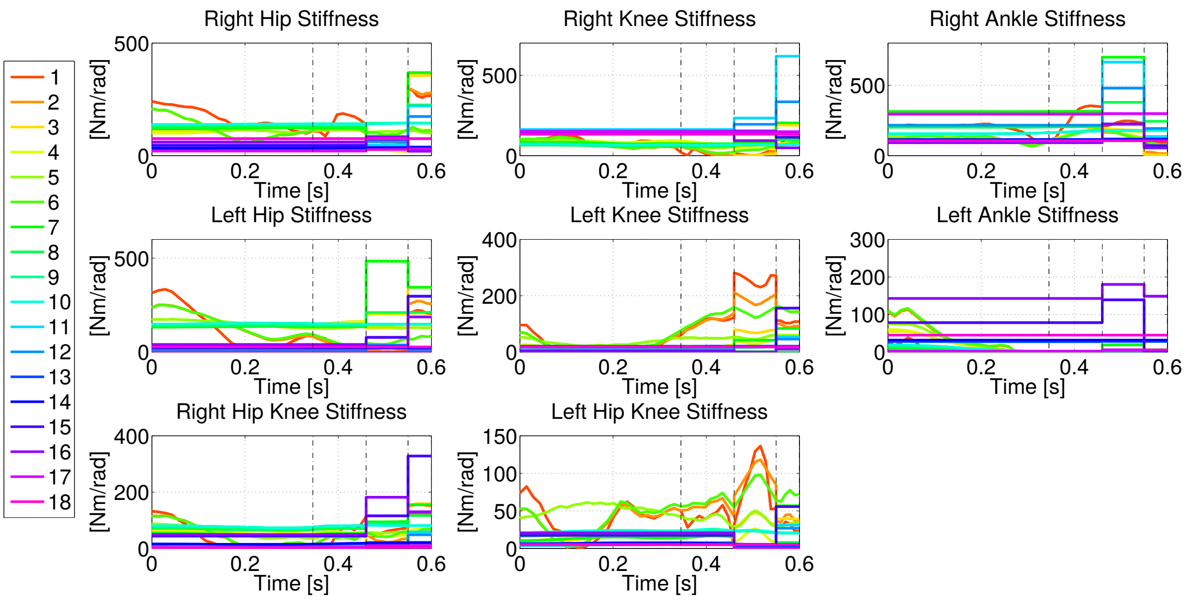


(b) Stiffness profiles

Subject M4



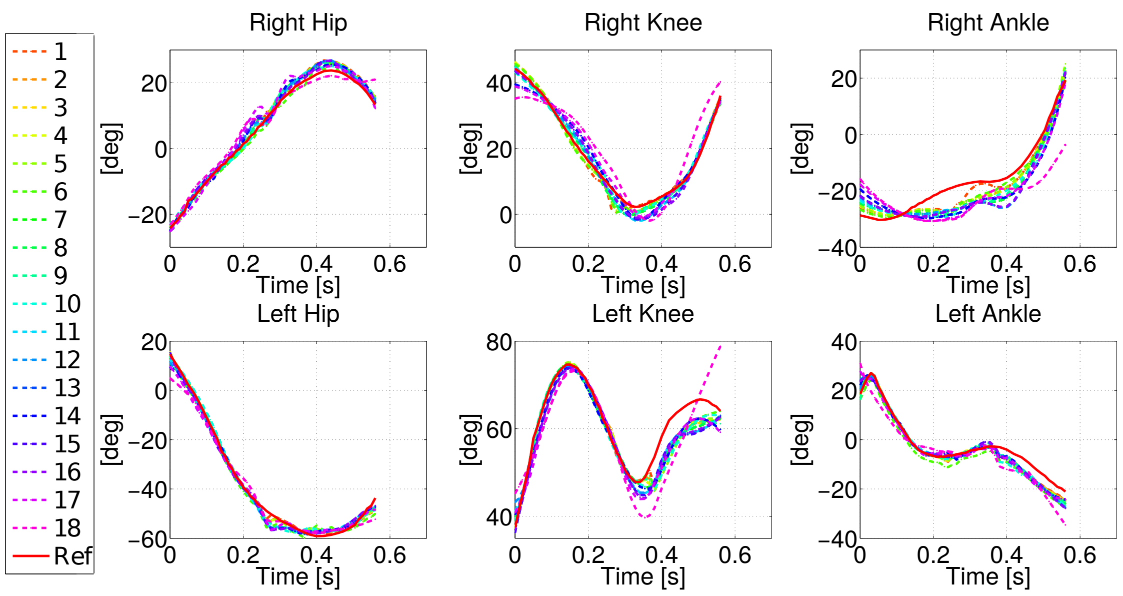
(a) Joint angles



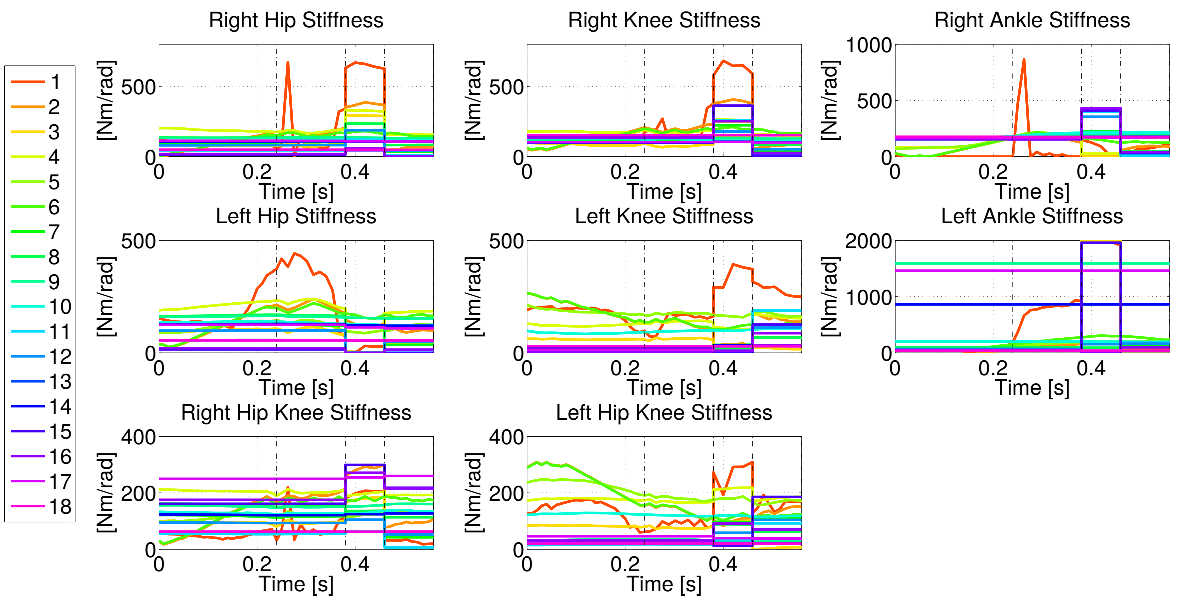
(b) Stiffness profiles

B.2 Slope up

Subject M1

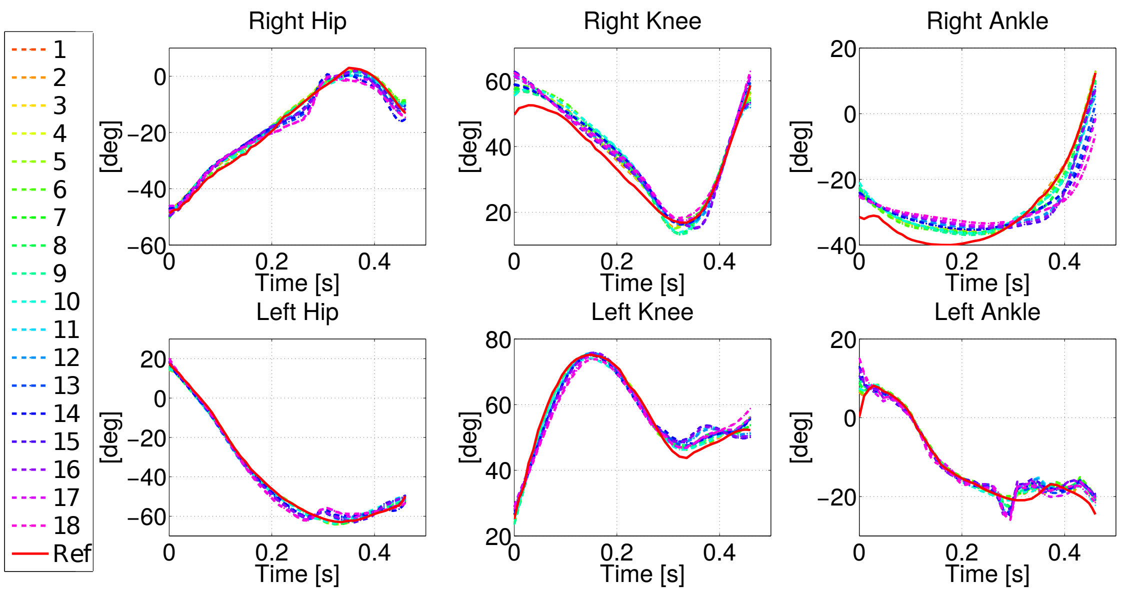


(a) Joint angles

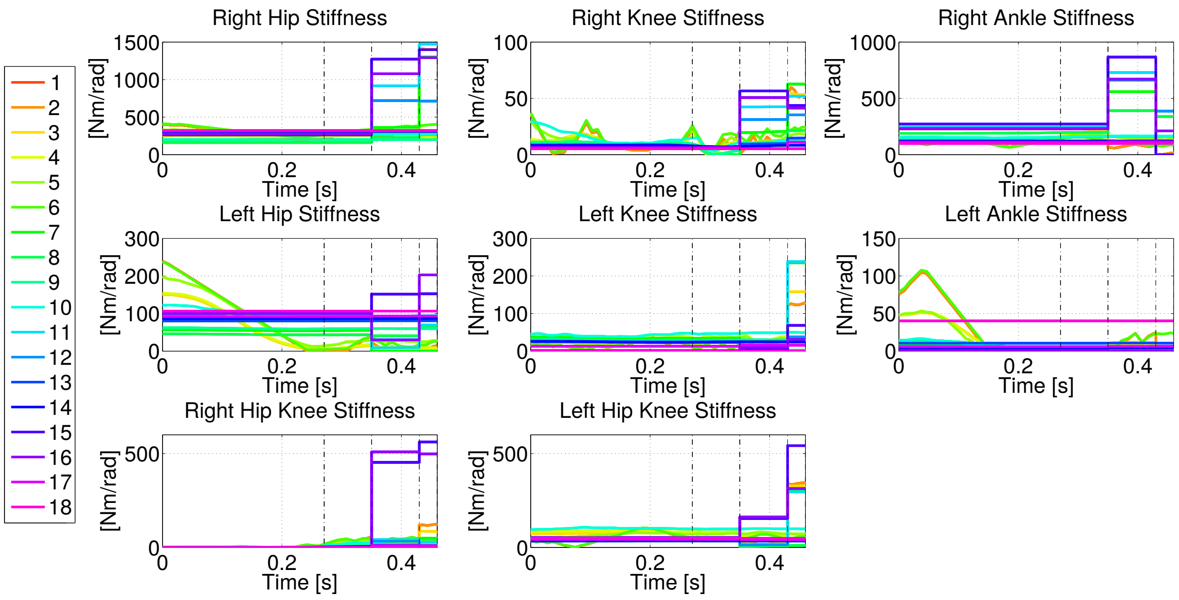


(b) Stiffness profiles

Subject M2

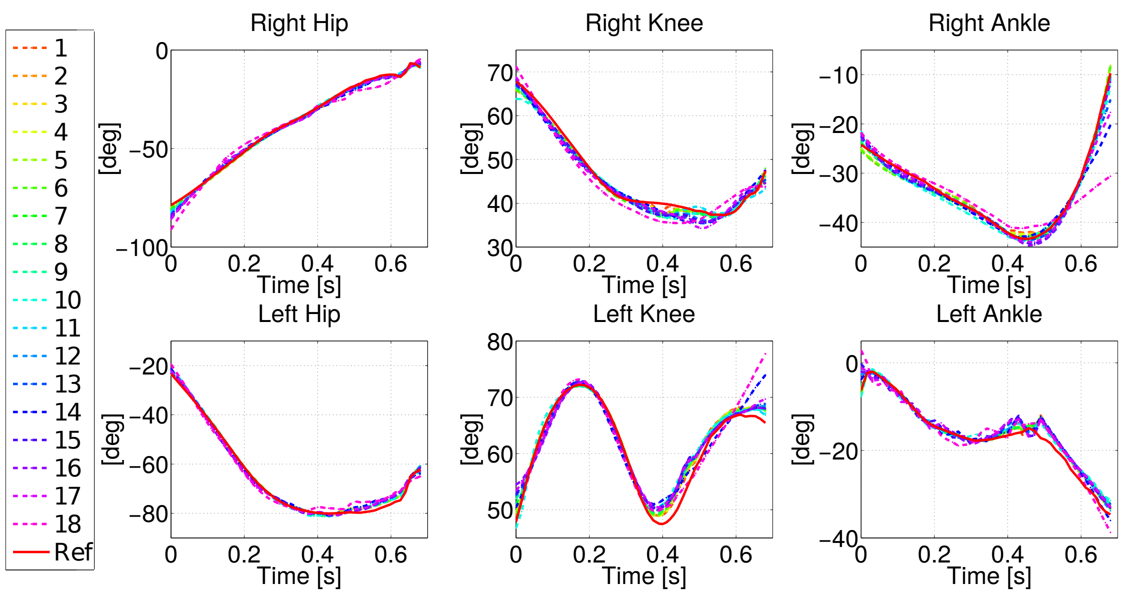


(a) Joint angles

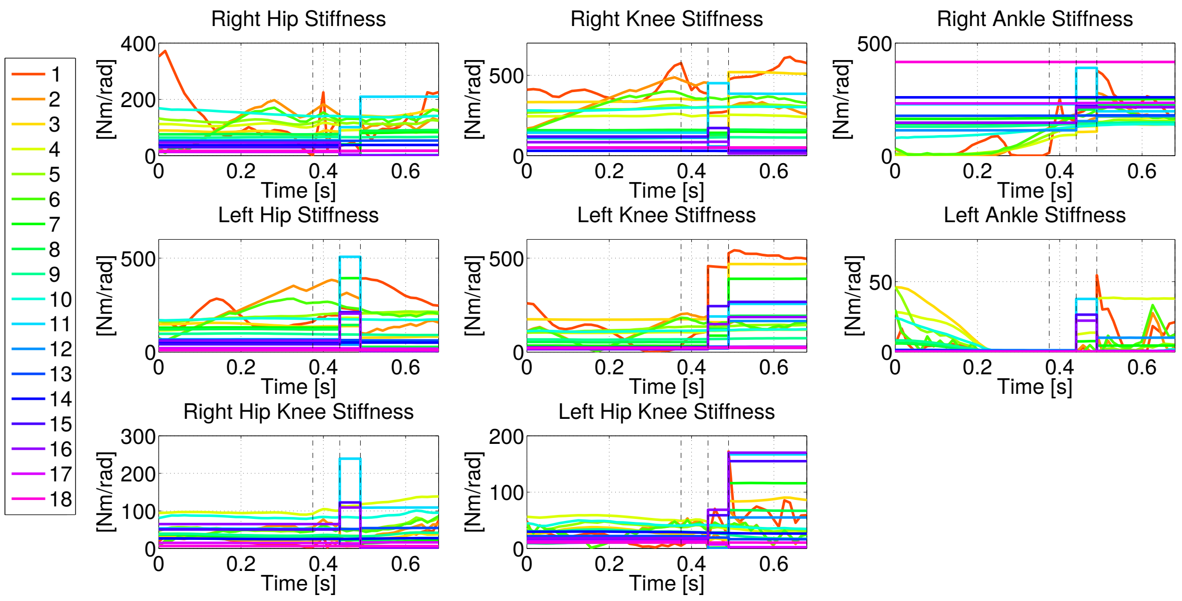


(b) Stiffness profiles

Subject M3

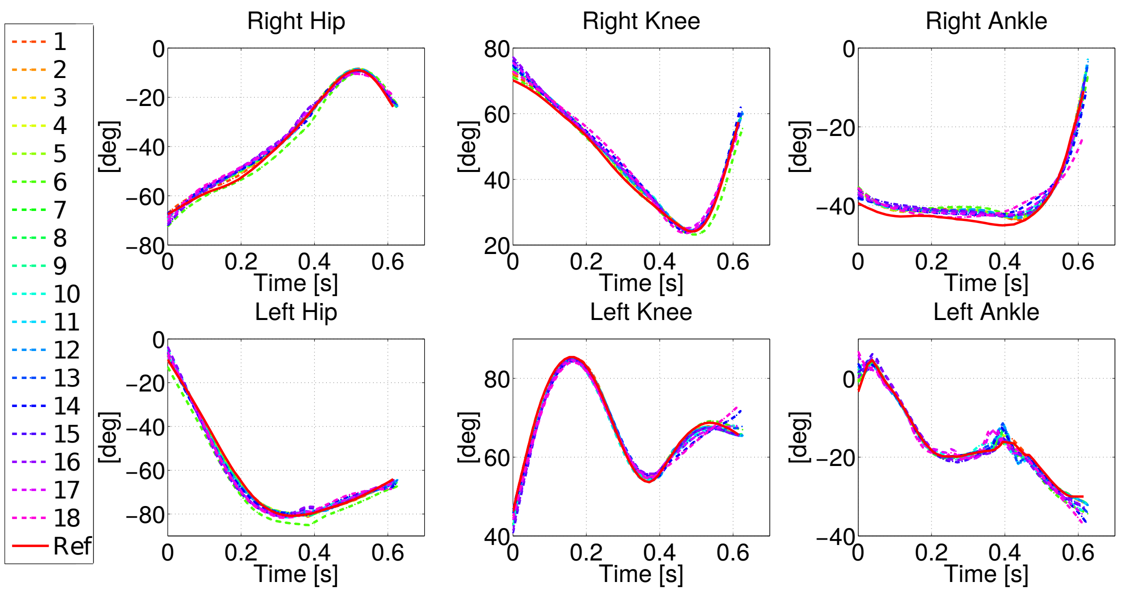


(a) Joint angles

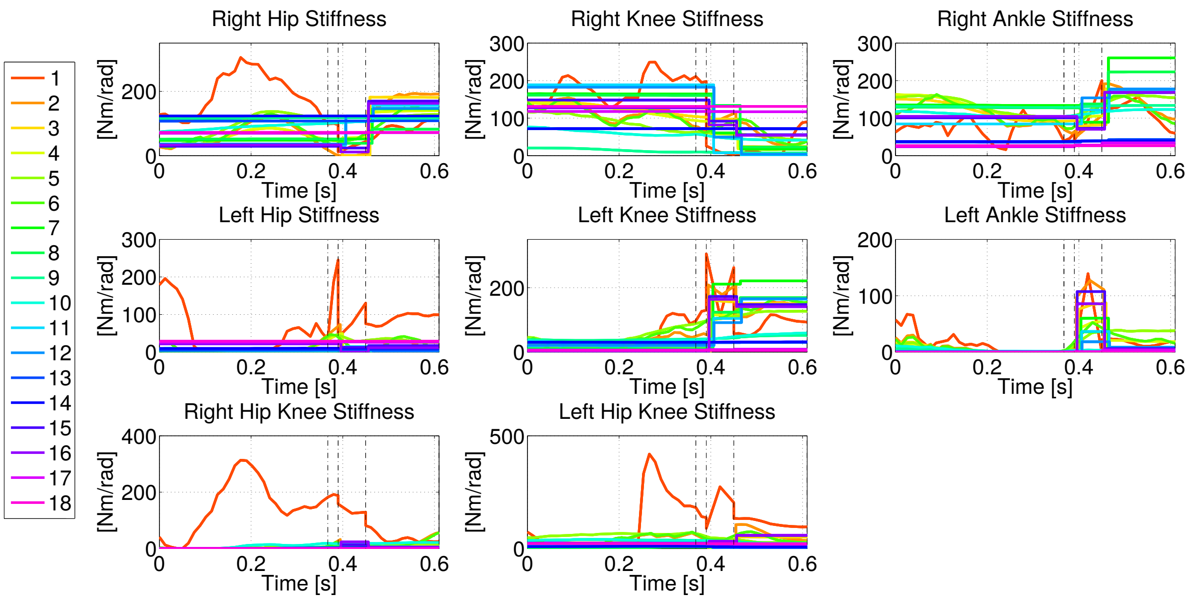


(b) Stiffness profiles

Subject M4



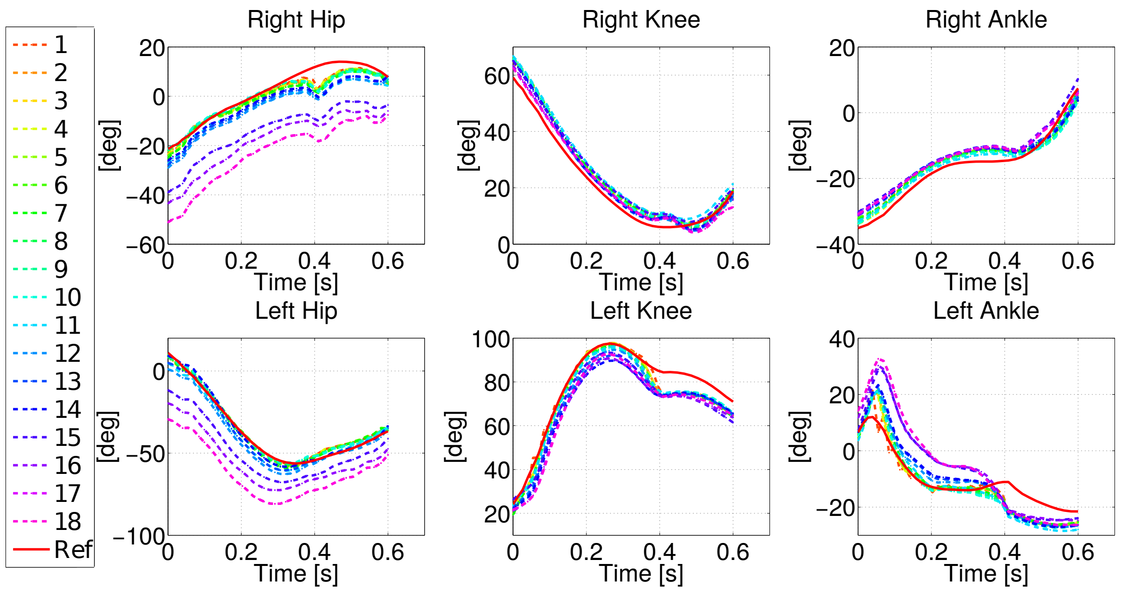
(a) Joint angles



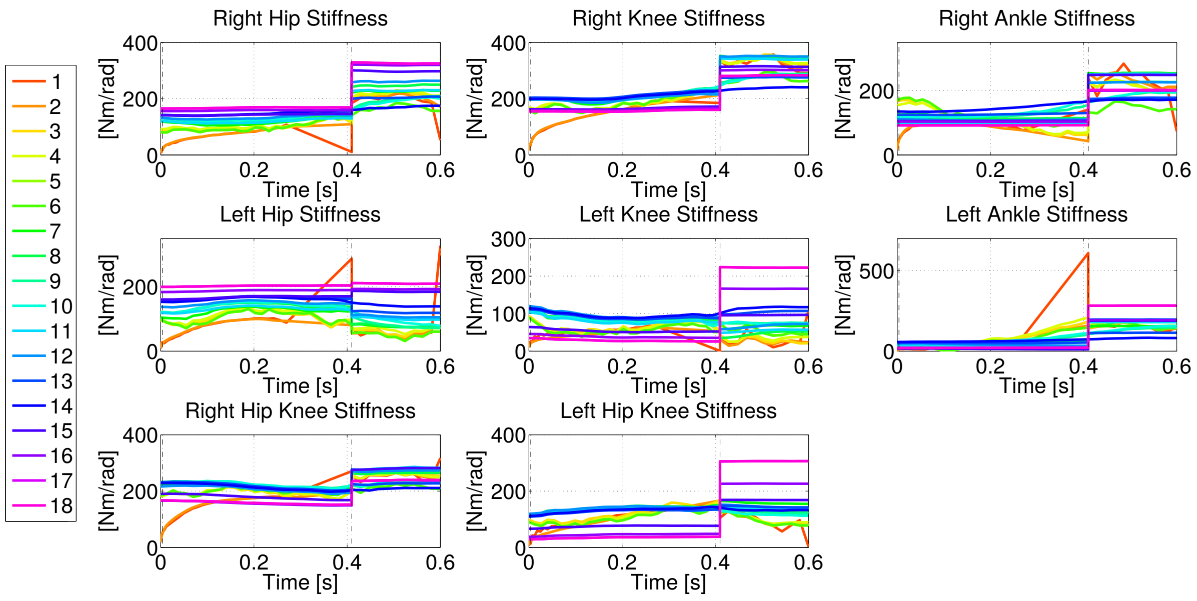
(b) Stiffness profiles

B.3 Stairs up

Subject M1

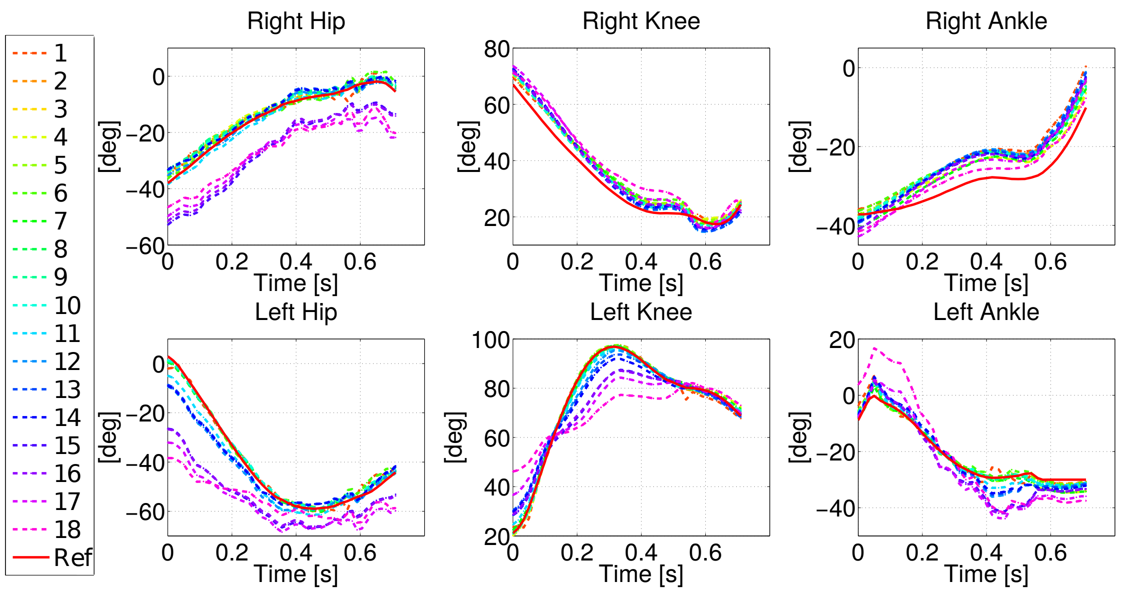


(a) Joint angles

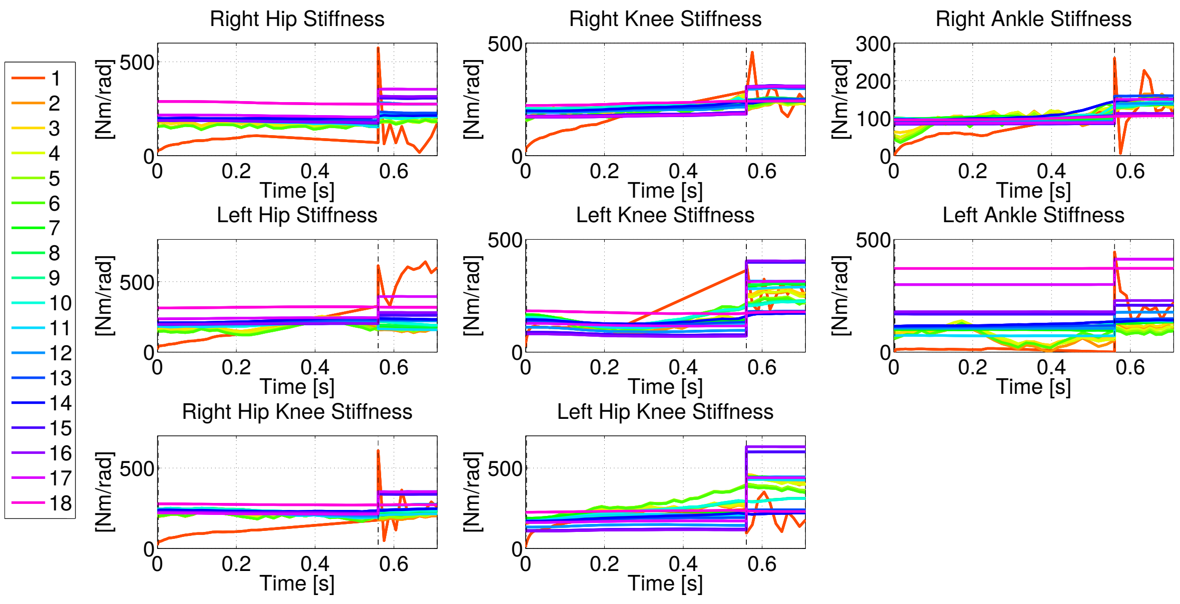


(b) Stiffness profiles

Subject M2

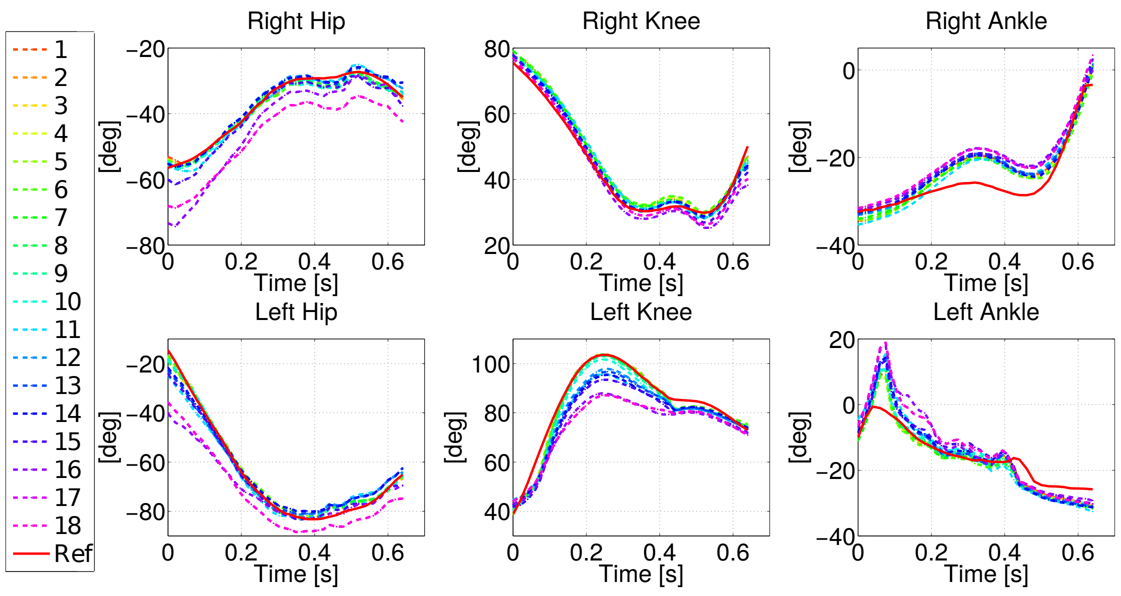


(a) Joint angles

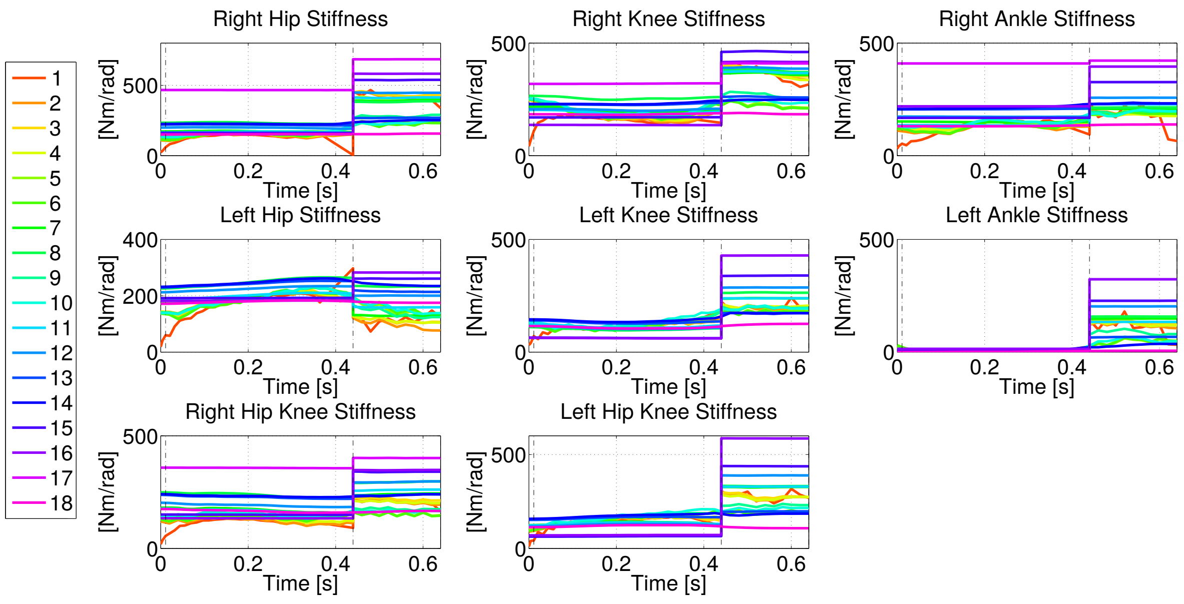


(b) Stiffness profiles

Subject M3

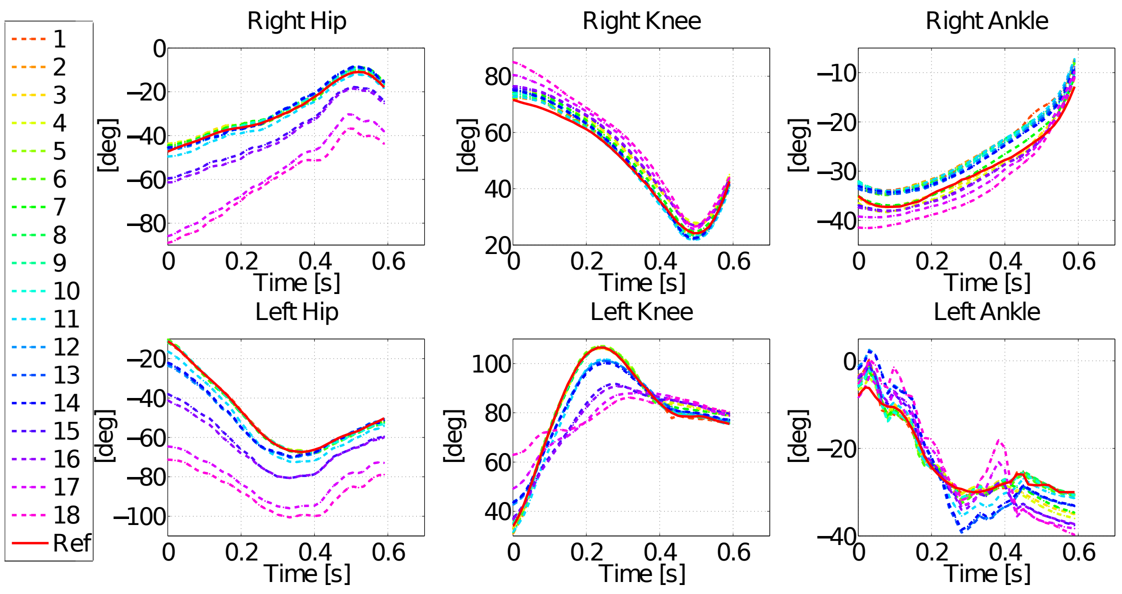


(a) Joint angles

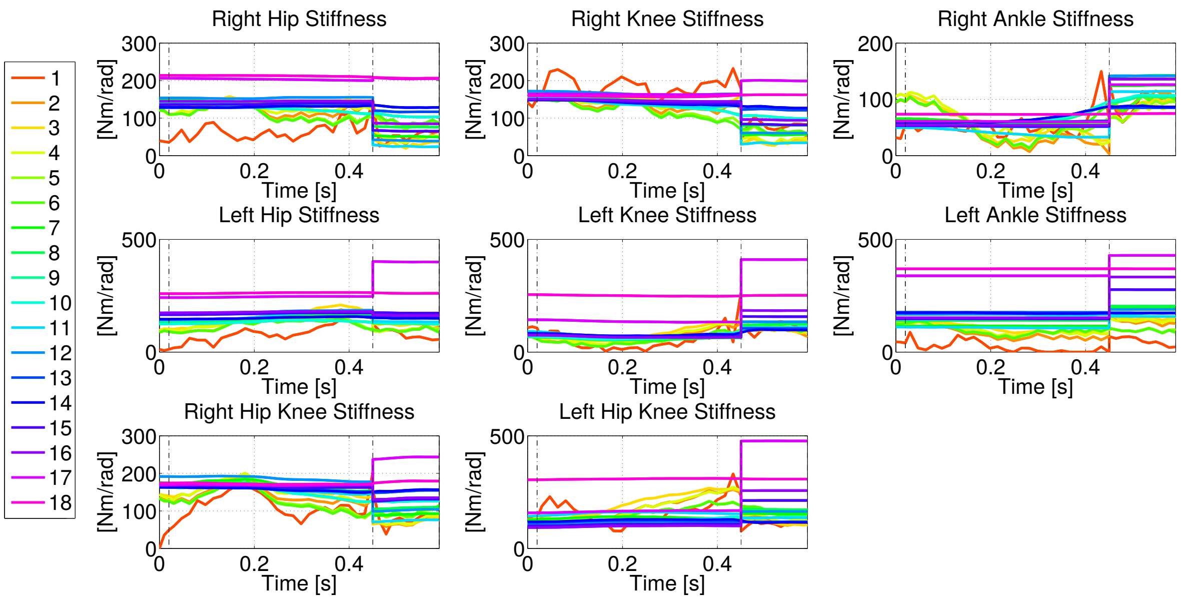


(b) Stiffness profiles

Subject M4



(a) Joint angles



(b) Stiffness profiles

Bibliography

[1] KoroiBot Motion Database. URL https://koroibot-motion-database.humanoids.kit.edu/.

[2] CoDyCo - Whole-body Compliant Dynamical Contacts in Cognitive Humanoids. URL http://codyco.eu/.

[3] codyco-modules. URL https://github.com/robotology/codyco-modules.

[4] DARPA Robotics Challenge. URL http://www.darpa.mil/program/darpa-robotics-challenge.

[5] Gazebo, robot simulation made easy. URL http://gazebosim.org/.

[6] Gazebo YARP Plugins. URL https://github.com/robotology/gazebo-yarp-plugins.

[7] KoroiBot EU project. URL http://www.koroibot.eu/.

[8] Unified Robot Description File (URDF). URL http://wiki.ros.org/urdf.

[9] Dynamic Interaction Control Lab - Fondazione Istituto Italiano di Tecnologia, Whole-BodyInterface. URL https://github.com/robotology/wholebodyinterface.

[10] T. Abuhashim and L. Natale. Robustness in view-graph SLAM. In *Information Fusion (FUSION), 2016 19th International Conference on*, pages 942–949. IEEE, 2016.

[11] R. M. Alexander. *Principles of animal locomotion*. Princeton University Press, 2003.

[12] A. N. Amirudin, S. Parasuraman, A. Kadirvel, M. A. Khan, and I. Elamvazuthi. Biomehcanics of hip, knee and ankle joint loading during ascent and descent walking. *Procedia Computer Science*, 42:336–344, 2014.

[13] T. Andriacchi, G. Andersson, R. Fermier, D. Stern, and J. Galante. A study of lower-limb mechanics during stair-climbing. *The Journal of Bone & Joint Surgery*, 62(5):749–757, 1980.

[14] R. Bellman. Dynamic programming and Lagrange multipliers. *Proceedings of the National Academy of Sciences*, 42(10):767–769, 1956.

[15] R. Bischoff, J. Kurth, G. Schreiber, R. Koeppe, A. Albu-Schäffer, A. Beyer, O. Eiberger, S. Haddadin, A. Stemmer, G. Grunwald, et al. The KUKA-DLR lightweight robot arm-a new reference platform for robotics research and manufacturing. In *Robotics (ISR), 2010 41st international symposium on and 2010 6th German conference on robotics (ROBOTIK)*, pages 1–8. VDE, 2010.

[16] H. Bock and K. Plitt. A multiple shooting algorithm for direct solution of optimal control problems. pages 243–247. Pergamon Press, 1984.

[17] D. Clever and K. Mombaur. A new template model for optimization studies of human walking on different terrains. In *IEEE/RAS International Conference on Humanoid Robots (Humanoids 2014)*, 2014.

[18] D. Clever, M. Harant, K. H. Koch, K. Mombaur, and D. M. Endres. A novel approach for the generation of complex humanoid walking sequences based on a combination of optimal control and learning of movement primitives. *Robotics and Autonomous Systems*, 2016. doi: 10.1016/j.robot.2016.06.001.

[19] L. Colasanto, N. Tsagarakis, and D. Caldwell. A compact model for the compliant humanoid robot COMAN. *IEEE International Conference on Biomedical Robotics and Biomechatronics*, pages 688–694, 2012.

[20] S. Dafarra, F. Romano, and F. Nori. Torque-controlled stepping-strategy push recovery: Design and implementation on the iCub humanoid robot. In *IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, pages 152–157. IEEE, 2016.

[21] H. Dallali, P. Kormushev, Z. Li, and D. Caldwell. On global optimization of walking gaits for the compliant humanoid robot, coman using reinforcement learning. *Cybernetics and Information Technologies*, 12(3):39–52, 2012.

[22] A. De Luca, B. Siciliano, and L. Zollo. PD control with on-line gravity compensation for robots with elastic joints: Theory and experiments. *Automatica*, 41(10):1809–1819, 2005.

[23] P. DeLeva. Adjustments to Zatsiorsky-Seluyanov's segment inertia parameters. *Journal of Biomechanics*, 29:1223–1230, 1996.

[24] J. Eljaik. *Multi-Modal Sensor Fusion in whole-body control for the iCub Humanoid Robot*. PhD thesis, University of Genoa, 2016.

[25] J. Eljaik, Z. Li, M. Randazzo, A. Parmiggiani, G. Metta, N. G. Tsagarakis, and F. Nori. Quantitative evaluation of standing stabilization using stiff and compliant actuators. In *Robotics: science and systems*, 2013.

[26] J. Englsberger, C. Ott, M. A. Roa, A. Albu-Schäffer, and G. Hirzinger. Bipedal walking control based on capture point dynamics. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4420–4427. IEEE, 2011.

[27] J. Englsberger, C. Ott, and A. Albu-Schäffer. Three-dimensional bipedal walking control based on divergent component of motion. *IEEE Transactions on Robotics*, 31(2):355–368, 2015.

[28] R. Featherstone. *Rigid Body Dynamics Algorithms*. Springer, 2007.

[29] A. G. Feldman and M. F. Levin. The equilibrium-point hypothesis–past, present and future. In *Progress in motor control*, pages 699–726. Springer, 2009.

[30] M. L. Felis. RBDL - Rigid Body Dynamics Library, 2012-2017. URL `http://rbdl.bitbucket.org/`.

[31] M. L. Felis, K. D. Mombaur, and A. Berthoz. An optimal control approach to reconstruct human gait dynamics from kinematic data. *IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, 2015.

[32] M. L. Felis. *Modeling Emotional Aspects in Human Locomotion*. PhD thesis, Heidelberg University, 2015.

[33] M. L. Felis. RBDL: an efficient rigid-body dynamics library using recursive algorithms. *Autonomous Robots*, pages 1–17, 2016. ISSN 1573-7527. doi: 10.1007/s10514-016-9574-0. URL http://dx.doi.org/10.1007/s10514-016-9574-0.

[34] M. L. Felis and K. Mombaur. Using optimal control methods to generate human walking motions. In M. Kallmann and K. Bekris, editors, *Motion in Games*, volume 7660 of *Lecture Notes in Computer Science*, pages 197–207. Springer Berlin Heidelberg, 2012. ISBN 978-3-642-34709-2.

[35] H. Ferreau, C. Kirches, A. Potschka, H. Bock, and M. Diehl. qpOASES: A parametric active-set algorithm for quadratic programming. *Mathematical Programming Computation*, 6(4):327–363, 2014.

[36] D. P. Ferris, M. Louie, and C. T. Farley. Running in the real world: adjusting leg stiffness for different surfaces. *Proceedings of the Royal Society of London B: Biological Sciences*, 265(1400):989–994, 1998.

[37] P. Fitzpatrick, G. Metta, and L. Natale. Towards Long- Lived Robot Genes. *Robotics and Autonomous Systems, Elsevier*, 56(1):29–45, 2008.

[38] J. R. Franz, N. E. Lyddon, and R. Kram. Mechanical work performed by the individual legs during uphill and downhill walking. *Journal of Biomechanics*, 45:257–262, 2012.

[39] R. C. Gabriel, J. Abrantes, K. Granata, J. Bulas-Cruz, P. Melo-Pinto, and V. Filipe. Dynamic joint stiffness of the ankle during walking: Gender-related differences. *Physical Therapy in sport*, 9:16–24, 2008.

[40] M. Garcia, A. Chatterjee, A. Ruina, and M. Coleman. The simplest walking model: stability, complexity, and scaling. *Journal of biomechanical engineering*, 120(2):281–288, 1998.

[41] H. Geyer, A. Seyfarth, and R. Blickhan. Compliant leg behaviour explains basic dynamics of walking and running. *Proceedings of the royal society B*, pages 2861–2867, 2006.

[42] M. Günther and R. Blickhan. Joint stiffness of the ankle and the knee in running. *Journal of Biomechanics*, 35:1459–1474, 2002.

[43] A. Herdt, N. Perrin, and P.-B. Wieber. Walking without thinking about it. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2010.

[44] H. Hirukawa, F. Kanehiro, K. Kaneko, S. Kajita, K. Fujiwara, Y. Kawai, F. Tomita, S. Hirai, K. Tanie, T. Isozumi, et al. Humanoid robotics platforms developed in HRP. *Robotics and Autonomous Systems*, 48(4):165–175, 2004.

[45] K. L. Ho-Hoang and K. Mombaur. Adjustments to De Leva-anthropometric regression data for the changes in body proportions in elderly humans. *Journal of Biomechanics*, 48(13):3732–3736, 2015.

[46] E. M. Hoffman, S. Traversaro, A. Rocchi, M. Ferrati, A. Settimi, F. Romano, L. Natale, A. Bicchi, F. Nori, and N. G. Tsagarakis. YARP based plugins for Gazebo simulator. In *Modelling and Simulation for Autonomous Systems*, pages 333–346. Springer, 2014.

[47] N. Hogan. Adaptive control of mechanical impedance by coactivation of antagonist muscles. *IEEE Transactions on Automatic Control*, 29(8):681–690, 1984.

[48] Y. Hu, K. Mombaur, and F. Nori. Using optimal control to generate squat motions for the humanoid robot icub with sea. In *Dynamic Walking Conference 2016 (Ohio)*. URL http://biomechanics.osu.edu/dynamic-walking/AbstractsFolder/Hu_2015_DW.pdf.

[49] Y. Hu, M. L. Felis, and K. Mombaur. Compliance analysis of human leg joints in level ground walking with an optimal control approach. In *IEEE-RAS International Conference on Humanoid Robots (Humanoids 2014)*, 2014.

[50] Y. Hu, F. Nori, and K. Mombaur. Squat motion generation for the humanoid robot iCub with Series Elastic Actuators. In *Biomedical Robotics and Biomechatronics (BioRob), 2016 6th IEEE International Conference on*, pages 207–212. IEEE, 2016.

[51] J. W. Hurst. *The role and implementation of compliance in legged locomotion*. ProQuest, 2008.

[52] F. Iida, J. Rummel, and A. Seyfarth. Bipedal walking and running with spring-like biarticular muscles. *Journal of Biomechanics*, 41:656–667, 2008.

[53] S. Ivaldi, J. Peters, V. Padois, and F. Nori. Tools for simulating humanoid robot dynamics: a survey based on user feedback. In *2014 IEEE-RAS International Conference on Humanoid Robots*, pages 842–849. IEEE, 2014.

[54] S. Kagami, T. Kitagawa, K. Nishiwaki, T. Sugihara, M. Inaba, and H. Inoue. A fast dynamically equilibrated walking trajectory generation method of humanoid robot. *Autonomous Robots*, 12(1):71–82, 2002.

[55] S. Kajita and K. Tani. Study of dynamic biped locomotion on rugged terrain-derivation and application of the linear inverted pendulum mode. In *Robotics and Automation, 1991. Proceedings., 1991 IEEE International Conference on*, pages 1405–1411. IEEE, 1991.

[56] S. Kajita, F. Kanehiro, K. Kaneko, K. Fujiwara, K. Yokoi, and H. Hirukawa. A Realtime Pattern Generator for Biped Walking. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2002.

[57] S. Kajita, F. Kanehiro, K. Kaneko, K. Fujiwara, K. Harada, K. Yokoi, and H. Hirukawa. Biped walking pattern generation by using preview control of zero-moment point. In *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*, volume 2, pages 1620–1626. IEEE, 2003.

[58] S. Kajita, H. Hirukawa, K. Harada, and K. Yokoi. *Introduction to Humanoid Robotics*, volume 101 of *Springer Tracts in Advanced Robotics*. Springer, 2014.

[59] I. Kato. Development of WABOT 1. *Biomechanism*, 2:173–214, 1973.

[60] J. Ketchel and P. Larochelle. Collision detection of cylindrical rigid bodies for motion planning. In *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, pages 1530–1535. IEEE, 2006.

[61] W. Khalil and E. Dombre. *Modeling, Identification and Control of Robots.* Taylor & Francis, Inc., Bristol, PA, USA,, 3rd edition, 2002.

[62] J. Kim, Y. Lee, S. Kwon, K. Seo, H. Kwak, H. Lee, and K. Roh. Development of the lower limbs for a humanoid robot. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 4000–4005. IEEE, 2012.

[63] K. H. Koch, K. Mombaur, P. Souères, and O. Stasse. Optimization based exploitation of the ankle elasticity of HRP-2 for overstepping large obstacles. In *IEEE/RAS International Conference on Humanoid Robots (Humanoids 2014)*, 2014.

[64] S. D. Lark, J. G. Buckley, S. Bennet, and A. J. Sargeant. Joint torques and dynamic joint stiffness in elderly and young men during stepping down. *Clinical Biomechanics*, 18(9): 848–855, 2003.

[65] M. L. Latash and V. M. Zatsiorsky. Joint stiffness: Myth or reality? *Human movement science*, 12:653–692, 1993.

[66] M. L. Latash, M. F. Levin, J. P. Scholz, and G. Schöner. Motor control theories and their applications. *Medicina (Kaunas, Lithuania)*, 46(6):382, 2010.

[67] S. Lee, J. Kim, F. Park, M. Kim, and J. Bobrow. Newton-type algorithms for dynamics-based robot movement optimization. *IEEE Trans. Robotics 21(4)*, pages 657–667, 2005.

[68] D. Leinweber, I. Bauer, H. Bock, and J. Schloeder. An efficient multiple shooting based reduced SQP strategy for large-scale dynamic process optimization. part I: Theoretical aspects. *Computers and Chemical Engineering*, 27:157–166, 2003.

[69] K. Levenberg. A method for the solution of certain non-linear problems in least squares. *Quarterly of applied mathematics*, 2(2):164–168, 1944.

[70] Z. Li, N. G. Tsagarakis, and D. G. Caldwell. Walking trajectory generation for humanoid robots with compliant joints: Experimentation with COMAN humanoid. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 836–841. IEEE, 2012.

[71] Y. Liu, P. M. Wensing, D. E. Orin, and Y. F. Zheng. Trajectory generation for dynamic walking in a humanoid over uneven terrain using a 3D-actuated dual-SLIP model. In *IEEE/RSJ IROS*, pages 374–380, Sept 2015. doi: 10.1109/IROS.2015.7353400.

[72] C. Loughlin, A. Albu-Schäffer, S. Haddadin, C. Ott, A. Stemmer, T. Wimböck, and G. Hirzinger. The DLR lightweight robot: design and control concepts for robots in human environments. *Industrial Robot: an international journal*, 34(5):376–385, 2007.

[73] D. W. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *Journal of the society for Industrial and Applied Mathematics*, 11(2):431–441, 1963.

[74] T. McGeer. Passive dynamic walking. *The international journal of robotics research*, 9 (2):62–82, 1990.

[75] G. Metta, L. Natale, F. Nori, G. Sandini, D. Vernon, L. Fadiga, C. Von Hofsten, K. Rosander, M. Lopes, J. Santos-Victor, et al. The iCub humanoid robot: An open-systems platform for research in cognitive development. *Neural Networks*, 23(8):1125–1134, 2010.

[76] M. Moltedo, T. Bacek, K. Junius, B. Vanderborght, and D. Lefeber. Mechanical design of a lightweight compliant and adaptable active ankle foot orthosis. In *Biomedical Robotics and Biomechatronics (BioRob), 2016 6th IEEE International Conference on*, pages 1224–1229. IEEE, 2016.

[77] K. Mombaur. A study on optimal compliance in running. In *Dynamic Walking Conference, Zürich*, 2014. URL http://dynamicwalking.org/ocs/index.php/dw2014/dw2014/paper/viewFile/129/83.

[78] K. Mombaur, R. Longman, H. Bock, and J. Schlöder. Open-loop stable running. *Robotica 23(1)*, 2005.

[79] F. L. Moro, N. G. Tsagarakis, and D. G. Caldwell. Walking in the resonance with the COMAN robot with trajectories based on human kinematic motion primitives (kmps). *Autonomous Robots*, 36(4):331–347, 2014.

[80] A. Mukovskiy, C. Vassallo, M. Naveau, O. Stasse, P. Soueres, and M. Giese. Adaptive synthesis of dynamically feasible full body movements for the humanoid robot HRP-2 by flexible combination of learned dynamic movement primitives. *Robotics and Autonomous Systems*, 2016, submitted.

[81] G. Nava, D. Pucci, S. Traversaro, F. Romano, L. Muratore, L. Natale, N. Tsagarakis, and F. Nori. Porting highly dynamic balancing from iCub to WALK-MAN. In *The use of dynamics in the field of humanoid robotics: identification, planning, perception and control, Workshop, IEEE-RAS International Conference on Humanoid Robots (Humanoids 2016)*, 2016.

[82] M. Naveau, M. Kudruss, O. Stasse, C. K. andKatja Mombaur, and P. Souéres. A reactive walking pattern generator based on nonlinear model predictive control. *IEEE Robotics and Automation Letters*, 2(1):10–17, 2017.

[83] F. Nori, S. Traversaro, J. Eljaik, F. Romano, A. Del Prete, and D. Pucci. iCub Whole-body Control through Force Regulation on Rigid Noncoplanar Contacts. *Frontiers in Robotics and AI*, 2(6), 2015. ISSN 2296-9144. doi: 10.3389/frobt.2015.00006.

[84] C. Ott, C. Baumgärtner, J. Mayr, M. Fuchs, R. Burger, D. Lee, O. Eiberger, A. Albu-Schäffer, M. Grebenstein, and G. Hirzinger. Development of a biped robot with torque controlled joints. In *2010 10th IEEE-RAS International Conference on Humanoid Robots*, pages 167–173. IEEE, 2010.

[85] A. Parmiggiani, G. Metta, and N. Tsagarakis. The mechatronic design of the new legs of the icub robot. In *Humanoid Robots (Humanoids), 2012 12th IEEE-RAS International Conference on*, pages 481–486. IEEE, 2012.

[86] L. S. Pontryagin. *Mathematical theory of optimal processes*. CRC Press, 1987.

[87] G. A. Pratt and M. M.Williamson. Series elastic actuators. *1995 IEEE/RSJ International Conference on Intelligent Robots and Systems. Human Robot Interaction and Cooperative Robots*, pages 399–406, 1995.

[88] J. Pratt and B. Krupp. Design of a bipedal walking robot. In *SPIE Defense and Security Symposium*, pages 69621F–69621F. International Society for Optics and Photonics, 2008.

[89] J. Pratt, C.-M. Chew, A. Torres, P. Dilworth, and G. Pratt. Virtual Model COntrol: An intuitive approach for bipedal locomotion. *International Journal of Robotics Research*, pages 129–143, 2001.

[90] J. Pratt, J. Carff, S. Drakunov, and A. Goswami. Capture point: A step toward humanoid push recovery. In *2006 6th IEEE-RAS international conference on humanoid robots*, pages 200–207. IEEE, 2006.

[91] J. Pratt, T. Koolen, T. D. Boer, J. Rebula, S. Cotton, J. Carff, M. Johnson, and P. Neuhaus. Capturability-based analysis and control of legged locomotion, part 2: Application to M2V2, a lower-body humanoid. *International Journal of Robotics Research*, pages 1117–1133, 2012.

[92] M. H. Raibert. *Legged robots that balance*. MIT press, 1986.

[93] F. Romano, N. Kuppuswamy, M. Ciocca, S. Traversaro, and F. Nori. Stable bipedal foot planting on uneven terrain through optimal ankle impedance. In *IEEE-RAS. International Conference. on Humanoid Robots*, 2016.

[94] F. Romano, S. Traversaro, D. Pucci, A. D. Prete, J. Eljaik, and F. Nori. A whole-body software abstraction layer for control design of free-floating mechanical systems. In *1st IEEE International Conference on Robotic Computing*, 2017.

[95] Y. Sakagami, R. Watanabe, C. Aoyama, S. Matsunaga, N. Higaki, and K. Fujimura. The intelligent ASIMO: System overview and integration. In *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, volume 3, pages 2478–2483. IEEE, 2002.

[96] G. Sandini, G. Metta, and D. Vernon. Robotcub: An open framework for research in embodied cognition. In *Humanoid Robots, 2004 4th IEEE/RAS International Conference on*, volume 1, pages 13–32. IEEE, 2004.

[97] R. M. Schemschat, D. Clever, M. L. Felis, E. Chiovetto, M. Giese, and K. Mombaur. Joint torque analysis of push recovery motions during human walking. In *Biomedical Robotics and Biomechatronics (BioRob), 2016 6th IEEE International Conference on*, pages 133–139. IEEE, 2016.

[98] R. M. Schemschat, D. Clever, M. L. Felis, and K. Mombaur. Optimal push recovery for periodic walking motions. *IFAC-PapersOnLine*, 49(14):93–98, 2016.

[99] A. Schubert, D. Clever, and K. Mombaur. Key performance indicators for humanoid walking motions defined in the KoroiBot project. In *International Workshop on Wearable Robotics (WeRob 2014)*, 2014.

[100] R. Sellaouti, O. Stasse, S. Kajita, K. Yokoi, and A. Kheddar. Faster and smoother walking of humanoid HRP-2 with passive toe joints. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4909–4914. IEEE, 2006.

[101] K. Shamaei, G. S. Sawicki, and A. M.Dollar. Estimation of quasi-stiffness and propulsion work of the human ankle in the stance phase of walking. *PLOS ONE*, 8, 2013.

[102] K. Shamaei, G. S. Sawicki, and A. M.Dollar. Estimation of quasi-stiffness of the human hip in the stance phase of walking. *PLOS ONE*, 8, 2013.

[103] K. Shamaei, G. S. Sawicki, and A. M.Dollar. Estimation of quasi-stiffness of the human knee in the stance phase of walking. *PLOS ONE*, 8, 2013.

[104] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo. *Robotics: modelling, planning and control*. Springer Science & Business Media, 2010.

[105] A. Silder, T. Besier, and S. L. Delp. Predicting the metabolic cost of incline walking from muscle activity and walking mechanics. *Journal of Biomechanics*, 45:1842–1849, 2012.

[106] K. Stein. *Walking Control Framework for the Humanoid Robot HeiCub - Implementation and Experimental Validation*. Master thesis, Heidelberg University, 2016.

[107] B. J. Stephens and C. G.Atkeson. Dynamic balance force control for compliant humanoid robots. *IEEE-International Conference on Intelligent Robots and Systems*, pages 1248–1255, 2010.

[108] S. Sugano and I. Kato. WABOT-2: Autonomous robot with dexterous finger-arm–finger-arm coordination control in keyboard performance. In *Robotics and Automation. Proceedings. 1987 IEEE International Conference on*, volume 4, pages 90–97. IEEE, 1987.

[109] T. Sugihara. Solvability-Unconcerned Inverse Kinematics by the Levenberg- Marquardt Method. *IEEE Transactions on Robotics*, 27:984–991, 2011.

[110] Ö. Terlemez, S. Ulbrich, C. Mandery, M. Do, N. Vahrenkamp, and T. Asfour. Master Motor Map (MMM) - framework and toolkit for capturing, representing, and reproducing human motion on humanoid robots. In *IEEE-RAS International Conference on Humanoid Robots (Humanoids 2014)*, 2014.

[111] D. Tolani, A. Goswami, and N. I. Badler. Real-time inverse kinematics techniques for anthropomorphic limbs. *Graphical models*, 62(5):353–388, 2000.

[112] G. Tonietti, R. Schiavi, and A. Bicchi. Design and control of a variable stiffness actuator for safe and fast physical human/robot interaction. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pages 526–531. IEEE, 2005.

[113] N. G. Tsagarakis, D. G. Caldwell, A. Bicchi, F. Negrello, M. Garabini, W. Choi, L. Baccelliere, V. Loc, J. Noorden, M. Catalano, et al. WALK-MAN: A high performance humanoid platform for realistic environments. *Journal of Field Robotics (JFR)*, 2016.

[114] R. Van Ham, B. Vanderborght, M. Van Damme, B. Verrelst, and D. Lefeber. Maccepa, the mechanically adjustable compliance and controllable equilibrium position actuator: Design and implementation in a biped robot. *Robotics and Autonomous Systems*, 55(10): 761–768, 2007.

[115] B. Verrelst, R. Van Ham, B. Vanderborght, F. Daerden, D. Lefeber, and J. Vermeulen. The pneumatic biped "Lucy" actuated with pleated pneumatic artificial muscles. *Autonomous Robots*, 18(2):201–213, 2005.

[116] B. Verrelst, O. Stasse, K. Yokoi, and B. Vanderborght. Dynamically stepping over obstacles by the humanoid robot hrp-2. In *2006 6th IEEE-RAS International Conference on Humanoid Robots*, pages 117–123. IEEE, 2006.

[117] M. Vukobratović and B. Borovac. Zero-moment point—thirty five years of its life. *International Journal of Humanoid Robotics*, 1(01):157–173, 2004.

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

[119] A. Wächter and L. T. Biegler. On the Implementation of a Primal-Dual Interior Point Filter Line Search Algorithm for Large-Scale Nonlinear Programming. *Mathematical Programming*, 106(1):25–27, 2006.

[120] P. L. Weiss, R. E. Kearney, and L. W. Hunter. Position dependence of ankle joint dynamics-I. Passive mechanics. *Biomechanics*, 19:727–735, 1986.

[121] P. L. Weiss, R. E. Kearney, and L. W. Hunter. Position dependence of ankle joint dynamics-II. Active mechanics. *Biomechanics*, 19:737–751, 1986.

[122] S. C.-B. Wieber. Stability and regulation of nonsmooth dynamical systems. 2004.

[123] D. Winter. *Biomechanics and Motor Control of Human Movement*. John Wiley & Sons, 2009.

[124] M. Wisse, A. L. Schwab, and F. C. van der Helm. Passive dynamic walking model with upper body. *Robotica*, 22(06):681–688, 2004.

[125] D.-G. Zhang and J. Angeles. Impact dynamics of flexible-joint robots. *Computers & structures*, 83(1):25–33, 2005.